Sustav za prikaz podataka o profilu vozaču vozila ispred vlastitog vozila zasnovan na prepoznavanju tablica vozila

Kovačević, Mihovil

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Diplomski sveučilišni studij Računarstvo

SUSTAV ZA PRIKAZ PODATAKA O PROFILU VOZAČA VOZILA ISPRED VLASTITOG VOZILA ZASNOVAN NA DETEKCIJI I PREPOZNAVANJU REGISTARKIH TABLICA VOZILA

Diplomski rad

Mihovil Kovačević

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 13.09.2023.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Mihovil Kovačević
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D-1214R, 07.10.2021.
OIB studenta:	35904356551
Mentor:	prof. dr. sc. Mario Vranješ
Sumentor:	,
Sumentor iz tvrtke:	Zvonimir Kaprocki
Predsjednik Povjerenstva:	izv. prof. dr. sc. Ratko Grbić
Član Povjerenstva 1:	prof. dr. sc. Mario Vranješ
Član Povjerenstva 2:	prof. dr. sc. Marijan Herceg
Naslov diplomskog rada:	Sustav za prikaz podataka o profilu vozaču vozila ispred vlastitog vozila zasnovan na prepoznavanju tablica vozila
Znanstvena grana diplomskog rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak diplomskog rada:	Za vrijeme vožnje vozaču mogu biti korisne razne informacije. Jedna od njih svakako je informacija o očekivanom ponašanju vozača vozila koje se nalazi ispred njega. Iako je vjerojatnost da neko vozilo tijekom određenog duljeg vremenskog perioda vozi samo jedan vozač mala, nije velika ni mogućnost da će ga voziti puno više vozača (možda 2 ili 3). Zadatak ovog diplomskog rada je izraditi sustav koji bi vozaču pružao informaciju o očekivanom ponašanju vozača vozila ispred njega, na način da prepozna tekst na registarskoj pločici vozila i da sa zadanog web poslužitelja dohvati podatke npr. o prosječnom stilu vožnje vozača tog vozila. U sklopu rada potrebno je i napraviti spomenuti web server ((omogućiti logiranje, registraciju korisnika, dodavanje recenzija za svakog vozača prema definiranim pitanjima) s kojeg će se čitati podaci o stilu vožnje koji će se ispisivati vozaču vlastitog vozila uz prepoznatu registarsku pločicu. Za izradu rješenja predlaže se korištenje Python Django web okvira otvorenog koda. Tema rezervirana za: Mihovil Kovačević Sumentor iz tvrtke: Zvonimir Kaprocki (TTTech Auto d.o.o.)
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	13.09.2023.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 27.09.2023.

Ime i prezime studenta:	Mihovil Kovačević
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1214R, 07.10.2021.
Turnitin podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: Sustav za prikaz podataka o profilu vozaču vozila ispred vlastitog vozila zasnovan na prepoznavanju tablica vozila**

izrađen pod vodstvom mentora prof. dr. sc. Mario Vranješ

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

U posljednjem desetljeću svjedočili smo ubrzanom napretku računalne tehnologije koji je značajno utjecao na mnoge industrije, a posebno na automobilsku industriju. Ugrađeni računalni sustavi u automobilima postali su neizostavan dio modernih vozila, pružajući poboljšano iskustvo vožnje i otvarajući prilike brojnim inovacijama. Očekuje se da će daljnji razvoj računalne tehnologije u automobilskoj industriji donijeti još više promjena u budućnosti, stvarajući prometno okruženje koje je pametnije, sigurnije i održivije.

Zadatak ovog diplomskog rada je razviti programsko rješenje sustava za prikaz podataka o profilu vozača vozila ispred vlastitog vozila zasnovanog na detekciji i prepoznavanju registarskih tablica vozila. Sustav je raspodijeljen na dvije aplikacije – aplikacija za detekciju registarske tablice vozila i prepoznavanje znakova na njoj te web aplikacija za prikaz podataka o profilu i recenziranje vozača. U svrhu razvoja aplikacije za detektiranje registarske tablice vozila i prepoznavanje znakova na njoj, testirano je više postojećih algoritama za detekciju objekata i algoritama za prepoznavanje znakova, te su odabrani najprikladniji. Web aplikacija pruža osnovne pozadinske funkcionalnosti i funkcionalnosti za prikaz podataka o profilu i recenziranje vozača. Web aplikacija je povezana s prethodno spomenutom aplikacijom za detektiranje registarske tablice i prepoznavanje znakova na njoj te zajedno čine cjelokupan sustav.

Ovaj diplomski rad sastoji se od šest poglavlja. U drugom poglavlju opisano je nekoliko primjena i nekoliko postojećih rješenja za detekciju registarske tablice vozila i prepoznavanje znakova na njoj. Treće poglavlje opisuje razvoj modela za detekciju registarske tablice vozila. Uključuje upotrebu YOLO (engl. *You Only Look Once*) algoritma za detekciju objekata i proces vlastitog treniranja i testiranja YOLO modela te odabir najprikladnijeg za ovaj slučaj upotrebe. Četvrto poglavlje fokusira se na odabir i podešavanje algoritma za prepoznavanje znakova na detektiranim registarskim tablicama vozila. Ovdje će se implementirati i testirati algoritmi otvorenog koda za prepoznavanje znakova. Peto poglavlje opisuje razvoj web aplikacije koja omogućuje prikaz podataka o profilu i recenziranje vozača ispred vlastitog vozila, njezinu implementaciju na javni web server i testiranje njezinih funkcionalnosti. U zaključku će biti sažeti glavni rezultati i predstavljene mogućnosti za daljnji razvoj i poboljšanje razvijenih rješenja.

2. PROBLEM DETEKCIJE REGISTARSKE TABLICE VOZILA I PREPOZNAVANJA ZNAKOVA NA NJOJ

Detekcija registarske tablice vozila i prepoznavanje znakova na njoj omogućuje jednostavnije i sigurnije upravljanje prometom. Jedna od poznatijih primjena je automatizirani sustav za naplatu cestarine, što rezultira kraćim čekanjem i manjim gužvama na naplatnim postajama. Također, ova tehnologija se koristi u sustavima praćenja i upravljanja parkiralištima. Osim toga, prepoznavanje znakova na registarskim tablicama vozila ima primjene u provođenju zakona, primjerice za praćenje ukradenih vozila ili identifikaciju vozila koja su uključena u prometne prekršaje.

Proces prepoznavanja znakova na registarskoj tablici vozila sastoji se od nekoliko ključnih zadataka. Prvi korak je detekcija registarske tablice vozila, što podrazumijeva automatsko pronalaženje i označavanje tablice na slici, svakoj pojedinačnoj slici videozapisa ili svakoj pojedinačnoj slici snimke u stvarnom vremenu. Detekcija registarske tablice vozila može se ostvariti korištenjem klasičnih tehnika i algoritama računalnog vida ili strojnog učenja. Nakon uspješne detekcije, slijedi zadatak prepoznavanja znakova na registarskoj tablici vozila. Algoritmi koji obavljaju ovaj posao najčešće su zasnovani na konvolucijskim neuronskim mrežama (engl. *Convolutional Neural Network*, CNN) i rekurentnim neuronskim mrežama (engl. *Reccurent Neural Network*, RNN).

2.1. Postojeća rješenja za detekciju registarske tablice vozila i prepoznavanje znakova na njoj

U nastavku je prikazan kratak pregled nekoliko najnovijih znanstvenih radova koji se bave detekcijom registarske tablice vozila i prepoznavanjem teksta na njoj. Neki od postojećih problema pri detekciji registarske tablice vozila uključuju nizak odziv na rubnim dijelovima slike, poteškoće pri lošim vremenskim uvjetima i s nedostatkom prirodnog dnevnog osvjetljenja, prilagodbu na različite formate registarskih tablica koje se koriste u različitim državama te različite fizičke oblike registarskih tablica na različitim vozilima.

U radu [1] predstavlja se rješenje za prepoznavanje multinacionalnih tablica vozila. Automatski sustav prepoznavanja je dizajniran da bude primjenjiv na multinacionalne tablice vozila s rasporedom znakova u jednom ili više redova, bez zahtjeva za promjenama u algoritmima.

Pristup segmentaciji znakova i prepoznavanju znakova tretira se kao jedinstveni problem prepoznavanja objekata. Sustav se sastoji od tri glavne faze: detekcija registarske tablice, jedinstveno prepoznavanje znakova i detekcija rasporeda multinacionalnih registarskih tablica. Prva faza, detekcija registarske tablice, koristi arhitekturu mreže YOLOv3 za detekciju registarske tablice. Druga faza, jedinstveno prepoznavanje znakova, koristi YOLOv3-*Spatial Pyramid Pooling* (YOLOv3-SPP) arhitekturu za prepoznavanje znakova na lokaliziranoj registarskoj tablici. Ovdje se znakovi tretiraju kao objekti, što omogućuje objedinjavanje koraka segmentacije i prepoznavanja znakova u jednu fazu. Treća faza, detekcija rasporeda multinacionalnih registarskih tablica, razvija univerzalni algoritam za određivanje ispravnog redoslijeda redova znakova na multinacionalnim registarskim tablicama koje sadrže više od jednog retka znakova. Rješenje je testirano na skupovima podataka koji sadrže tablice vozila iz Južne Koreje [2] (KarPlate database), Tajvana [3] (AOLP database), Sjedinjenih Američkih Država [4] (Caltech Cars (Rear) 1999 database), Grčke [5] (MediaLab LPR database) i Hrvatske [6] (University of Zagreb database), koristeći osobno računalo koje sadrži Intel Core i7-4770, NVIDIA Titan X Pascal i 24GB RAM-a. Predloženo rješenje daje bolje rezultate točnosti pri procesu detekcije registarske tablice i prepoznavanja znakova na njoj u odnosu na OpenALPR [7], Sighthound [8], BRNNs [9], YOLOv2 + WPOD-Net [10] i YOLOv2 + CR-Net [11]. Najveću točnost ima za AOLP skup podataka [3] koja iznosi 99.51%, a najmanju za MediaLab skup podataka [5] koja iznosi 96.98%. Uz ranije navedenu konfiguraciju, prosječno vrijeme izvršavanja za prepoznavanje jedne registarske tablice iznosi 63.62 ms, za dvije registarske tablice na istoj slici 78.39 ms, a za tri registarske tablice na istoj slici 93.10 ms. Među nedostacima predloženog rješenja spominju se mogući problemi s novim rasporedima tablica vozila i potreba za poboljšanjem vremena izvođenja sustava. Također se navodi potreba za ručnim označavanjem javno dostupnih skupova podataka koji nemaju prethodno dostupne granične okvire. Ističe se mogućnost daljnjeg poboljšanja vremena izvođenja sustava kroz implementaciju u C++ programski jezik i integraciju višenitnosti.

U sklopu istraživanja [12] predstavljen je sustav za detekciju i prepoznavanje registarskih tablica vozila koji se temelji na poboljšanom YOLOv5m algoritmu [13] i mreži za prepoznavanje registarskih oznaka koju je razvila Nvidia [14] (engl. *License Plate Recognition Network*, LPRNet). Poboljšanje izvornog YOLOv5m algoritma izvršeno je na nekoliko načina. Prvo, dubina i širina mreže su povećane kako bi se postigla veća preciznost detekcije. Povećana dubina omogućuje bolje modeliranje značajki, dok veća širina omogućuje modelu da nauči složenije

obrasce i detalje. Drugo, primijenjena je *Mosaic* metoda za poboljšanje podudaranja između referentnih okvira i ciljnih objekata, što rezultira generiranjem dodatnih podataka za trening i poboljšanom robusnošću mreže. Treće, metoda potiskivanja nemaksimuma (engl. *Non-Maximum Suppression*, NMS) je poboljšana primjenom gubitka udaljenosti presjeka preko unije (engl. *Distance Intersection over Union*, DIOU), koji uzima u obzir geometrijske informacije i poboljšava postprocesiranje rezultata detekcije. Naposljetku, primijenjen je poboljšani postupak odabira referentnih okvira (engl. *anchor box*) koristeći *K-means*++ algoritam za bolje grupiranje i odabir referentnih okvira koji bolje odgovaraju oblicima ciljnih objekata. Uz to, LPRNet koristi CNN s malim brojem parametara koja omogućava prepoznavanje znakova registarske tablice bez segmentacije znakova. Evaluacija rješenja provedena je na Chinese City Parking Dataset skupu podataka s otvorenim pristupom [15] koji sadrži oko 250.000 slika tablica vozila. Testiranje je obuhvatilo različite složene uvjete poput različitih osvjetljenja, kutova snimanja i vremenskih uvjeta. Kao eksperimentalna konfiguracija, korišteni su Intel CPU R Core i7-10875h @ 2.60GHz×8, GeForce GTX 3060, 12GB memorije i 16GB RAM-a. Predloženo poboljšano rješenje za prepoznavanje registarskih tablica vozila ima veliku brzinu rada, što omogućava rad u stvarnom vremenu. LPRNet mreža pruža dobru robusnost i prilagodljivost u složenim okruženjima. Nadmašuje YOLOv3-LPRNet, YOLOv4-LPRNet i YOLOv5s-LPRNet modele, pri čemu ostvaruje F1 mjeru u rasponu od 0.9734 do 0.9985 i brzinu izvođenja mjerenu brojem obrađenih sličica u sekundi (engl. *frames per second*, FPS) u rasponu od 31 FPS do 42 FPS, ovisno o vremenskim uvjetima, uvjetima osvjetljenja i kutova snimanja. Međutim, treba napomenuti da je sustav testiran samo na skupu podataka koji sadrži tablice vozila jedne države, što može ograničiti njegovu primjenu u multinacionalnim scenarijima.

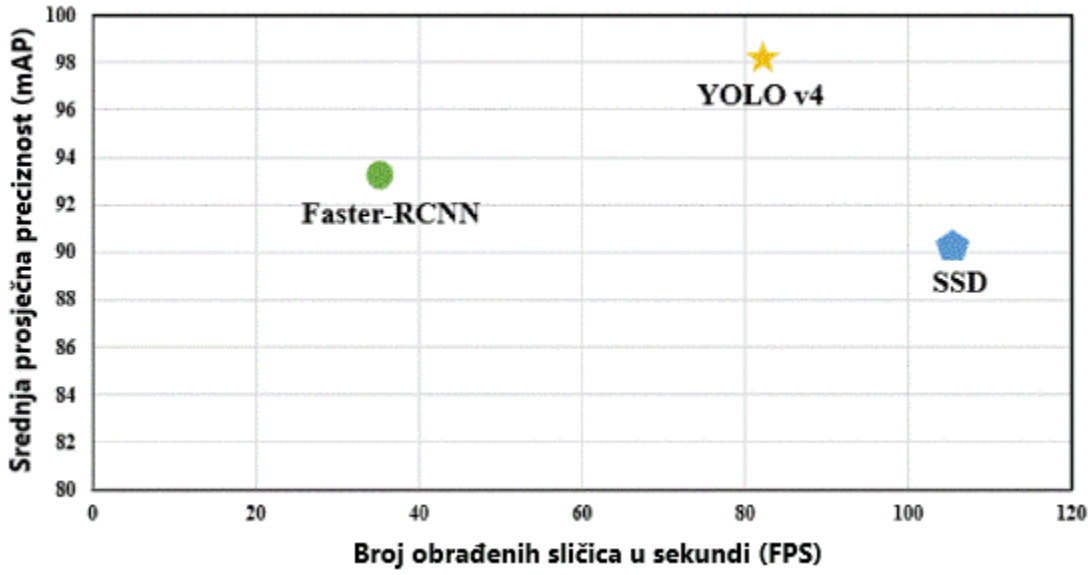
U znanstvenom radu [16] proučava se detekcija i prepoznavanje nagnutih kineskih registarskih tablica u realnim okruženjima. Predlaže se robusna metoda koja omogućava detekciju i ispravljanje izobličenih slika registarskih tablica, te njihovo prepoznavanje kako bi se postigao konačni rezultat. Metoda uključuje tri ključna modula: detekcija vozila, detekcija registarske tablica vozila i prepoznavanje znakova detektirane registarske tablice vozila. Za detekciju vozila koristi se algoritam *EfficientDet* [17]. Ovaj algoritam je odabran jer ima visoku preciznost detekcije, brzu obradu i niske računalne zahtjeve. Ovdje se koristi izvorni *EfficientDet* model bez modifikacija. Za detekciju registarske tablice koristi se mrežna struktura LPD (engl. *License Plate Detection*) za ispravljanje distorzije registarske tablice. Ova mreža uči detektirati registarske

tablice u prirodnim scenama s različitim stupnjevima distorzije i ispravlja ih u pravokutni oblik sličan pogledu sprijeda. Za prepoznavanje znakova detektirane registarske tablice koristi se kombinacija triju nivoa mrežnog okvira. Prvo se koristi CNN za izvlačenje značajki slika registarskih tablica vozila. Zatim se koristi *Bi-directional long short-term memory* (BiLSTM) za modeliranje sekvence znakova. Na kraju se koristi metoda predikcije, uključujući i *Connectionist temporal classification* (CTC) za dekodiranje i predviđanje konačnih rezultata. Za potrebe istraživanja, korišten je CCPD skup podataka [15] koji sadrži više od 250 tisuća jedinstvenih slika registarskih pločica kineskih vozila, uz detaljne oznake. Eksperimenti su provedeni koristeći Intel Xeon E5-2680v4 procesor i Nvidia Tesla K80∗8 grafičku karticu. Rezultati eksperimenata pokazuju da predložena metoda brzo i točno otkriva te identificira nagnute i iskrivljene registarske tablice vozila. U usporedbi s postojećim metodama, ova metoda pokazuje značajno poboljšanje u izvedbi prepoznavanja registarskih tablica vozila. Navedeni rezultati testiranja su točnost prepoznavanja 98.9% i vrijeme izvršavanja 53 ms. Iako je ova metoda uspješna u prepoznavanju kineskih registarskih tablica, važno je napomenuti da se može suočiti s izazovima kod registarskih tablica vozila s drugačijim rasporedom i multinacionalnim registarskim tablicama vozila. Nadalje, postoje mogućnosti istraživanja nove arhitekture neuronske mreže kako bi se optimizirala brzina otkrivanja vozila.

Postoje još mnoga rješenja za dani problem, a više detalja o njima može se naći u [18]–[22]. Cilj svih iznad opisanih rješenja je bio prilagoditi tada najnaprednije tehnologije slučaju detekcije i prepoznavanja registarskih tablica vozila i time ostvariti veću brzinu izvođenja i kvalitetnije rezultate. Unatoč ostvarenim visokim performansama i kvaliteti rezultata, budući razvoj tehnologije će uvijek omogućavati izradu novih rješenja koja će nadmašiti trenutna. Također, zbog značajnih varijacija multinacionalnih registarskih tablica vozila, za rješenja čiji su modeli prilagođeni samo registarskim tablicama vozila jedne države očekivan je pad kvalitete rezultata detekcije i prepoznavanja multinacionalnih registarskih tablica vozila. Povećanje skupa podataka za trening modela je još jedan od načina kojim se uvijek može unaprijediti svako rješenje za detekciju i prepoznavanje registarskih tablica vozila. Za razliku od navedenih rješenja u kojima su spomenuti, ali ne i uključeni, konkretni slučajevi upotrebe zbog kojih se nastoji poboljšati performanse detekcije i prepoznavanja registarskih tablica vozila, u sklopu ovog rada će se osim toga razviti i web aplikacija za prikaz podataka o profilu i recenziranje vozača ispred vlastitog vozila.

3. RAZVOJ MODELA ZA DETEKCIJU REGISTARSKIH TABLICA VOZILA

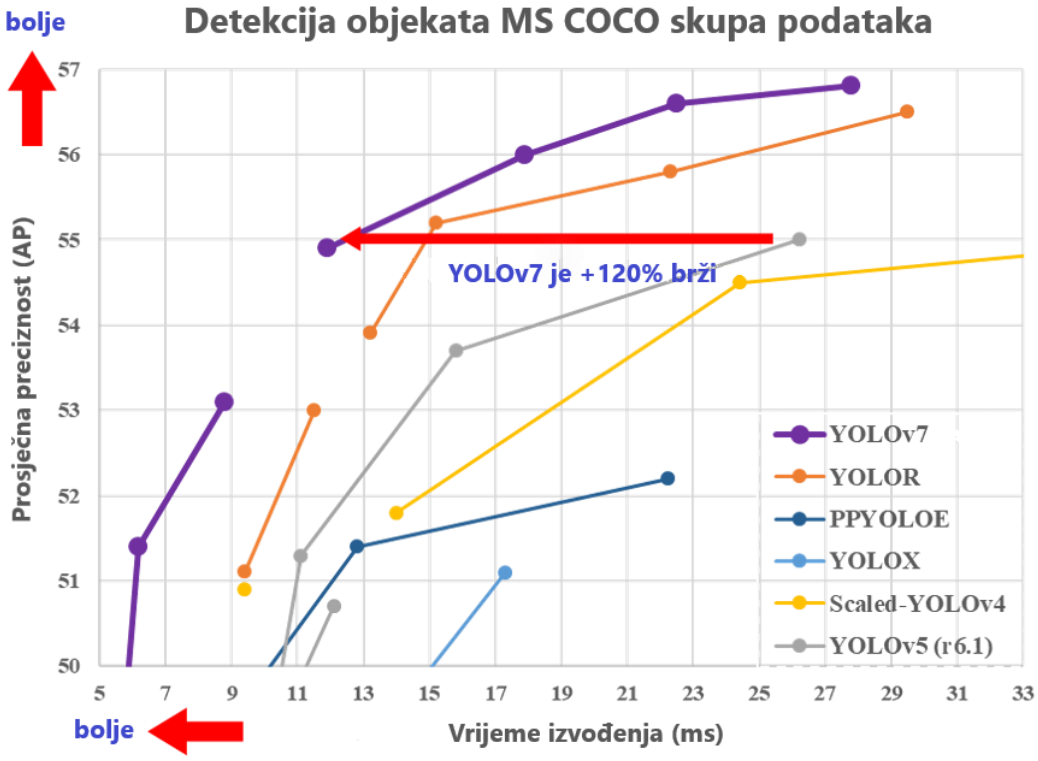
Prvi korak za ostvarenje sustava za prikaz podataka o profilu vozača vozila ispred vlastitog vozila zasnovanog na detekciji i prepoznavanju registarskih tablica vozila, je vlastiti razvoj modela za detekciju registarskih tablica vozila. Iz prvog znanstvenog rada opisanog u prethodnom poglavlju [1] vidljivo je da YOLOv3 algoritam pri zadatku detekcije registarske tablice vozila nadmašuje po performansama i brzini izvođenja komercijalne proizvode OpenALPR [7] i Slighthound [8]. Nadalje, u znanstvenom istraživanju [23] YOLOv4 algoritam ostvaruje najbolji rezultat srednje prosječne preciznosti (mAP) 98.19% te najbolji rezultat F1 mjere 96% u odnosu na algoritme *Faster Region-based*-CNN (R-CNN) [24] i *Single Shot Detector* (SSD) [25] pri zadatku detekcije regije interesa vozila kako bi se naknadno prepoznao tip vozila. SSD algoritam nadmašuje YOLOv4 algoritam u brzini izvođenja pri čemu YOLOv4 ima mogućnost obrade 82.1 FPS-a, a SSD 105.14 FPS-a na istoj hardverskoj konfiguraciji. Na slici 3.1. prikazan je dijagram izvedbi modela dubokog učenja spomenutog znanstvenog istraživanja [23].



Slika 3.1. Dijagram izvedbi modela dubokog učenja [23]

Konačno, u znanstvenom radu [26] prema kojem je implementiran YOLOv7 algoritam testirane su i uspoređene verzije YOLO algoritma YOLOv5 [13], *Scaled*-YOLOv4 [27], YOLOX [28], PPYOLOE [29], YOLOR [30] i YOLOv7 [26] za detekciju objekata MSCOCO skupa

podataka [31]. Na slici 3.2. vidljivo je da YOLOv7 algoritam u kompleksnijim izvedbama nadmašuje sve ostale prema mjeri prosječne preciznosti detekcije objekata, a u slučaju usporedbe rezultata prosječne preciznosti detekcije objekata neke od izvedbi YOLOv7 algoritma i neke od izvedbi drugih YOLO algoritama, vidljivo je da izvedba YOLOv7 algoritma ostvaruje značajno brže izvođenje.



Slika 3.2. Dijagram usporedbe izvedbi više YOLO algoritama testiranih na MS COCO skupu podataka [26]

Sudeći prema rezultatima prethodno navedenih istraživanja, za razvoj vlastitog modela za detekciju registarskih tablica vozila bit će odabrana jedna od podverzija YOLOv7 ili YOLOv5 algoritama nakon vlastitog treniranja, implementacije, testiranja i usporedbe.

3.1. YOLO algoritam za detekciju objekata

YOLO je trenutno smatran najnaprednijim algoritmom za detekciju objekata. Odlikuju ga velika brzina izvođenja i preciznost u čemu nadmašuje ostale popularne algoritme za detekciju objekata R-CNN [32], *Fast* R-CNN [33], *Faster* R-CNN [24] i SDD [25]. Prva verzija YOLO

algoritma (YOLOv1) predstavljena je 2016. godine, a danas je njegova najnovija verzija u području računalnog vida standard za detekciju objekata. Autori prve verzije YOLO algoritma (YOLOv1) problem detekcije objekta postavljaju kao problem regresije umjesto klasifikacije, tako da prostorno odvajaju granične okvire (engl. *bounding boxes*) koji označavaju detektirani objekt i pridružuju pouzdanost svakom detektiranom objektu koristeći samo jednu CNN, po čemu je algoritam dobio naziv [34].

Nadalje, prema području umjetne inteligencije, YOLO algoritam može se promatrati kao kombinacija nadziranog dubokog učenja (metoda regresije + CNN), računalnog vida i prikupljanja te analize podataka. Ove kategorizacije međusobno imaju preklapanja, a detaljnije o tome kako se ona reflektiraju u YOLO algoritmu bit će riječi u nastavku.

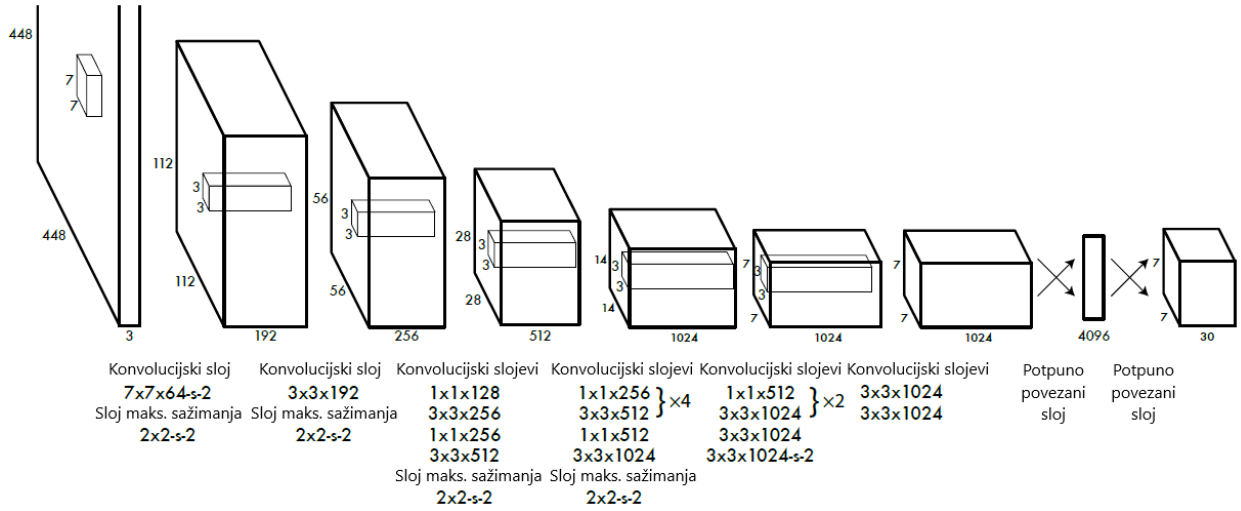
3.1.1. Klasifikacija i regresija kao metode nadziranog dubokog učenja

Klasifikacija se odnosi na postupak predviđanja diskretnih klasa na temelju ulaznih podataka, koji se temelji na statističkim i vjerojatnosnim metodama [35]. U klasifikaciji, model bi završavao s izlaznim vektorom koji predstavlja vjerojatnosti različitih klasa. Primjerice ukoliko se radi o problemu klasifikacije slike gdje se želi odlučiti nalazi li se na slici čovjek ili pas, idealni izlaz modela je 'Čovjek = 1, Pas = 0' ukoliko se na slici nalazi čovjek, a 'Čovjek = 0, Pas = 1' ukoliko se na slici nalazi pas.

Regresija se odnosi na postupak predviđanja kontinuiranih izlaznih vrijednosti na temelju ulaznih podataka. Općenito, regresija se temelji na statističkim modelima i matematičkim tehnikama koje pokušavaju pronaći najbolju funkcionalnu vezu između ulaznih i izlaznih varijabli. Primjerice linearna regresija koristi linearnu funkciju za modeliranje veze između ulaznih i izlaznih podataka. Cilj je minimizirati razlike između stvarnih i predviđenih vrijednosti [35]. Kada primijenimo regresiju na problem detekcije objekta, model će kao izlaz davati kontinuiranu vrijednost, odnosno izlaz će biti izlazni vektori detektiranih objekata, od kojih svaki sadrži pouzdanost graničnog okvira, vrijednosti pozicije i veličine graničnog okvira tog objekta i vjerojatnosti svih klasa koje model može detektirati [34].

3.1.2. Arhitektura YOLOv1 algoritma

YOLOv1 arhitektura se sastoji od 24 konvolucijska sloja, nakon kojih slijede 2 potpuno povezana sloja [34] (slika 3.3.).



Slika 3.3. Arhitektura originalnog YOLO modela [34]

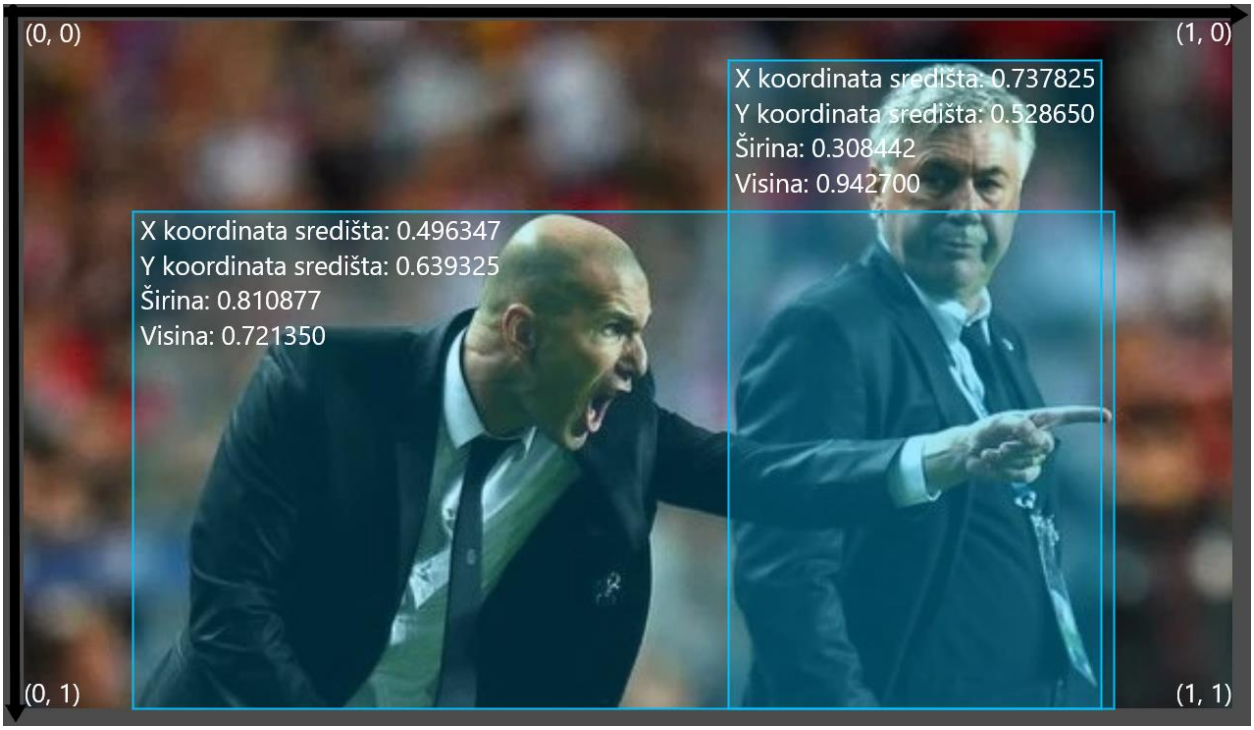
Općenito, neuronska mreža je metoda u području umjetne inteligencije koja podučava računala obradi podataka na način koji je inspiriran ljudskim mozgom. To je vrsta postupka strojnog učenja, nazvana duboko učenje, koja koristi međusobno povezane čvorove ili neurone u slojevitoj strukturi koja podsjeća na ljudski mozak. Stvara adaptivni sustav koji računala koriste za učenje iz ulaznih podataka, vlastitih pogrešaka i neprekidnog poboljšanja. Neuronske mreže imaju nekoliko skrivenih slojeva s milijunima umjetnih međusobno povezanih neurona (čvorova). Vrijednost nazvana težina predstavlja veze između jednog čvora nekog sloja i drugog čvora sljedećeg sloja. Težina je pozitivan broj ako jedan čvor potiče drugi, ili negativan ako jedan čvor suzbija drugi. Čvorovi s većim vrijednostima težine imaju veći utjecaj na ostale čvorove.

CNN su vrsta neuronskih mreža koje se koriste za obradu i analizu slika. Inspirirane su načinom na koji funkcionira ljudski vid, odnosno načinom na koji naš mozak obrađuje vizualne informacije. CNN koristi slojeve filtara za izdvajanje značajki iz slika. Ovi slojevi obavljaju operaciju konvolucije na slici kako bi detektirali razne vizualne obrasce i značajke poput rubova, uglova i tekstura, te na višoj razini veličine, izgleda i okruženja objekata. Broj filtera u svakom konvolucijskom sloju određuje broj značajki koje se mogu detektirati. Tada se dobiva aktivacijska mapa koja prikazuje prisutnost određenih značajki. Na kraju CNN-a koriste se potpuno povezani slojevi koji su slični tradicionalnim neuronskim mrežama i koriste se za donošenje konačne odluke na temelju izdvojenih značajki [36]. U slučaju YOLO arhitekture, potpuno povezani slojevi na kraju mreže uzimaju izlaz iz prethodnih konvolucijskih slojeva i predviđaju pouzdanosti graničnih

okvira, vrijednosti pozicije i veličine graničnih okvira i vjerojatnosti svih klasa koje model može detektirati [34].

3.1.3. Općeniti postupak treniranja YOLO modela

Prije započinjanja postupka treniranja, potrebno je pripremiti ulazne podatke, odnosno ručno označiti granične okvire za proizvoljne objekte na dovoljno velikom skupu slika. YOLO algoritam za raspoznavanje pozicije i veličine graničnog okvira koristi *x* i *y* vrijednosti centra graničnog okvira i širinu te visinu graničnog okvira normalizirane na raspon od 0 do 1 u odnosu na izvornu širinu i visinu slike, odnosno širinu i visinu slike prije skaliranja na određenu rezoluciju (prema zadanom 448x448 piksela za YOLOv1) koje YOLO algoritam provodi kao dio postupka treniranja (slika 3.4.) [37].



Slika 3.4. Anotacija graničnog okvira u YOLO formatu [37]

Prvi korak postupka treniranja je propagacija unaprijed. Prethodno ručno označene slike se skaliraju na određenu rezoluciju (prema zadanom 448x448 piksela za YOLOv1) i prosljeđuju slojevima konvolucije kako bi se izvršilo predviđanje, odnosno detekcija željenih objekata. Nakon propagacije unaprijed, izračunava se gubitak (eng. *loss*) koji se izračunava na temelju preklapanja predikcija sa stvarnim oznakama objekata. Gubitak se propagira unatrag kroz neuronsku mrežu

kako bi se ažurirale težine veza. Ovaj postupak se ponavlja za sve slike u danom skupu podataka. Nadalje, sve zajedno se ponavlja kroz više epoha (eng. *epochs*) kako bi se poboljšala kvaliteta detekcije [34].

3.1.4. Općeniti postupak predviđanja na temelju YOLO modela

Algoritam prima ulaznu sliku i mijenja joj rezoluciju u onu koja je definirana prilikom treniranja modela. Ulazna slika se dijeli na mrežu pravokutnih ćelija i svaka ćelija u mreži odgovorna je za detekciju objekata koji se nalaze unutar nje, odnosno svakoj ćeliji će za svaki detektirani objekt biti pridružen vektor detekcije objekta (3-1).

$$\begin{bmatrix} C \\ B_x \\ B_y \\ B_w \\ B_h \\ P_1 \\ \vdots \\ P_n \end{bmatrix}, \tag{3-1}$$

gdje je:

- C – pouzdanost graničnog okvira, odnosno procjena koliko je model siguran u svoje predikcije graničnih okvira
- B_x – normalizirana x-koordinata centra graničnog okvira objekta u odnosu na širinu ćelije
    - Primjerice ako je širina ćelije 100 piksela, a centar graničnog okvira objekta je na udaljenosti 20 piksela od lijeve rubne točke ćelije, tada će se x-koordinata normalizirati kao 0.2 (20/100).
- B_y – normalizirana y-koordinata centra graničnog okvira objekta u odnosu na visinu ćelije
- B_w – normalizirana širina detektiranog graničnog okvira objekta u odnosu na ukupnu širinu slike
    - Primjerice ako je širina slike 800 piksela, a širina graničnog okvira je 100 piksela, tada će se širina normalizirati kao 0.125 (100/800).
- B_h – normalizirana visina detektiranog graničnog okvira objekta u odnosu na ukupnu visinu slike
- P_n – vjerojatnosti pripadnosti objekta različitim klasama modela (1,…, n)

To omogućava detaljno određivanje položaja objekta i učinkovito izvršavanje YOLO algoritma. Umjesto da obradi svaki piksel slike zasebno, mreža podijeljena na ćelije smanjuje broj obrada koje je potrebno provesti [34].

Pouzdanost u kontekstu detekcije objekata označava procjenu koliko je model siguran u svoje predikcije graničnih okvira. Tijekom treniranja, za vrijednost pouzdanosti, računa se mjera presjeka preko unije (engl. *Intersection over Union*, IoU) kao omjer presjeka i unije površine detektiranog graničnog okvira i površine stvarnog graničnog okvira (3-2).

$$IoU = \frac{A \cap B}{A \cup B}, \tag{3-2}$$

gdje je:

- A - površina detektiranog graničnog okvira
- B - površina stvarnog (ručno označenog) graničnog okvira (engl. *ground truth*)

Pouzdanost također sudjeluje u procesu propagacije gubitka (engl. *loss*) unatrag kroz mrežu. To omogućava CNN-u modela da na izlazu vrati pouzdanost predikcije graničnog okvira kao vrijednost vektora detekcije objekta. Ovo je nužno jer, kada se radi predikcija pomoću prethodno treniranog modela, nisu dostupni stvarni granični okviri pa se stoga ne može računati IoU, a prema tome ni pouzdanost.

Algoritam prilikom predviđanja koristi prag pouzdanosti (engl. *confidence threshold*), čija zadana vrijednost iznosi 0.5, a moguće ju je proizvoljno izmijeniti. U slučaju da neke detekcije imaju iznos pouzdanosti manji od određenog praga, bit će odbačene.

Vjerojatnosti P_n da detektirani objekti pripadaju određenoj klasi označava mjeru sigurnosti u vezi s prisutnošću određenog objekta unutar graničnog okvira. Algoritam određuje takvu vjerojatnost na temelju faktora kao što su veličina objekta, njegov izgled i okruženje koje model nauči iz skupa podataka tijekom treniranja.

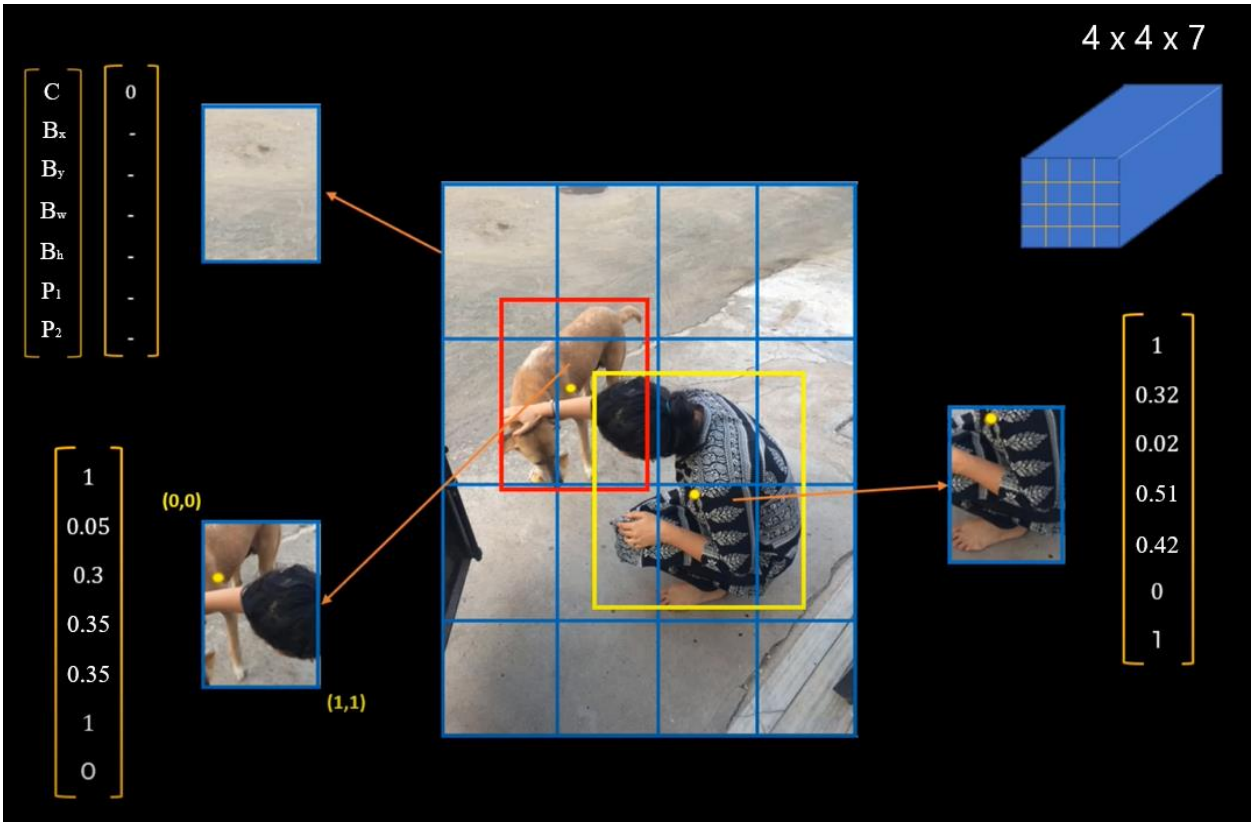
Slika 3.5. prikazuje detektirane objekte i vektore detektiranih objekata koji su pridruženi ćelijama odgovornim za detekciju. U ovom primjeru je mreža ćelija veličine 4x4x7, a u originalnom YOLO algoritmu je veličine 7x7x30 (3-3).

$$\mathrm{S} \times \mathrm{S} \times (\mathrm{B} * \mathrm{V} + \mathrm{P}), \tag{3-3}$$

gdje je:

- SxS – mreža ćelija
- B – broj mogućih detekcija pojedine ćelije
- V – dio vektora detekcije objekta koji uvijek sadrži istu količinu elemenata
- P – dio vektora detekcije objekta koji sadrži vjerojatnosti pripadnosti objekta različitim klasama modela

Dakle, za model koji je predstavljen u radu originalnog YOLO algoritma, SxS je 7x7, B je 2, V je 5 (C, B_x, B_y, B_w, B_h), a P je 20 jer se trenirao model za 20 klasa objekata [34].



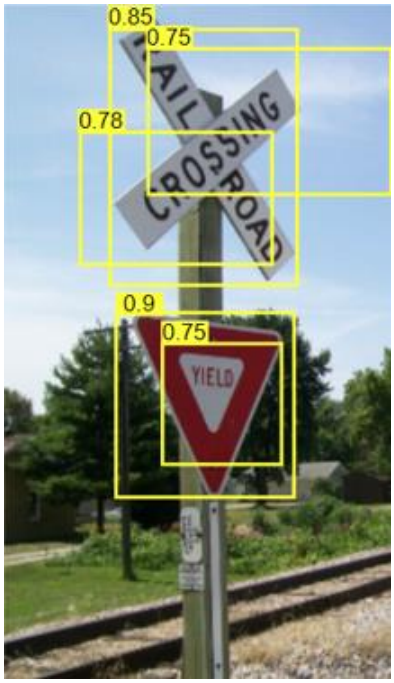
Slika 3.5. Detektirani objekti i vektori detektiranih objekata koji su pridruženi ćelijama odgovornim za detekciju [38]

Kao što je vidljivo sa slike 3.5., detektirani objekt 'pas' i detektirani objekt 'čovjek' se protežu kroz više ćelija. U ovom slučaju, svaka ćelija daje svoje predikcije na temelju dijela objekta koji se nalazi unutar nje. Međutim, nakon što se filtriraju detekcije više ćelija tog objekta pomoću metode potiskivanja nemaksimuma (engl. *Non Maximum Supression*, NMS) ostaje samo

jedna detekcija ćelije koja od ponuđenih ima najveću pouzdanost i najbolje smješten centar graničnog okvira objekta [38].

3.1.5. Metoda potiskivanja nemaksimuma

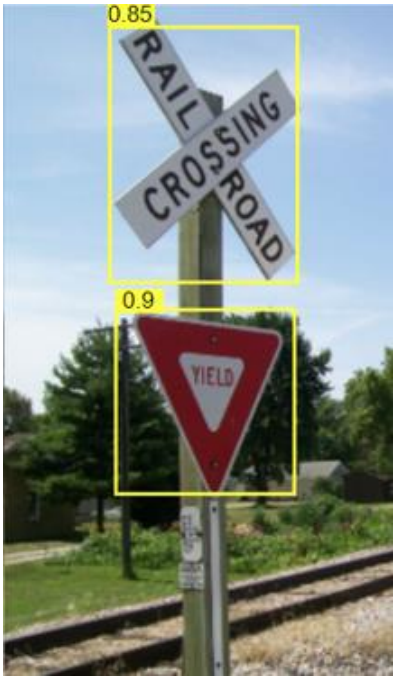
U slučaju slike koja sadrži objekte koji se protežu kroz više ćelija, YOLO algoritam će vratiti rezultat više graničnih okvira za svaki od objekata. Cilj je ostaviti samo jedan granični okvir s najvećom pouzdanošću za svaki pojedinačni objekt. To bi se moglo učiniti filtriranjem prema najvećem iznosu pouzdanosti svake klase, ali ukoliko se na slici nalazi više objekata iste klase koji se protežu kroz više ćelija, rezultat će biti samo jedan granični okvir ukupno. Razlog tomu je što je moguće raspoznati samo pojedinačnu klasu, ali ne i svaki pojedinačni objekt te klase. Slika 3.6. prikazuje dva objekta klase prometni znak i rezultat 5 graničnih okvira YOLO algoritma.



Slika 3.6. Granični okviri prije primjene metode potiskivanja nemaksimuma [38]

U ovom slučaju koristi se metoda NMS, kako bi se zadržao po jedan granični okvir s najvećom pouzdanošću za svaki pojedinačni objekt neke klase. Detekcije se sortiraju prema vrijednosti pouzdanosti. Nakon sortiranja, NMS metoda prolazi kroz sve detekcije od najviše pouzdanih prema manje pouzdanim. Za svaku detekciju, provjerava se preklapanje te detekcije s nekom drugom detekcijom koja ima veću pouzdanost pomoću mjere IoU kao omjer presjeka i unije površina detektiranih graničnih okvira. Ako se granični okviri detekcija preklapaju više nego

što je definirano pragom preklopa (engl. *overlap threshold*), manje pouzdana detekcija se odbacuje. Ako se ne preklapaju dovoljno, obje detekcije se zadržavaju. Zadana vrijednost praga preklopa iznosi 0.5, a moguće ju je proizvoljno izmijeniti. Rezultat primjene NMS metode na rezultate graničih okvira YOLO algoritma sa slike 3.6. vidljiv je na slici 3.7. [34].



Slika 3.7. Granični okviri nakon primjene metode potiskivanja nemaksimuma [38]

3.1.6. Metoda sidrenih okvira

Sidreni okvir (engl. *anchor box*) može se opisati kao nadskup (referenca) graničnih okvira koje generiraju ćelije u mreži. Sidreni okviri daju smjernice o približnoj poziciji objekata i o tome kako predviđati veličinu i oblik graničnih okvira oko objekata. Tijekom postupka treniranja, model prilagođava parametre sidrenih okvira kako bi što bolje odgovarali stvarnim oblicima objekata u skupu slika za treniranje. Samo mrežne ćelije kroz koje prolazi sidreni okvir su odgovorne za korištenje tog sidrenog okvira u postupku detekcije objekata. Ostale ćelije u mreži ne koriste taj sidreni okvir za svoje predikcije.

Pregledavanje skupa podataka koji će se koristiti za treniranje može pomoći u razumijevanju različitih veličina, oblika i pozicija objekata prisutnih na slikama. Ovo može pružiti smijernice pri odabiru odgovarajućih sidrenih okvira. Prirodno je da neuronske mreže bolje predviđaju mala odstupanja od velikih odstupanja te stoga, što se bolje odaberu sidreni okviri,

manje "posla" će obaviti neuronska mreža, a rezultati predikcije će biti bolji. Ako korisnik ne postavi ručno sidrene okvire, originalni YOLO algoritam ih neće automatski postaviti tijekom treniranja. Moguća su i detaljnija podešavanja poput granica koje određuje da objekt ne smije biti toliko puta manji ili veći [39], [40]. Izvorni YOLO algoritam ne koristi ovu metodu, odnosno uvedena je tek u verziji YOLOv2.

3.1.7. Općeniti postupak testiranja YOLO modela

Postupak testiranja YOLO modela provodi predviđanje nad zadanim ulaznim skupom slika te vlastite rezultate detekcije uspoređuje s prethodno ručno označenim. Performanse modela se očituju kroz sljedeće metrike: preciznost, odziv, mAP@0.5 i mAP@0.5:.95. U kontekstu testiranja također se koristi mjera IoU (3-2) [41].

Preciznost mjeri udio ispravnih detekcija modela među svim detekcijama modela (3-4). Ispravne detekcije su detekcije čiji je IoU veći od 0.5. Vrijednost 1 preciznosti označava da su sve detekcije modela ispravne, a vrijednost manja od 1 označava da neke detekcije modela nisu ispravne.

Odziv mjeri udio ispravnih detekcija modela među svim mogućim ispravnim detekcijama (3-5). Ispravne detekcije su detekcije čiji je IoU veći od 0.5. Vrijednost 1 odziva označava da su svi ručno označeni granični okviri pronađeni detekcijom modela, a vrijednost manja od 1 označava da neki od ručno označenih graničnih okvira nisu detektirani modelom.

F1 mjera (engl. *F1 score*) je metrika koja se koristi za procjenu ravnoteže između preciznosti i odziva. F1 mjera je harmonijska sredina (geometrijska sredina) preciznosti i odziva (3-6).

$$preciznost = \frac{TP}{TP+FP}, \tag{3-4}$$

$$odziv = \frac{TP}{TP+FN}, \tag{3-5}$$

$$F1 = 2 * \frac{preciznost * odziv}{preciznost + odziv}, \tag{3-6}$$

gdje je:

- TP – *true positive,* detekcije koje je model proglasio objektom, a one to jesu,
- FP – *false positive,* detekcije koje je model proglasio objektom, a one to nisu,
- FN – *false negative,* detekcije koje model nije proglasio objektom, a one to jesu,
- TN – *true negative,* detekcije koje model nije proglasio objektom, a one to nisu.

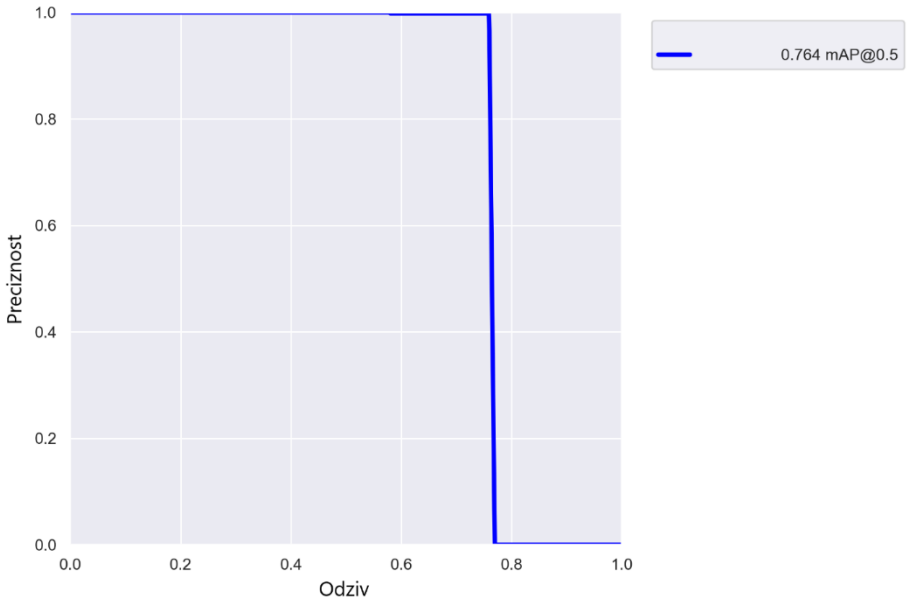
mAP@0.5 označava prosječnu vrijednost svih prosječnih preciznosti (engl. *average* precision, AP) pojedinih klasa. Za svaku klasu objekta izračunava se AP, što je površina ispod krivulje preciznost-odziv (engl. *precision-recall curve*) (3-7).

$$AP = \int_0^1 p(r)dr, \tag{3-7}$$

gdje je:

- AP – prosječna preciznost
- p(r) – preciznost u ovisnosti o odzivu pri određenom pragu
- dr – inkrement odziva

U obzir se uzimaju samo detekcije s IoU iznad 0.5. Na x-osi krivulje nalaze se odzivi, a na y-osi preciznosti. Krivulja pokazuje kako se preciznost mijenja s odzivom pri različitim pragovima IoU (slika 3.8.). Koriste se pragovi IoU koji se kreću iznad 0.5, na primjer 0.55, 0.6, 0.65, itd.



Slika 3.8. Primjer krivulje preciznost-odziv

mAP@0.5:.95 označava prosječnu vrijednost svih prosječnih preciznosti (AP) pojedinih klasa, uz dodatak u odnosu na mAP@0.5, u obzir se uzima više skupina detekcija s IoU od iznad 0.5 do iznad 0.95 s korakom 0.05. Dakle, za svaku skupinu će se izračunavati AP, što znači da će svaka klasa imati više AP vrijednosti.

3.1.8. Nedostaci izvornog YOLO algoritma i kako su suvremene verzije unaprijeđene

YOLOv1 donio je značajne inovacije, ali imao je i određene nedostatke. Bio je brz u usporedbi s mnogim drugim algoritmima za detekciju objekata, ali se suočavao s problemom brzine kod detekcije objekata na videozapisima i u stvarnom vremenu. Učitavanje cijele slike i njezino jednokratno proslijeđivanje kroz neuronsku mrežu uzrokovalo je određeni zastoj, što je posebno bilo primjetno u slučaju velikog broja objekata. Također je imao poteškoća s preciznom detekcijom malih objekata. Ovo je bilo posebno problematično u scenarijima gdje su objekti zauzimali samo mali dio slike. Uz to, postojale su poteškoće u prepoznavanju objekata sličnih, ali nešto drugačijih oblika ili tekstura [34].

Nakon YOLOv1, uslijedila su kontinuirana poboljšanja i evolucija algoritma. YOLOv2 (2017), YOLOv3 (2018) i YOLOv4 (2020) ostvarili su povećanje preciznosti detekcije, brzine izvođenja i bolje su se nosili s raznolikim uvjetima. U interesu ovog rada je koristiti što recentnije verzije YOLO algoritama, stoga će u nastavku biti opisana unaprjeđenja koja donose YOLOv5 [13] i YOLOv7 [26] algoritmi.

YOLOv5 verzija uključuje niz ključnih unaprjeđenja koja su riješila neke od nedostataka prethodnih YOLO algoritama. YOLOv5 koristi mrežu sa smanjenim brojem filtara koja omogućuje brže izvođenje detekcije bez značajnih gubitaka u smislu preciznosti. Također, YOLOv5 koristi optimizacije poput CUDA (engl. *Compute Unified Device Architecture*) jezgre kako bi iskoristio GPU (engl. *Graphics Processing Unit*) arhitekturu za brže izračune. Uz to, omogućava bolje postavljanje sidrenih okvira kako bi poboljšao detekciju malih objekata, koji su ranije bili izazov za YOLO. Također donosi unaprijeđeni postupak treniranja, uključujući preneseno učenje na raznim pred-treniranim modelima. Ovo omogućuje brže treniranje i bolju generalizaciju modela za različite vrste objekata i scene [13].

YOLOv7 je najnovija verzija YOLO algoritma u trenutku izrade ovog rada koja se predstavlja kao još naprednija i učinkovitija. Koristi veći broj kanala za detekciju, što omogućuje bolje razlikovanje između različitih klasa objekata. Također, koristi dublje neuronske mreže s većim brojem slojeva, što povećava kapacitet modela i omogućuje bolje učenje kompleksnih značajki u slikama. Uz to, koristi napredne tehnike paralelizacije kako bi maksimalno iskoristio mogućnosti modernih višejezgrenih procesora i grafičkih kartica, što rezultira bržim izvođenjem algoritma [26].

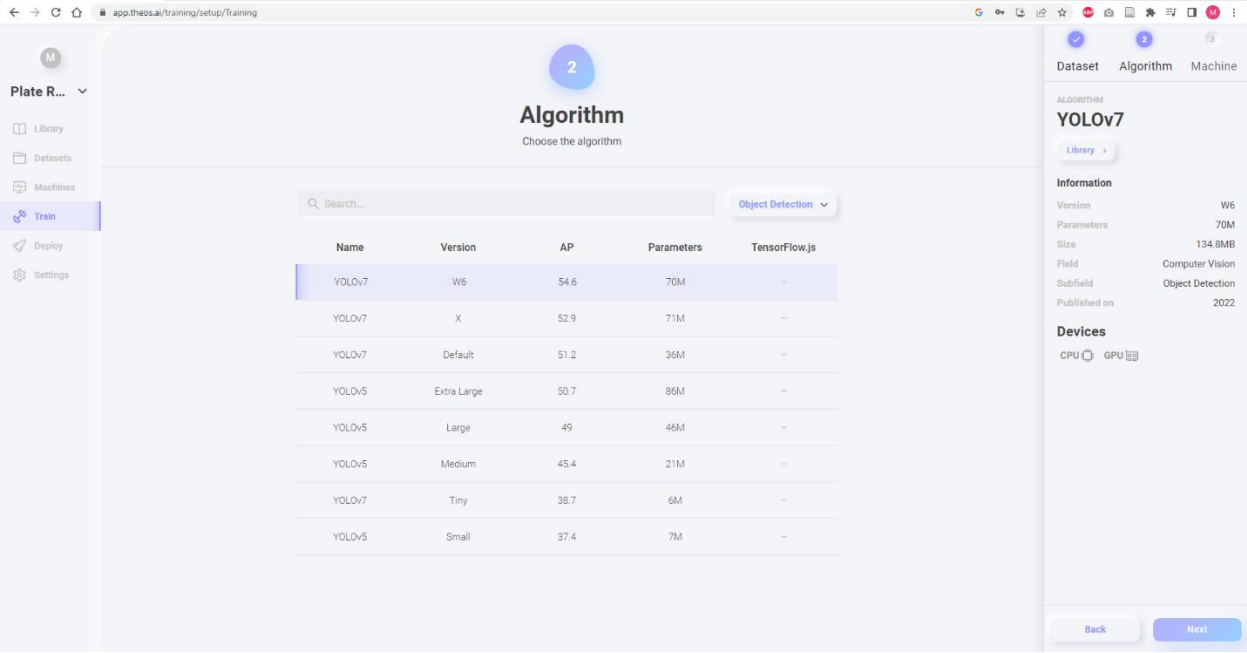
Za potrebe ovog diplomskog rada, testirat će se podverzije YOLOv5 i YOLOv7 algoritama. Općenito, podverzije YOLO algoritma predstavljaju taj isti algoritam, ali s drugačijim brojem parametara u neuronskoj mreži, što može rezultirati bržom i manje preciznom mrežom ili sporijom i više preciznom mrežom.

3.2. Vlastito treniranje YOLO modela

Za treniranje modela koristit će se podverzije YOLOv5 i YOLOv7 algoritama. Podverzije s malim brojem parametara koje će se trenirati su YOLOv5 *small* i YOLOv7 *tiny*, podverzije sa srednjim brojem parametara YOLOv5 *medium* i YOLOv7 *default* i podverzije s velikim brojem parametara YOLOv5 *extra large* i YOLOv7 *W6*. U nastavku će biti opisano prikupljanje i označavanje slika za vlastito treniranje YOLO modela, postupak vlastitog treniranja YOLO modela te radno okruženje i implementacija YOLO modela.

3.2.1. Alati za vlastito treniranje YOLO modela

Za vlastito treniranje YOLO modela korištena je web usluga *Theos AI* koja implementira izvorni kod YOLOv5 i YOLOv7 algoritama (slika 3.9.) [42]. Modele je također moguće trenirati izravno pomoću skripti YOLOv5 i YOLOv7 algoritama, ali zbog prednosti koje nudi *Theos AI* odabran je taj pristup. *Theos AI* omogućava pripremu skupa podataka, treniranje, spremanje i preuzimanje modela svih podverzija YOLOv5 i YOLOv7 algoritama na jednom mjestu, što značajno poboljšava organiziranost. Osim toga, u području strojnog učenja poznato je da treniranje modela za detekciju objekata može trajati jako dugo, stoga još jedna od prednosti korištenja *Theos AI* je mogućnost jednostavnog povezivanja s uslugom *Google Colab* koja ima funkciju infrastrukture kao usluge, odnosno korištenje grafičke kartice Nvidia A100 kao usluge oblaka za treniranje vlastitih modela [43].



Slika 3.9. Web usluga Theos AI i YOLO algoritmi za koje omogućava treniranje vlastitih modela [42]

3.2.2. Prikupljanje i označavanje slika za vlastito treniranje YOLO modela

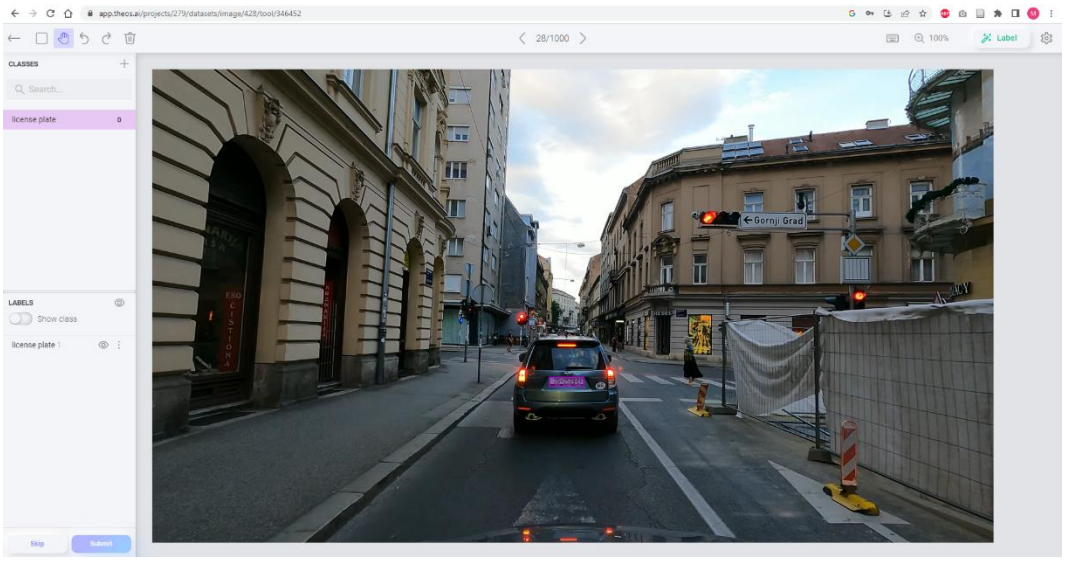
Kao izvor za prikupljanje skupa slika, poslužio je *Youtube* kanal City Drive [44]. Kanal nudi videozapise vožnje automobilom hrvatskim gradovima iz perspektive vozača, snimane kamerom GoPro Hero 9 te izvorne rezolucije 3840x2160 piksela. Prikupljeno je 1000 slika za treniranje uzimanjem snimke zaslona u situacijama kada se ispred vozača nalazilo vozilo pri različitim udaljenostima. Bez obzira na izvornu rezoluciju videozapisa, snimke zaslona ograničene su na rezoluciju korištenog monitora, u ovom slučaju 1920x1080 piksela.

Svaka slika koja sadrži osim čitljivih i jako udaljenu registarsku tablicu (slika 3.10.), izrezana je tako da ostanu samo čitljive (slika 3.11.). Znakovi jako udaljenih registarskih tablica nisu razlučivi ljudskom oku, stoga ni algoritam prepoznavanja znakova ne bi u ovom slučaju ostvario točan rezultat. Osim toga, ovakvi slučajevi bi mogli narušiti kvalitetu modela zbog njihove male količine, jer u slučaju neoznačavanja jako udaljenih registarskih tablica model bi ih mogao uračunati kao FP, a u slučaju označavanja kao FN. Skup slika koji se koristi za treniranje YOLO modela može se pronaći na DVD-u priloženom uz ovaj rad u mapi Prilog P.3.1.

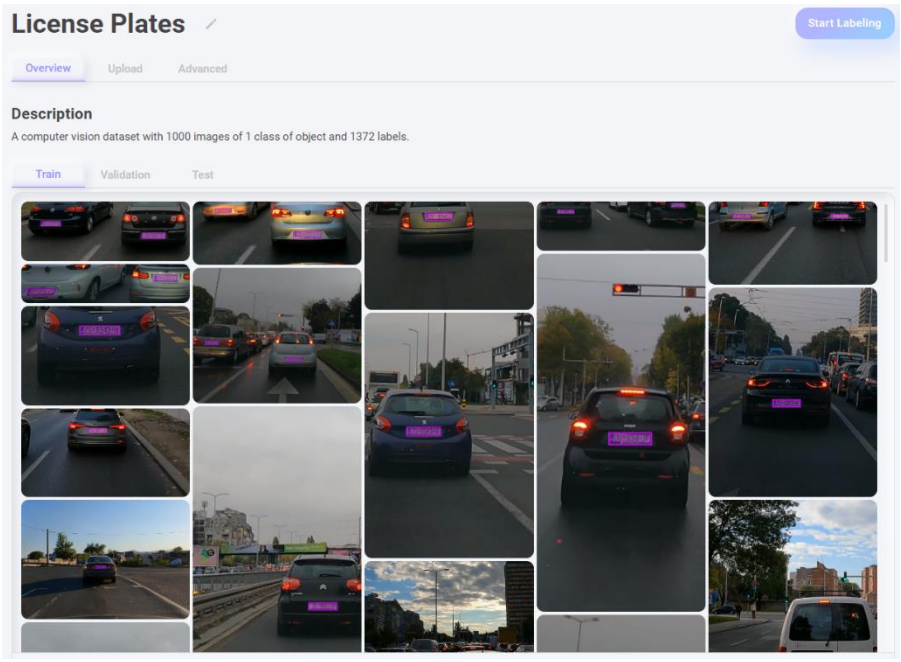
Slika 3.10. Slika koja sadrži jako udaljene registarske tablice vozila

Slika 3.11. Izrezana slika koja sadrži samo poželjnu registarsku tablicu vozila

Za ručno označavanje graničnih okvira na skupu slika za treniranje korištena je web usluga *Theos AI* (slika 3.12.). Na svakoj od 1000 slika ručno su označeni granični okviri svih registarskih tablica vozila, što je rezultiralo iznosom 1372 graničnih okvira (slika 3.13.) [42]. 1322 graničnih okvira označavaju hrvatske registarske tablice vozila, a 50 graničnih okvira označavaju strane registarske tablice vozila.



Slika 3.12. Ručno označena registarska tablica vozila na web usluzi Theos AI [42]

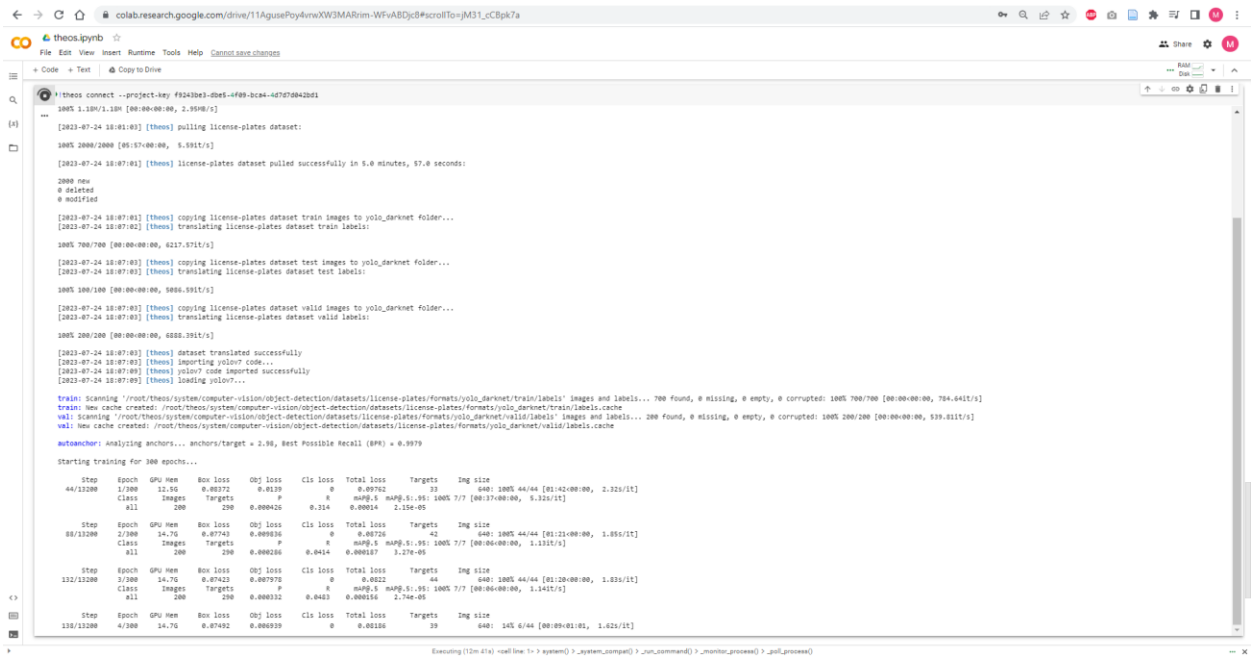


Slika 3.13. Skup ručno označenih slika na web usluzi Theos AI [42]

3.2.3. Vlastito treniranje YOLO modela

Odabire se prethodno označeni skup slika, algoritam pomoću kojeg se želi trenirati i grafička kartica koju se želi koristiti. Tijekom treniranja korišteno je svih 1372 ručno označenih graničnih okvira registarskih tablica vozila u 300 epoha uz količinu uzoraka 32 (engl. *batch size* 32) iz skupa podataka za svaku iteraciju. Epoha se odnosi na jedno potpuno prolazno treniranje kroz kompletan skup podataka, a težine modela se ažuriraju nakon svake iteracije na temelju izračunate funkcije gubitka (*mini-batch* metoda). *Batch size* odabire se proizvoljno prije postupka treniranja. Veći *batch size* omogućava brže treniranje jer se više podataka obrađuje istovremeno, ali zahtjeva više memorije.

Za potrebe ovog rada, istrenirano je šest modela: YOLOv5 *small*, YOLOv5 *medium*, YOLOv5 *extra large*, YOLOv7 *tiny*, YOLOv7 *default* i YOLOv7 *W6*, od kojih svaki ima različit broj parametara te će ponuditi jedinstven rezultat prilikom testiranja u vidu preciznosti i brzine izvođenja. Za treniranje je korištena grafička kartica Nvidia A100 koju omogućava *Google Colab* kao uslugu oblaka (slika 3.14.) [43].



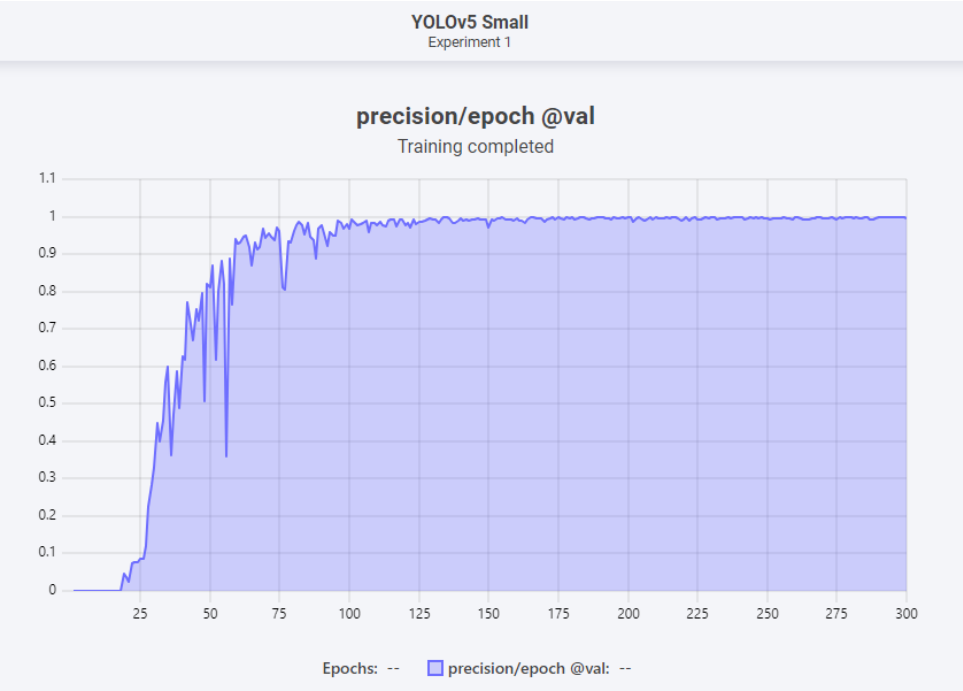
Slika 3.14. Primjer treniranja koristeći uslugu Google Colab [43]

Nakon treniranja, moguće je pregledati rezultate svakog modela na web usluzi *Theos AI*, te preuzeti *.weights* datoteku kako bi se koristila na lokalnom sustavu. *Theos AI* ne sprema

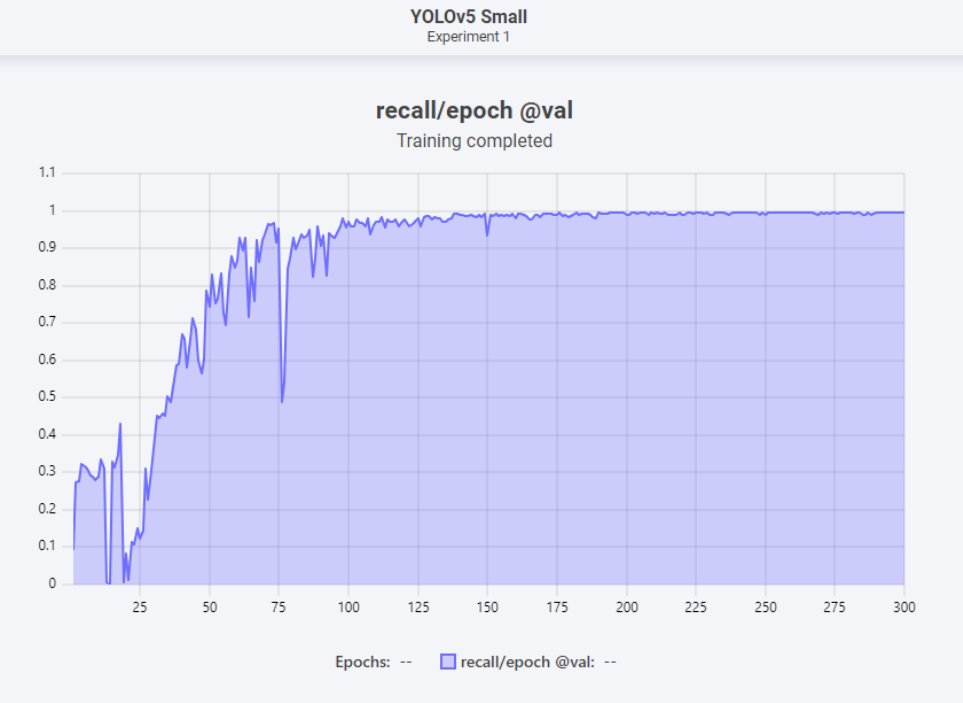
confusion matrix i *batch* slike nakon treniranja, stoga to neće ovdje biti pokazano. Osim metrika preciznosti, odziva, mAP@0.5 i mAP@0.5:95 može se pregledati i metrika *fitness*. Metrika *fitness* se računa kao zbroj 10% vrijednosti metrike mAP@0.5 i 90% vrijednosti metrike mAP@0.5:95 (3-8).

$$fitness = 0.1 * mAP@0.5 + 0.9 * mAP@0.5{:}95 \quad (3\text{-}8)$$

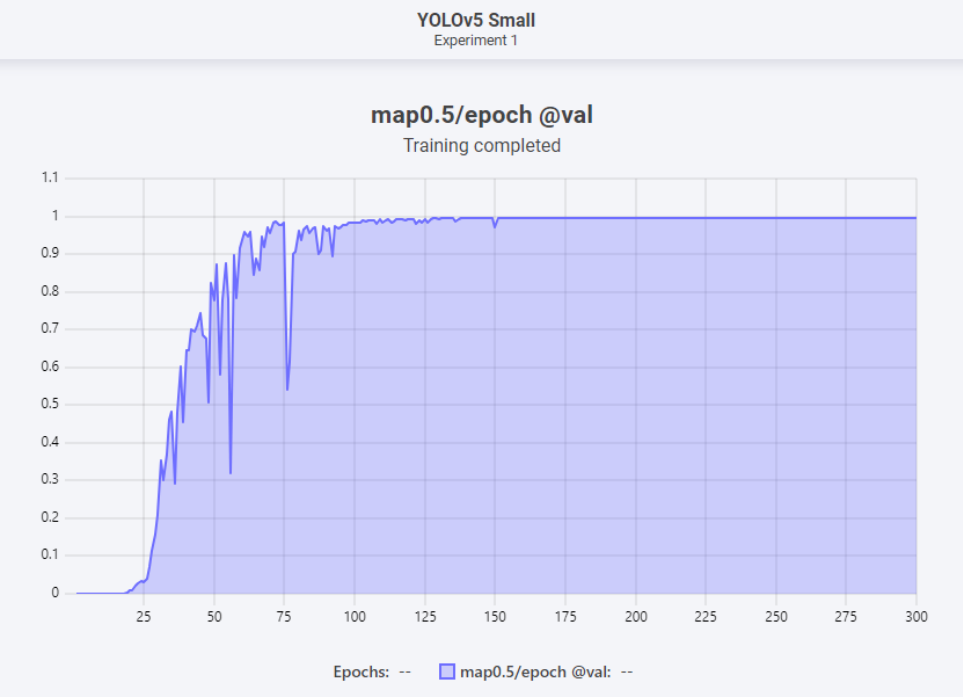
U nastavku su prikazani kao primjer ostvareni rezultati u obliku grafa metrika preciznosti (slika 3.15.), odziva (slika 3.16.), mAP@0.5 (slika 3.17.), mAP@0.5:95 (slika 3.18.) i *fitness*-a (slika 3.19.) uz epohe nakon treniranja modela YOLOv5 *small*. U tablici 3.1. navedeni su rezultati metrike *fitness* za epohu u kojoj je svaki pojedinačni prethodno trenirani model ostvario najvišu vrijednost metrike *fitness* tijekom treniranja. Slike ostvarenih rezultata u obliku grafa metrika preciznosti, odziva, mAP@0.5, mAP@0.5:95 i *fitness*-a uz epohe za trenirane modele YOLOv5 *small*, YOLOv5 *medium*, YOLOv5 *extra large*, YOLOv7 *tiny*, YOLOv7 *default* i YOLOv7 *W6* mogu se pronaći na DVD-u priloženom uz ovaj rad u mapi Prilog P.3.2.



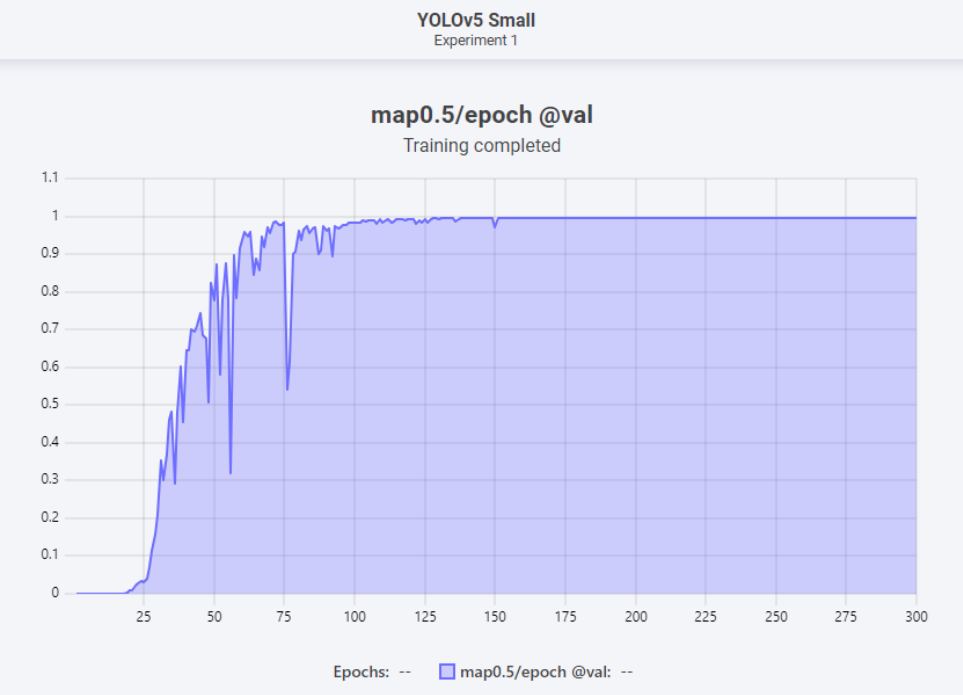
Slika 3.15. Graf preciznost-epoha nakon treniranja YOLOv5 small modela na web usluzi Theos AI [42]



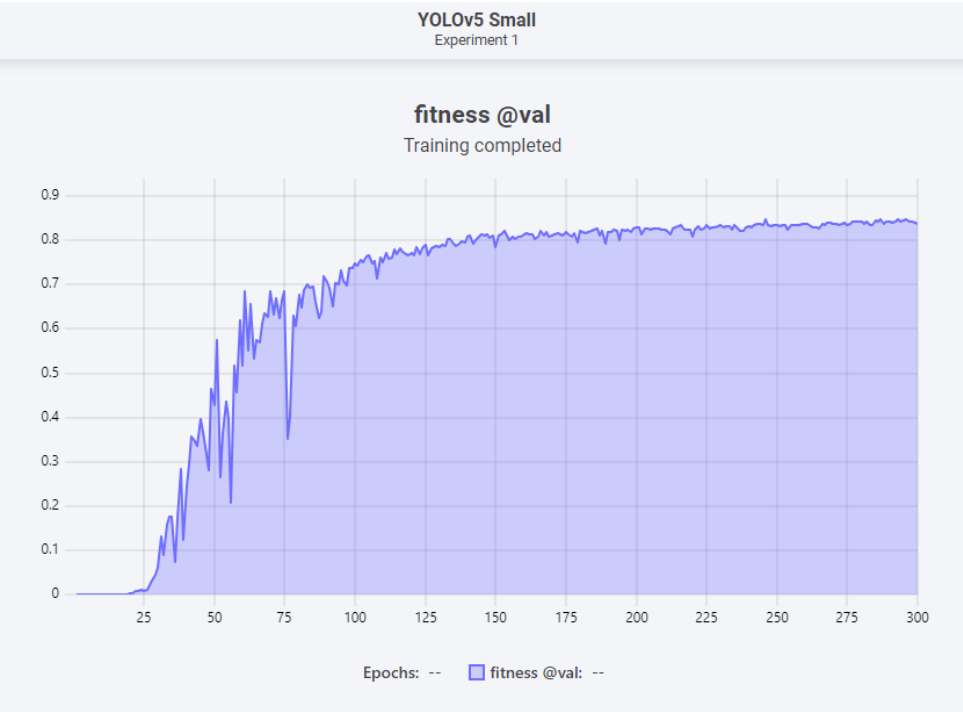
Slika 3.16. Graf odziv-epoha nakon treniranja YOLOv5 small modela na web usluzi Theos AI [42]



Slika 3.17. Graf mAP@0.5-epoha nakon treniranja YOLOv5 small modela na web usluzi Theos AI [42]



Slika 3.18. Graf mAP@0.5:95-epoha nakon treniranja YOLOv5 small modela na web usluzi Theos AI [42]



Slika 3.19. Graf fitness-epoha nakon treniranja YOLOv5 small modela na web usluzi Theos AI [42]

Tablica 3.1. Rezultati metrike fitness za epohu u kojoj je svaki pojedinačni prethodno trenirani model ostvario najvišu vrijednost metrike fitness tijekom treniranja

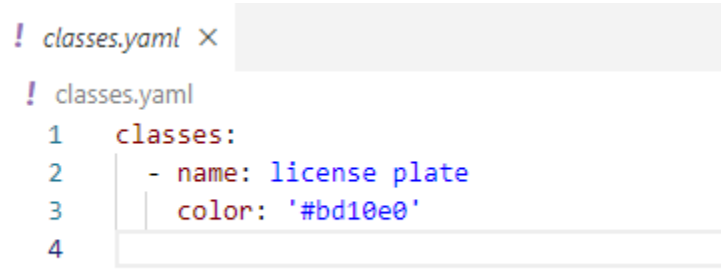
Trenirani model	**Najviša vrijednost metrike *fitness* pojedinačnog modela unutar 300 epoha**
YOLOv5 *small*	0.8490 (epoha 293/300)
YOLOv5 *meidum*	0.8591 (epoha 292/300)
YOLOv5 *extra large*	0.8520 (epoha 251/300)
YOLOv7 *tiny*	0.8162 (epoha 275/300)
YOLOv7 *default*	0.8360 (epoha 261/300)
YOLOv7 *W6*	0.8884 (epoha 300/300)

3.2.4. Radno okruženje i implementacija vlastitih YOLO modela

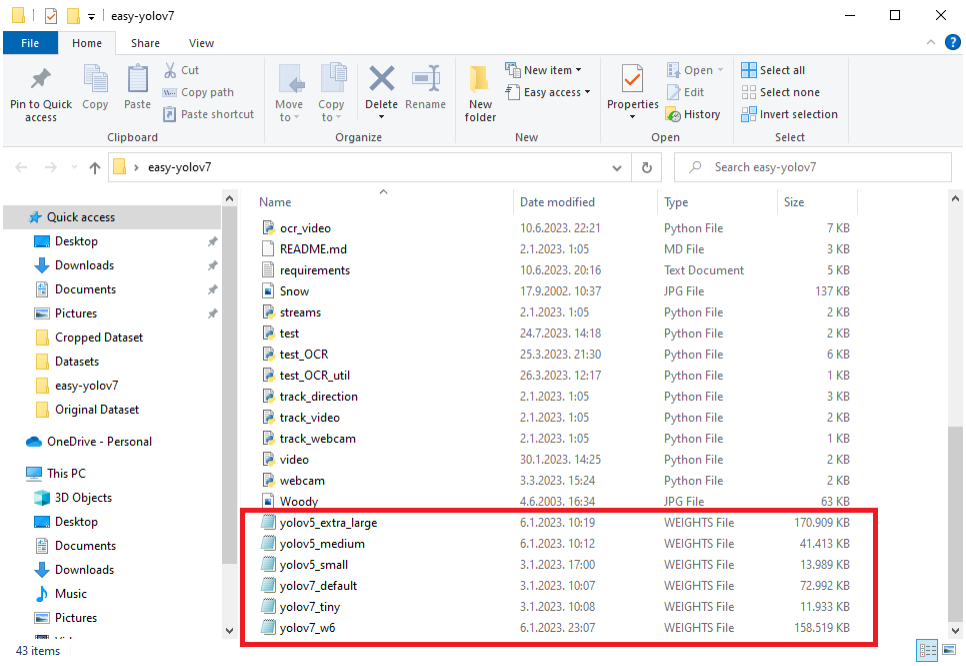
Za radno okruženje korištena je jednostavna implementacija YOLOv7 algoritma u *PyTorch* okviru koja je dostupna kao otvoreni kod na platformi *GitHub* [45]. *PyTorch* je okvir izrađen u programskom jeziku *Python* za duboko učenje koji se koristi za olakšavanje izgradnje, treniranja i evaluacije dubokih neuronskih mreža [46]. Prije korištenja ovog okruženja, potrebno je instalirati zahtjevane pakete koji se nalaze u *requirements.txt* datoteci pozivom sljedeće naredbe unutar naredbenog retka:

```
pip install -r requirements.txt
```

Uz to, potrebno je izmijeniti *classes.yaml* datoteku prema vlastitoj potrebi (slika 3.20.) i dodati preuzete *.weights* datoteke istreniranih modela u mapu radnog okruženja (slika 3.21.).

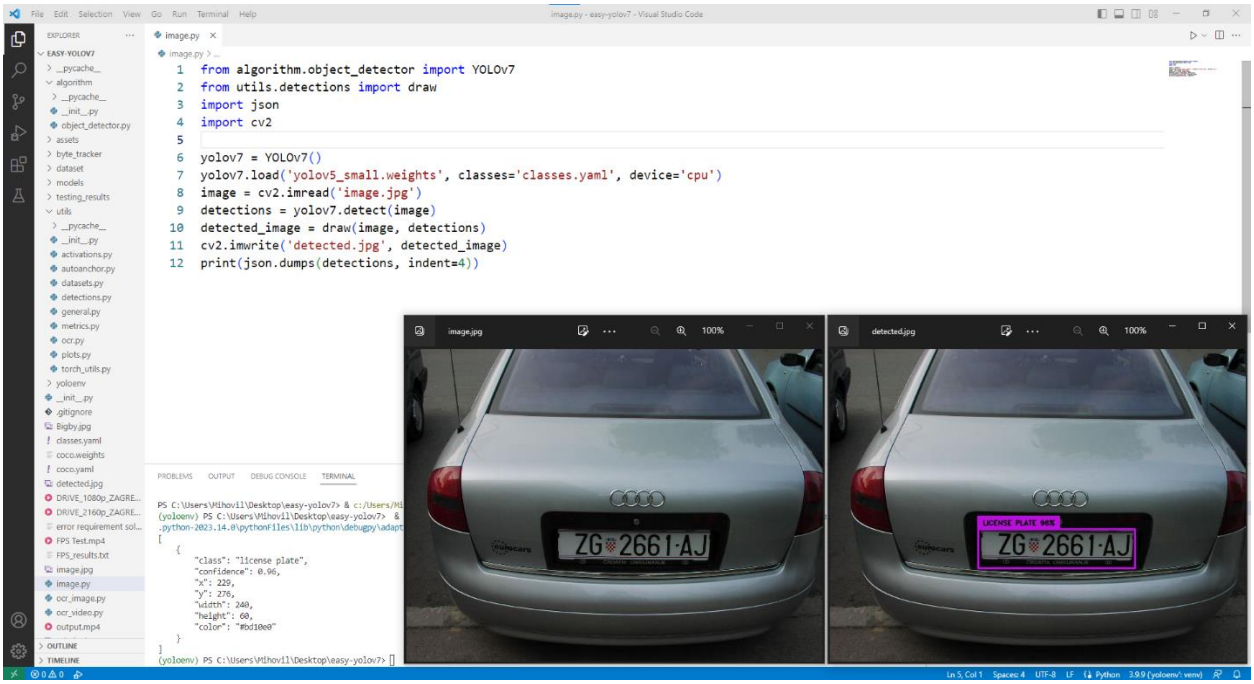


Slika 3.20. Datoteka classes.yaml



Slika 3.21. Mapa radnog okruženja implementacije YOLOv7 algoritma u PyTorch okviru

Nakon toga moguće je pomoću skripte *image.py* detektirati objekte za koje je prethodno treniran model (slika 3.22.). Vrijednost izražena u postotku nakon imena klase graničnog okvira, u ovom slučaju *license plate*, označava pouzdanost detekcije.



Slika 3.22. Skripta image.py i rezultat njezinog izvođenja

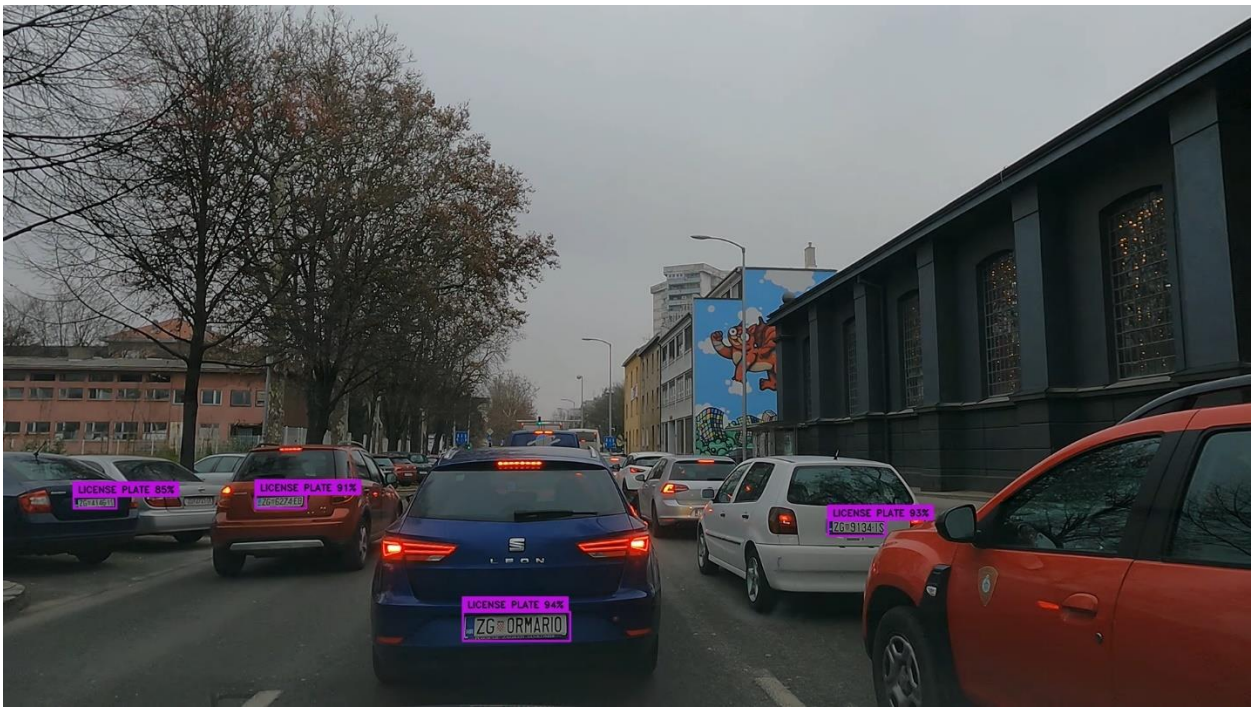
U modulu *object_detector.py* implementiran je izvorni YOLOv7 algoritam, a modul *detections.py* omogućava crtanje graničnih okvira. Učita se željeni model, odabere odgovarajuća *.yaml* datoteka i uređaj na kojem će se izvoditi skripta. Slika na kojoj se vrši detekcija mora biti sadržana unutar mape radnog okruženja i učitana pomoću naredbe *imread()* CV2 biblioteke. U ovom slučaju bit će detektirane sve registarske tablice vozila koje se nalaze na slici, što ne odgovara slučaju upotrebe ovog rada. Stoga su u funkciji *detect()* modula *object_detector.py* dodana ograničenja nakon kojih će se filtrirati detekcije graničnih okvira registarskih tablica vozila na one koje se nalaze unutar područja ispred vozača, odnosno na one čije se točke nalaze unutar ograničenja. Ograničenja su ručno prilagođena sličicama videozapisa ranije spomenutog *Youtube* kanala City Drive [44] (slika 3.23.). Na slici 3.24. vidljiv je rezultat predviđanja bez primjene ograničenja, a na slici 3.25. i slici 3.26. vidljiv je rezultat predviđanja uz primjenu ograničenja.

object_detector.py

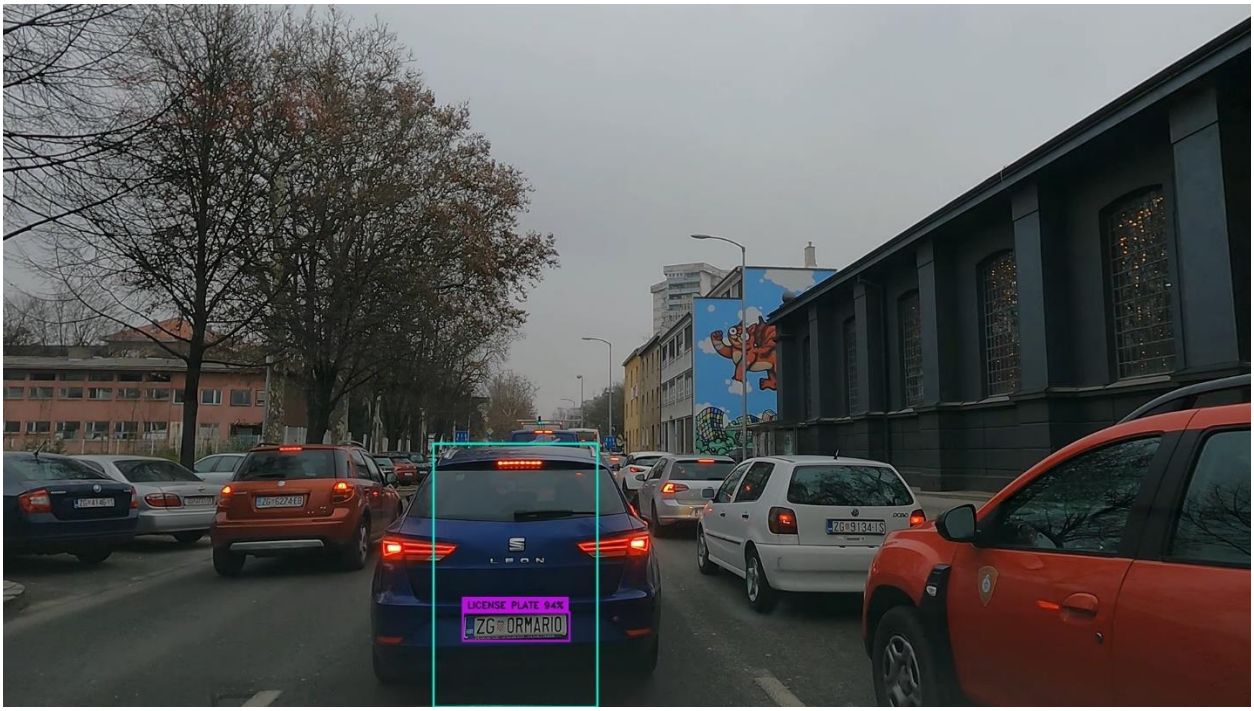
algorithm > object_detector.py > YOLOv7 > detect

```
71      def detect(self, img, track=False):
72          with torch.no_grad():
73              im0, img = self.__parse_image(img)
74              pred = self.model(img)[0]
75              pred = non_max_suppression(pred, self.settings['conf_thres'], self.settings['iou_thres'])
76              raw_detection = np.empty((0,6), float)
77
78              height = im0.shape[0]
79              width = im0.shape[1]
80
81              top_border = height / 1.5
82              left_border = width / 2.9
83              right_border = width / 2.1
84
85              for det in pred:
86                  if len(det) > 0:
87                      det[:, :4] = scale_coords(img.shape[2:], det[:, :4], im0.shape).round()
88                      for *xyxy, conf, cls in reversed(det):
89                          if ((int(xyxy[0]) > left_border and
90                               xyxy[0] < right_border) and
91                              (int(xyxy[2]) > left_border and
92                               xyxy[2] < right_border) and
93                               (int(xyxy[1]) > top_border)):
94                              raw_detection = np.concatenate((raw_detection,
95                                                              [[int(xyxy[0]),
96                                                                int(xyxy[1]),
97                                                                int(xyxy[2]),
98                                                                int(xyxy[3]),
99                                                                round(float(conf), 2),
100                                                               int(cls)]]))
101
102             if track:
103                 raw_detection = self.tracker.update(raw_detection)
104
105             detections = Detections(raw_detection, self.classes, tracking=track).to_dict()
```

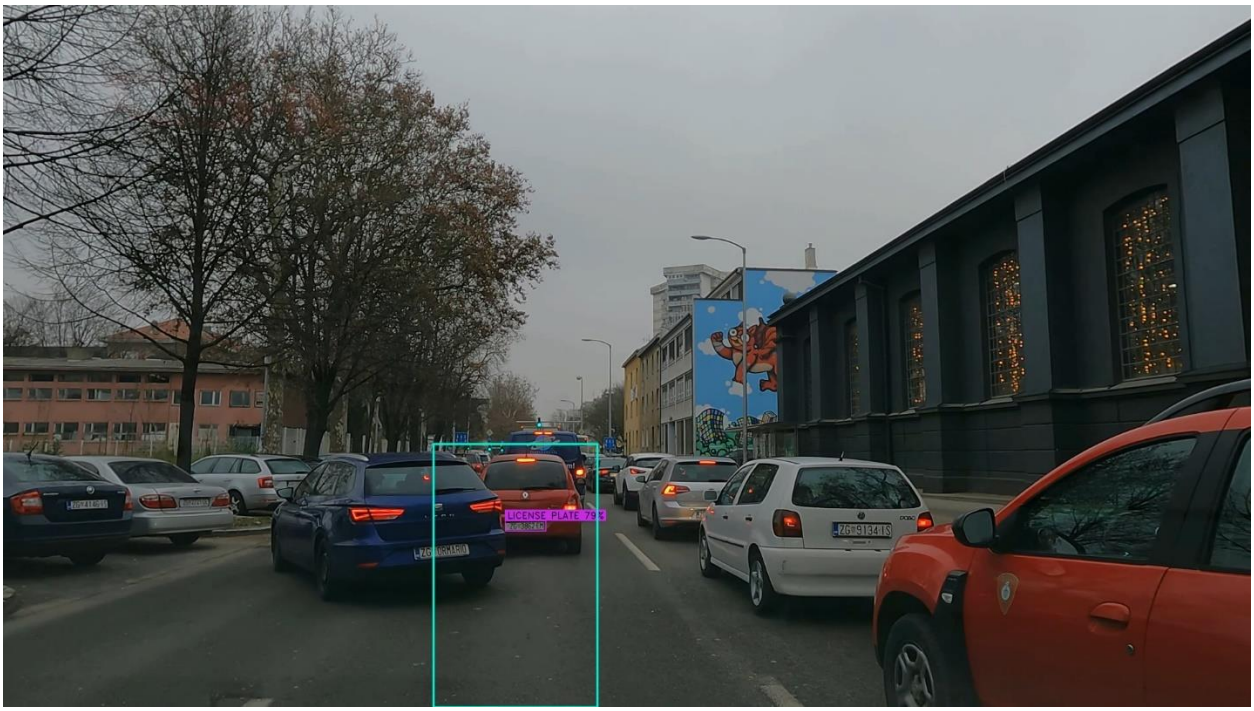
Slika 3.23. Kod za filtriranje detekcija graničnog okvira registarskih tablica vozila na one koje se nalaze unutar područja ispred vozača



Slika 3.24. Detekcije graničnih okvira registarskih tablica vozila bez filtriranja na one koje se nalaze unutar područja ispred vozača



Slika 3.25. Filtriranje detekcija graničnog okvira registarske tablice vozila na one koje se nalaze unutar područja ispred vozača – 1. primjer



Slike 3.26. Filtriranje detekcija graničnog okvira registarske tablice vozila na one koje se nalaze unutar područja ispred vozača – 2. primjer

Ograničenje na slikama se inače ne iscrtava, nego je prikazano samo na ovim slikama kao primjer pomoću funkcije *rectangle()* CV2 biblioteke, pripadajući *Python* kod naveden je u nastavku:

```
height = im0.shape[0]
width = im0.shape[1]

top_border = int(height / 1.5)
left_border = int(width / 2.9)
right_border = int(width / 2.1)

cv2.rectangle(image, (left_border, top_border, right_border, height),
(210,240,0), thickness=2, lineType=cv2.LINE_AA)
```

Ukoliko bi se za snimanje koristila kamera drugačije širine kuta snimanja ili drugačijeg položaja snimanja, morale bi se provesti izmjene ograničenja nakon kojih će se filtrirati detekcije graničnih okvira registarskih tablica vozila na one koje se nalaze unutar područja ispred vozača, odnosno na one čije se točke nalaze unutar ograničenja.

3.3. Vlastito testiranje i usporedba YOLO modela

Testiranje vlastito treniranih modela za detekciju registarskih tablica vozila bit će provedeno na skupu slika koji je ručno označen i sadrži 500 slika. S obzirom da web usluga *Theos AI* ne nudi mogućnost testiranja, testiranje će biti provedeno na lokalnom računalu. Za krajnju usporedbu u obzir će se uzeti F1 mjera i brzina obrade sličica u sekundi (FPS).

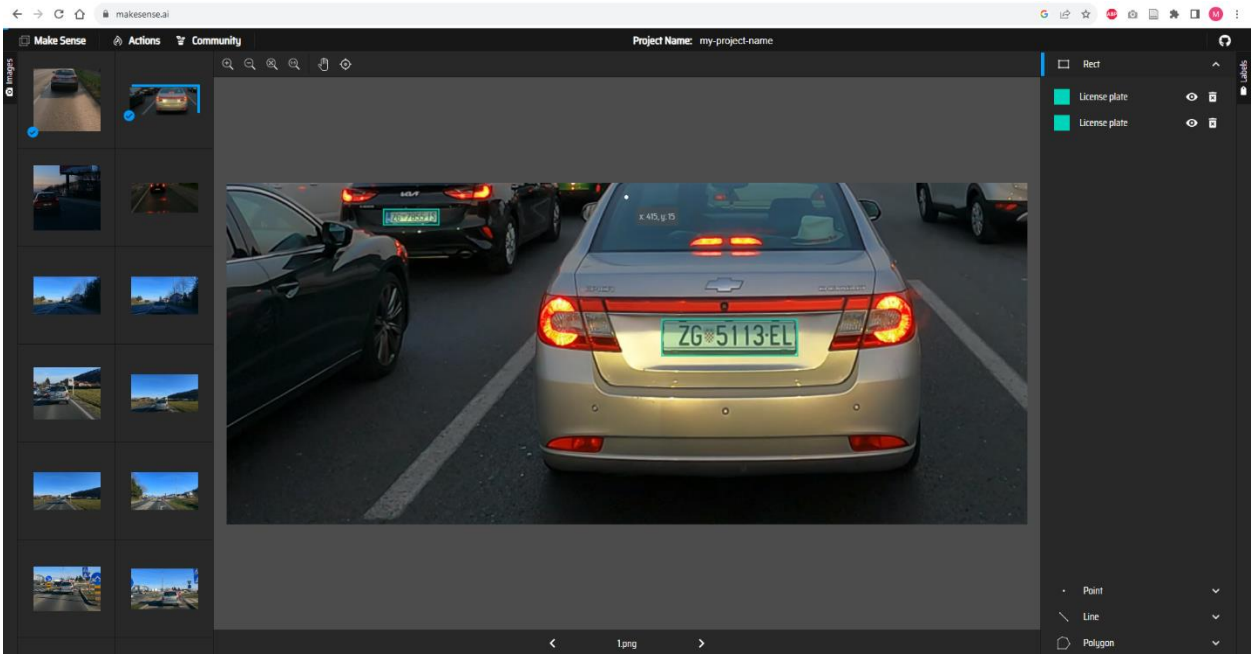
3.3.1. Prikupljanje i označavanje slika za testiranje YOLO modela

Skup slika za testiranje prikupljen je s istog *Youtube* kanala kao i skup slika za treniranje [44]. Kanal pruža videozapise vožnje automobilom hrvatskim gradovima iz perspektive vozača, snimane pomoću kamere GoPro Hero 9, s izvornom rezolucijom 3840x2160 piksela [44]. Prikupljeno je 500 slika koje prikazuju situacije kada se ispred vozača nalazi vozilo na različitim udaljenostima. Snimke zaslona su uzete u tim situacijama, ali su ograničene na rezoluciju monitora 1920x1080 piksela, bez obzira na izvornu rezoluciju videozapisa.

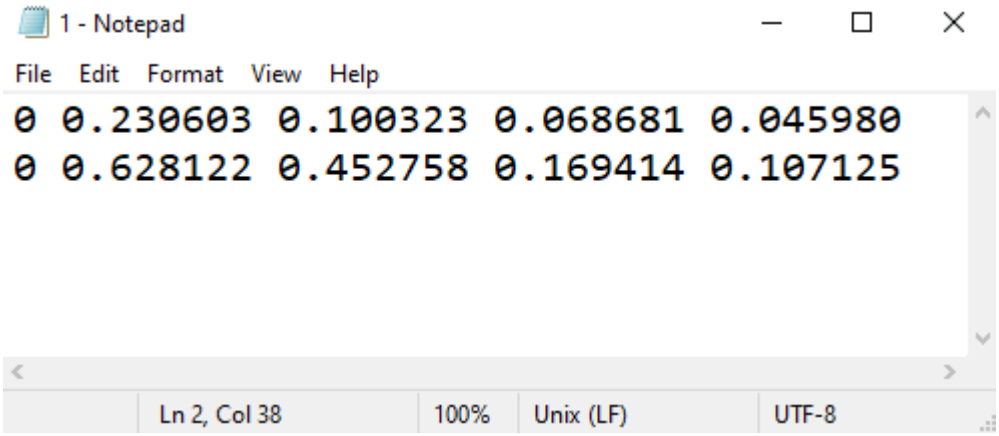
Kao i kod skupa za treniranje, prije testiranja slike su izrezane tako da sadrže samo registarske tablice vozila na poželjnoj udaljenosti, odnosno one koje je moguće pročitati ljudskim okom. Testiranje će se provesti bez ranije spomenutog ograničenja, jer se na mnogim slikama može pronaći više čitljivih registarskih tablica vozila.

Skup slika koji se koristi za testiranje YOLO modela može se pronaći na DVD-u priloženom uz ovaj rad u mapi Prilog P.3.3. Osim skupa slika, sa spomnutog *Youtube* kanala preuzet je i dio videozapisa u trajanju 5 sekundi koji će poslužiti za testiranje YOLO modela za brzinu izvođenja. Videozapis se može pronaći na DVD-u priloženom uz ovaj rad u mapi Prilog P.3.4.

Budući da *Theos AI* web usluga ne omogućuje testiranje i preuzimanje označenog skupa slika, skup slika za testiranje ručno je označen pomoću web usluge *Make Sense* (slika 3.27.) [47]. Na svakoj od 500 slika ručno su označeni granični okviri svih registarskih tablica vozila, što je rezultiralo iznosom 636 graničnih okvira. 618 graničnih okvira označavaju hrvatske registarske tablice, a 18 graničnih okvira označavaju strane registarske tablice. Nakon završetka ručnog označavanja graničnih okvira registarskih tablica vozila, mogu se preuzeti *.txt* datoteke koje sadrže vrijednosti koje opisuju granične okvire (slika 3.28.).



Slika 3.27. Ručno označavanje graničnih okvira na web usluzi Make Sense [47]

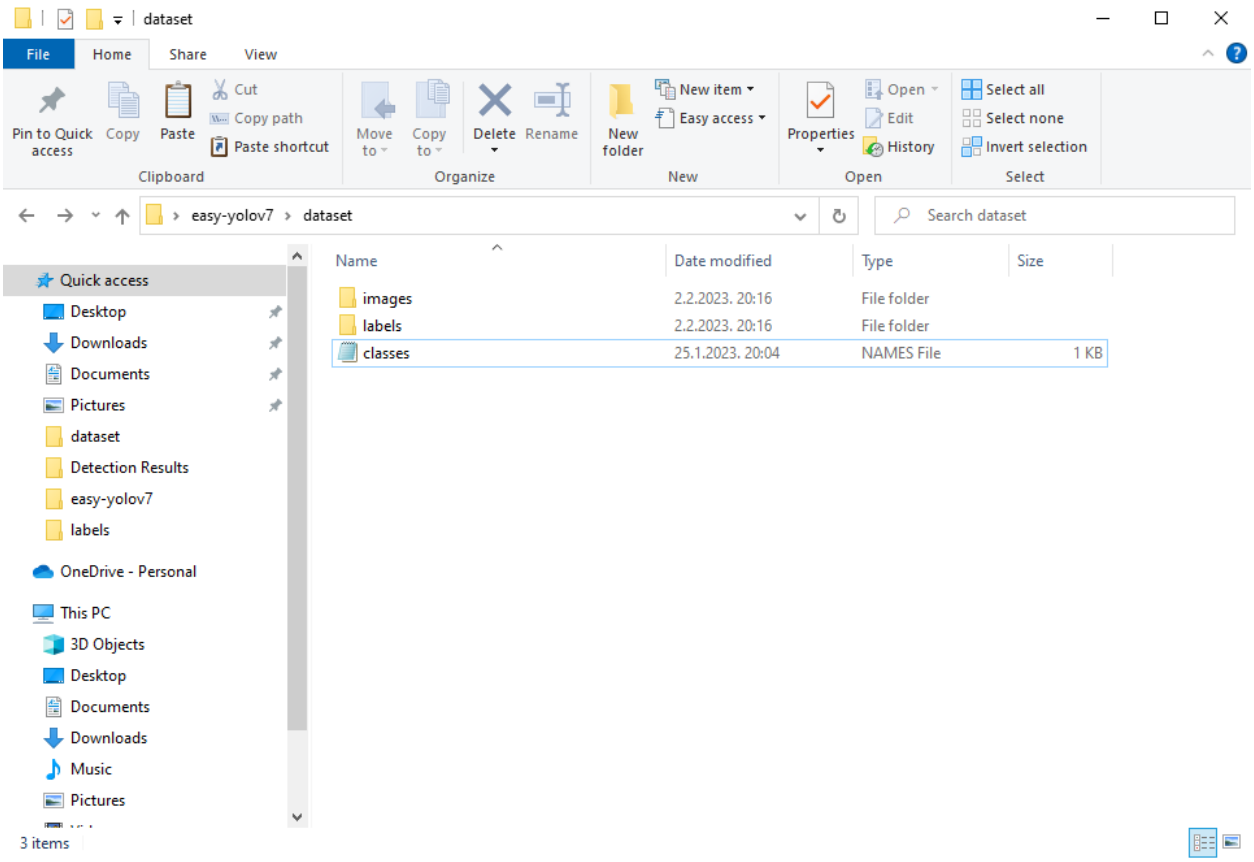


Slika 3.28. YOLO anotacija graničnih okvira

U svakom redu nalaze se vrijednosti jednog graničnog okvira. Prva vrijednost označava broj klase, u ovom slučaju koristi se samo klasa registarske tablice vozila pa će svaki granični okvir za prvu vrijednost imati 0. Druga i treća vrijednost označavaju *x* i *y* vrijednosti centra graničnog okvira normalizirane od 0 do 1 u odnosu na širinu i visinu slike. Četvrta i peta vrijednost označavaju širinu i visinu graničnog okvira normalizirane od 0 do 1 u odnosu na izvornu širinu i visinu slike, odnosno širinu i visinu slike prije skaliranja na rezoluciju koja je definirana prilikom treniranja YOLO modela.

3.3.2. Skripte za vlastito testiranje YOLO modela

Za testiranje vlastito treniranih modela za detekciju registarskih tablica vozila bit će korišten modul za testiranje službenog YOLOv7 *GitHub* repozitorija [26]. Kako bi skripta ispravno funkcionirala, potrebno je dodati skup slika na kojem će se provesti testiranje i odgovarajuće informacije u radno okruženje. Kreira se mapa *dataset*, a unutar nje mape *images* i *labels* te NAMES datoteka koja sadrži imena klasa, u ovom slučaju jedna klasa *license plate* (slika 3.29.). Tijekom testiranja koristit će se svih 636 ručno označenih graničnih okvira registarskih tablica vozila.



Slika 3.29. Mapa dataset radnog okruženja implementacije YOLO algoritma

Cilj ovog testiranja je za svaku kombinaciju prethodno vlastito treniranih modela YOLOv5 *small*, YOLOv5 *medium*, YOLOv5 *extra large*, YOLOv7 *tiny*, YOLOv7 *default*, YOLOv7 *W6* s pragovima pouzdanosti 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 dobiti F1 mjeru. Stoga je izlaz modula za testiranje modificiran tako da vraća samo vrijednosti preciznosti i odziva što je potrebno kako bi se izračunala F1 mjera.

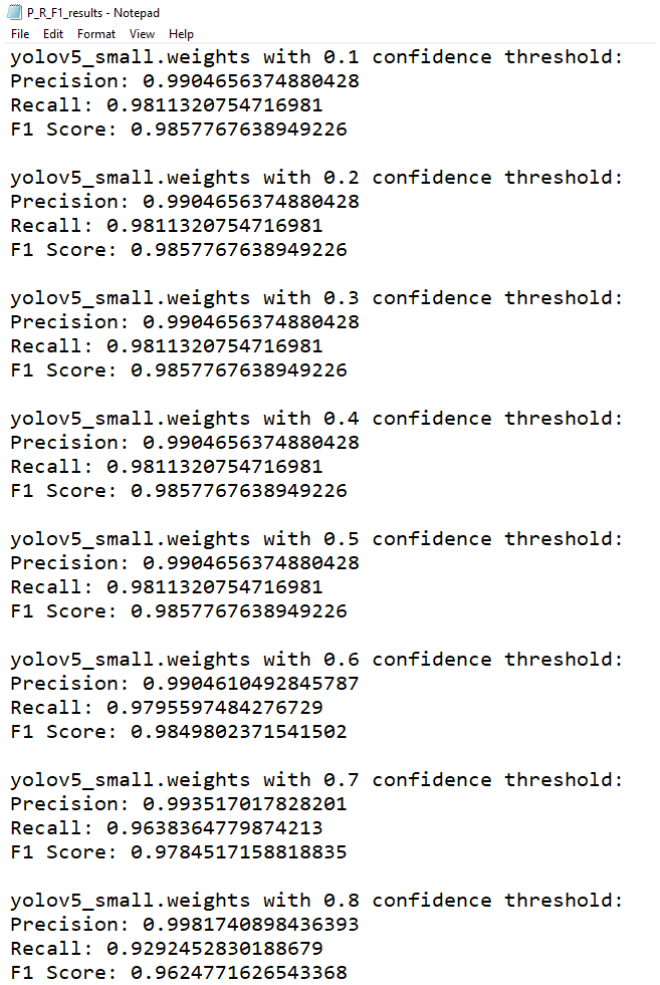
Kako bi se ubrzalo cjelokupno testiranje, izrađena je skripta koja automatizira testiranje svih kombinacija vlastito treniranih modela podverzija YOLOv5 i YOLOv7 algoritama s određenim pragovima pouzdanosti te računa F1 mjeru i sprema rezultate u *.txt* datoteku (slika 3.30.).

test.py

```python
from algorithm.object_detector import YOLOv7

def test():
    yolov7 = YOLOv7()
    model_weights = ['yolov5_small.weights',
                     'yolov5_medium.weights',
                     'yolov5_extra_large.weights',
                     'yolov7_tiny.weights',
                     'yolov7_default.weights',
                     'yolov7_w6.weights']
    confidence_thresholds = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]

    for item in model_weights:
        weights = item
        if weights == 'yolov7_w6.weights':
            yolov7.set(img_size=1280)
        else:
            yolov7.set(img_size=640)

        for value in confidence_thresholds:

            yolov7.set(conf_thres=value)

            yolov7.load(weights, classes='classes.yaml', device='gpu')
            P, R = yolov7.test()

            F1 = 2*((P*R)/(P+R))
            conf_thres = str(yolov7.settings['conf_thres'])
            result = weights + " with " + conf_thres + " confidence threshold:\n" + \
                "Precision: " + str(P) + "\n" + \
                    "Recall: " + str(R) + "\n" + \
                        "F1 Score: " + str(F1) + "\n\n"
            print(result)

            text_file = open("F1_results.txt", "a")
            text_file.write(result)
            text_file.close()

if __name__ == '__main__':
    test()
```

Slika 3.30. Skripta za testiranje vlastito treniranih modela podverzija YOLOv5 i YOLOv7 algoritama uz pragove pouzdanosti od 0.1 do 0.9 za metriku F1

Na slici 3.31. prikazan je dio rezultata spremljenih u *.txt* datoteku nakon izvođenja skripte za testiranje vlastito treniranih modela podverzija YOLOv5 i YOLOv7 algoritama uz pragove pouzdanosti od 0.1 do 0.9 za metriku F1.



Slika 3.31. Dio rezultata spremljenih u .txt datoteku nakon izvođenja skripte za testiranje vlastito treniranih modela podverzija YOLOv5 i YOLOv7 algoritama uz pragove pouzdanosti od 0.1 do 0.9 za metriku F1

Iz rezultata vidljivih u tablicama 3.2. – 3.7. se očituje da ovi modeli pri nižim pragovima pouzdanosti imaju veću vrijednost F1 mjere. To znači da ovi modeli unatoč prisutnosti *false positive*-a pri nižim pragovima pouzdanosti daju bolje rezultate, nego pri višim pragovima pouzdanosti čiji je smisao eliminirati što više *false positive*-a uz rizik gubitka *true positive*-a, odnosno rizik prisutstva *false negative*-a. Iznimka su modeli YOLOv5 *extra large* i YOLOv7 *W6*,

za koje se uz prag pouzdanosti 0.7 mogu iščitati najbolji rezultati. U svakoj tablici označeni su retci koji sadrže najveću vrijednost F1 mjere.

Tablica 3.2. Rezultati testiranja vlastito treniranog modela YOLOv5 small uz pragove pouzdanosti od 0.1 do 0.9 za metrike preciznosti, odziva i F1 mjere – označeni redovi sadrže najveći iznos F1 mjere

YOLOv5 *small*			
Prag pouzdanosti	**Preciznost**	**Odziv**	**F1**
0.1	0.9904656374880428	0.9811320754716981	0.9857767638949226
0.2	0.9904656374880428	0.9811320754716981	0.9857767638949226
0.3	0.9904656374880428	0.9811320754716981	0.9857767638949226
0.4	0.9904656374880428	0.9811320754716981	0.9857767638949226
0.5	0.9904656374880428	0.9811320754716981	0.9857767638949226
0.6	0.9904610492845787	0.9795597484276729	0.9849802371541502
0.7	0.9935170178282010	0.9638364779874213	0.9784517158818835
0.8	0.9981740898436393	0.9292452830188679	0.9624771626543368
0.9	1.0000000000000000	0.8081761006289309	0.8939130434782608

Tablica 3.3. Rezultati testiranja vlastito treniranog modela YOLOv5 medium uz pragove pouzdanosti od 0.1 do 0.9 za metrike preciznosti, odziva i F1 mjere – označeni redovi sadrže najveći iznos F1 mjere

YOLOv5 *medium*			
Prag pouzdanosti	**Preciznost**	**Odziv**	**F1**
0.1	0.9902873623687088	0.9669811320754716	0.9784954874218832
0.2	0.9902873623687088	0.9669811320754716	0.9784954874218832
0.3	0.9902873623687088	0.9669811320754716	0.9784954874218832
0.4	0.9902873623687088	0.9669811320754716	0.9784954874218832
0.5	0.9902873623687088	0.9669811320754716	0.9784954874218832
0.6	0.9902873623687088	0.9669811320754716	0.9784954874218832
0.7	0.9918831168831169	0.9606918238993710	0.9760383386581470
0.8	0.9916943521594684	0.9386792452830188	0.9644588045234248
0.9	1.0000000000000000	0.8632075471698113	0.9265822784810126

Tablica 3.4. Rezultati testiranja vlastito treniranog modela YOLOv5 extra large uz pragove pouzdanosti od 0.1 do 0.9 za metrike preciznosti, odziva i F1 mjere – označeni redovi sadrže najveći iznos F1 mjere

YOLOv5 *extra large*			
Prag pouzdanosti	**Preciznost**	**Odziv**	**F1**
0.1	0.9810208123245149	0.9764150943396226	0.9787125348562751
0.2	0.9810208123245149	0.9764150943396226	0.9787125348562751
0.3	0.9810208123245149	0.9764150943396226	0.9787125348562751
0.4	0.9810208123245149	0.9764150943396226	0.9787125348562751
0.5	0.9810208123245149	0.9764150943396226	0.9787125348562751
0.6	0.9810208123245149	0.9764150943396226	0.9787125348562751
0.7	0.9810426540284360	0.9764150943396226	0.9787234042553191
0.8	0.9884297520661157	0.9402515723270440	0.9637389202256245
0.9	0.9980657640232108	0.8113207547169812	0.8950563746747614

Tablica 3.5. Rezultati testiranja vlastito treniranog modela YOLOv7 tiny uz pragove pouzdanosti od 0.1 do 0.9 za metrike preciznosti, odziva i F1 mjere – označeni redovi sadrže najveći iznos F1 mjere

YOLOv7 *tiny*			
Prag pouzdanosti	**Preciznost**	**Odziv**	**F1**
0.1	0.9838186754982042	0.9559691634462688	0.9696940019075525
0.2	0.9838186754982042	0.9559691634462688	0.9696940019075525
0.3	0.9838186754982042	0.9559691634462688	0.9696940019075525
0.4	0.9838186754982042	0.9559691634462688	0.9696940019075525
0.5	0.9837925445705025	0.9544025157232704	0.9688747007182761
0.6	0.9900990099009901	0.9433962264150944	0.9661835748792270
0.7	0.9949664429530202	0.9323899371069182	0.9626623376623377
0.8	0.9982668977469671	0.9056603773584906	0.9497114591920858
0.9	0.9980158730158730	0.7908805031446541	0.8824561403508773

Tablica 3.6. Rezultati testiranja vlastito treniranog modela YOLOv7 default uz pragove pouzdanosti od 0.1 do 0.9 za metrike preciznosti, odziva i F1 mjere – označeni redovi sadrže najveći iznos F1 mjere

YOLOv7 *default*			
Prag pouzdanosti	**Preciznost**	**Odziv**	**F1**
0.1	0.9839498586363404	0.9669811320754716	0.9753917004132687
0.2	0.9839498586363404	0.9669811320754716	0.9753917004132687
0.3	0.9839498586363404	0.9669811320754716	0.9753917004132687
0.4	0.9839498586363404	0.9669811320754716	0.9753917004132687
0.5	0.9839498586363404	0.9669811320754716	0.9753917004132687
0.6	0.9839498586363404	0.9669811320754716	0.9753917004132687
0.7	0.9870967741935484	0.9622641509433962	0.9745222929936306
0.8	0.9949324324324325	0.9261006289308176	0.9592833876221498
0.9	0.9979338842975206	0.7594339622641509	0.8624999999999999

Tablica 3.7. Rezultati testiranja vlastito treniranog modela YOLOv7 W6 uz pragove pouzdanosti od 0.1 do 0.9 za metrike preciznosti, odziva i F1 mjere – označeni redovi sadrže najveći iznos F1 mjere

YOLOv7 *W6*			
Prag pouzdanosti	**Preciznost**	**Odziv**	**F1**
0.1	0.9905641749581124	0.9937106918238994	0.9921349386337138
0.2	0.9905641749581124	0.9937106918238994	0.9921349386337138
0.3	0.9905641749581124	0.9937106918238994	0.9921349386337138
0.4	0.9905641749581124	0.9937106918238994	0.9921349386337138
0.5	0.9905641749581124	0.9937106918238994	0.9921349386337138
0.6	0.9905641749581124	0.9937106918238994	0.9921349386337138
0.7	0.9905956112852664	0.9937106918238994	0.9921507064364207
0.8	0.9967897271268058	0.9764150943396226	0.9864972200158856
0.9	0.9982110912343470	0.8773584905660378	0.9338912133891213

Za iduće testiranje YOLO modela za brzinu izvođenja, testirat će se svaki model uz prag pouzdanosti pri kojem je dao najbolje rezultate F1 mjere, odnosno koji je reprezentativan za

performanse modela. Izrađena je skripta za automatsko testiranje svih 6 modela za metriku FPS. Svaki model detektira registarske tablice vozila na videozapisu trajanja 5 sekundi. Prosječni FPS se računa kao *ukupan broj obrađenih sličica / proteklo vrijeme obrade videa* (slika 3.32., slika 3.33.).

test_fps.py

```python
from algorithm.object_detector import YOLOv7
from utils.detections import draw
from tqdm import tqdm
import cv2
import time

yolov7 = YOLOv7()
model_weights = ['yolov5_small.weights',
                 'yolov5_medium.weights',
                 'yolov5_extra_large.weights',
                 'yolov7_tiny.weights',
                 'yolov7_default.weights',
                 'yolov7_w6.weights']
confidence_thresholds = [0.5, 0.6, 0.7, 0.4, 0.6, 0.7]

for i in range(len(model_weights)):
    weights = model_weights[i]
    if weights == 'yolov7_w6.weights':
        yolov7.set(img_size=1280)
    else:
        yolov7.set(img_size=640)
    yolov7.set(conf_thres=confidence_thresholds[i])

    yolov7.load(weights, classes='classes.yaml', device='cpu')

    video = cv2.VideoCapture('FPS Test.mp4')
    width  = int(video.get(cv2.CAP_PROP_FRAME_WIDTH))
    height = int(video.get(cv2.CAP_PROP_FRAME_HEIGHT))
    fps = int(video.get(cv2.CAP_PROP_FPS))
    frames_count = int(video.get(cv2.CAP_PROP_FRAME_COUNT))
    fourcc = cv2.VideoWriter_fourcc('m', 'p', '4', 'v')
    output = cv2.VideoWriter('output.mp4', fourcc, fps, (width, height))

    if video.isOpened() == False:
        print('[!] error opening the video')
```

Slika 3.32. Skripta za testiranje vlastito treniranih modela podverzija YOLOv5 i YOLOv7 algoritama uz reprezentativne pragove pouzdanosti za metriku FPS – 1. dio

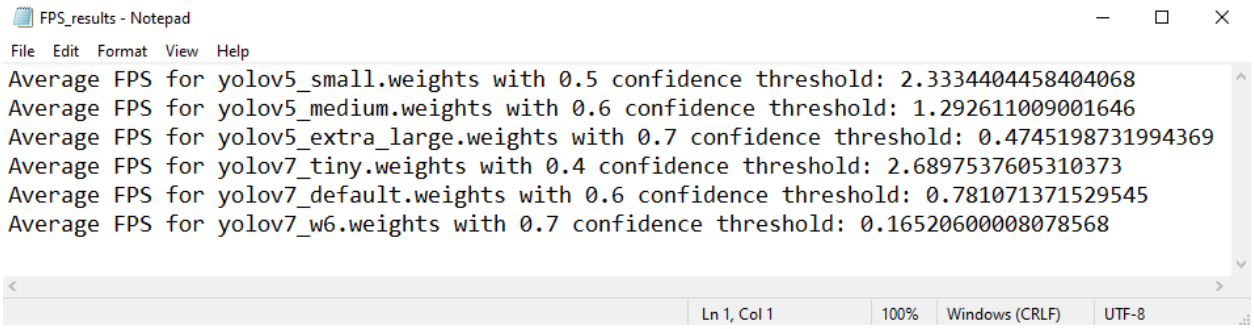
test_fps.py

```
    print('[+] detecting video...\n')
    pbar = tqdm(total=frames_count, unit=' frames', dynamic_ncols=True, position=0, leave=True)

    average_fps = 0
    counter = 0

    first_frame = True

    try:
        start = time.time()
        while video.isOpened():
            ret, frame = video.read()
            if ret == True:
                detections = yolov7.detect(frame)
                detected_frame = draw(frame, detections)
                output.write(detected_frame)
                pbar.update(1)

                counter += 1

            else:
                break
    except KeyboardInterrupt:
        pass

    end = time.time()
    pbar.close()
    video.release()
    output.release()

    total_time = end - start
    average_fps = counter/total_time
    average_fps = str(average_fps)

    conf_thres = str(yolov7.settings['conf_thres'])

    result = "Average FPS for " + weights + " with " + conf_thres + " confidence threshold: " + average_fps + "\n"
    print(result)

    text_file = open("FPS_results.txt", "a")
    text_file.write(result)
    text_file.close()

yolov7.unload()
```

Slike 3.33. Skripta za testiranje vlastito treniranih modela podverzija YOLOv5 i YOLOv7 algoritama uz reprezentativne pragove pouzdanosti za metriku FPS – 2. dio

Iz rezultata se očituje da modeli s manje parametara brže provode predviđanje (slika 3.34. i tablica 3.8.). Iznos obrađenih sličica u sekundi kod svih testiranih modela je nizak samo zato što se testiranje izvodilo na procesoru Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz. Ukoliko se predviđanje izvršava na grafičkoj kartici novije generacije, iznos obrađenih sličica u sekundi će biti puno veći i dovoljan za upotrebu u stvarnom vremenu, ali odnos između modela bi ostao jednak, stoga je i testiranje na slabijoj konfiguraciji koje je ovdje provedeno relevantno.

Primjerice, na *Youtube* kanalu TheCodingBug [48] dostupan je videozapis testiranja svih podverzija YOLOv7 algoritma za metriku FPS na grafičkoj kartici srednjeg cjenovnog ranga Nvidia GTX1060 6GB. U tom slučaju, broj obrađenih sličica u sekundi za YOLOv7 *tiny* model iznosi oko 50, za YOLOv7 *default* model iznosi oko 16, a za YOLOv7 *W6* model iznosi oko 5 [48].



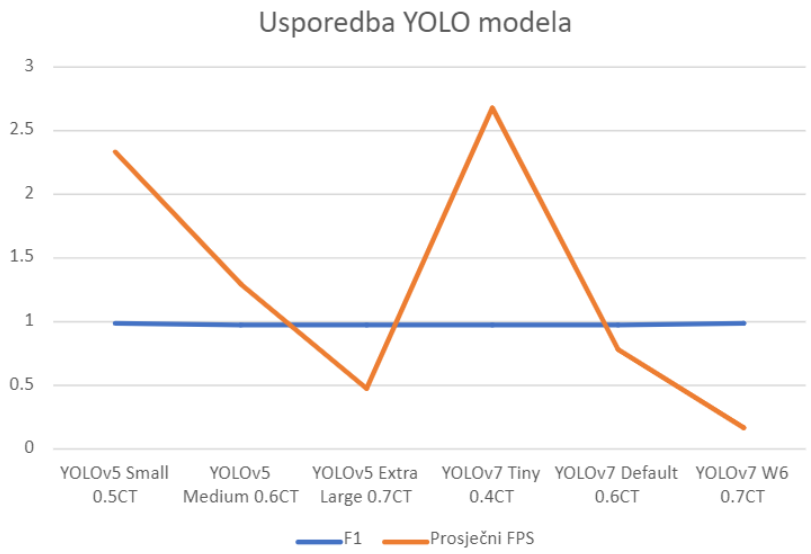
Slika 3.34. Rezultati testiranja vlastito treniranih modela podverzija YOLOv5 i YOLOv7 algoritama uz reprezentativne pragove pouzdanosti za metriku FPS

Tablica 3.8. Rezultati testiranja vlastito treniranih modela podverzija YOLOv5 i YOLOv7 algoritama uz reprezentativne pragove pouzdanosti za metriku FPS

Model	**FPS**
YOLOv5 *small* uz prag pouzdanosti 0.5	2.33344044584040680
YOLOv5 *medium* uz prag pouzdanosti 0.6	1.29261100900164600
YOLOv5 *extra large* uz prag pouzdanosti 0.7	0.47451987319943690
YOLOv7 *tiny* uz prag pouzdanosti 0.4	2.68975376053103730
YOLOv7 *default* uz prag pouzdanosti 0.6	0.78107137152954500
YOLOv7 *W6* uz prag pouzdanosti 0.7	0.16520600008078568

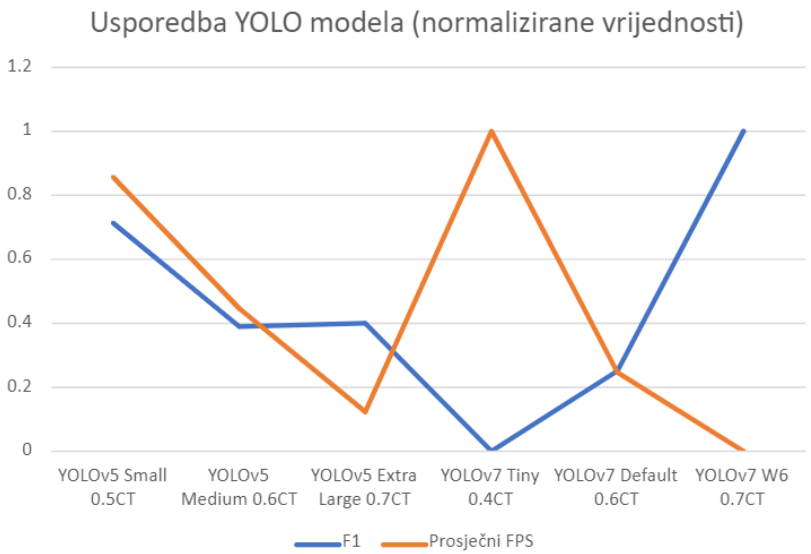
3.3.3. Usporedba YOLO modela

Za krajnju usporedbu u obzir će se uzeti F1 mjera i FPS. Usporedit će se prethodno trenirani modeli prema rezultatima F1 mjere i FPS-a. U tu svrhu iscrtava se linijski graf (slika 3.35.).



Slika 3.35. Usporedba vlastito treniranih modela podverzija YOLOv5 i YOLOv7 algoritama uz reprezentativne pragove pouzdanosti prema metrikama F1 i FPS

Iz priložene slike 3.35. nije vidljiva značajna razlika rezultata F1 mjere. Normalizacijom svih vrijednosti na raspon od 0 do 1 relativno na najveću i najmanju vrijednost rezultata F1 metrike te relativno na najveću i najmanju vrijednost rezultata prosječnog FPS-a, dobiva se čitljiviji graf od prethodnog (slika 3.36.).



Slika 3.36. Normalizirana usporedba vlastito treniranih modela podverzija YOLOv5 i YOLOv7 algoritama uz reprezentativne pragove pouzdanosti prema metrikama F1 i FPS

Za korištenje detekcije registarskih tablica vozila u stvarnom vremenu poželjni su velika brzina i visoka preciznost. Iz priložene slike 3.36. vidljivo je da model YOLOv5 *small* najbolje ostvaruje taj kompromis, stoga će se upravo on koristiti u modulu detekcije kao dio sustava za prikaz podataka o profilu vozača vozila ispred vlastitog vozila zasnovanog na detekciji i prepoznavanju registarskih tablica vozila.

4. ODABIR I PODEŠAVANJE ALGORITMA ZA PREPOZNAVANJE ZNAKOVA NA DETEKTIRANIM REGISTARSKIM TABLICAMA VOZILA

Nakon detektiranog područja registarske tablice vozila, idući zadatak za ostvarenje sustava za prikaz podataka o profilu vozača vozila ispred vlastitog vozila zasnovanog na detekciji i prepoznavanju registarkih tablica vozila, je odabir i podešavanje algoritma za prepoznavanje znakova na detektiranim registarskim tablicama vozila. S obzirom da su za potpoglavlje 2.1. ovog diplomskog rada pronađeni i predstavljeni radovi koji uz algoritme detekcije opisuju samo korištenu vrstu neuronske mreže za prepoznavanje znakova, a ne konkretan algoritam otvorenog koda, neće moći biti testirana i uspoređena ta rješenja zbog nedostatka informacija. Kao alternativa testirat će se i usporediti najmodernija rješenja otvorenog koda koja se mogu implementirati pomoću *Python* programskog jezika zbog kompatibilnosti s YOLO algoritmom za detekciju objekata, odnosno *KerasOCR* [49], *EasyOCR* [50], *Pytesseract* [51] i *PaddleOCR* [52]. Testirat će se točnost prepoznavanja znakova u odnosu na ručno zapisane znakove unutar *.txt* datoteke za svaku sliku. Nakon pronalaska najtočnijeg algoritma za ovaj slučaj upotrebe, bit će ukratko opisan način rada odabranog algoritma i provest će se daljnja testiranja kako bi se utvrdila najveća udaljenost registarske tablice vozila od kamere na kojoj ostvaruje prihvatljivu točnost.

4.1. Implementacija OCR algoritama unutar postojećeg radnog okruženja implementacije YOLO algoritma

Algoritmi koji će se testirati, *KerasOCR* (slika 4.1.), *EasyOCR* (slika 4.2.), *Pytesseract* (slika 4.3.) i *PaddleOCR* (slika 4.4.), implementirani su unutar modula *ocr.py*. Navedeni algoritmi imaju jednostavno sučelje za programiranje aplikacije (engl. *application programming interface*, API), odnosno koriste se u nekoliko linija koda.

Ostavljanje velikih slova engleske abecede i brojki od 0 do 9 te odbacivanje svih ostalih prepoznatih znakova povećava konzistentnost prepoznavanja za ovaj slučaj upotrebe jer hrvatski grb i znak '-' na registarskim tablicama vozila budu često prepoznati kao drugačiji znakovi na slikama koje sadrže različite složene uvjete poput različitih osvjetljenja, kutova snimanja i vremenskih uvjeta. Prethodno navedeni algoritmi koji će se testirati nemaju hrvatsku verziju, stoga hrvatski dijakritički znakovi 'Č', 'Ć', 'Đ', 'Š', 'Ž' bit će prepoznati kao 'C', 'C', 'D', 'S', 'Z'. Ovo ograničenje nije u dosegu ovog rada, ali moglo bi biti riješeno dodatnim treniranjem postojećih

modela prethodno navedenih algoritama ili razvitkom vlastitog algoritma za prepoznavanje znakova.

ocr.py

utils > ocr.py > ...

```python
import keras_ocr

pipeline = keras_ocr.pipeline.Pipeline()

def read(image):
  image = keras_ocr.tools.read(image)
  predictions = pipeline.recognize([image])

  for prediction in predictions[0]:
    text = prediction[0]

  text = text.upper()

  characters = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9",
                "Q", "W", "E", "R", "T", "Z", "U", "I", "O", "P",
                "A", "S", "D", "F", "G", "H", "J", "K", "L", "Y",
                "X", "C", "V", "B", "N", "M"]
  for character in text:
    if character not in characters:
      text = text.replace(character, "")

  return text
```

Slika 4.1. API KerasOCR algoritma

ocr.py

utils > ocr.py > ...

```python
import easyocr

easy_ocr = None

def read(image):
  global easy_ocr

  if easy_ocr is None:
    easy_ocr = easyocr.Reader(['en'], gpu=False)

  result = easy_ocr.readtext(image, detail = 0)

  text = ''
  text = result[0]

  characters = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9",
                "Q", "W", "E", "R", "T", "Z", "U", "I", "O", "P",
                "A", "S", "D", "F", "G", "H", "J", "K", "L", "Y",
                "X", "C", "V", "B", "N", "M"]
  for character in text:
    if character not in characters:
      text = text.replace(character, "")

  return text
```

Slika 4.2. API EasyOCR algoritma

ocr.py

utils > ocr.py > ...

```python
import pytesseract
import cv2
from PIL import Image

pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'

def read(image):
  image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
  pil_image = Image.fromarray(image)

  text = pytesseract.image_to_string(pil_image)

  characters = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9",
                "Q", "W", "E", "R", "T", "Z", "U", "I", "O", "P",
                "A", "S", "D", "F", "G", "H", "J", "K", "L", "Y",
                "X", "C", "V", "B", "N", "M"]
  for character in text:
    if character not in characters:
      text = text.replace(character, "")

  return text
```

Slika 4.3. API Pytesseract algoritma

ocr.py

utils > ocr.py > ...

```python
from paddleocr import PaddleOCR

paddle_ocr = None

def read(image):
  global paddle_ocr

  if paddle_ocr is None:
    paddle_ocr = PaddleOCR(use_angle_cls=True, lang='ch', show_log=False)

  result = paddle_ocr.ocr(image, cls=True)

  text = ''
  for idx in range(len(result)):
    res = result[idx]
    for line in res:
      text += f'{line[1][0]}'

  characters = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9",
                "Q", "W", "E", "R", "T", "Z", "U", "I", "O", "P",
                "A", "S", "D", "F", "G", "H", "J", "K", "L", "Y",
                "X", "C", "V", "B", "N", "M"]
  for character in text:
    if character not in characters:
      text = text.replace(character, "")

  return text
```

Slika 4.4. API PaddleOCR algoritma

Modul *object_detector.py* koji sadrži YOLO algoritam uključuje modul *ocr.py*. Detekcije su spremljene u *Python* riječniku (engl. *dictionary*), te se za svaku detekciju poziva funkcija *read()* modula *ocr.py* kojoj se predaje detektirano područje registarske tablice vozila kao posebna slika. Prepoznati znakovi se spremaju kao atribut odgovarajuće detekcije riječnika (slika 4.5.).

```
97          detections = Detections(raw_detection, self.classes, tracking=track).to_dict()
98          if len(self.settings['ocr_classes']) > 0:
99              for detection in detections:
100                 if detection['class'] in self.settings['ocr_classes']:
101                     cropped_box = crop(im0, detection)
102                     text = ''
103                     try:
104                         text = ocr.read(cropped_box)
105                     except:
106                         pass
107                     detection['text'] = text
108                     if(detection['width'] < 120): #test_OCR.py only
109                         detection['text'] = "IGNORED" #test_OCR.py only
110
111         return detections
```

Slika 4.5. Implementacija modula ocr.py unutar modula object_detector.py

Linija 108 koda koji je vidljiv na slici 4.5. će služiti kao ograničenje za širinu detektiranog graničnog okvira registarske tablice vozila na kojoj će se prihvatiti prepoznati znakovi, uz pretpostavku da je veći detektirani granični okvir registarske tablice vozila bliže vozaču te sadrži čitljivije znakove i obrnuto. Primjer koji je naveden na slici 4.5. označava da će se za detektirane granične okvire registarskih tablica vozila manje od 120 piksela prepoznati znakovi zamijeniti nizom znakova 'IGNORED'

4.2. Testiranje i usporedba OCR algoritama

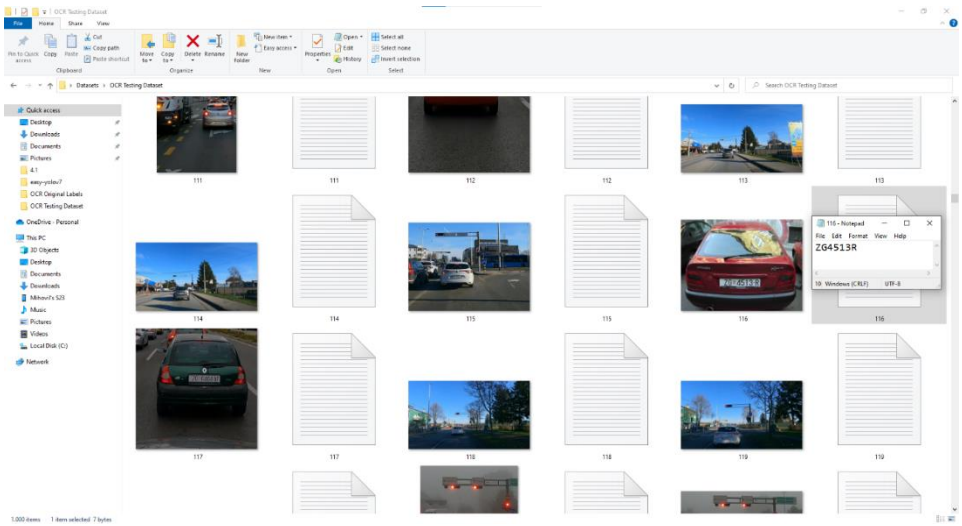
Testiranje će se prvo provesti za sve odabrane algoritme, *KerasOCR*, *EasyOCR*, *Pytesseract* i *PaddleOCR*, za prepoznavanje znakova samo na detektiranim graničnim okvirima registarskih tablica vozila širine veće od i uključujući 120 piksela. To ograničenje je odabrano s ciljem da se pri prvom testiranju svih odabranih algoritama za prepoznavanje znakova koriste registarske tablice vozila koje su što veće i čitljivije uz broj testnih slika veći od 100, odnosno broj registarskih tablica vozila, jer u ovom skupu slika svaka slika sadrži jednu registarsku tablicu vozila. Nakon što se utvrdi koji je algoritam najtočniji za ovaj slučaj upotrebe, taj će se testirati na još nekoliko širina detektiranih graničnih okvira registarskih tablica vozila kako bi se pronašla

najveća udaljenost na kojoj ostvaruje prihvatljivu točnost. Za skup slika na kojem će se testirati bit će ručno zapisani znakovi registarke tablice vozila unutar *.txt* datoteke za svaku sliku.

4.2.1. Prikupljanje i označavanje slika za testiranje OCR algoritama

Skup slika za testiranje prepoznavanja znakova prikupljen je s istog *Youtube* kanala kao i skupovi slika za treniranje i testiranje YOLO modela [44]. Za testiranje prepoznavanja znakova, prikupljeno je 500 slika koje prikazuju situacije kada se ispred vozača nalazi vozilo na različitim udaljenostima, uz uvjet da registarske tablice vozila moraju biti čitljive ljudskim okom. Ovih 500 slika nisu među onih 1000 korištenih za treniranje YOLO modela i 500 korištenih za testiranje YOLO modela. Snimke zaslona su uzete u tim situacijama, ali su ograničene na rezoluciju monitora 1920x1080 piksela, bez obzira na izvornu rezoluciju videozapisa. Slike su izrezane tako da sadrže samo registarske tablice vozila ispred vozača, što znači da svaka slika sadrži jednu registarsku tablicu vozila.

Za svaku sliku kreirana je istoimena *.txt* datoteka koja sadrži ručno upisane znakove registarske tablice vozila koja se nalazi na toj slici. Dakle, skup podataka za vlastito testiranje OCR algoritama se sastoji od 500 slika od kojih svaka sadrži jednu registarsku tablicu vozila i 500 *.txt* datoteka od kojih svaka sadrži ručno upisane znakove registarske tablice vozila pripadajuće slike (slika 4.6.). 478 graničnih okvira označavaju hrvatske registarske tablice vozila, a 22 graničnih okvira označavaju strane registarske tablice vozila. Skup slika i oznaka koji se koristi za testiranje OCR algoritama može se pronaći na DVD-u priloženom uz ovaj rad u mapi Prilog P.4.1.



Slika 4.6. Skup podataka za testiranje OCR algoritama

4.2.2. Vlastita skripta za testiranje OCR algoritama

Na slikama 4.7. i 4.8. vidljiva su prva dva dijela skripte za testiranje *test_OCR.py*.

test_OCR.py

```python
from algorithm.object_detector import YOLOv7
import cv2
import os
from test_OCR_util import evaluate

yolov7 = YOLOv7()
yolov7.load('yolov5_small.weights', classes='classes.yaml', device='cpu')
yolov7.set(ocr_classes=['license plate'])

max_sample_width = len('Sample') + 2
max_actual_width = len('Actual License Plate') + 2
max_predicted_width = len('Predicted License Plate') + 2
max_passed_width = len('Passed') + 2

text_result = open("OCR_results.txt", "w")
string = "{1:<{0}s}  {3:<{2}s}  {5:<{4}}  {7:<{6}}".format(max_sample_width, 'Sample',
                                                          max_actual_width, 'Actual License Plate',
                                                          max_predicted_width, 'Predicted License Plate',
                                                          max_passed_width, 'Accuracy')
text_result.write(string)
text_result.write("\n")

string = "{1:<{0}s}  {3:<{2}s}  {5:<{4}}  {7:<{6}}".format(max_sample_width, '------',
                                                          max_actual_width, '--------------------',
                                                          max_predicted_width, '------------------------',
                                                          max_passed_width, '--------')
text_result.write(string)
text_result.write("\n")

text_result.close()

dir_path = r"C:\\Users\Mihovil\\Desktop\\Datasets\\OCR Testing Dataset\\"

totalFiles = 0

for path in os.listdir(dir_path):
    if os.path.isfile(os.path.join(dir_path, path)):
        totalFiles += 1

totalSamples = int(totalFiles/2)
totalCorrect = 0
totalOneWrong = 0
totalTwoWrong = 0
totalThreeWrong = 0
totalIncorrect = 0
totalIgnored = 0
totalUndetected = 0
totalEmpty = 0
```

Slika 4.7. Skripta za testiranje točnosti prepoznavanja znakova – 1. dio

test_OCR.py

```python
for sample in range(totalSamples):
    image_path = dir_path + str(sample) + ".png"
    print(image_path)
    text_path = dir_path + str(sample) + ".txt"
    image = cv2.imread(image_path)
    detections = yolov7.detect(image)

    if(detections):
        predicted = detections[0]['text']
    else:
        predicted = "UNDETECTED"

    text_file = open(text_path, "r")
    actual = text_file.read()
    text_file.close()

    if(predicted == actual):
        accuracy = "100%"
        totalCorrect += 1
    elif(predicted == "IGNORED"):
        accuracy = "/"
        totalIgnored += 1
    elif(predicted == "UNDETECTED"):
        accuracy = "/"
        totalUndetected += 1
    else:
        incorrect_characters = evaluate(actual, predicted)
        if(len(actual) >= len(predicted)):
            accuracy_percent = ((len(actual) - incorrect_characters) / len(actual)) * 100
        elif(len(actual) < len(predicted)):
            accuracy_percent = ((len(predicted) - incorrect_characters) / len(predicted)) * 100
        accuracy_percent = round(accuracy_percent, 2)
        accuracy = str(accuracy_percent) + "%"

        if(incorrect_characters == 1):
            totalOneWrong += 1
        elif(incorrect_characters == 2):
            totalTwoWrong += 1
        elif(incorrect_characters == 3):
            totalThreeWrong += 1

        totalIncorrect += 1
    if(predicted == ""):
        totalEmpty += 1
```

Slika 4.8. Skripta za testiranje točnosti prepoznavanja znakova – 2. dio

Učitava se željeni YOLO model za detekciju, te se navode klase čiji će se znakovi prepoznavati, u ovom slučaju *license plate*. Otvara se *.txt* datoteka i ispisuju četiri stupca u koje će se za svako prepoznavanje unositi redni broj slike na kojoj se detektira i prepoznaje registarska tablica vozila, referentni (engl. *actual*) *string* (ručno upisani znakovi), prepoznati (engl. *predicted*)

string i točnost. Podaci unutar skupa imenovani su brojkama od 0 do 499, stoga se pomoću *for* petlje pronalazi pripadajući par *.png* datoteke i *.txt* datoteke.

Slika se obrađuje i ukoliko nema detekcije, rezultat se evidentira kao 'UNDETECTED'. Otvara se odgovarajuća *.txt* datoteka i iščitava referentni *string*. Ukoliko je prepoznati *string* potpuno jednak referentnom *string*-u, u varijablu *accuracy* (točnost) se zapisuje '100%', a varijabli *totalCorrect* (točna prepoznavanja) se dodaje 1. Ukoliko je modul *object_detector.py* pronašao detektirani granični okvir registarske tablice vozila manje širine od navedene, rezultat se evidentira kao 'IGNORED', u varijablu *accuracy* se zapisuje '/', a varijabli *totalIgnored* (zanemarene registarske tablice vozila) se dodaje 1. Za rezultat 'UNDETECTED' se također u varijablu *accuracy* zapisuje '/', a varijabli *totalUndetected* (nedetektirane registarske tablice vozila) se dodaje 1. Ukoliko nijedan znak nije prepoznat, a detekcija postoji, varijabli *totalEmpty* (registarske tablice vozila za koje je prepoznat prazan *string*) se dodaje 1. U svakom drugom slučaju evaluacija se vrši nad referentnim *string*-om i prepoznatim *string*-om pomoću modula *test_OCR_util.py* (slika 4.9.).

test_OCR_util.py

test_OCR_util.py > isSubstring

```
1   def evaluate(actual, predicted):
2       correct = 0
3
4       for character in actual:
5           predicted_index = isSubstring(character, predicted)
6           if(predicted_index >= 0):
7               correct += 1
8               predicted = predicted[:predicted_index] + "/" + predicted[predicted_index + 1:]
9       if(len(predicted) > len(actual)):
10          return len(predicted) - correct
11      return len(actual) - correct
12
13  def isSubstring(s1, s2):
14      if s1 in s2:
15          return s2.index(s1)
16      return -1
```

Slika 4.9. Modul za evaluaciju nad referentnim (engl. actual) string-om i prepoznatim (engl. predicted) string-om

Za svaki znak referentnog *string*-a, provjerava se nalazi li se unutar prepoznatog *string*-a i ukoliko se nalazi dohvaća se njegova pozicija (engl. *index*), varijabla *correct* (točni znakovi) se uvećava za 1, te se taj znak unutar prepoznatog *string*-a zamjenjuje znakom '/', kako se ne bi u

idućem prolazu petlje ponovno pronašao njegov indeks ukoliko postoji još istih znakova (slika 4.10.).

```
C:\\Users\Mihovil\\Desktop\\Datasets\\OCR Testing Dataset\\95.png
/GI665HF
//I665HF
///665HF
////65HF
/////5HF
//////HF
///////F
```

Slika 4.10. Primjer ispisa u terminalu tijekom evaluacije nad referentnim string-om (ZGI665HP) i prepoznatim string-om (ZGI665HF)

Ako su referentni *string* i prepoznati *string* jednake duljine ili je duljina referentnog *string*-a veća od duljine prepoznatog *string*-a, vraća se razlika duljine pravog *string*-a i varijable *correct*, što predstavlja broj netočnih znakova. Ukoliko je duljina prepoznatog *string*-a veća od duljine referentnog *string*-a, broj netočnih znakova se predstavlja kao razlika duljine prepoznatog *string*-a i varijable *correct*. Primjerice, ukoliko je duljina prepoznatog *string*-a 6, a duljina referentnog *string*-a 5, te su svi znakovi osim viška točno prepoznati, prethodno navedenom razlikom ovo će se računati kao jedan pogrešan znak.

Nakon evaluacije nad referentnim *string*-om i prepoznatim *string*-om, broj netočnih znakova se vraća skripti *test_OCR.py* koja računa točnost u postotcima. Također se izračunava ukupan broj prepoznatih *string*-ova s jednim, dvama i trima netočnima znakovima (*totalOneWrong*, *totalTwoWrong*, *totalThreeWrong*) te njihove pripadajuće točnosti u postotcima (*oneWrongAccuracy*, *twoWrongAccuracy*, *threeWrongAccuracy*). Uz to računa se ukupan broj netočno prepoznatih *string*-ova (*totalIncorrect*) i ukupan broj slika koje nisu ignorirane zbog toga što sadrže detektirani granični okvir registarske tablice vozila čija je širina u pikselima u skladu s ograničenjem (*totalSamples*). Na kraju se sve izračunate vrijednosti ispisuju (slika 4.11.).

Na taj način testirani su algoritmi *KerasOCR* (slika 4.12.), *EasyOCR* (slika 4.13.), *Pytesseract* (slika 4.14.) i *PaddleOCR* (slika 4.15.) uz detektirane granične okvire registarskih tablica vozila širine veće od i uključujući 120 piksela. U tablici 3.8. dani su rezultati za sve prethodno navedene algoritme zajedno radi bolje preglednosti.

test_OCR.py

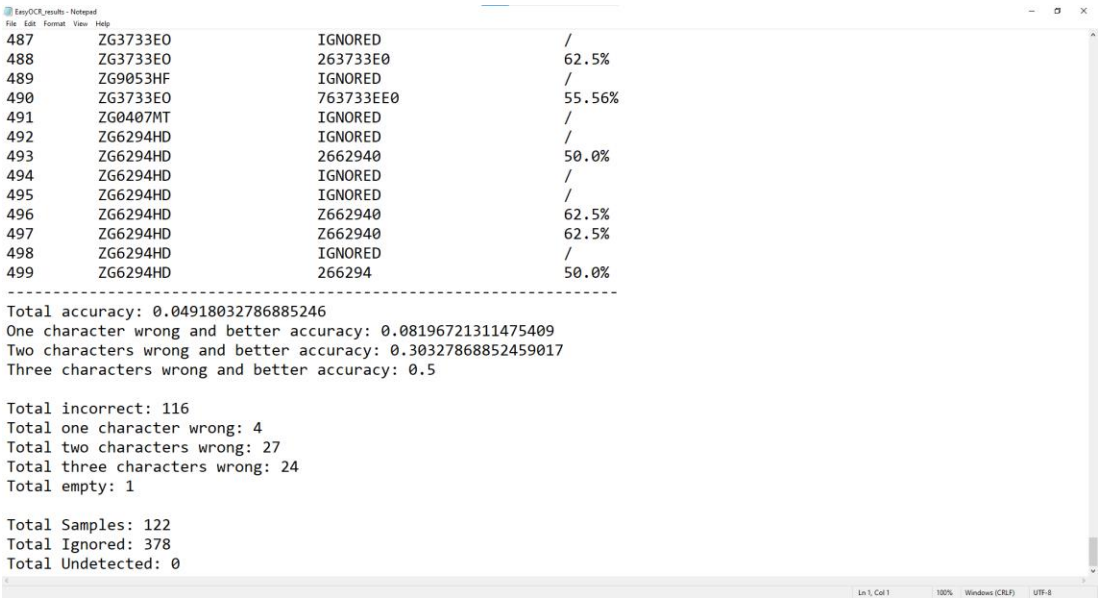
```
    text_result = open("OCR_results.txt", "a")
    string = "{1:<{0}s}  {3:<{2}s}  {5:<{4}}  {7:<{6}}".format(max_sample_width, str(sample),
                                                              max_actual_width, actual,
                                                              max_predicted_width, predicted,
                                                              max_passed_width, accuracy)
    text_result.write(string)
    text_result.write("\n")
    text_result.close()

totalAccuracy = totalCorrect / (totalCorrect + totalIncorrect)
oneWrongAccuracy = (totalCorrect + totalOneWrong) / (totalCorrect + totalIncorrect)
twoWrongAccuracy = (totalCorrect + totalOneWrong + totalTwoWrong) / (totalCorrect + totalIncorrect)
threeWrongAccuracy = (totalCorrect + totalOneWrong + totalTwoWrong + totalThreeWrong) / (totalCorrect + totalIncorrect)

print("Total accuracy: " + str(totalAccuracy))

text_result = open("OCR_results.txt", "a")
text_result.write("----------------------------------------------------------------------\n")
text_result.write("Total accuracy: " + str(totalAccuracy) + "\n")
text_result.write("One character wrong and better accuracy: " + str(oneWrongAccuracy) + "\n")
text_result.write("Two characters wrong and better accuracy: " + str(twoWrongAccuracy) + "\n")
text_result.write("Three characters wrong and better accuracy: " + str(threeWrongAccuracy) + "\n\n")

text_result.write("Total incorrect: " + str(totalIncorrect) + "\n")
text_result.write("Total one character wrong: " + str(totalOneWrong) + "\n")
text_result.write("Total two characters wrong: " + str(totalTwoWrong) + "\n")
text_result.write("Total three characters wrong: " + str(totalThreeWrong) + "\n")
text_result.write("Total empty: " + str(totalEmpty) + "\n\n")

text_result.write("Total Samples: " + str(totalCorrect + totalIncorrect) + "\n")
text_result.write("Total Ignored: " + str(totalIgnored) + "\n")
text_result.write("Total Undetected: " + str(totalUndetected))
text_result.close()
```

Slika 4.11. Skripta za testiranje točnosti prepoznavanja znakova – 3. dio

KerasOCR_results - Notepad

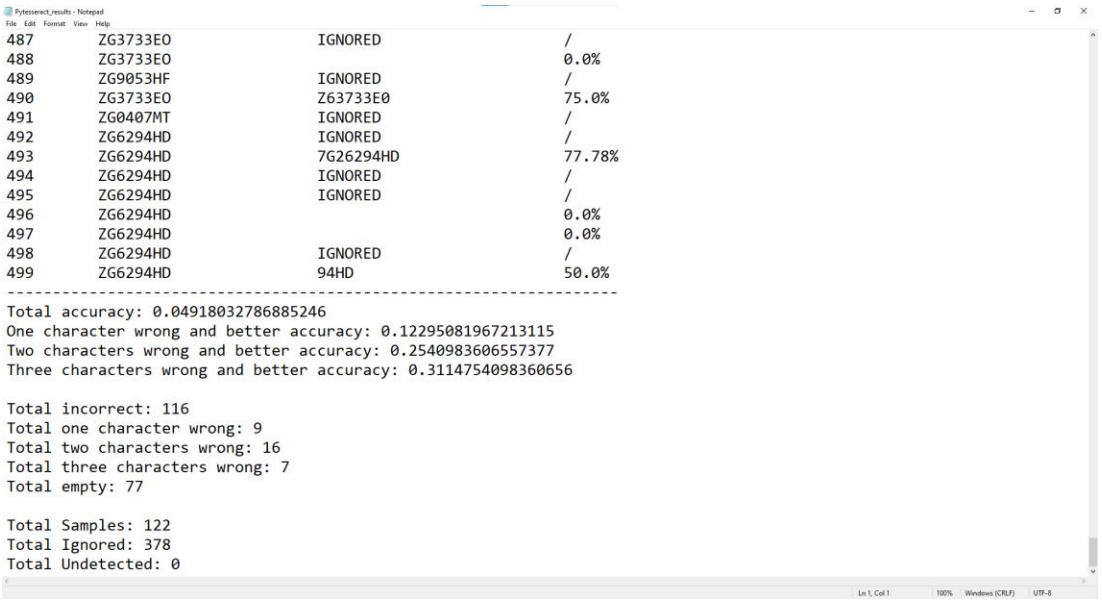
```
487       ZG3733EO            IGNORED             /
488       ZG3733EO            ZG                  25.0%
489       ZG9053HF            IGNORED             /
490       ZG3733EO            ZG                  25.0%
491       ZG0407MT            IGNORED             /
492       ZG6294HD            IGNORED             /
493       ZG6294HD            6Z9AHD              62.5%
494       ZG6294HD            IGNORED             /
495       ZG6294HD            IGNORED             /
496       ZG6294HD            CR                  0.0%
497       ZG6294HD            HR                  12.5%
498       ZG6294HD            IGNORED             /
499       ZG6294HD            ZG                  25.0%
----------------------------------------------------------------------
Total accuracy: 0.00819672131147541
One character wrong and better accuracy: 0.01639344262295082
Two characters wrong and better accuracy: 0.1885245901639344
Three characters wrong and better accuracy: 0.28688524590163933

Total incorrect: 121
Total one character wrong: 1
Total two characters wrong: 21
Total three characters wrong: 12
Total empty: 0

Total Samples: 122
Total Ignored: 378
Total Undetected: 0
```

Slika 4.12. Rezultati testiranja algoritma KerasOCR pri širini detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući 120 piksela



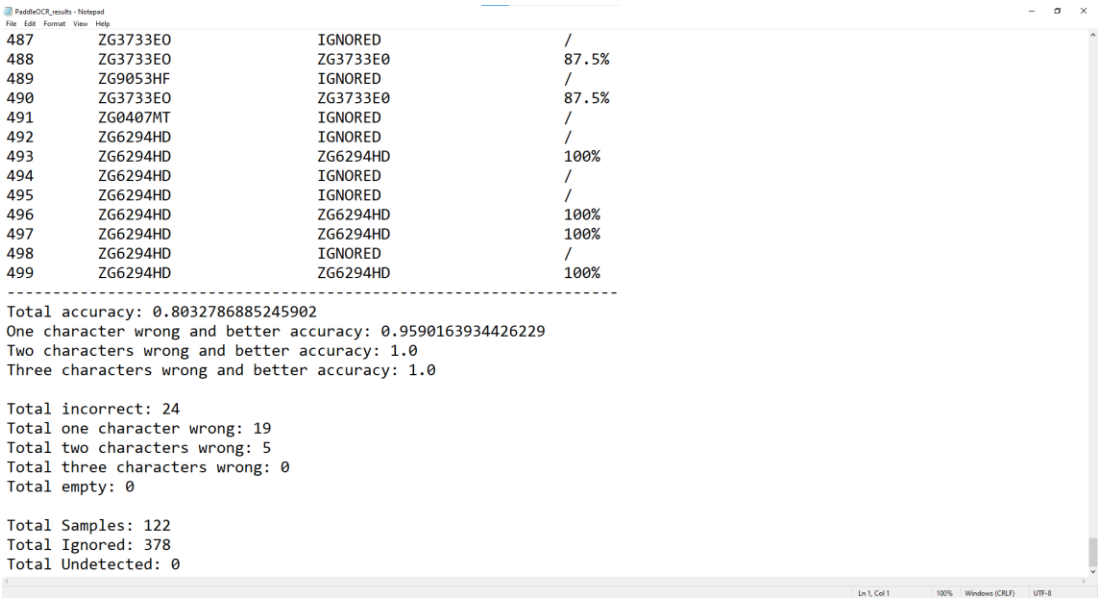
```
EasyOCR_results - Notepad
File  Edit  Format  View  Help
487        ZG3733EO                IGNORED                 /
488        ZG3733EO                263733E0                62.5%
489        ZG9053HF                IGNORED                 /
490        ZG3733EO                763733EE0               55.56%
491        ZG0407MT                IGNORED                 /
492        ZG6294HD                IGNORED                 /
493        ZG6294HD                2662940                 50.0%
494        ZG6294HD                IGNORED                 /
495        ZG6294HD                IGNORED                 /
496        ZG6294HD                Z662940                 62.5%
497        ZG6294HD                Z662940                 62.5%
498        ZG6294HD                IGNORED                 /
499        ZG6294HD                266294                  50.0%
--------------------------------------------------------------------------
Total accuracy: 0.04918032786885246
One character wrong and better accuracy: 0.08196721311475409
Two characters wrong and better accuracy: 0.30327868852459017
Three characters wrong and better accuracy: 0.5

Total incorrect: 116
Total one character wrong: 4
Total two characters wrong: 27
Total three characters wrong: 24
Total empty: 1

Total Samples: 122
Total Ignored: 378
Total Undetected: 0
```

Slika 4.13. Rezultati testiranja algoritma EasyOCR pri širini detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući 120 piksela



```
Pytesseract_results - Notepad
File  Edit  Format  View  Help
487        ZG3733EO                IGNORED                 /
488        ZG3733EO                                        0.0%
489        ZG9053HF                IGNORED                 /
490        ZG3733EO                Z63733E0                75.0%
491        ZG0407MT                IGNORED                 /
492        ZG6294HD                IGNORED                 /
493        ZG6294HD                7G26294HD               77.78%
494        ZG6294HD                IGNORED                 /
495        ZG6294HD                IGNORED                 /
496        ZG6294HD                                        0.0%
497        ZG6294HD                                        0.0%
498        ZG6294HD                IGNORED                 /
499        ZG6294HD                94HD                    50.0%
--------------------------------------------------------------------------
Total accuracy: 0.04918032786885246
One character wrong and better accuracy: 0.12295081967213115
Two characters wrong and better accuracy: 0.2540983606557377
Three characters wrong and better accuracy: 0.3114754098360656

Total incorrect: 116
Total one character wrong: 9
Total two characters wrong: 16
Total three characters wrong: 7
Total empty: 77

Total Samples: 122
Total Ignored: 378
Total Undetected: 0
```

Slika 4.14. Rezultati testiranja algoritma Pytesseract pri širini detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući 120 piksela



Slika 4.15. Rezultati testiranja algoritma PaddleOCR pri širini detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući 120 piksela

Tablica 4.1. Rezultati testiranja algoritama KerasOCR, EasyOCR, Pytesseract i PaddleOCR za točnost prepoznavanja registarskih tablica vozila pri širini detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući 120 piksela

	KerasOCR	**EasyOCR**	**Pytesseract**	**PaddleOCR**
Točnost	0.81%	4.92%	4.92%	80.33%
Točnost za jedan netočan znak i bolje	1.64%	8.2%	12.3%	95.9%
Točnost za dva netočna znaka i bolje	18.85%	30.33%	25.41%	100%
Točnost za tri netočna znaka i bolje	28.69%	50%	31.15%	100%
Netočno prepoznate registarske tablice vozila	121	116	116	24
Prepoznate registarske tablice vozila s jednim netočnim znakom	1	4	9	19
Prepoznate registarske tablice vozila s dva netočna znaka	21	27	16	5
Prepoznate registarske tablice vozila s tri netočna znaka	12	24	7	0

Registarske tablice vozila za koje je prepoznat prazan *string*	0	1	77	0
Registarske tablice vozila na kojima je provedeno testiranje	122	122	122	122
Zanemarene registarske tablice vozila	378	378	378	378
Nedetektirane registarske tablice vozila	0	0	0	0

Izračunate metrike prema kojima se očituju rezultati testiranja su *total accuracy*, *one character wrong and better accuracy*, *two characters wrong and better accuracy*, *three characters wrong and better accuracy*, *total incorrect*, *total one character wrong*, *total two characters wrong*, *total three characters wrong*, *total samples*, *total ignored* i *total undetected*. *Total accuracy* predstavlja ukupan iznos u postotcima potpuno točno prepoznatih oznaka registarskih tablica vozila. *One character wrong and better accuracy* predstavlja ukupan iznos u postotcima potpuno točno prepoznatih oznaka registarskih tablica vozila i prepoznatih oznaka registarskih tablica vozila s jednim netočnim znakom. *Two characters wrong and better accuracy* predstavlja ukupan iznos u postotcima potpuno točno prepoznatih oznaka registarskih tablica vozila i prepoznatih oznaka registarskih tablica vozila s jednim i dva netočna znaka. *Three characters wrong and better accuracy* predstavlja ukupan iznos u postotcima potpuno točno prepoznatih oznaka registarskih tablica vozila i prepoznatih oznaka registarskih tablica vozila s jednim, dva i tri netočna znaka. *Total incorrect* predstavlja ukupan broj netočno prepoznatih oznaka registarskih tablica vozila, odnosno ukoliko prepoznata oznaka sadrži jedan ili više netočno prepoznatih znakova, računa se kao netočno prepoznata. *Total one character wrong* predstavlja iznos prepoznatih oznaka registarskih tablica vozila s jednim netočnim znakom. *Total two characters wrong* predstavlja iznos prepoznatih oznaka registarskih tablica vozila s dva netočno prepoznata znaka. *Total three characters wrong* predstavlja iznos prepoznatih oznaka registarskih tablica vozila s tri netočna znaka. *Total empty* predstavlja iznos oznaka registarskih tablica oznaka za koje algoritam prepoznavanja nije prepoznao niti jedan znak. *Total samples* predstavlja iznos slika na kojima je provedeno testiranje, odnosno iznos registarskih tablica vozila jer u ovom skupu slika svaka slika sadrži jednu registarsku tablicu vozila. *Total ignored* predstavlja iznos slika na kojima nije provedeno testiranje jer detektirani granični okviri registarskih tablica vozila koje se nalaze na tim slikama nisu zadovoljili uvjet određene širine. *Total undetected* predstavlja iznos slika na kojima

algoritam za detekciju nije detektirao granični okvir registarske tablice vozila pa stoga nije mogao biti proveden postupak prepoznavanja.

Iz priloženih rezultata vidljivo je da *PaddleOCR* ima daleko najviše točnih prepoznavanja u odnosu na ostala tri testirana algoritma, pa će se on koristiti u konačnom sustavu u sklopu ovog diplomskog rada. Ovi rezultati ne označavaju ostala tri algoritma generalno lošim, već samo za ovaj slučaj upotrebe. Iz navedenih razloga, u nastavku je dano više informacija o *PaddleOCR*.

PaddleOCR je algoritam za prepoznavanje znakova razvijen pomoću otvorenog okvira za duboko učenje *PaddlePaddle*. Glavna svrha algoritma je automatsko prepoznavanje znakova u slikama i konverzija teksta u tekstualni format. Algoritam koristi duboke neuronske mreže za obavljanje OCR zadataka. *PaddleOCR* koristi kombinaciju različitih modela, kao što su CNN za detekciju regija s tekstom na slici, a zatim rekurentne neuronske mreže (RNN) ili transformatorske mreže za prepoznavanje znakova unutar svake detektirane regije. *PaddleOCR* nudi nekoliko prednosti u odnosu na druge OCR algoritme. Prvo, zbog upotrebe dubokih neuronskih mreža, može postići impresivne rezultate u prepoznavanju teksta na različitim vrstama slika. Također, algoritam je razvijen na *PaddlePaddle* platformi koja nudi efikasnost i brzinu za veliku količinu podataka. *PaddleOCR* također podržava nekoliko jezika, što ga čini prilagodljivim za različite lokalizacije i jezičke zahtjeve [52].

CNN su posebno dizajnirane za obradu podataka koji imaju prostornu strukturu, kao što su slike i videozapisi. Ova vrsta mreža postigla je izvanredne rezultate u zadacima prepoznavanja obrazaca, klasifikaciji slika i detekciji objekata [53]. RNN su posebno dizajnirane za rad sa sekvencijalnim podacima, kao što su nizovi riječi u rečenici, vremenske serije i govorni signali. Glavna karakteristika RNN-a je prisutnost povratne petlje koja omogućuje prenošenje informacija između trenutaka u vremenu unutar sekvence [53]. Transformatorske mreže su relativno novi tip neuronskih mreža koji su postigli značajne uspjehe u obradi prirodnog jezika (engl. *neuro-linguistic programming*) i drugim sekvencijalnim zadacima. Umjesto korištenja rekurentnih slojeva, transformatorske mreže se oslanjaju na pažnju (engl. *self-attention*) kako bi naučile međuzavisnosti između različitih elemenata u sekvenci [53].

4.2.3. Dodatno testiranje *PaddleOCR* algoritma

Svrha dodatnog testiranja *PaddleOCR* algoritma je utvrđivanje maksimalne udaljenosti registarske tablice vozila na kojoj algoritam ostvaruje točnost veću od 75%, odnosno za određenu udaljenost testiranjem se mora utvrditi da su barem 75% prepoznatih oznaka registarskih tablica vozila potpuno točne. Testirat će se na širinama detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući 120, 100, 80 i 60 piksela. Na slikama 4.16., 4.17., 4.18., 4.19. prikazane su redom navedene širine detektiranih graničnih okvira registarskih tablica vozila. Na slikama su širine točno određene na navedenim veličinama tako da se odredila granica širine detektiranog graničnog okvira za svaku veličinu u modulu *object_detector.py*, provela detekcija za cijeli video, te u situaciji gdje se vozač s veće udaljenosti približava vozilu ispred, zaustavio video na prvoj sličici gdje se pojavila detekcija. U ovom primjeru rezolucije videa 1920x1080 piksela, jedino pri detektiranom graničnom okviru registarske tablice vozila širine 60 piksela nisu prepoznati znakovi, dok se pri širinama 120 piksela, 100 piksela i 80 piksela svi znakovi prepoznaju točno. Korišteni videozapis rezolucije 1920x1080 piskela se može pronaći na DVD-u priloženom uz ovaj rad u mapi Prilog P.4.2.

Procjene udaljenosti registarske tablice vozila od kamere za širine detektiranih graničnih okvira registarskih tablica vozila 120 piksela, 100 piksela, 80 piksela i 60 piksela određene su pomoću formule navedene u (4-1). Ova formula zahtjeva podatke stvarne širine objekta (registarske tablice vozila) u metrima, širine slike u pikselima, širine objekta (detektiranog graničnog okvira registarske tablice vozila) na slici u pikselima i horizontalnog vidnog polja.

$$Procijenjena\ udaljenost\ [m] = \frac{Širina\ objekta\ [m] * Širina\ slike\ [piksel]}{2 * Širina\ objekta\ [piksel] * \tan\frac{Horizontalno\ vidno\ polje\ [°]}{2}}, \qquad (4\text{-}1)$$

gdje je:

- Procijenjena udaljenost [m] – procijenjena udaljenost objekta (registarske tablice vozila) od kamere u metrima
- Širina objekta [m] – stvarna širina objekta (registarske tablice vozila) u metrima
- Širina slike [piksel] – širina slike u pikselima
- Širina objekta [piksel] – širina objekta (detektiranog graničnog okvira registarske tablice vozila) na slici u pikselima

- Horizontalno vidno polje [°] – horizontalno vidno polje kamere u stupnjevima, odnosno horizontalni kut snimanja kamere u stupnjevima

Tangens polovine kuta omogućava izračunavanje udaljenosti objekta od kamere, jer pola kuta horizontalnog vidnog polja kamere tvori pravokutni trokut sa udaljenošću objekta od kamere koji je centriran u kadru kamere. S obzirom da detektirani granični okviri registarskih tablica vozila na slikama koje su odabrane za primjer različitih širina detektiranih graničnih okvira registarskih tablica vozila u pikselima (slike 4.16., 4.17., 4.18., 4.19) imaju odmak od središta kadra kamere, rezultati dobiveni korištenjem ove formule smatrat će se procjenom. Dobivena kutna veličina izračunata pomoću umnoška širine objekta u pikselima i tangensa polovine kuta horizontalnog vidnog polja uključuje kut koji pokriva objekt s obje strane središta kadra kamere. Stoga je potrebno cijeli nazivnik razlomka pomnožiti s 2 kako bi se pravilno uzela u obzir cijela širina objekta na slici. Ako se ne bi koristio faktor 2, izračun bi vrijedio samo za polovinu širine objekta, što bi rezultiralo pogreškom u procjeni udaljenosti.

Stvarna širina registarske tablice automobila u hrvatskoj iznosi 0.52 metra [54]. Širina slika koje će se koristiti pri izračunu iznosi 1920 piksela. Širine detektiranih graničnih okvira registarskih tablica vozila koje će se koristiti pri izračunu iznose 120 piksela, 100 piksela, 80 piksela i 60 piksela.

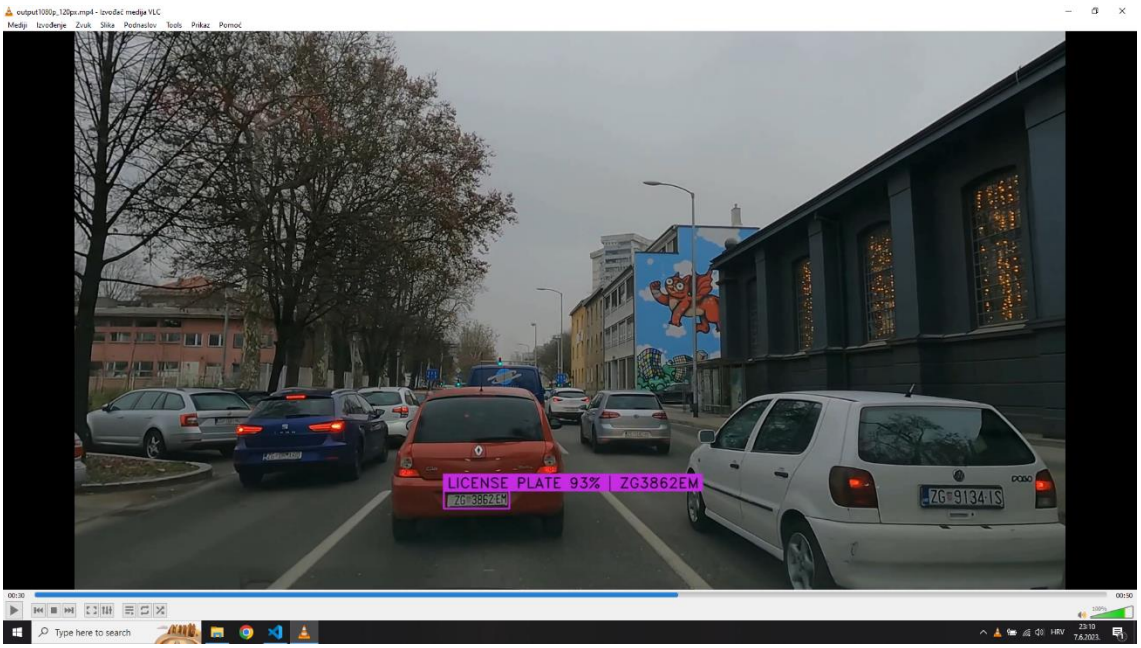
Korištena kamera GoPro Hero 9 pri snimanju videozapisa iz kojeg su prikupljene slike koje će se koristiti pri izračunu je bila postavljena na opciju *linear* postavke digitalnog objektiva (engl. *digital lens*) i na opciju *boost* postavke razine stabilizacije (engl. *hypersmooth level*). Na službenoj stranici GoPro Hero 9 navedena su pripadajuća horizontalna vidna polja za svaku kombinaciju opcija postavki digitalnog objektiva i razine stabilizacije [55]. Stoga će se pri izračunu koristiti horizontalno vidno polje 75 stupnjeva.

U tablici 4.2. su prikazane vrijednosti procijenjene udaljenosti registarske tablice vozila od kamere u metrima dobivene izračunom pri horiznotalnom vidnom polju 75 stupnjeva za širine detektiranog graničnog okvira registarske tablice vozila 120 piksela, 100 piksela, 80 piksela i 60 piksela.

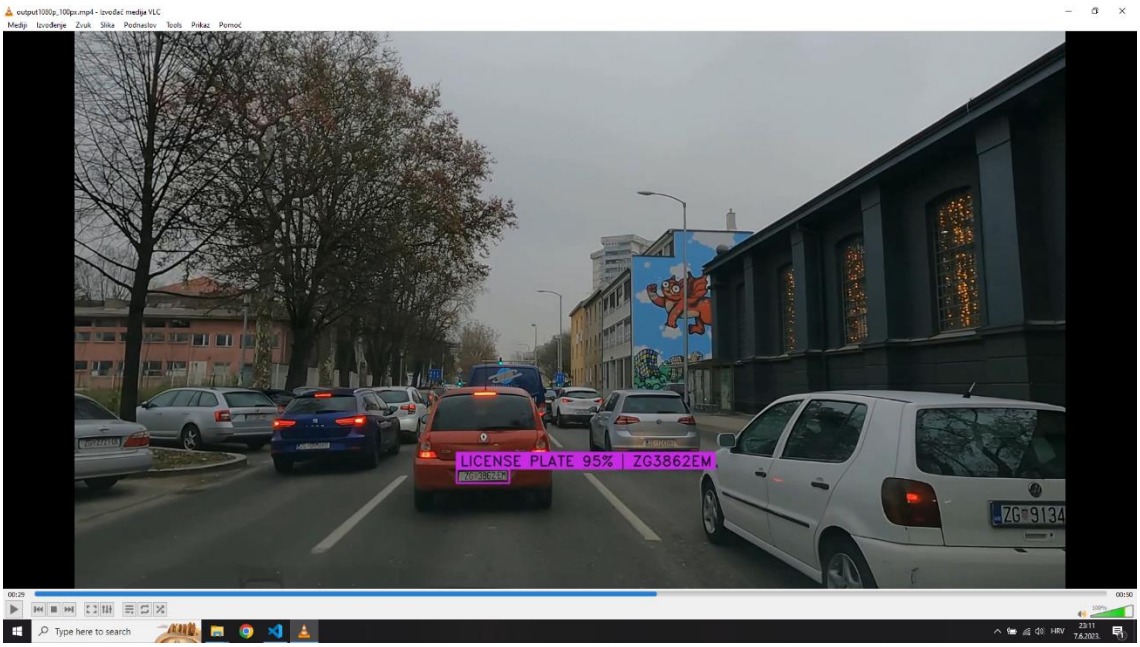
Tablica 4.2. Vrijednosti procijenjene udaljenosti registarske tablice vozila od kamere u metrima dobivene izračunom pri horizontalnom vidnom polju 75 stupnjeva za širine detektiranog graničnog okvira registarske tablice vozila 120 piksela, 100 piksela, 80 piksela i 60 piksela

	Širina detektiranog graničnog okvira registarske tablice vozila			
	120 piksela	**100 piksela**	**80 piksela**	**60 piksela**
Horizontalno vidno polje 75°	5.42142 m	6.50570 m	8.13213 m	10.84284 m

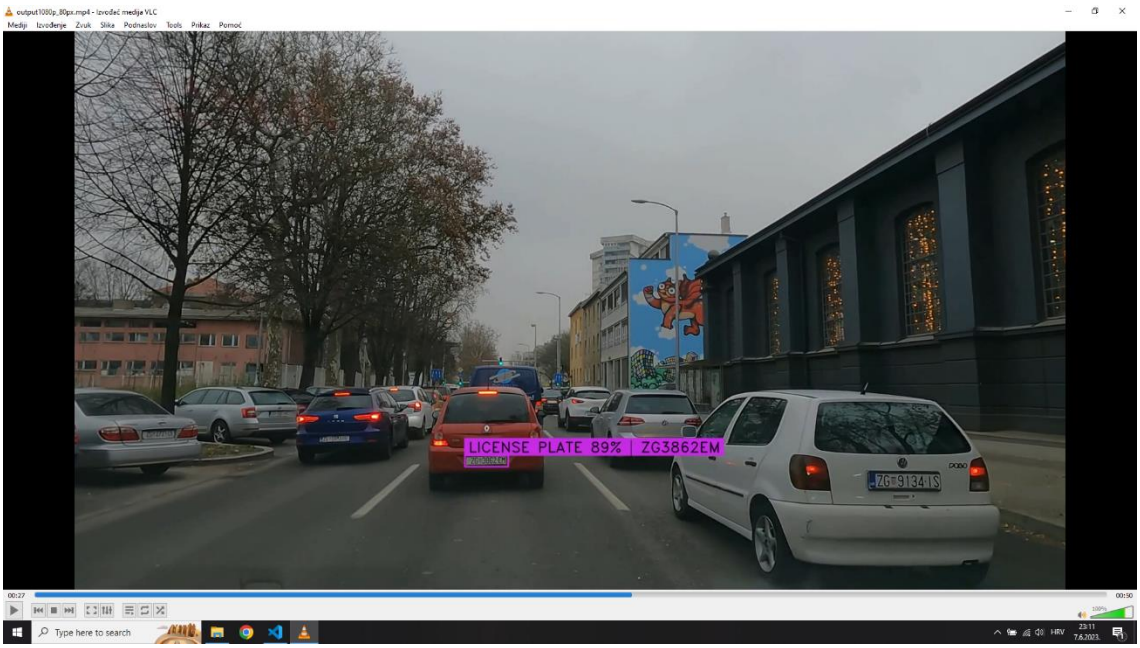
Na slici 4.16. vidi se registarska tablica vozila vlastito procijenjene udaljenosti od kamere približno 5.42 metara, na slici 4.17 registarska tablica vozila vlastito procijenjene udaljenosti od kamere približno 6.51 metara, na slici 4.18. registarska tablica vozila vlastito procijenjene udaljenosti od kamere približno 8.13 metara i na slici 4.19 registarska tablica vozila vlastito procijenjene udaljenosti od kamere približno 10.84 metara.



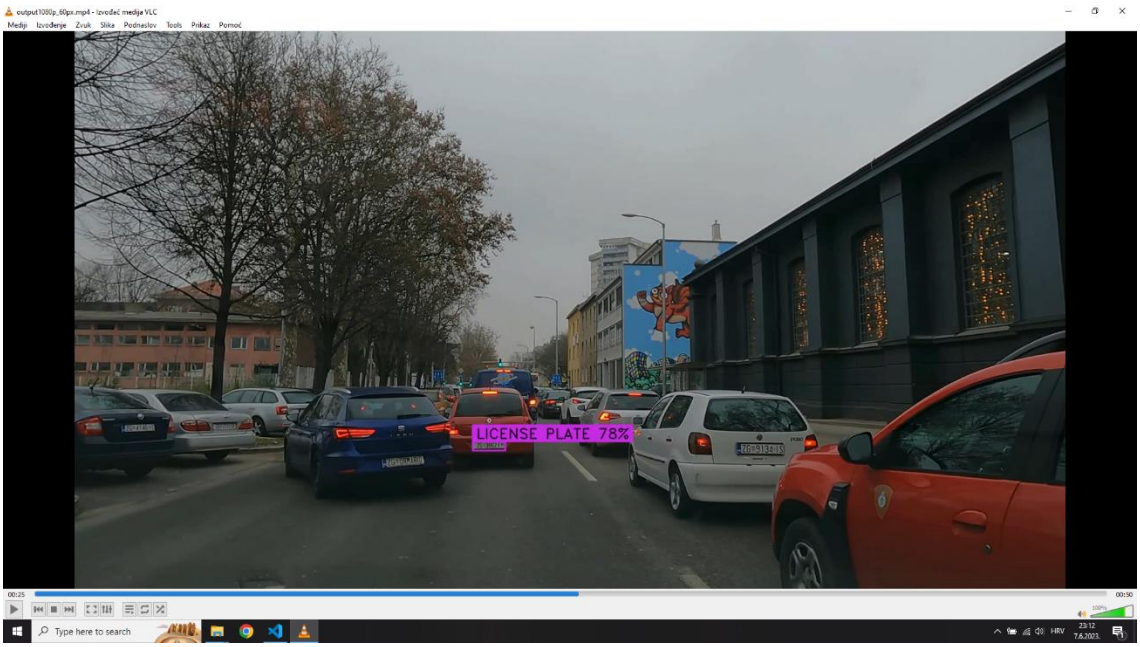
Slika 4.16. Širina detektiranog graničnog okvira registarske tablice vozila 120 piksela pri rezoluciji videa 1920x1080 piksela – vlastito procijenjena udaljenost približno 5.42 metara



Slika 4.17. Širina detektiranog graničnog okvira registarske tablice vozila 100 piksela pri rezoluciji videa 1920x1080 piksela – vlastito procijenjena udaljenost približno 6.51 metara



Slika 4.18. Širina detektiranog graničnog okvira registarske tablice vozila 80 piksela pri rezoluciji videa 1920x1080 piksela – vlastito procijenjena udaljenost približno 8.13 metara

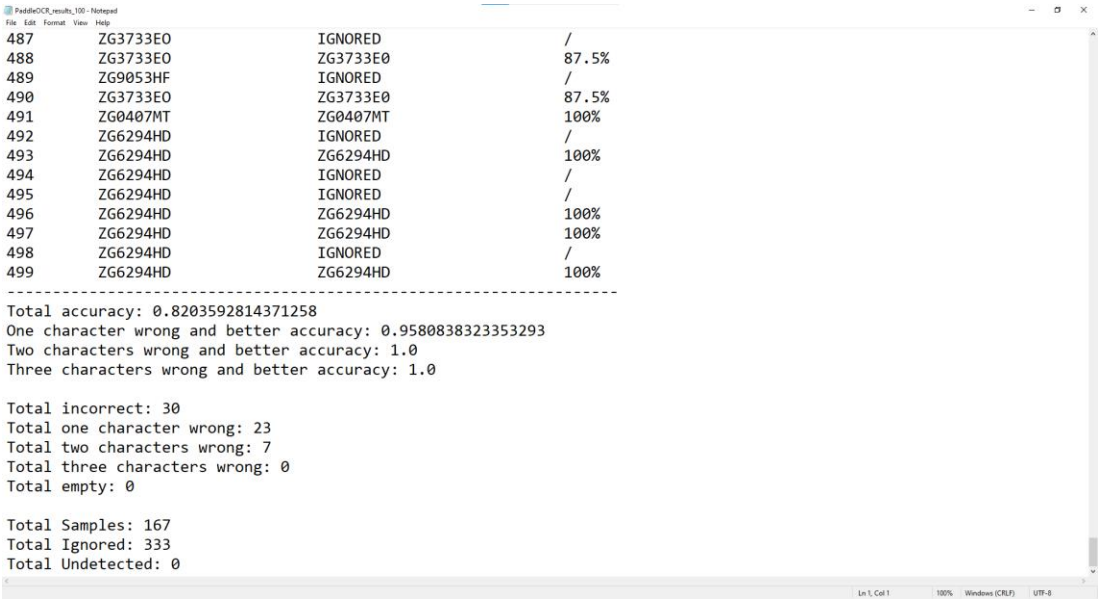


Slika 4.19. Širina detektiranog graničnog okvira registarske tablice vozila 60 piksela pri rezoluciji videa 1920x1080 piksela – vlastito procijenjena udaljenost približno 10.84 metara

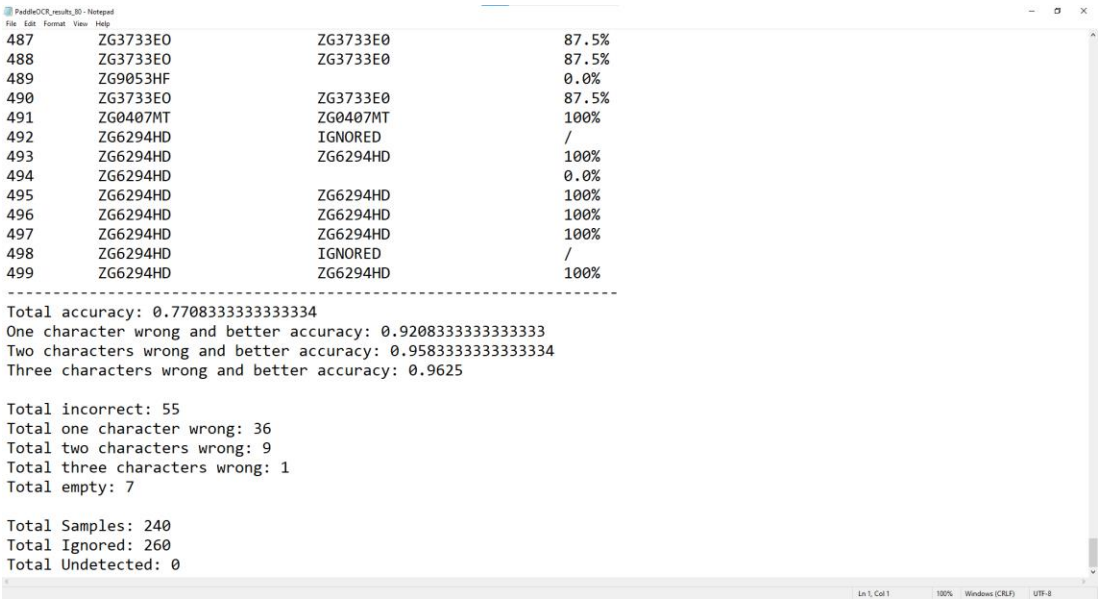
Rezultati u nastavku predstavljaju provedeno testiranje za *PaddleOCR* algoritam pri širinama detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući 120 piksela (slika 4.20.), 100 piskela (slika 4.21.), 80 piksela (slika 4.22.) i 60 piksela (slika 4.23.).

```
487         ZG3733EO                IGNORED                 /
488         ZG3733EO                ZG3733E0                87.5%
489         ZG9053HF                IGNORED                 /
490         ZG3733EO                ZG3733E0                87.5%
491         ZG0407MT                IGNORED                 /
492         ZG6294HD                IGNORED                 /
493         ZG6294HD                ZG6294HD                100%
494         ZG6294HD                IGNORED                 /
495         ZG6294HD                IGNORED                 /
496         ZG6294HD                ZG6294HD                100%
497         ZG6294HD                ZG6294HD                100%
498         ZG6294HD                IGNORED                 /
499         ZG6294HD                ZG6294HD                100%
----------------------------------------------------------------------
Total accuracy: 0.8032786885245902
One character wrong and better accuracy: 0.9590163934426229
Two characters wrong and better accuracy: 1.0
Three characters wrong and better accuracy: 1.0

Total incorrect: 24
Total one character wrong: 19
Total two characters wrong: 5
Total three characters wrong: 0
Total empty: 0

Total Samples: 122
Total Ignored: 378
Total Undetected: 0
```

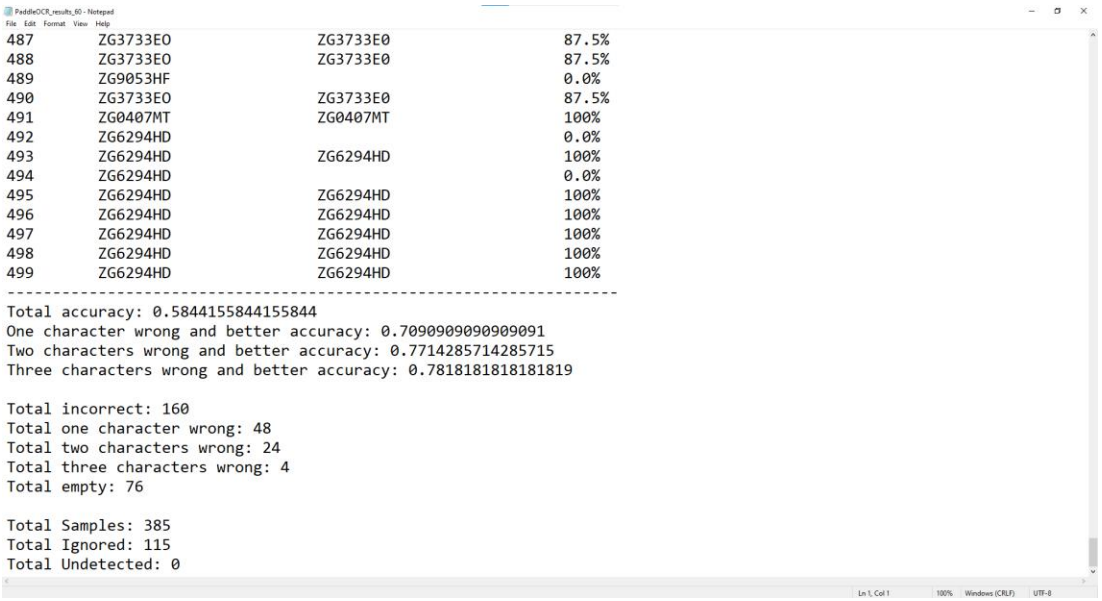
Slika 4.20. Rezultati testiranja PaddleOCR algoritma pri širini detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući 120 piksela



Slika 4.21. Rezultati testiranja PaddleOCR algoritma pri širini detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući 100 piksela



Slika 4.22. Rezultati testiranja PaddleOCR algoritma pri širini detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući 80 piksela



Slika 4.23. Rezultati testiranja PaddleOCR algoritma pri širini detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući 60 piksela

U tablici 4.3. dani su rezultati testiranja *PaddleOCR* algoritma pri svim prethodno navedenim širinama detektiranih graničnih okvira registarskih tablica vozila zajedno radi bolje preglednosti.

Tablica 4.3. Rezultati testiranja PaddleOCR algoritma za točnost prepoznavanja registarskih tablica vozila pri detektiranim graničnim okvirima registarskih tablica vozila širine veće od i uključujući 120 piksela, 100 piksela, 80 piksela i 60 piksela

	***PaddleOCR* pri širinama detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući *X* piksela**			
	120	**100**	**80**	**60**
Točnost	80.33%	82.04%	77.08%	58.44%
Točnost za jedan netočan znak i bolje	95.9%	95.81%	92.01%	70.91%
Točnost za dva netočna znaka i bolje	100%	100%	95.83%	77.14%
Točnost za tri netočna znaka i bolje	100%	100%	96.25%	78.18%
Netočno prepoznate registarske tablice vozila	24	30	55	160
Prepoznate registarske tablice vozila s jednim netočnim znakom	19	23	36	48

Prepoznate registarske tablice vozila s dva netočna znaka	5	7	9	24
Prepoznate registarske tablice vozila s tri netočna znaka	0	0	1	4
Registarske tablice vozila za koje je prepoznat prazan *string*	0	0	7	76
Registarske tablice vozila na kojima je provedeno testiranje	122	167	240	385
Zanemarene registarske tablice vozila	378	333	260	115
Nedetektirane registarske tablice vozila	0	0	0	0

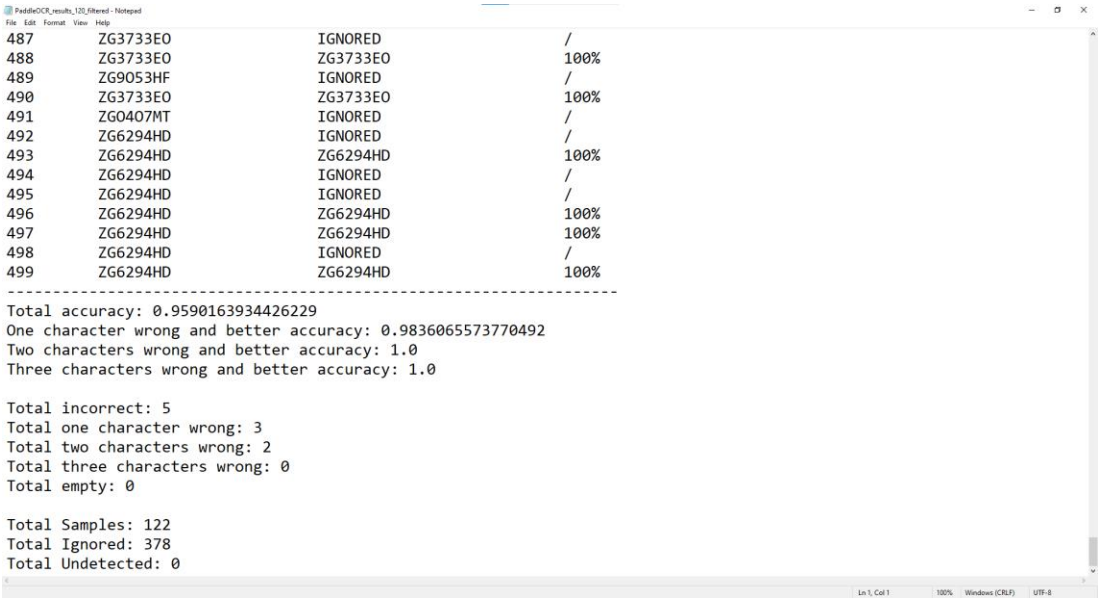
Nakon svih provedenih testiranja, za korištenje u krajnjem sustavu odabrana je udaljenost detekcije koja se odnosi na širinu detektiranog graničnog okvira registarske tablice vozila većeg od i uključujući 80 piksela za rezoluciju slike 1920x1080 piksela, zato što pruža najbolji kompromis između točnosti prepoznavanja znakova koja u ovom slučaju iznosi 77.08% i širine detektiranog graničnog okvira registarske tablice vozila koja u ovom slučaju iznosi najmanje 80 piksela. Vlastito procijenjena udaljenost registarske tablice vozila od kamere za detektirani granični okvir registarske tablice vozila širine 80 piksela iznosi približno 8.13 metara.

Iz tablice 4.3. vidljivo je da točnost *PaddleOCR* algoritma uz širine detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući 100 piksela nadmašuje točnost *PaddleOCR* algoritma uz širine detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući 120 piksela. Ovo ne bi trebao biti slučaj jer bi *PaddleOCR* algoritam trebao točnije prepoznavati registarske tablice vozila koje se nalaze bliže vozaču, odnosno koje su veće na slici a time i čitljivije. Razlog ove nepravilnosti je što *PaddleOCR* često ne razlikuje slovo 'O' i broj '0', te slovo 'I' i broj '1'. Dio skupa podataka korištenog za testiranje OCR algoritama koji sadrži širine detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući 100 piksela i manjih od 120 piksela je sadržavao puno manje oznaka koje imaju slovo 'O' i/ili broj '0' i/ili slovo 'I' i/ili broj '1'. S obzirom da je skup podataka nastao uzimanjem slika ekrana videa *Youtube* kanala City Drive [44], nije bilo moguće pronaći četiri podjednake udaljenosti za svaku registarsku tablicu vozila, što je rezultiralo gore navedenom nepravilnošću. Zbog toga, ne može se s punom pouzdanošću tvrditi da su zaključci doneseni na osnovu dobivenih rezultata do kraja točni, ali opet s druge strane zasigurno ukazuju na ono najbitnije.

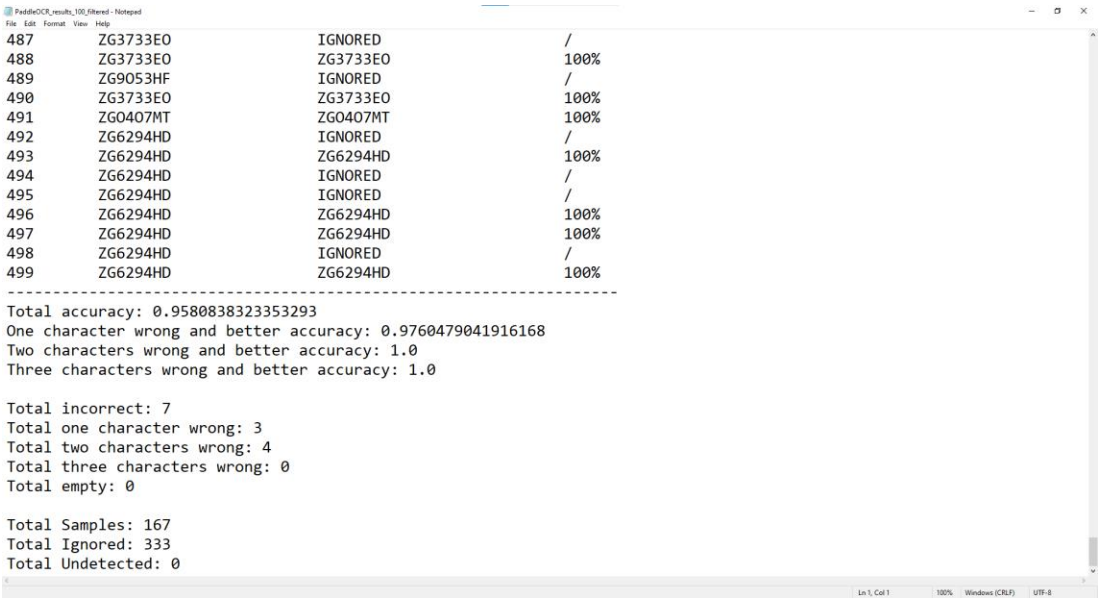
Ukoliko se nakon prepoznavanja oznaka broj '0' uvijek zamijeni slovom 'O', a broj '1' uvijek zamijeni slovom 'I', i ukoliko ručno upisivanje referentnih oznaka također slijedi ova pravila, može se postići veća točnost prepoznavanja standarnih hrvatskih registarskih tablica vozila jer je na njima redoslijed slova i brojeva uvijek isti. Ali kod vlastito prilagođenih hrvatskih registarskih tablica vozila može se dogoditi jednako prepoznavanje dviju različitih oznaka, primjerice 'ORAO' i '0RA0' bi bili u ovom slučaju oboje prepoznati kao 'ORAO'.

Rezultati u nastavku predstavljaju provedeno testiranje za *PaddleOCR* algoritam uz zamjenu broja '0' slovom 'O' i uz zamjenu broja '1' slovom 'I' pri širinama detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući 120 piksela (slika 4.24.), 100 piskela (slika 4.25.), 80 piksela (slika 4.26.) i 60 piksela (slika 4.27.).

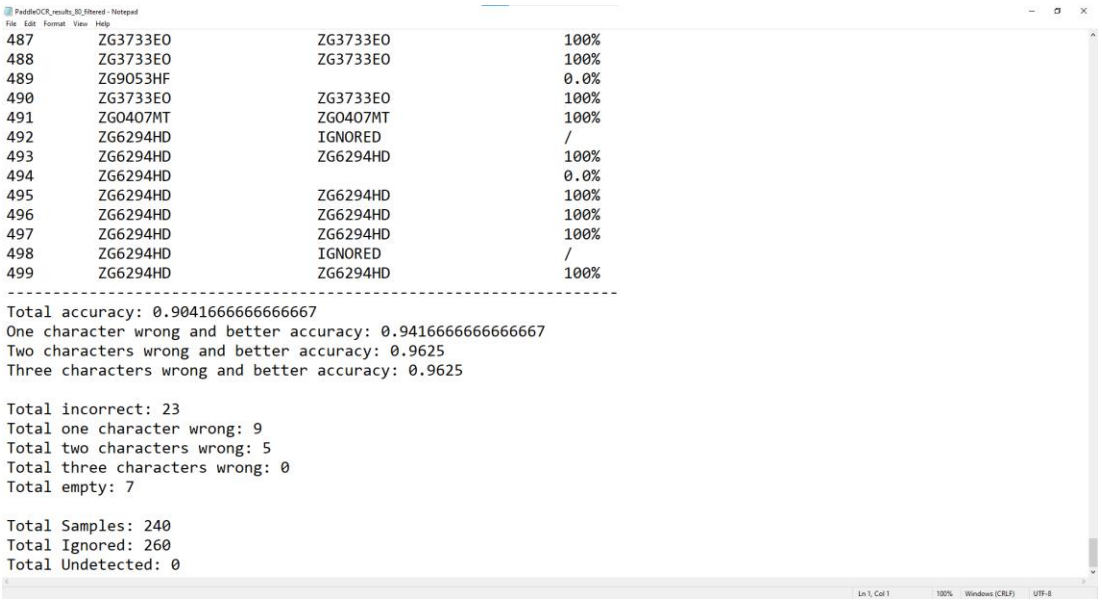
U tablici 4.4. dani su rezultati testiranja *PaddleOCR* algoritma uz zamjenu broja '0' slovom 'O' i uz zamjenu broja '1' slovom 'I' pri svim prethodno navedenim širinama detektiranih graničnih okvira registarskih tablica vozila zajedno radi bolje preglednosti.



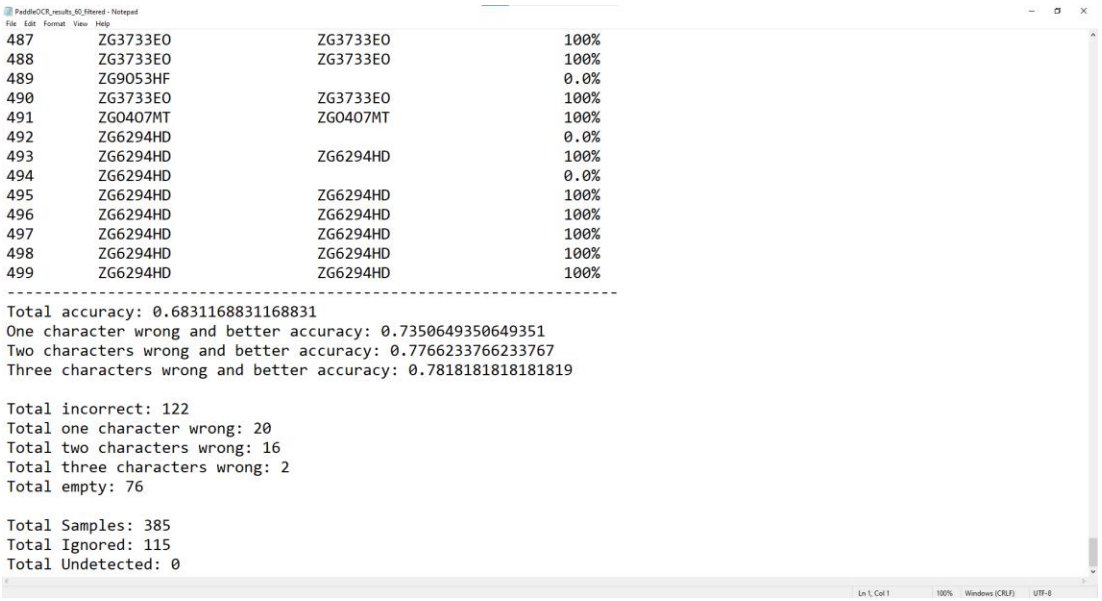
Slika 4.24. Rezultati testiranja PaddleOCR algoritma uz zamjenu broja '0' slovom 'O' i uz zamjenu broja '1' slovom 'I' pri širini detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući 120 piksela



Slika 4.25. Rezultati testiranja PaddleOCR algoritma uz zamjenu broja '0' slovom 'O' i uz zamjenu broja '1' slovom 'I' pri širini detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući 100 piksela



Slika 4.26. Rezultati testiranja PaddleOCR algoritma uz zamjenu broja '0' slovom 'O' i uz zamjenu broja '1' slovom 'I' pri širini detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući 80 piksela



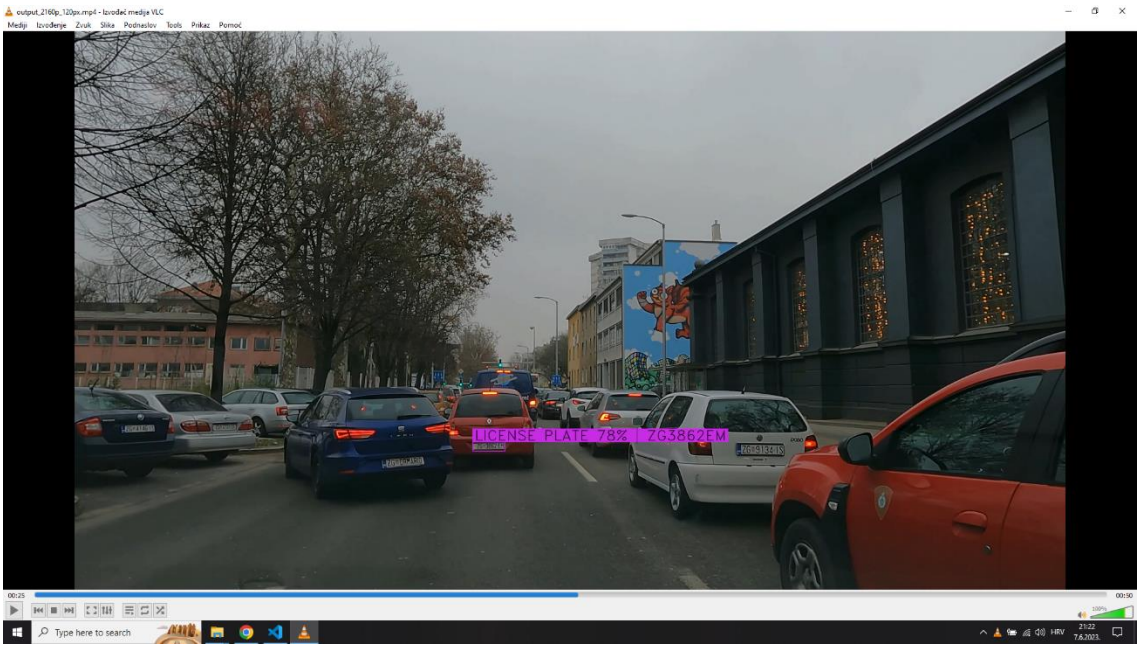
Slika 4.27. Rezultati testiranja PaddleOCR algoritma uz zamjenu broja '0' slovom 'O' i uz zamjenu broja '1' slovom 'I' pri širini detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući 60 piksela

Tablica 4.4. Rezultati testiranja PaddleOCR algoritma za točnost prepoznavanja registarskih tablica vozila uz zamjenu broja '0' slovom 'O' i uz zamjenu broja '1' slovom 'I' pri širinama detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući 120 piksela, 100 piksela, 80 piksela i 60 piksela

	***PaddleOCR* pri širinama detektiranih graničnih okvira registarskih tablica vozila većih od i uključujući *X* piksela**			
	120	**100**	**80**	**60**
Točnost	95.9%	95.81%	90.42%	68.31%
Točnost za jedan netočan znak i bolje	98.36%	97.6%	94.17%	73.51%
Točnost za dva netočna znaka i bolje	100%	100%	96.25%	77.66%
Točnost za tri netočna znaka i bolje	100%	100%	96.25%	78.18%
Netočno prepoznate registarske tablice vozila	5	7	23	122
Prepoznate registarske tablice vozila s jednim netočnim znakom	3	3	9	20

Prepoznate registarske tablice vozila s dva netočna znaka	2	4	5	16
Prepoznate registarske tablice vozila s tri netočna znaka	0	0	0	2
Registarske tablice vozila za koje je prepoznat prazan *string*	0	0	7	76
Registarske tablice vozila na kojima je provedeno testiranje	122	167	240	385
Zanemarene registarske tablice vozila	378	333	260	115
Nedetektirane registarske tablice vozila	0	0	0	0

Za rezoluciju 3840x2160 piksela ili veću, moguća su točna prepoznavanja registarskih tablica vozila na većim udaljenostima nego za rezoluciju 1920x1080 piksela, no sustav će se ravnati prema rezoluciji 1920x1080 piksela jer su takve kamere dostupnije na tržištu od onih koje nude veću rezoluciju. Na slici 4.28. prikazan je primjer točnog prepoznavanja znakova za detekciju veću od i uključujući 120 piksela za rezoluciju 3840x2160 piksela, što je jednako 60 piksela za rezoluciju 1920x1080 piksela.



Slika 4.28. Širina detektiranog graničnog okvira registarske tablice vozila 120 piksela pri rezoluciji videa 3840x2160 piksela (ekvivalentno 60 piksela pri rezoluciji videa 1920x1080 piksela) – vlastito procijenjena udaljenost približno 10.84 metara

Korišteni videozapis rezolucije 3840x2160 piksela se može pronaći na DVD-u priloženom uz ovaj rad u mapi Prilog P.4.3. Konačni repozitorij aplikacije za detekciju i prepoznavanje znakova registarskih tablica vozila dostupan je na DVD-u priloženom uz ovaj rad u mapi Prilog P.4.4.

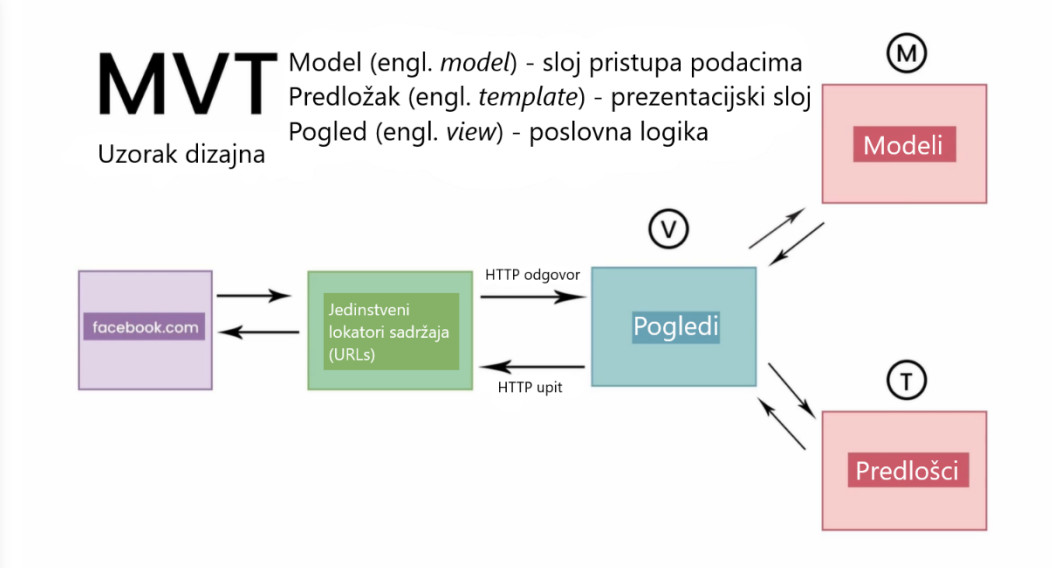
5. RAZVOJ WEB APLIKACIJE ZA PRIKAZ PODATAKA O PROFILU I RECENZIRANJE VOZAČA ISPRED VLASTITOG VOZILA

Posljednji zadatak za ostvarenje sustava za prikaz podataka o profilu vozača ispred vlastitog vozila zasnovan na detekciji i prepoznavanju registarkih tablica vozila, je razvoj web aplikacije za prikaz podataka o profilu i recenziranje vozača ispred vlastitog vozila. Web aplikacija će imati radno okruženje odvojeno od radnog okruženja sustava za detekciju i prepoznavanje registarskih tablica vozila te će međusobno komunicirati kao server-klijent. Web aplikacija će biti izgrađena pomoću *Django* okvira za razvoj pozadinskih funkcionalnosti koji se temelji na programskom jeziku *Python*. Ovo nije nužno za kompatibilnost s radnim okruženjem za detekciju i prepoznavanje registarskih tablica vozila te se mogla koristiti bilo koja tehnologija za razvoj pozadinskih funkcionalnosti. Web aplikacija će biti implementirana na javni server pomoću *AWS* usluga i *Heroku* platforme. Konačno, testirat će se funkcionalnosti web aplikacije i cjelokupni sustav.

5.1. Django okvir za razvoj pozadinskih funkcionalnosti web aplikacije

Django je web okvir (eng. *web framework*) otvorenog koda napisan u *Python*-u. Web okvir općenito omogućuje razvoj složenih i skalabilnih web aplikacija brže i jednostavnije. *Django* je stvoren kako bi smanjio ponavljanje koda i promicao čistu, organiziranu i održivu arhitekturu [56].

Django ima sustav usmjerevanja koji omogućuje mapiranje jedinstvenih lokatora sadržaja (engl. *Uniform Resource Locator*, URL) na odgovarajuće poglede (engl. *views*). To pomaže u organiziranju i upravljanju aplikacijom. Pogledi u *Django*-u predstavljaju *Python* funkcije koje obrađuju zahtjeve i generiraju odgovore koji se vraćaju korisniku. Ovdje se nalazi glavna poslovna logika aplikacije. Osim toga, *Django* pruža dodatni sloj (engl. *models*) koji omogućuje programerima da manipuliraju bazom podataka putem *Python* objekata i upita, umjesto da koriste SQL izravno. Ovo olakšava rad s bazom podataka i čini ga manje osjetljivim na promjene u samoj bazi podataka. Uz to, *Django* dolazi sa šablonskim sustavom koji omogućuje programerima da razdvajaju logiku od prezentacije. To znači da HTML i *Python* kod mogu biti jasno odvojeni, što olakšava rad dizajnera i programera zajedno, odnosno programeri mogu lako iskoristiti gotov predložak dizajna (engl. *templates*). Tri prethodno navedene komponente čine MVT (engl. *Model-View-Controller*) oblikovni uzorak (slika 5.1.).



Slika 5.1. MVT oblikovni uzorak [57]

5.2. Radno okruženje za razvoj web aplikacije

Prije instalacije okvira *Django*, kreira se virtualno radno okruženje kako bi se osigurala kompatibilnost svih paketa koji će biti instalirani. Virtualno radno okruženje kreira se pomoću *Python* modula *venv* pozivom sljedeće naredbe unutar naredbenog retka:

```
python -m venv /path/to/new/virtual/environment
```

Nakon toga se instalira okvir *Django* pomoću naredbe *Python* modula *pip* pozivom sljedeće naredbe unutar naredbenog retka:

```
pip install django
```

Django projekt se kreira pozivom sljedeće naredbe unutar naredbenog retka:

```
django-admin startproject driverprofile
```

Django aplikacija se kreira pozivom sljedeće naredbe unutar naredbenog retka:

```
django-admin startapp users
```

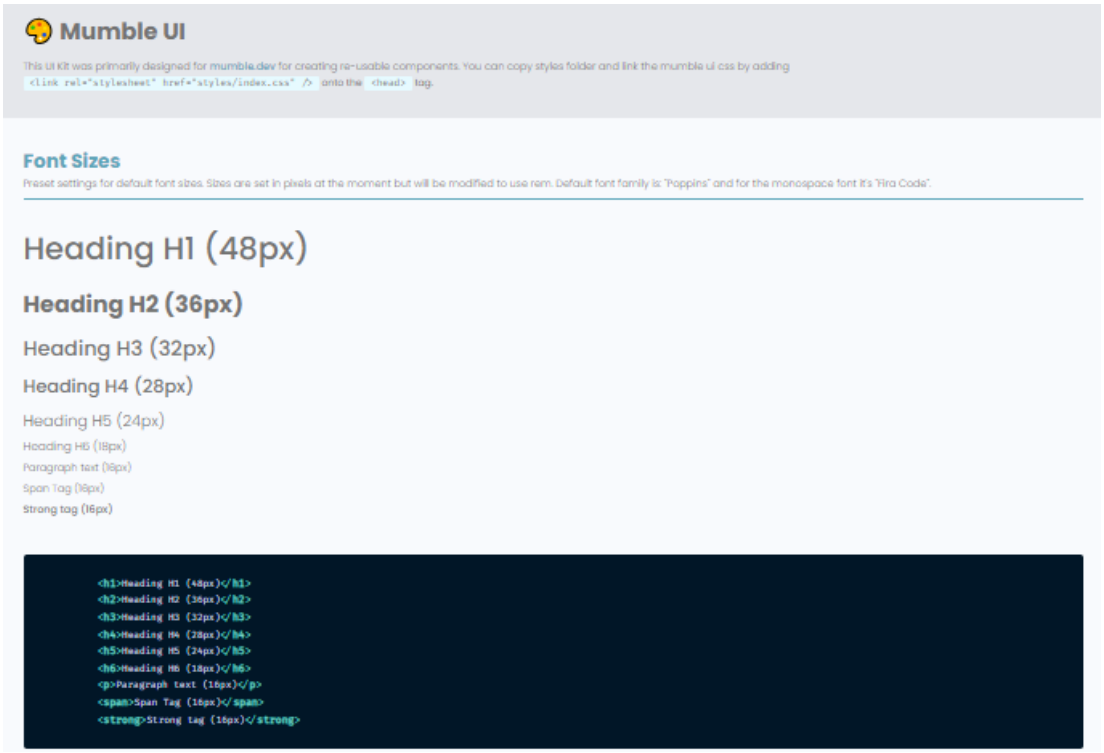
Posljednje dvije naredbe kreiraju sve standardne datoteke koje su nužne za lokalno pokretanje servera, pa se nakon toga može odmah prijeći na vlastito razvijanje izgleda i pozadinskih funkcionalnosti.

Django projekt može sadržavati više aplikacija ukoliko ima više namjena, odnosno svaka aplikacija obavlja specifične zadatke vezane za svoju funkcionalnost. Jedina glavna funkcionalnost *Django* projekta za ovaj slučaj upotrebe je prikaz podataka o profilu i recenziranje vozača, stoga će sadržavati samo jednu aplikaciju.

Ostvarene pozadinske funkcionalnosti u nastavku neće biti popraćene naredbama terminala i potpunim kodom. Konačni repozitorij web aplikacije dostupan je na DVD-u priloženom uz ovaj rad u mapi Prilog P.5.1., a za detaljnije razumijevanje *Django* okvira preporuča se tečaj dostupan na [57].

5.3. Predložak za izgled web aplikacije

Zahvaljujući šablonskom sustavu *Django* okvira, HTML i Python kod mogu biti jasno odvojeni u različitim datotekama. Na taj način, programeri mogu raditi na poslovnoj logici, obradi podataka i manipulaciji sa sadržajem u *Python* kodu, dok dizajneri mogu usmjeriti svoj fokus na izradu atraktivnih i funkcionalnih predložaka dizajna u HTML-u. Za uređivanje izgleda web aplikacije ovog projekta, korišten je predložak dostupan u *Django* tečaju [57]. Na slici 5.2. prikazan je dio kompleta za uređivanje korisničkog sučelja.



Slika 5.2. Prikaz dijela kompleta za uređivanje korisničkog sučelja [58]

5.4. Funkcionalni zahtjevi web aplikacije

Analizirat će se funkcionalni zahtjevi web aplikacije. Funkcionalni zahtjevi opisuju što aplikacija treba raditi, koje akcije korisnici mogu izvršavati te kako će aplikacija obraditi te akcije. Osnovni funkcionalni zahtjevi web aplikacije koji će biti analizirani su registracija korisnika, prijava i odjava korisnika, uređivanje informacija vlastitog korisničkog računa i pretraživanje profila vozača. Funkcionalni zahtjevi web aplikacije specifični za glavni slučaj upotrebe cjelokupnog sustava ovog rada koji će biti analizirani su omogućavanje recenziranja samo prijavljenim korisnicima koji koriste aplikaciju za detekciju i prepoznavanje registarskih tablica vozila, te funkcionalnost recenziranja i izračun statistike postojećih recenzija.

5.4.1. Registracija korisnika

Korisnici će moći izvršiti registraciju na aplikaciji putem obrasca. Registracija će zahtijevati sljedeće informacije: ime, e-mail adresa, korisničko ime, lozinka i potvrda lozinke. E-mail adresa mora biti oblika 'naziv@email.domena'. Lozinka mora sadržavati najmanje 8 znakova i kombinaciju slova i brojeva. Nakon neuspješne registracije, korisnici će biti automatski preusmjereni na istu stranicu registracije uz dodatne informacije o unešenim informacijama koje nisu zadovoljile zahtjeve registracije i dobiti jednokratnu obavijest o neuspješnosti registracije na web aplikaciji. Nakon uspješne registracije, korisnici će dobiti jednokratnu obavijest o uspješnosti registracije na web aplikaciji i biti će automatski prijavljeni i preusmjereni na stranicu uređivanja korisničkog računa.

5.4.2. Prijava korisnika i odjava korisnika

Registrirani korisnici će se moći prijaviti na aplikaciju koristeći svoje korisničko ime i lozinku. Nakon neuspješne prijave, korisnici će dobiti jednokratnu obavijest o neuspješnosti prijave na web aplikaciji i biti će automatski preusmjereni na istu stranicu prijave. Nakon uspješne prijave, korisnici će biti preusmjereni na početnu stranicu i dobiti pristup stranici uređivanja vlastitog korisničkog računa.

Prijavljeni korisnici se mogu odjaviti nakon čega više nemaju pristup stranici uređivanja vlastitog korisničkog računa. Nakon odjave korisnici će biti preusmjereni na početnu stranicu web aplikacije i dobiti jednokratnu obavijest o uspješnosti odjave.

5.4.3. Uređivanje informacija vlastitog korisničkog računa

Korisnici će imati mogućnost uređivanja informacija svojih korisničkih računa. To uključuje ažuriranje imena, e-mail adrese, korisničkog imena, lokacije i slike profila. Uz to, u odvojenoj sekciji moći će ažurirati ili brisati oznaku vlastite registarske tablice vozila. Nakon svakog ažuriranja, dobit će jednokratnu obavijest o uspješnosti ili neuspješnosti ažuriranja informacija svojih korisničkog računa.

5.4.4. Pretraživanje profila vozača

Korisnici će moći pretraživati profile vozača koristeći oznaku registarske tablice vozila. Za pretraživanje profila vozača neće biti zahtjevana prijava korisnika. Ukoliko se u pretraživač unese i pošalje oznaka registarske tablice vozila koja nije sadržana unutar baze podataka povezane s ovom web aplikacijom, korisnici će dobiti jednokratnu obavijest o nepostojanju rezultata pretrage.

5.4.5. Omogućavanje recenziranja samo prijavljenim korisnicima koji koriste aplikaciju za detekciju i prepoznavanje registarskih tablica vozila

Samo prijavljeni korisnici koji pošalju oznaku registarske tablice vozila iz aplikacije za detekciju i prepoznavanje registarskih tablica vozila moći će recenzirati vozača. Aplikacija za detekciju i prepoznavanje registarskih tablica vozila otvara novi prozor za recenziranje u web aplikaciji samo kada se prepozna različita registarska tablica vozila od prethodne.

5.4.6. Funkcionalnost recenziranja i izračun statistike postojećih recenzija

Web aplikacija će omogućiti funkcionalnost recenziranja vozača prema kategorijama brzine vožnje, poštivanja semafora, poštivanja prometnih znakova, naglog mijenjanja vozne trake, korištenja smjerokaznog svjetla i poštivanja pješačkog prijelaza. Kategorija brzine vožnje će pružati tri opcije odabira: sporo, normalno, brzo. Ostale kategorije će pružati dvije opcije odabira: pozitivan i negativan odabir prilagođen za svaku kategoriju.

Aplikacija će automatski izračunavati statistike temeljene na postojećim recenzijama vozača. Za svaku prethodno navedenu kategoriju će prikazati omjer odabranih opcija svih recenzija korisnika u obliku traka različite boje za svaku opciju i različite duljine s obzirom na izračunati postotak odabranih opcija svih recenzija korisnika.

5.5. Osnovne pozadinske funkcionalnosti web aplikacije

Nakon generiranja nužnih standardnih datoteka za *Django* projekt, vlastiti razvoj odvijat će se za projektnu aplikaciju unutar datoteka *urls.py*, *models.py*, *forms.py*, *signals.py*, *views.py*.

5.5.1. Rute web aplikacije

Unutar datoteke *urls.py* svaka ruta definirana je pomoću *path()* funkcije iz *Django* modula *django.urls*, koja povezuje određeni URL s odgovarajućim *view* funkcijama. *View* funkcije predstavljaju *Python* funkcije koje obrađuju HTTP zahtjeve (engl. *requests*) i generiraju HTTP odgovore (engl. *responses*). U ovoj aplikaciji definirane su rute:

- Početna stranica – "
- Prijava korisnika – 'login/'
- Odjava korisnika – 'logout/'
- Registracija korisnika – 'register/'
- Korisnički račun – 'account/'
- Uređivanje korisničkog računa – 'edit-account/'
- Uređivanje registarske tablice korisničkog računa – 'edit-license-plate/'
- Brisanje registarske tablice korisničkog računa – 'delete-license-plate/'

Također su definirane i rute za prikaz podataka o profilu i recenziranje vozača, sesiju vožnje i brisanje sesije vožnje, o kojima će uz njihove funkcionalnosti biti riječi u posebnom poglavlju.

5.5.2. Modeli web aplikacije

Django modeli koriste se za definiranje struktura tablica u bazi podataka, a svaki model klasa predstavlja tablicu s njezinim atributima (stupcima). Za potrebe osnovnih pozadinskih funkcionalnosti stvoreni su modeli *Profile* i *License Plate*. Model *Profile* nasljeđuje postojeći Django model *User* i sadrži atribute imena, elektroničke adrese, slike profila i slično. Model *License Plate* osim znakova registarske tablice sadrži i atribut *owner* koji definira vezu s profilom (*Profile* model). Koristi se strani ključ (engl. *Foreign key*) i postavlja se na *null* ako profil bude izbrisan. Definirana veza *Profile* modela i *License Plate* modela je jedan na više u smislu da jedan *Profile* objekt može biti pridružen više *License Plate* objekata. Jedini razlog zašto se koristi ova veza, a ne veza jedan na jedan, je da korisnik može promijeniti registarsku tablicu na profilu.

Dakle, unatoč vezi jedan na više, unutar datoteke *views.py* bit će ograničeno pridruživanje *License Plate* objekta *Profile* objektu na jedan *License Plate* objekt u određenom trenutku.

5.5.3. Forme web aplikacije

U datoteci *forms.py* definira se i prilagođava korisnički unos na temelju prethodno definiranih modela. Prva forma je *CustomUserCreationForm*, koja je izvedena iz ugrađene Django forme *UserCreationForm*. Druga forma je *ProfileForm*, koja se temelji na modelu *Profile*. Treća forma je *LicensePlateForm*, koja se temelji na modelu *LicensePlate*.

5.5.4. Signali web aplikacije

Signali su mehanizam u Django okviru koji omogućuju praćenje događaja koji se izvršavaju na objektima modela i izvršavanje određenih akcija kao odgovor na te događaje. Za ovu aplikaciju definirane su tri funkcije: *createProfile()*, *updateUser()*, *deleteUser()*.

Funkcija *createProfile()* je povezana sa signalom *post_save*, koji se okida nakon što je objekt modela *User* stvoren. Ako je objekt stvoren, tada se koristi informacija iz povezanog *User* objekta kako bi se stvorio novi *Profile* objekt.

Funkcija *updateUser()* je povezana s istim signalom *post_save*, ali za model *Profile*. Kada se *Profile* objekt ažurira, signal će se aktivirati. U ovoj funkciji, ažuriraju se podaci *User* objekta na temelju novih informacija iz *Profile* objekta.

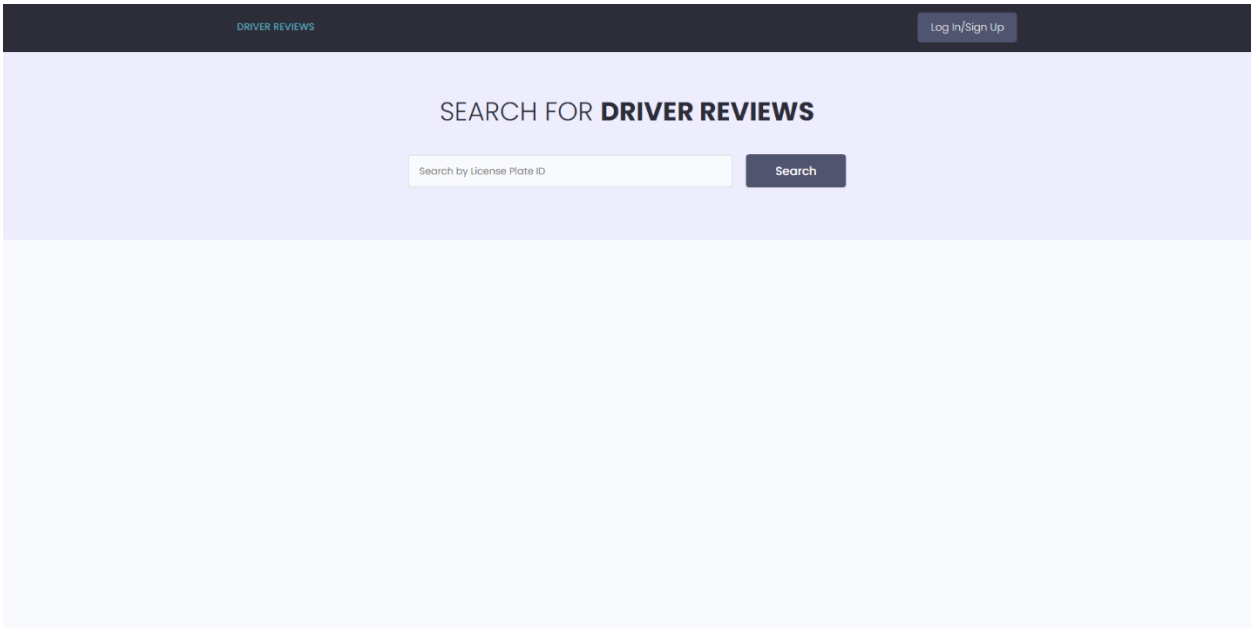
Funkcija *deleteUser()* je povezana sa signalom *post_delete*, koji se okida nakon što je model *Profile* obrisan. Kada se *Profile* objekt obriše, ovaj signal će se aktivirati. U ovoj funkciji, briše se i pripadajući *User* objekt koji je bio povezan s tim *Profile* objektom.

5.5.5. Pogledi web aplikacije

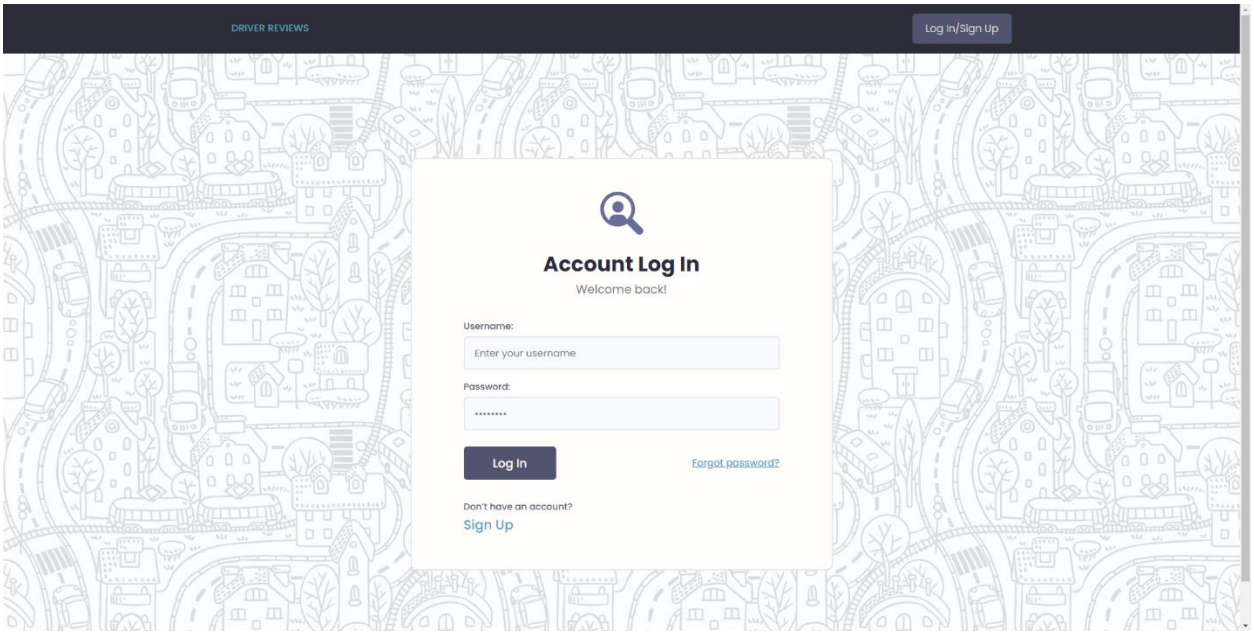
U Django okviru, pogledi (engl. *views*) predstavljaju Python funkcije koje obrađuju HTTP zahtjeve i vraćaju HTTP odgovore. Pogledi su ključna komponenta u obradi zahtjeva korisnika jer kontroliraju logiku aplikacije i određuju što će se prikazati korisniku u odgovoru. Kada korisnik posjeti neku stranicu na web aplikaciji, Django će prepoznati URL putanju koju je korisnik zatražio i proslijediti taj zahtjev odgovarajućem pogledu koji će ga obraditi. Pogled zatim obrađuje zahtjev, pristupa podacima iz baze podataka ili drugih izvora, obavlja potrebnu logiku i generira odgovor koji se šalje natrag korisniku.

Prema tome su za ovu aplikaciju napravljeni pogledi koji obrađuju zahtjeve za prijavu korisnika, odjavu korisnika, registraciju korisnika, prikaz podataka na računu korisnika, uređivanje podataka na računu korisnika, uređivanje registarske tablice na računu korisnika, brisanje registarske tablice na računu korisnika i prikaz početne stranice te mogućnost pretraživanja profila po registarskoj tablici. Odjava korisnika, prikaz i uređivanje podataka, uređivanje i brisanje registarske tablice imaju postavljeno ograničenje koje dopušta pristup samo prijavljenim korisnicima, a u suprotnom se korisniku vraća greška i preusmjerava ga se na početnu stranicu. Također se obrađuju slučajevi poput toga da se vraća greška ukoliko su korisničko ime ili lozinka neispravni, dva korisnika ne mogu imati pridružen isti *License Plate* objekt, da se ne stvara novi *License Plate* objekt ukoliko već postoji nego da se samo pridruži korisniku ako je slobodan, atribut oznake *License Plate* objekta mora imati određen raspon broja znakova i mnogi drugi.

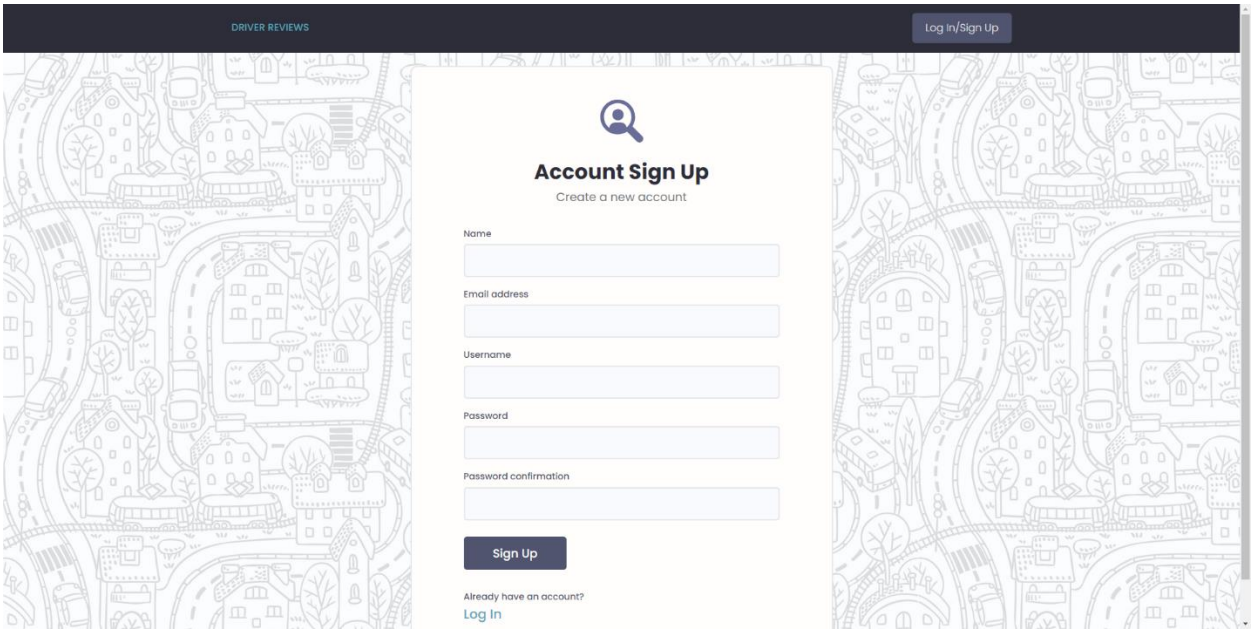
U nastavku je prikazan zajednički rezultat opisanih modula (slika 5.3., slika 5.4., slika 5.5., slika 5.6., slika 5.7.).



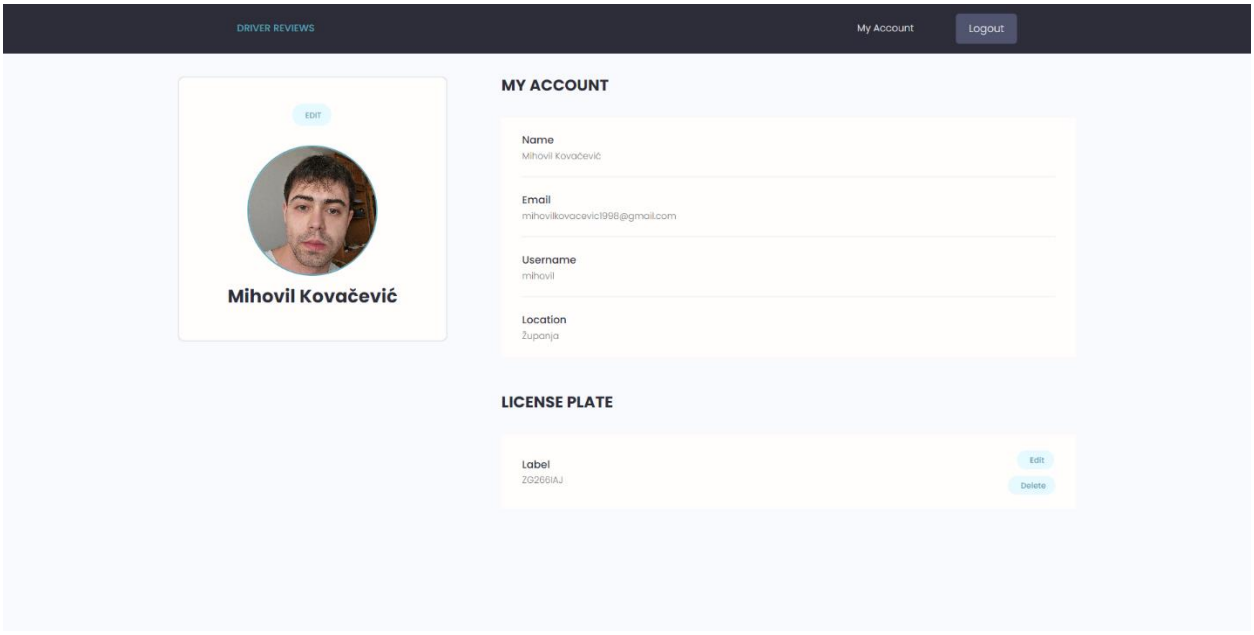
Slika 5.3. Početna stranica web aplikacije za prikaz podataka o profilu i recenziranje vozača ispred vlastitog vozila



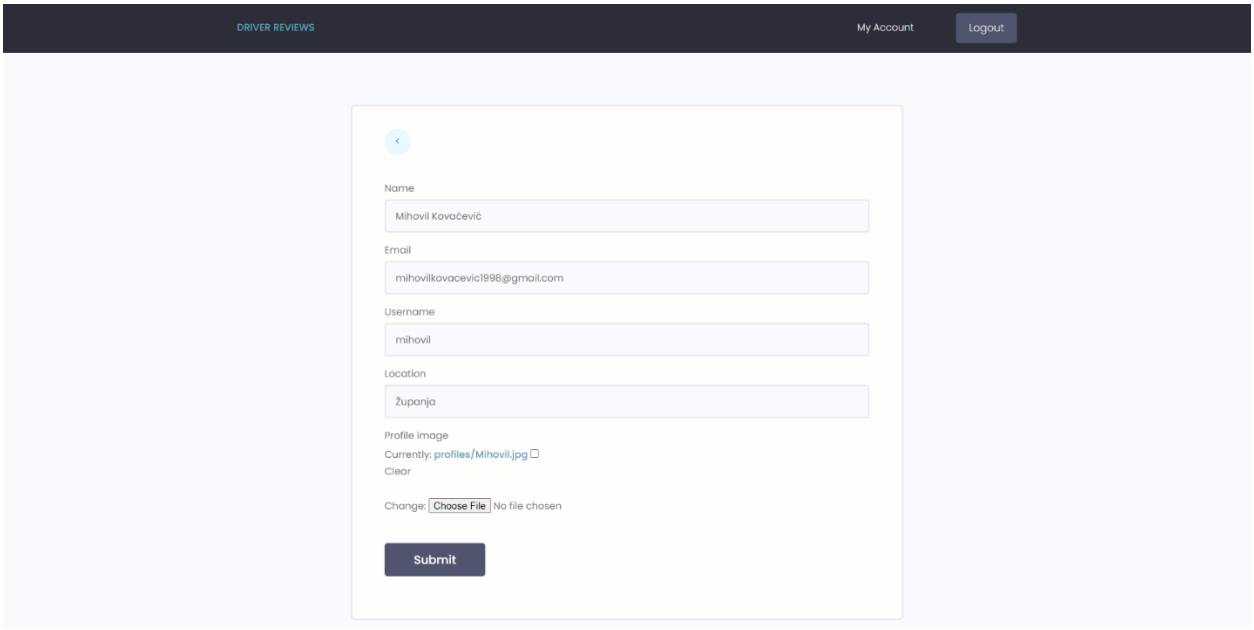
Slika 5.4. Stranica za prijavu korisnika web aplikacije za prikaz podataka o profilu i recenziranje vozača ispred vlastitog vozila



Slika 5.5. Stranica za registaraciju korisnika web aplikacije za prikaz podataka o profilu i recenziranje vozača ispred vlastitog vozila



Slika 5.6. Stranica korisničkog računa web aplikacije za prikaz podataka o profilu i recenziranje vozača ispred vlastitog vozila



Slika 5.7. Stranica uređivanja korisničkog računa web aplikacije za prikaz podataka o profilu i recenziranje vozača ispred vlastitog vozila

5.6. Dodatne pozadinske funkcionalnosti web aplikacije potrebne za recenziranje vozača

Osnovna ideja veze između web aplikacije i aplikacije za detekciju i prepoznavanje registarske tablice vozila je da se nakon detekcije i prepoznavanja registarske tablice vozila ispred vozača u stvarnom vremenu pošalju prepoznati znakovi web aplikaciji koja će obraditi taj zahtjev, prikazati korisniku trenutne podatke i omogućiti recenziranje. U ovom slučaju će web aplikacija predstavljati server koji obrađuje zahtjeve, a aplikacija za detekciju i prepoznavanje registarske tablice klijenta koji šalje zahtjeve i dobiva odgovor.

5.6.1. Strana web aplikacije

Na strani web aplikacije definirane su rute za prikaz podataka o profilu i recenziranje vozača, sesiju vožnje i brisanje sesije vožnje. Kao i kod osnovnih pozadinskih funkcionalnosti, svaka ruta ima pripadajući pogled (engl. *view*) koji obrađuje zahtjeve.

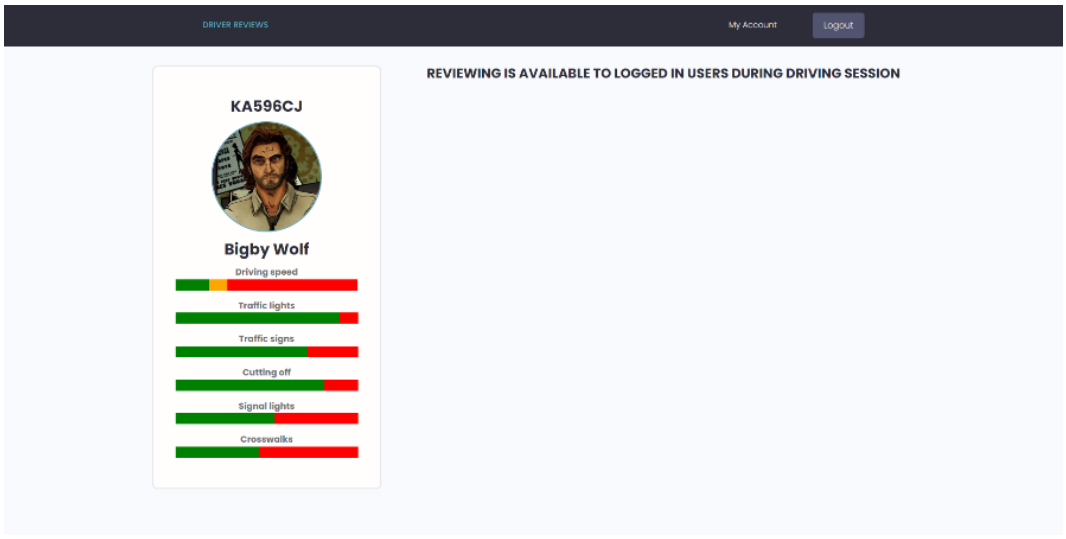
Recenzije imaju model *Review* koji sadrži mogućnosti odabira za atribute brzine vožnje (naziv unutar web aplikacije – *driving speed*), poštivanja semafora (naziv unutar web aplikacije – *traffic lights*), poštivanja prometnih znakova (naziv unutar web aplikacije – *traffic signs*), naglog mijenjanje vozne trake (naziv unutar web aplikacije – *cutting off*), korištenja smjerokaznog svjetla (naziv unutar web aplikacije – *signal lights*) i poštivanja pješačkog prijelaza (naziv unutar web aplikacije – *crosswalks*). Osim atributa koji čine recenziju, sadrži i atribut *owner* u kojem je pomoću stranog ključa definirana veza *Review* modela s *Profile* modelom te atribut *license plate* u kojem je pomoću stranog ključa definirana veza *Review* modela s *License plate* modelom. Sesije vožnje imaju model *Session* koji sadrži atribute za pohranu ključa i enkriptiranih podataka. Kako bi se opcije recenziranja mogle prikazati korisniku, za svaki od atributa modela *Review* koji omogućuju odabir, napravljena je prilagođena forma u obliku *radio button* selekcije koja je dodatno uređena pomoću CSS koda.

Funkcija pogleda za recenzije omogućava korisnicima ocjenjivanje ponašanja vozača i prikaz dosadašnjih ocjena. Recenziranje je omogućeno samo prijavljenim korisnicima za tablicu vozila koja je detektirana tijekom sesije vožnje. Koncept sesije se sastoji od dviju odvojenih funkcija pogleda koje će biti opisane u nastavku. Osim toga, funkcija pogleda za recenzije izvršava pronalaženje postojeće recenzije i njezino postavljanje u formu, izračunavanje statistike ocjena,

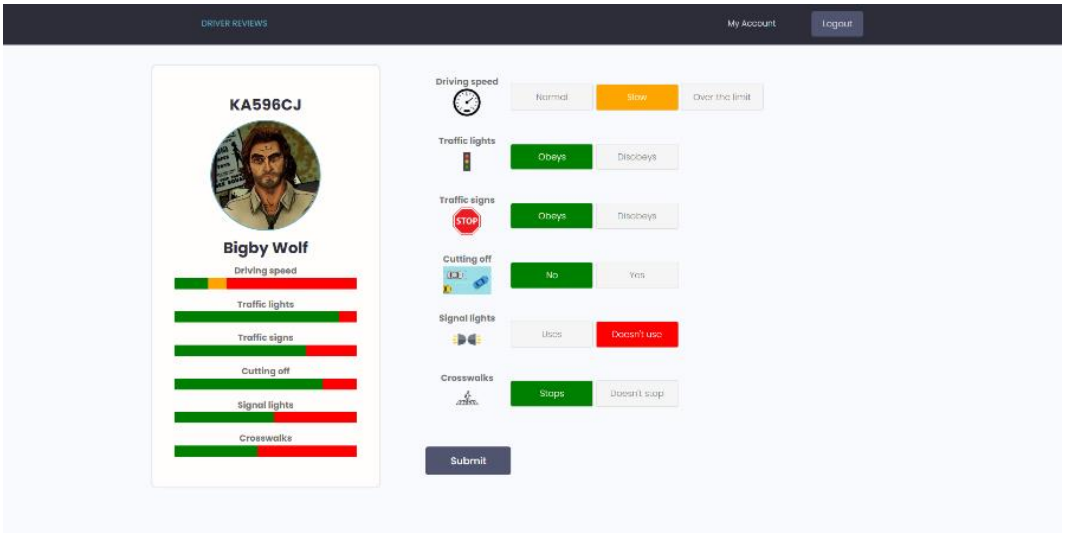
obradu zahtjeva recenzije pri čemu prati je li sesija istekla i ostalo. Za iscrtavanje prikaza dosadašnjih ocjena, ocjene se u obliku postotka prosljeđuju *html* datoteci u kojoj se pomoću oznaka predloška (engl. *template tags*, {%%}) poziva *for* petlja koja iscrtava pravokutnik širine 1% onoliko puta koliki je postotak određene ocjene.

5.6.2. Veza između web aplikacije i aplikacije za detekciju i prepoznavanje registarske tablice

URL vezan uz funkciju pogleda za sesije služi kao krajnja točka (engl. *endpoint*) kojoj aplikacija za detekciju i prepoznavanje registarske tablice šalje ključ i enkriptirane prepoznate znakove (POST zahtjev). Funkcija pogleda za sesije iščitava primljene vrijednosti i sprema ih u bazu podataka. Nakon toga aplikacija za detekciju i prepoznavanje registarske tablice otvara URL profil s dodatkom enkriptiranih detektiranih znakova, pri čemu funkcija pogleda za sesije može pronaći sesiju na temelju enkriptiranih znakova jer je ranije spremljena u bazu podataka. Na taj način recenziranje je omogućeno isključivo prijavljenom korisniku koji je pristupio profilu vozača pomoću aplikacije za detekciju i prepoznavanje registarske tablice. Nakon posljednje detekcije neke registarske tablice, recenziranje tog vozača je omogućeno dvadeset sekundi, te se u nastavku ta sesija briše iz baze podataka pomoću funkcije pogleda za brisanje sesije. Na slici 5.8. je prikazan profil vozača do kojeg se došlo pomoću pretraživanja, a na slici 5.9. je prikazan profil vozača do kojeg se došlo pomoću sesije vožnje.



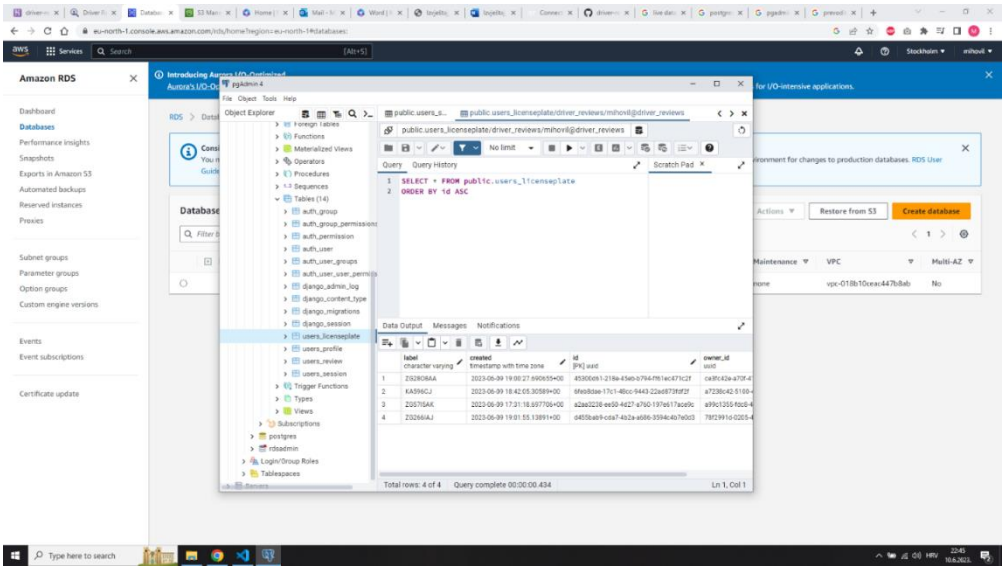
Slika 5.8. Stranica profila vozača bez mogućnosti recenziranja web aplikacije za prikaz podataka o profilu i recenziranje vozača ispred vlastitog vozila



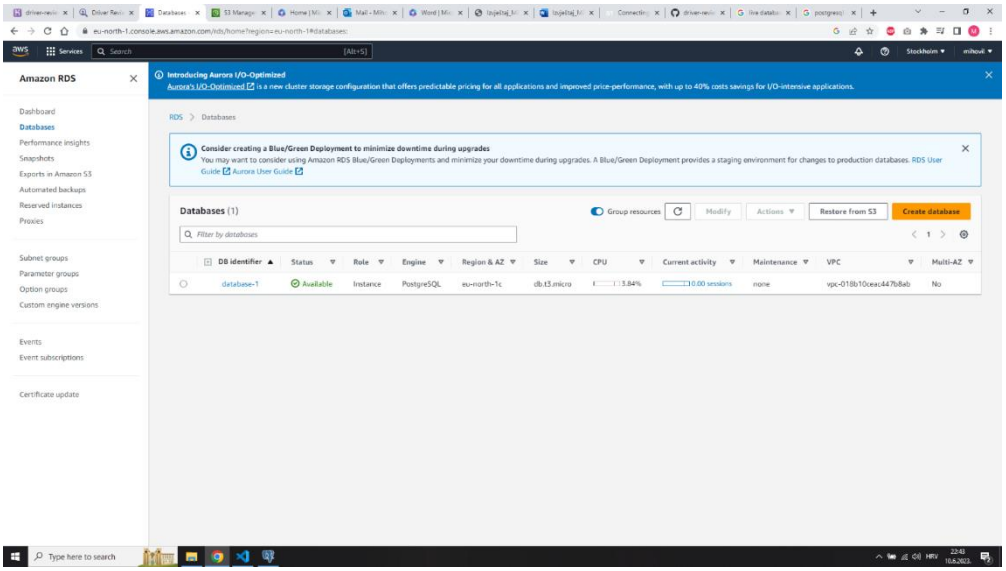
Slika 5.9. Stranica profila vozača s mogućnošću recenziranja web aplikacije za prikaz podataka o profilu i recenziranje vozača ispred vlastitog vozila

5.7. Implementacija web aplikacije na javni server

Za implementaciju javne baze podataka korišten je sustav za upravljanje bazama podataka *PostgreSQL* te alat za praćenje i upravljanje *pgAdmin 4* (slika 5.10.) i *AWS* distribuirana usluga relacijske baze podataka *Amazon RDS* (slika 5.11.).

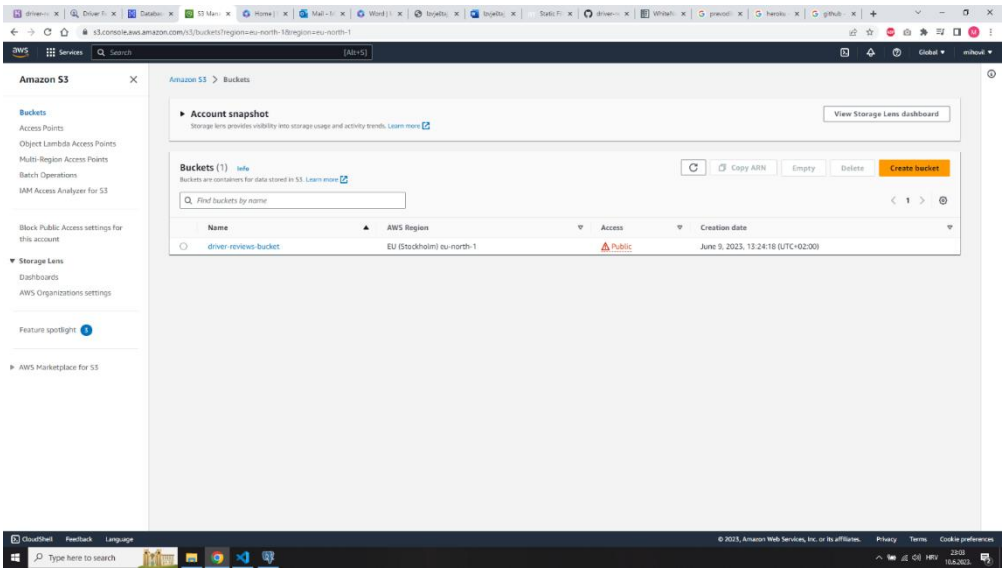


Slika 5.10. Prozor alata pgAdmin 4 za praćenje i upravljanje bazom podataka

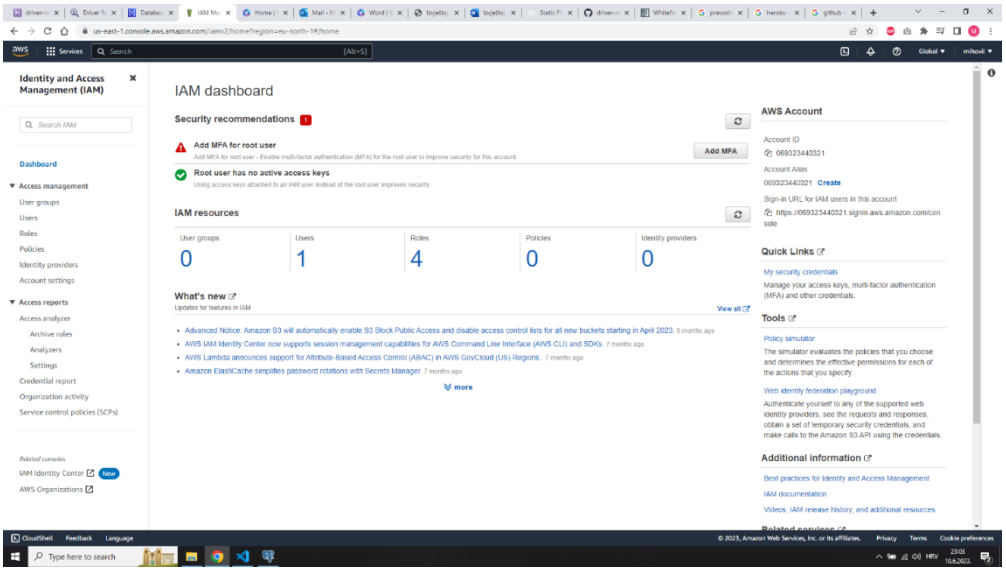


Slika 5.11. Prozor AWS distribuirane usluge relacijske baze podataka Amazon RDS

Za posluživanje statičkih objekata koji nisu vezani uz pojedinačne korisnike (*css*, *java*, slike koje nisu vezane uz pojedinačne korisnike) korišten je *middleware WhiteNoise*. Za posluživanje statičkih objekata koji jesu vezani uz pojedinačne korisnike korištena je *AWS* usluga za pohranu resursa *Amazon S3* (slika 5.12.) te *AWS* usluga za sigurno kontroliranje pristupa resursima *Amazon IAM* (slika 5.13.) i *boto3 AWS SDK* za *Python*.

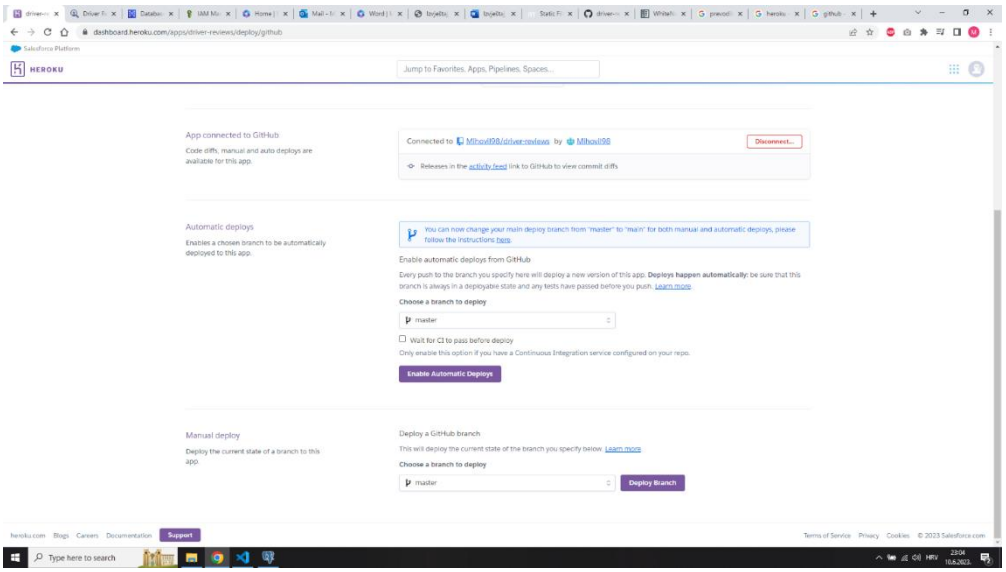


Slika 5.12. Prozor AWS usluge Amazon S3 za pohranu resursa

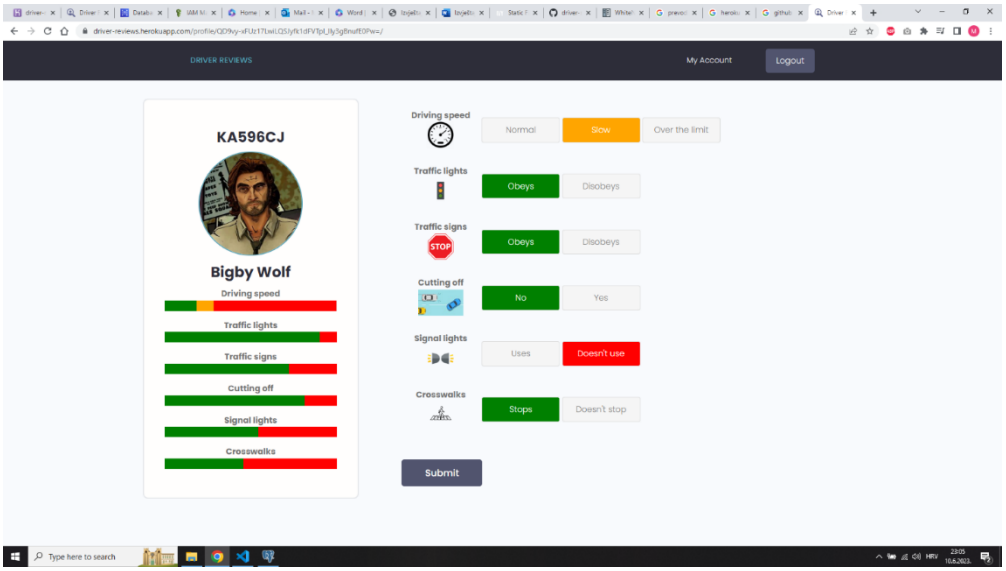


Slika 5.13. Prozor AWS usluge Amazon IAM za sigurno kontroliranje pristupa resursima

Django aplikacija učitana je na uslugu Github jer platforma Heroku omogućuje direktnu implementaciju Github repozitorija na javni server (slika 5.14.). Također je korišten *Web Server Gateway Interface Gunicorn* koji omogućava pokretanje Django aplikacije na više radničkih procesa (engl. *worker processes*), što poboljšava performanse i omogućava bolje skaliranje za višekorisničke ili opterećene web aplikacije. Pristup konačnoj javnoj web aplikaciji moguć je na poveznici https://driver-reviews.herokuapp.com (slika 5.15.).



Slika 5.14. Prozor platforme Heroku za implementaciju web aplikacije na javni server



Slika 5.15. Pristup konačnoj javnoj web aplikaciji

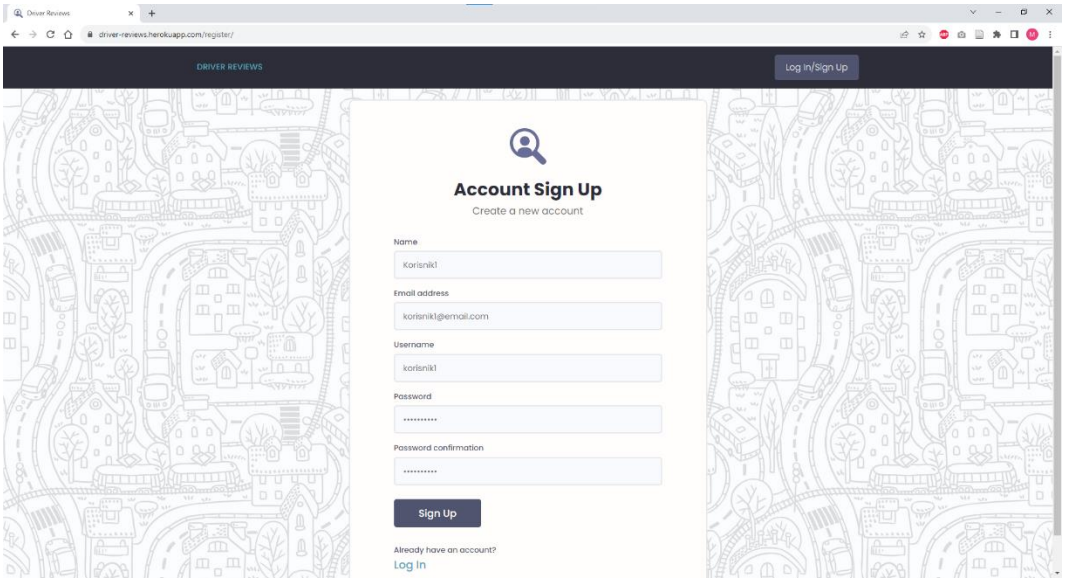
5.8. Testiranje funkcionalnosti web aplikacije

Testirat će se osnovne funkcionalnosti web aplikacije i funkcionalnosti specifične za glavni slučaj upotrebe cjelokupnog sustava ovog rada. Osnovne funkcionalnosti web aplikacije koje će biti testirane su registracija korisnika, prijava i odjava korisnika, uređivanje informacija vlastitog korisničkog računa i pretraživanje profila vozača. Funkcionalnosti web aplikacije specifične za glavni slučaj upotrebe cjelokupnog sustava ovog rada koje će biti testirane su omogućavanje recenziranja samo prijavljenim korisnicima koji koriste aplikaciju za detekciju i prepoznavanje registarskih tablica vozila, te funkcionalnost recenziranja i izračun statistike postojećih recenzija.

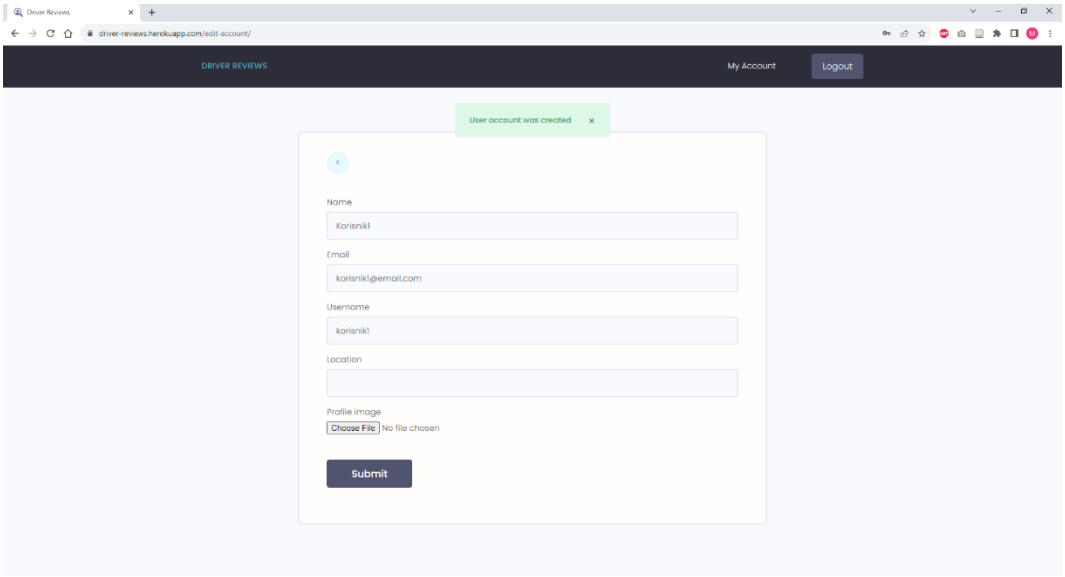
5.8.1. Testiranje registracije korisnika

U svrhu testiranja registracije korisnika provest će se ručno unošenje informacija unutar formi za registraciju na web aplikaciji koje zadovoljavaju zahtjeve registracije, odnosno e-mail adresa će biti oblika 'naziv@email.domena', te će lozinka sadržavati najmanje 8 znakova i kombinaciju slova i brojeva. Na ovaj način će se testirati registracija 5 korisnika. Također će se provesti ručno unošenje informacija unutar formi za registraciju na web aplikaciji koje ne zadovoljavaju zahtjeve registracije, odnosno e-mail adresa neće biti oblika 'naziv@email.domena', te lozinka ne će sadržavati najmanje 8 znakova i kombinaciju slova i brojeva. Na ovaj način će se također testirati registracija 5 korisnika. Ponašanje web aplikacije će se usporediti s očekivanim ponašanjem.

Na slici 5.16. prikazan je primjer unešenih informacija unutar formi za registraciju na web aplikaciji koje zadovoljavaju zahtjeve registracije. Na slici 5.17. prikazan je primjer preusmjeravanja na stranicu uređivanja korisničkog računa nakon uspješne registracije i jednokratna poruka o uspješnosti registracije.

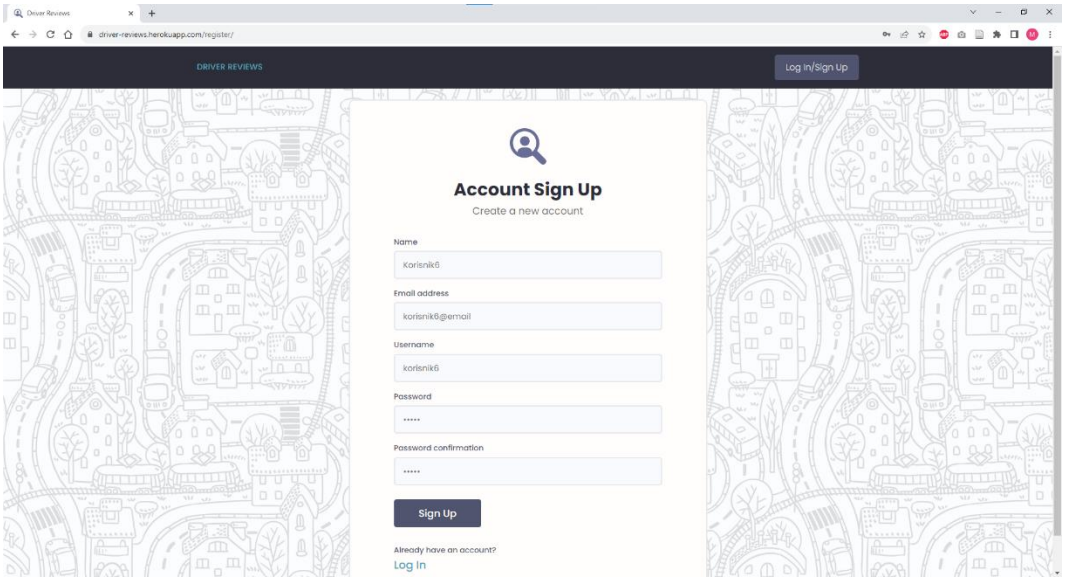


Slika 5.16. Primjer unešenih informacije unutar formi za registraciju korisnika na web aplikaciji koje zadovoljavaju zahtjeve registracije

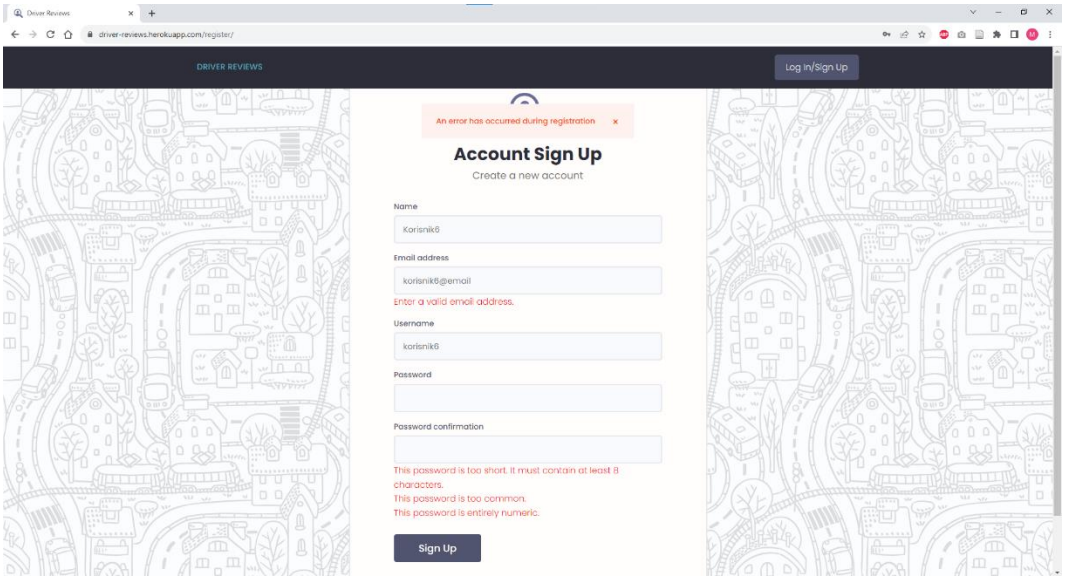


Slika 5.17. Primjer preusmjeravanja na stranicu uređivanja korisničkog računa nakon uspješne registracije i jednokratna poruka o uspješnosti registracije

Na slici 5.18. prikazan je primjer unešenih informacija unutar formi za registraciju na web aplikaciji koje ne zadovoljavaju zahtjeve registracije. Na slici 5.19. prikazan je primjer dodatnih informacija o unešenim informacijama koje nisu zadovoljile zahtjeve registracije i jednokratna obavijest o neuspješnosti registracije.



Slika 5.18. Primjer unešenih informacija unutar formi za registraciju korisnika na web aplikaciji koje ne zadovoljavaju zahtjeve registracije



Slika 5.19. Primjer dodatnih informacija o unešenim informacijama koje nisu zadovoljile zahtjeve registracije i jednokratna obavijest o neuspješnosti registracije

Tablica 5.1. prikazuje informacije unošene u forme za registraciju na web aplikaciji za svakog od 10 korisnika i ponašanja web aplikacije provjerena testiranjem. Iz priloženog je vidljivo da nema odstupanja. Očekivano ponašanje web aplikacije za unešene informacije unutar formi za registraciju korisnika na web aplikaciji koje zadovoljavaju zahtjeve registracije je uspješna registracija, preusmjeravanje na stranicu uređivanja korisničkog računa i obavijest o uspješnoj registraciji. Očekivano ponašanje web aplikacije za unešene informacije unutar formi za registraciju korisnika na web aplikaciji koje ne zadovoljavaju zahtjeve registracije je neuspješna registracija, obavijest o neuspješnoj registraciji i dodatne informacije o unešenim informacijama koje nisu zadovoljile zahtjeve registracije.

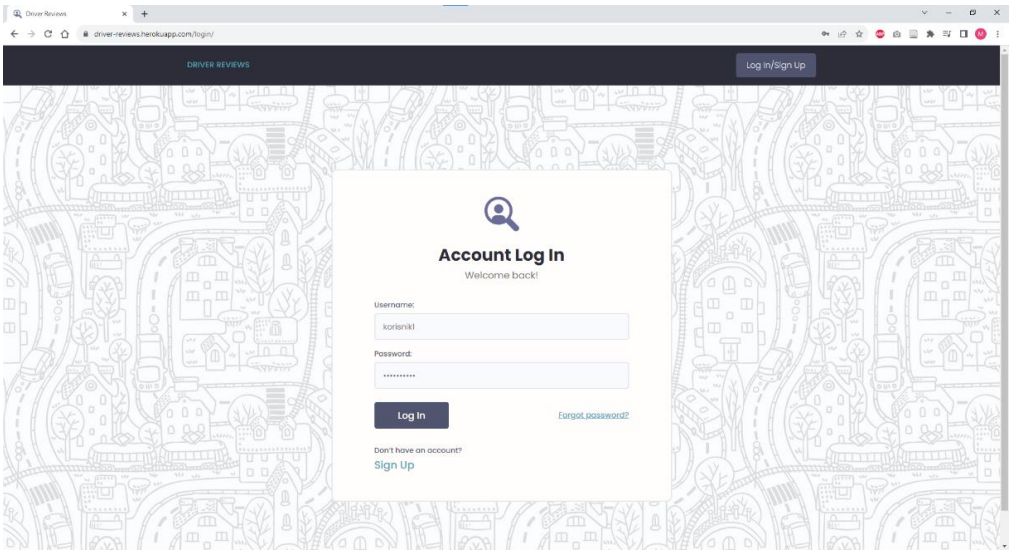
Tablica 5.1. Informacije unošene u forme za registraciju na web aplikaciji za svakog od 10 korisnika i ponašanja web aplikacije provjerena testiranjem

	Name	***Email address***	***Username***	***Password***	***Password confirmation***	**Ponašanje web aplikacije provjereno testiranjem**	**Odstupanje**
Test 1	Korisnik1	korisnik1@email.com	korisnik1	lozinka123	lozinka123	Očekivano, uspješna registracija	Nema
Test 2	Korisnik2	korisnik2@email.com	korisnik2	weffev5342	weffev5342	Očekivano, uspješna registracija	Nema
Test 3	Korisnik3	korisnik3@eemai.com	korisnik3	mfiew523	mfiew523	Očekivano, uspješna registracija	Nema
Test 4	Korisnik4	korisnik4@email.com	korisnik4	ewofn0208	ewofn0208	Očekivano, uspješna registracija	Nema
Test 5	Korisnik5	korisnik5@email.com	korisnik5	vonen445	vonen445	Očekivano, uspješna registracija	Nema
Test 6	Korisnik6	korisnik6@email	korisnik6	12345	12345	Očekivano, neuspješna registracija	Nema
Test 7	Korisnik7	korisnik7@email.com	korisnik7	123	1234	Očekivano, neuspješna registracija	Nema

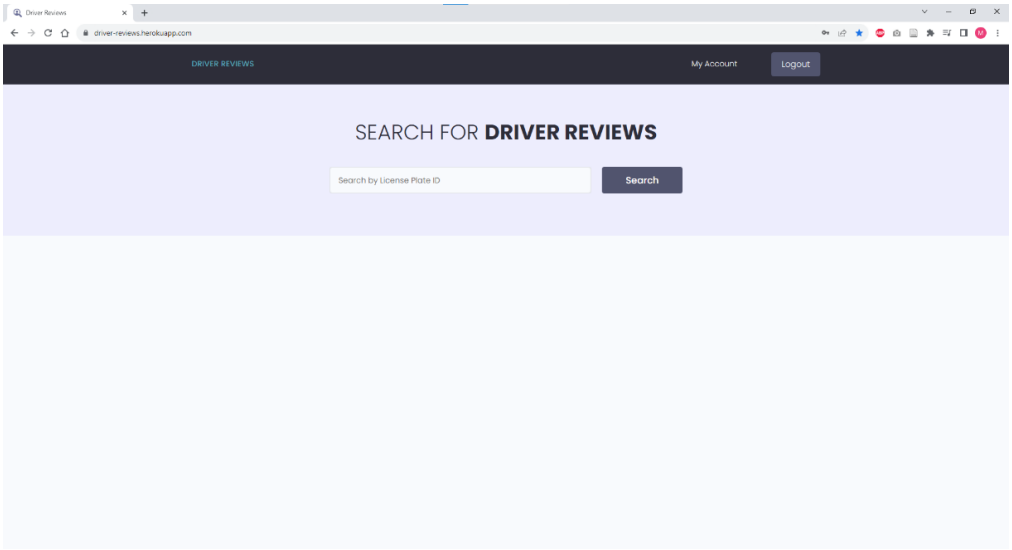
Test 8	Korisnik8	korisnik8@email	korisnik8	lozinkaa	lozinkaa	Očekivano, neuspješna registracija	Nema
Test 9	Korisnik9	korisnik9@email.com	korisnik9	rvervw	rvervw	Očekivano, neuspješna registracija	Nema
Test 10	Korisnik10	korisnik10@email	korisnik10	pwenvwn086	pwenvwn086	Očekivano, neuspješna registracija	Nema

5.8.2. Testiranje prijave i odjave korisnika

U svrhu testiranja prijave korisnika provest će se ručno unošenje informacija unutar formi za prijavu na web aplikaciji koje zadovoljavaju zahtjeve prijave, odnosno unešeno korisničko ime i lozinka će se podudarati s onima u bazi podataka. Na ovaj način će se testirati prijava 5 korisnika. Također će se provesti ručno unošenje informacija unutar formi za prijavu na web aplikaciji koje ne zadovoljavaju zahtjeve prijave, odnosno unešeno korisničko ime i lozinka ne će se podudarati s onima u bazi podataka. Na ovaj način će se također testirati prijava 5 korisnika. Ponašanje web aplikacije će se usporediti s očekivanim ponašanjem. Na slici 5.20. prikazan je primjer unešenih informacija unutar formi za prijavu na web aplikaciji koje zadovoljavaju zahtjeve prijave. Na slici 5.21. prikazan je primjer preusmjeravanja na početnu stranicu nakon uspješne prijave gdje je omogućen pristup stranici uređivanja vlastitog korisničkog računa.

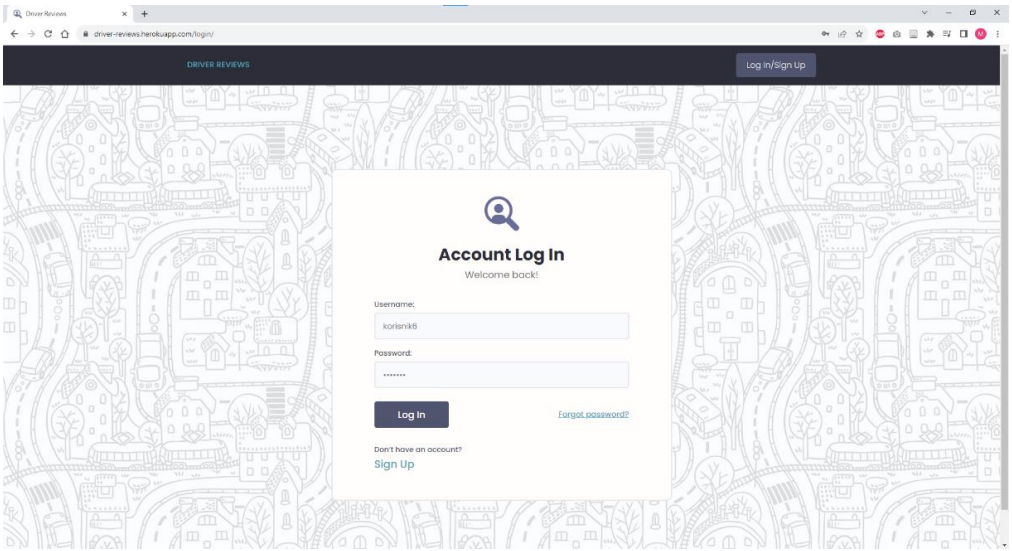


Slika 5.20. Primjer unešenih informacija unutar formi za prijavu na web aplikaciji koje zadovoljavaju zahtjeve prijave

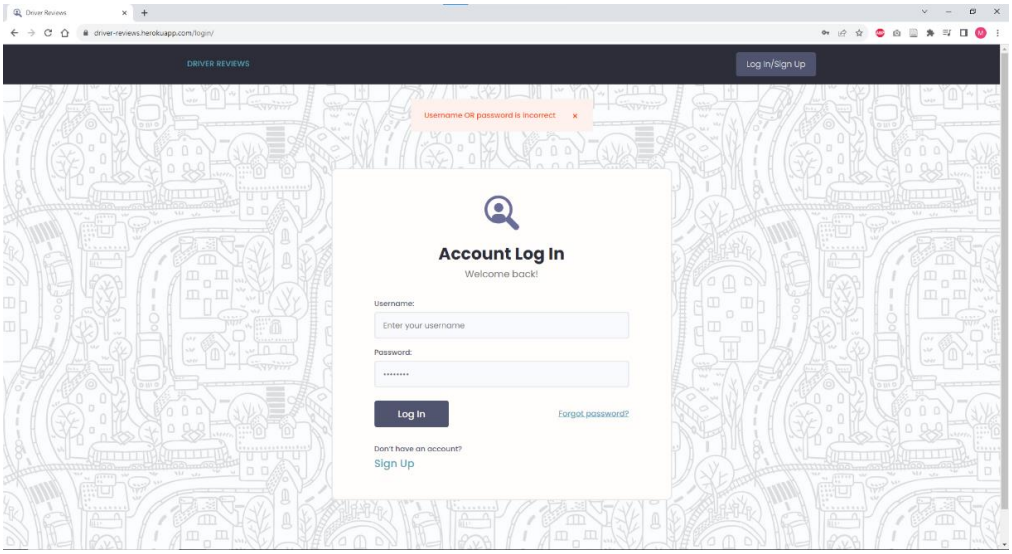


Slika 5.21. Primjer preusmjeravanja na početnu stranicu nakon uspješne prijave gdje je omogućen pristup stranici uređivanja vlastitog korisničkog računa

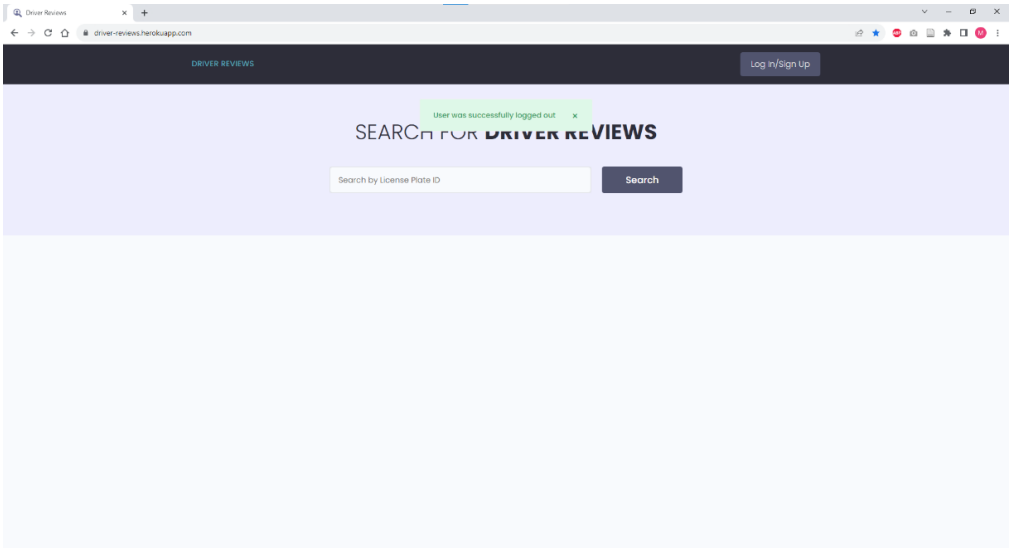
Na slici 5.22. prikazan je primjer unešenih informacija unutar formi za prijavu na web aplikaciji koje ne zadovoljavaju zahtjeve prijave. Na slici 5.23. prikazan je primjer stranice na kojoj se vidi jednokratna obavijest o neuspješnosti prijave. Na slici 5.24. prikazan je primjer preusmjeravanja na početnu stranicu nakon uspješne odjave gdje se vidi jednokratna obavijest o uspješnosti odjave.



Slika 5.22. Primjer unešenih informacija unutar formi za prijavu na web aplikaciji koje ne zadovoljavaju zahtjeve prijave



Slika 5.23. Primjer stranice na kojoj se vidi jednokratna obavijest o neuspješnosti prijave



Slika 5.24. Primjer preusmjeravanja na početnu stranicu nakon uspješne odjave gdje se vidi jednokratna obavijest o uspješnosti odjave

Tablica 5.2. prikazuje informacije unošene u forme za prijavu na web aplikaciji za svakog od 10 korisnika, očekivana ponašanja web aplikacije za prijavu i odjavu i ponašanja web aplikaciju za prijavu i odjavu provjerena testiranjem. Iz priloženog je vidljivo da nema odstupanja. Očekivano ponašanje web aplikacije za unešene informacije unutar formi za prijavu korisnika na web aplikaciji koje zadovoljavaju zahtjeve prijave je uspješna prijava, preusmjeravanje na početnu stranicu, omogućen pristup stranici uređivanja vlastitog korisničkog računa i omogućena odjava.

Očekivano ponašanje web aplikacije za unešene informacije unutar formi za prijavu korisnika na web aplikaciji koje ne zadovoljavaju zahtjeve prijave je neuspješna prijava, preusmjeravanje na istu stranicu prijave, jednokratna obavijest o neuspješnosti prijave i odjava nije omogućena.

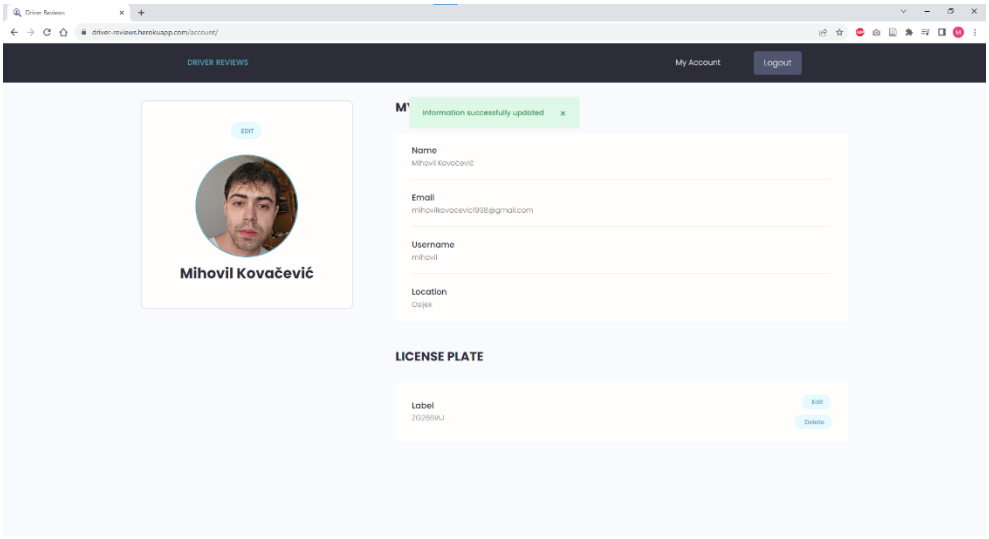
Tablica 5.2. Informacije unošene u forme za prijavu na web aplikaciji za svakog od 10 korisnika i odjavu i ponašanja web aplikacije za prijavu i odjavu provjerena testiranjem

	Username	***Password***	**Unešeni *password***	**Ponašanje web aplikacije provjereno testiranjem**	**Odstupanje**
Test 1	korisnik1	lozinka123	lozinka123	Očekivano, uspješna prijava	Nema
Test 2	korisnik2	weffev5342	weffev5342	Očekivano, uspješna prijava	Nema
Test 3	korisnik3	mfiew523	mfiew523	Očekivano, uspješna prijava	Nema
Test 4	korisnik4	ewofn0208	ewofn0208	Očekivano, uspješna prijava	Nema
Test 5	korisnik5	vonen445	vonen445	Očekivano, uspješna prijava	Nema
Test 6	korisnik6	ewvev134	lozinka	Očekivano, neuspješna prijava	Nema
Test 7	korisnik7	hnret4653	ewvnwv1	Očekivano, neuspješna prijava	Nema
Test 8	korisnik8	stbh4wr13	evbearv54	Očekivano, neuspješna prijava	Nema
Test 9	korisnik9	tbwr3542	12345	Očekivano, neuspješna prijava	Nema
Test 10	korisnik10	qtbqeea14	raebrv	Očekivano, neuspješna prijava	Nema

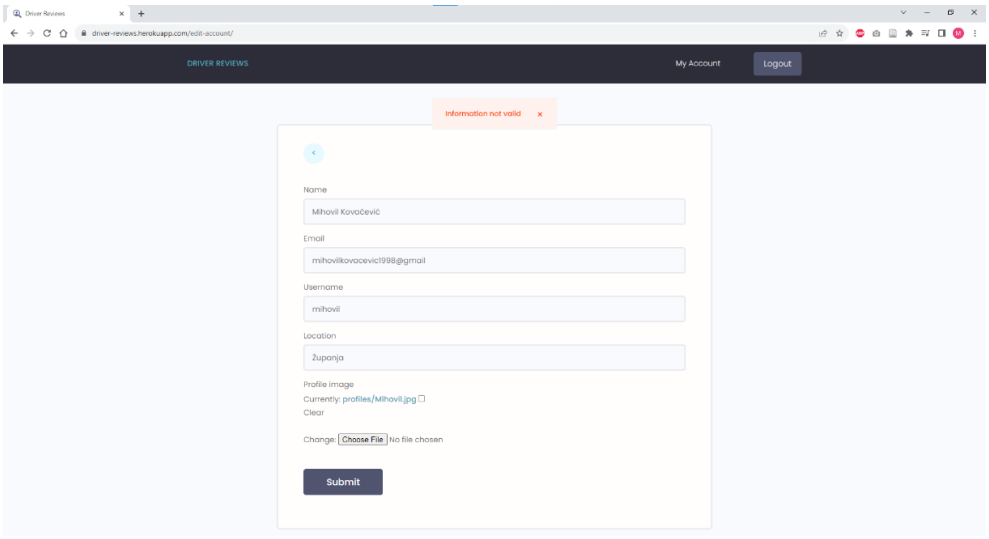
5.8.3. Testiranje uređivanja informacija vlastitog korisničkog računa

U svrhu testiranja uređivanja informacija vlastitog korisničkog računa provest će se ručno unošenje informacija unutar formi za uređivanje imena, e-mail adrese, korisničkog imena, lokacije i slike profila. Za svaku od navedenih formi napravit će se i spremiti izmjena. Uz to, testirat će se uređivanje i brisanje oznake vlastite registarske tablice vozila. Provest će se promjena oznake na oznaku koja nije dodijeljena drugom korisničkom računu, promjena oznake na oznaku koja jeste dodijeljena drugom korisničkom računu i brisanje oznake. Ponašanje web aplikacije će se

usporediti s očekivanim ponašanjem. Na slici 5.25. prikazan je primjer preusmjeravanja na stranicu vlastitog korisničkog računa nakon uspješnog uređivanja informacija vlastitog korisničkog računa gdje se vidi jednokratna obavijest o uspješnosti uređivanja informacija. Na slici 5.26. prikazan je primjer jednokratne obavijesti o neuspješnosti uređivanja informacija vlastitog korisničkog računa.

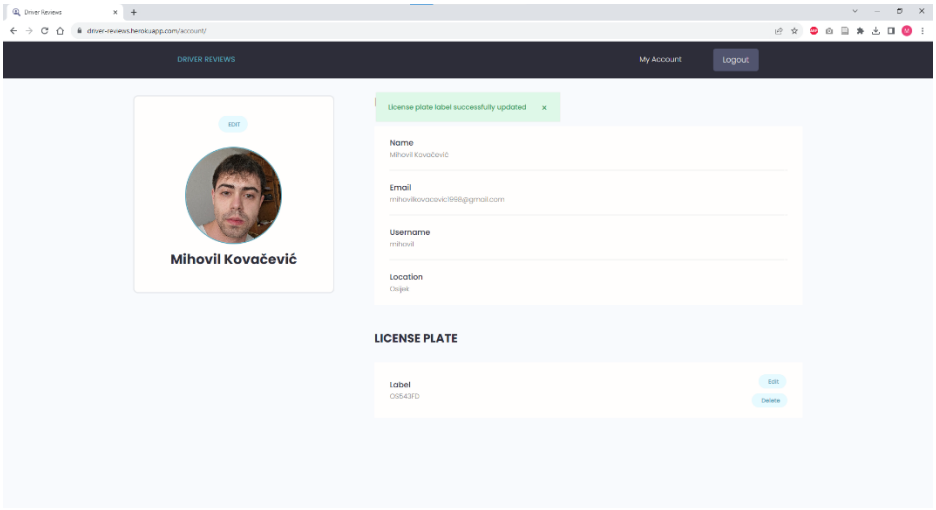


Slika 5.25. Primjer preusmjeravanja na stranicu vlastitog korisničkog računa nakon uspješnog uređivanja informacija vlastitog korisničkog računa gdje se vidi jednokratna obavijest o uspješnosti uređivanja informacija

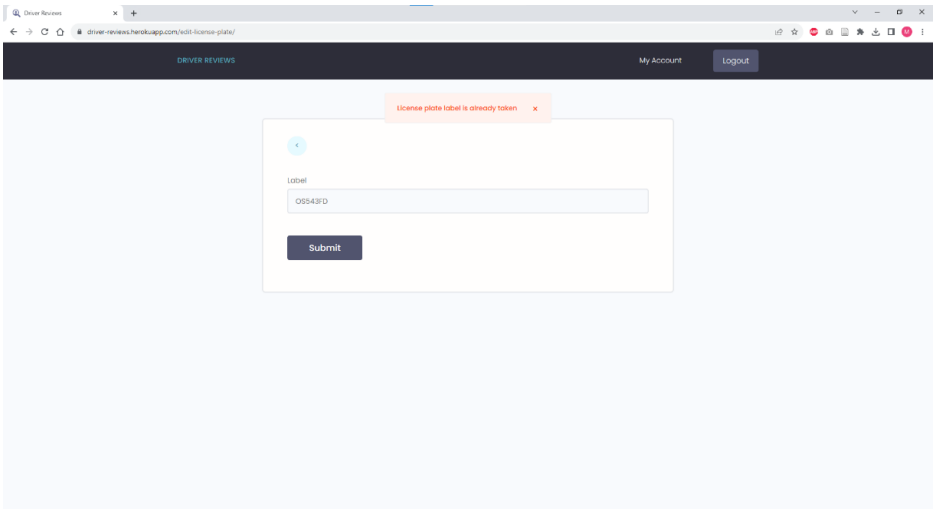


Slika 5.26. Primjer jednokratne obavijesti o neuspješnosti uređivanja informacija vlastitog korisničkog računa

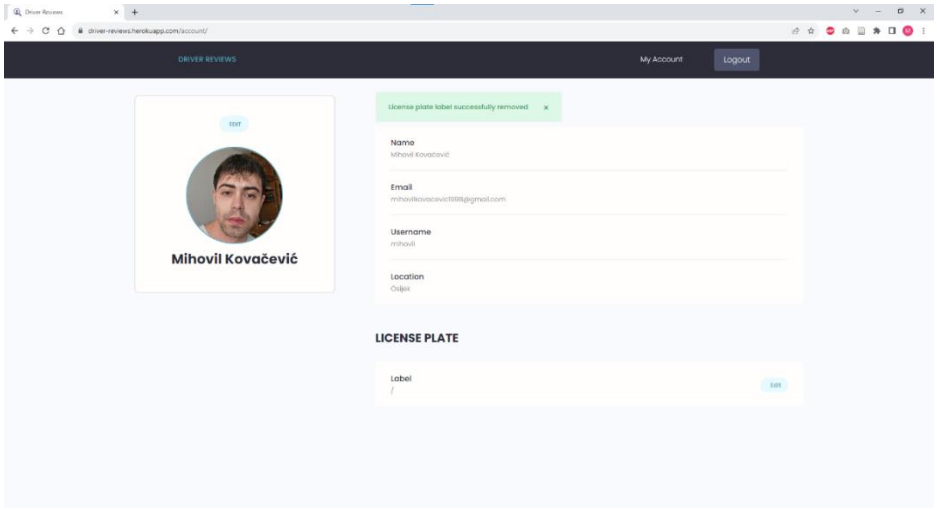
Na slici 5.27. prikazan je primjer preusmjeravanja na stranicu vlastitog korisničkog računa nakon uspješne promjene vlastite oznake registarske tablice vozila na oznaku koja nije dodijeljena drugom korisničkom računu gdje se vidi jednokratna obavijest o uspješnosti promjene oznake. Na slici 5.28. prikazan je primjer jednokratne obavijesti o neuspješnosti promjene vlastite oznake registarske tablice vozila. Na slici 5.29. prikazan je primjer jednokratne obavijesti o uspješnosti brisanja vlastite oznake registarske tablice vozila.



Slika 5.27. Primjer preusmjeravanja na stranicu vlastitog korisničkog računa nakon uspješne promjene vlastite oznake registarske tablice vozila na oznaku koja nije dodijeljena drugom korisničkom računu gdje se vidi jednokratna obavijest o uspješnosti promjene oznake



Slika 5.28. Primjer jednokratne obavijesti o neuspješnosti promjene vlastite oznake registarske tablice vozila

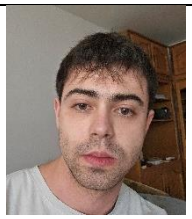


Slika 5.29. Primjer jednokratne obavijesti o uspješnosti brisanja vlastite oznake registarske tablice vozila

Tablica 5.3. prikazuje trenutne i unešene informacije u forme za uređivanje informacija vlastitog korisničkog profila i promjenu te brisanje vlastite oznake registarske tablice vozila na web aplikaciji i ponašanja web aplikacije za uređivanje informacija vlastitog korisničkog profila i promjenu te brisanje vlastite oznake registarske tablice vozila na web aplikaciji provjerena testiranjem. Iz priloženog je vidljivo da nema odstupanja. Očekivano ponašanje web aplikacije za uređivanje informacija vlastitog korisničkog profila na web aplikaciji za unešene informacije unutar odgovarajućih formi koje zadovoljavaju pripadajuće zahtjeve je uspješno uređivanje informacije, preusmjeravanje na stranicu vlastitog korisničkog računa, prikaz jednokratne obavijesti o uspješnosti uređivanja informacije. Očekivano ponašanje web aplikacije za uređivanje informacija vlastitog korisničkog profila na web aplikaciji za unešene informacije unutar odgovarajućih formi koje ne zadovoljavaju pripadajuće zahtjeve je neuspješno uređivanje informacije, prikaz jednokratne obavijesti o neuspješnosti uređivanja informacije. Očekivano ponašanje web aplikacije za promjenu vlastite oznake registarske tablice vozila na web aplikaciji za unešene informacije unutar odgovarajućih formi koje zadovoljavaju pripadajuće zahtjeve je uspješna promjena vlastite oznake registarske tablice vozila, preusmjeravanja na stranicu vlastitog korisničkog računa, prikaz jednokratne obavijesti o uspješnosti promjene vlastite oznake registarske tablice vozila. Očekivano ponašanje web aplikacije za promjenu vlastite oznake registarske tablice vozila na web aplikaciji za unešene informacije unutar odgovarajućih formi koje ne zadovoljavaju pripadajuće zahtjeve je neuspješna promjena vlastite oznake registarske

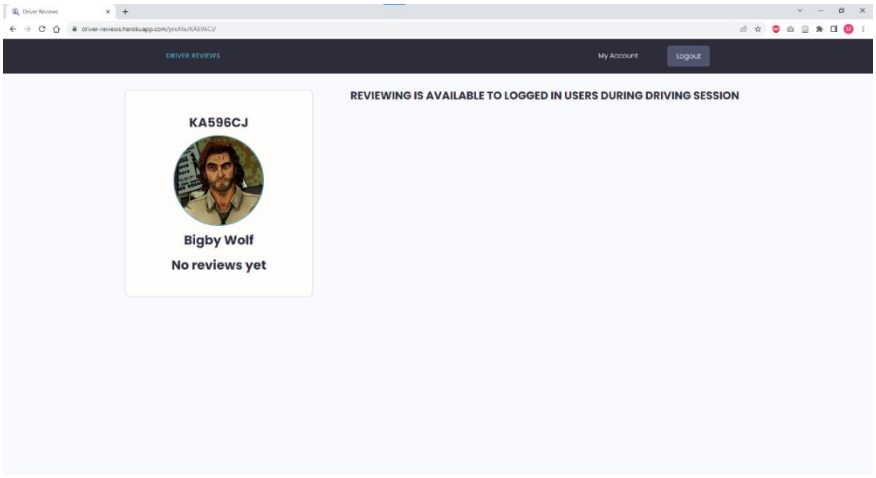
tablice vozila, obavijesti o neuspješnosti promjene vlastite oznake registarske tablice vozila. Očekivano ponašanje web aplikacije za brisanje vlastite oznake registarske tablice vozila na web aplikaciji je uspješno brisanje vlastite oznake registarske tablice vozila, preusmjeravanja na stranicu vlastitog korisničkog računa, prikaz jednokratne obavijesti o uspješnsti brisanja vlastite oznake registarske tablice vozila.

Tablica 5.3. Trenutne i unešene informacije u forme za uređivanje informacija vlastitog korisničkog profila i promjenu te brisanje vlastite oznake registarske tablice vozila na web aplikaciji i ponašanja web aplikacije za uređivanje informacija vlastitog korisničkog profila i promjenu te brisanje vlastite oznake registarske tablice vozila na web aplikaciji provjerena testiranjem

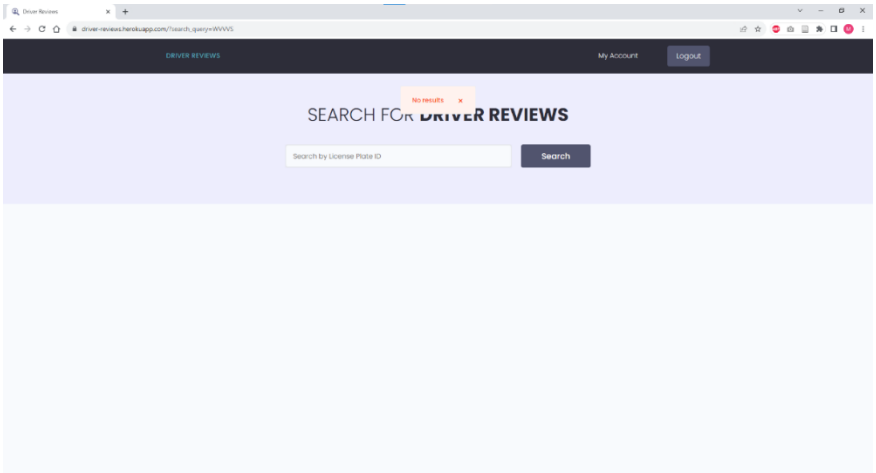
	Trenutna informacija	**Unešena informacija**	**Ponašanje web aplikacije provjereno testiranjem**	**Odstupanje**
Name	Mihovil	Mihovil Kovačević	Očekivano, uspješno uređivanje	Nema
Email	mihovilkovacevic1998@gmail.com	mihovil.kovacevic@tttech-auto.com	Očekivano, uspješno uređivanje	Nema
Email	mihovil.kovacevic@tttech-auto.com	mihovilkovacevic1998@gmail	Očekivano, neuspješno uređivanje	Nema
Username	mihovil	mihovil1998	Očekivano, uspješno uređivanje	Nema
Location	Županja	Osijek	Očekivano, uspješno uređivanje	Nema
Profile image			Očekivano, uspješno uređivanje	Nema
License plate	ZG2661AJ	OS543FD	Očekivano, uspješno uređivanje	Nema
License plate	OS543FD	KA596CJ	Očekivano, neuspješno uređivanje zbog već dodijeljene oznake registarske tablice vozila	Nema
License plate	OS543FD	*delete*	Očekivano, uspješno brisanje	Nema

5.8.4. Testiranje pretraživanja profila vozača

U svrhu testiranja pretraživanja profila vozača provest će se ručno unošenje oznaka registarskih tablica vozila u pretraživač koji se nalazi na početnoj stranici. Testirat će se pretraživanje 5 oznaka registarskih tablica vozila koje su spremljene u bazi podataka s kojom komunicira web aplikacija i pretraživanje 5 oznaka registarskih tablica vozila koje nisu spremljene u bazi podataka s kojom komunicira web aplikacija. Ponašanje web aplikacije će se usporediti s očekivanim ponašanjem. Na slici 5.30. prikazan je primjer preusmjeravanja na stranicu profila vozača nakon uspješnog pretraživanja koristeći oznaku registarske tablice vozila. Na slici 5.31. prikazan je primjer jednokratne obavijesti o neuspješnosti pretraživanja profila vozača.



Slika 5.30. Primjer preusmjeravanja na stranicu profila vozača nakon uspješnog pretraživanja koristeći oznaku registarske tablice vozila



Slika 5.31. Primjer jednokratne obavijesti o neuspješnosti pretraživanja profila vozača

Tablica 5.4. prikazuje unešene oznake registarskih tablica vozila u pretraživač za profile vozača na web aplikaciji, prisutstvo tih oznaka u bazi podataka povezanom s web aplikacijom, očekivana ponašanja web aplikacije za pretraživanje profila vozača za različite oznake registarskih tablica vozila i ponašanja web aplikacije za pretraživanje profila vozača za različite oznake registarskih tablica vozila provjerena testiranjem. Iz priloženog je vidljivo da nema odstupanja. Očekivano ponašanje web aplikacije za unešene oznake registarskih tablica vozila u pretraživač za profile vozača na web aplikaciji uz prisutstvo tih oznaka registarskih tablica vozila u bazi podataka povezanoj s web aplikacijom je uspješno pretraživanje profila vozača i preusmjeravanja na stranicu profila vozača. Očekivano ponašanje web aplikacije za unešene oznake registarskih tablica vozila u pretraživač za profile vozača na web aplikaciji bez prisutstva tih oznaka registarskih tablica vozila u bazi podataka povezanoj s web aplikacijom je neuspješno pretraživanje profila vozača i prikaz jednokratne obavijesti o neuspješnosti pretraživanja profila vozača.

Tablica 5.4. Unešene oznake registarskih tablica vozila u pretraživač za profile vozača na web aplikaciji, prisutstvo tih oznaka u bazi podataka povezanom s web aplikacijom i ponašanja web aplikacije za pretraživanje profila vozača za različite oznake registarskih tablica vozila provjerena testiranjem

	Unešena oznaka	**Prisutstvo oznake registarske tablice vozila u bazi podataka povezanoj s web aplikacijom**	**Ponašanje web aplikacije provjereno testiranjem**	**Odstupanje**
Test 1	ZG2661AJ	Da	Očekivano, uspješno pretraživanje	Nema
Test 2	OS543FD	Da	Očekivano, uspješno pretraživanje	Nema
Test 3	KA596CJ	Da	Očekivano, uspješno pretraživanje	Nema
Test 4	ZG2808AA	Da	Očekivano, uspješno pretraživanje	Nema
Test 5	ZG5715AK	Da	Očekivano, uspješno pretraživanje	Nema
Test 6	OS672MD	Ne	Očekivano, neuspješno pretraživanje	Nema
Test 7	ZG234HS	Ne	Očekivano, neuspješno pretraživanje	Nema

Test 8	ZG623JZ	Ne	Očekivano, neuspješno pretraživanje	Nema
Test 9	OS5532HP	Ne	Očekivano, neuspješno pretraživanje	Nema
Test 10	OS231MS	Ne	Očekivano, neuspješno pretraživanje	Nema

5.8.5. Testiranje omogućavanja recenziranja samo prijavljenim korisnicima koji koriste aplikaciju za detekciju i prepoznavanje registarskih tablica vozila

U svrhu testiranja omogućavanja recenziranja samo prijavljenim korisnicima koji koriste aplikaciju za detekciju i prepoznavanje registarskih tablica vozila, 5 puta će se ručno pristupiti istom profilu vozača pomoću pretraživača te nakon toga 5 puta pomoću aplikacije za detekciju i prepoznavanje registarskih tablica vozila koristeći 5 slika iste registarske tablice vozila na različitim udaljenostima. Ponašanje web aplikacije će se usporediti s očekivanim ponašanjem. Za testiranje ponašanja web aplikacije za pristup profilu vozača pomoću aplikacije za detekciju i prepoznavanje registarskih tablica vozila koristit će se vlastita skripta. Skripta simulira rad sustava u stvarnom vremenu na temelju 5 ranije spomenutih slika (slika 5.32., slika 5.33., slika 5.34., slika 5.35., slika 5.36.). Na slici 5.37. prikazan je *Python* kod spomenute skripte.

Slika 5.32. Prva slika korištena za testiranje ponašanja web aplikacije za pristup profilu vozača pomoću aplikacije za detekciju i prepoznavanje registarskih tablica vozila

Slika 5.33. Druga slika korištena za testiranje ponašanja web aplikacije za pristup profilu vozača pomoću aplikacije za detekciju i prepoznavanje registarskih tablica vozila

Slika 5.34. Treća slika korištena za testiranje ponašanja web aplikacije za pristup profilu vozača pomoću aplikacije za detekciju i prepoznavanje registarskih tablica vozila

Slika 5.35. Četvrta slika korištena za testiranje ponašanja web aplikacije za pristup profilu vozača pomoću aplikacije za detekciju i prepoznavanje registarskih tablica vozila

Slika 5.36. Peta slika korištena za testiranje ponašanja web aplikacije za pristup profilu vozača pomoću aplikacije za detekciju i prepoznavanje registarskih tablica vozila

test_system.py

```python
from algorithm.object_detector import YOLOv7
from utils.detections import draw
import cv2
import webbrowser
import requests
from cryptography.fernet import Fernet
import time

yolov7 = YOLOv7()
yolov7.load('yolov5_small.weights', classes='classes.yaml', device='cpu')
yolov7.set(ocr_classes=['license plate'])
images = ['test1.png', 'test2.png', 'test3.png', 'test4.png', 'test5.png']
previous_text = ''
payload = dict(key = '', encrypted = '')

for image in images:
    image_str = image
    image = cv2.imread(image)
    detections = yolov7.detect(image)
    detected_image = draw(image, detections)
    text = ''
    for detection in detections:
        text = detection['text']
    if text == '':
        print(image_str + " skipped because the label was not recognized\n")
        continue
    elif text != '' and text != previous_text:
        url = "https://driver-reviews.herokuapp.com/delete-session/"
        response = requests.post(url, data=payload)

        url = "https://driver-reviews.herokuapp.com/set-session/"
        data = text
        key = Fernet.generate_key()
        fernet = Fernet(key)
        encrypted = fernet.encrypt(data.encode())
        payload = dict(key = key, encrypted = encrypted)
        response = requests.post(url, data=payload)
        if response.status_code == 200:
            url = 'https://driver-reviews.herokuapp.com/profile/' + key.decode('utf-8') + '/'
            webbrowser.get('windows-default').open(url)
            print("Opened review page for " + image_str + " - " + text + "\n")
            previous_text = text
        else:
            print("Request failed with status code:", response.status_code)
    elif text == previous_text:
        print(image_str + " skipped because the label is the same as the previous one - " + text + "\n")

time.sleep(5)

url = "https://driver-reviews.herokuapp.com/delete-session/"
response = requests.post(url, data=payload)
```

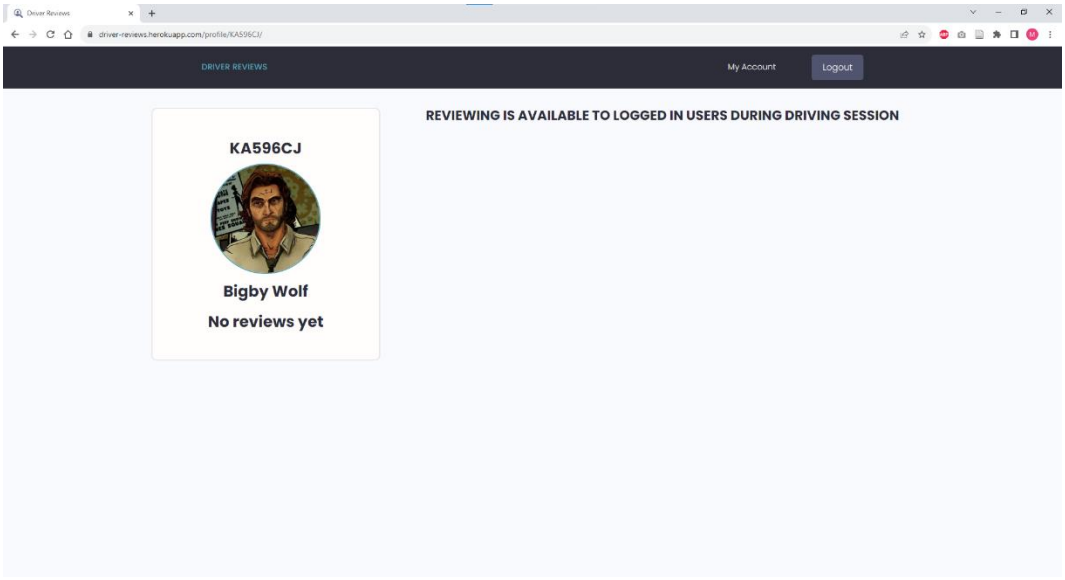
Slika 5.37. Skripta za testiranje ponašanja web aplikacije za pristup profilu vozača pomoću aplikacije za detekciju i prepoznavanje registarskih tablica vozila

Na slici 5.38. prikazan je rezultat poziva skripte sa slike 5.32. koja se izvršila za pet prethodno navedenih slika za testiranje (slika 5.33., slika 5.34., slika 5.35., slika 5.36., slika 5.37.). Vidljivo je da je za sliku 5.33. registarska tablica vozila detektirana, ali prepoznat je samo prazan *string*, za sliku 5.34. prepoznata je oznaka registarske tablice vozila s jednim netočnim znakom, a za sliku 5.35., sliku 5.36. i sliku 5.37. točno je prepoznata oznaka registarske tablice vozila.

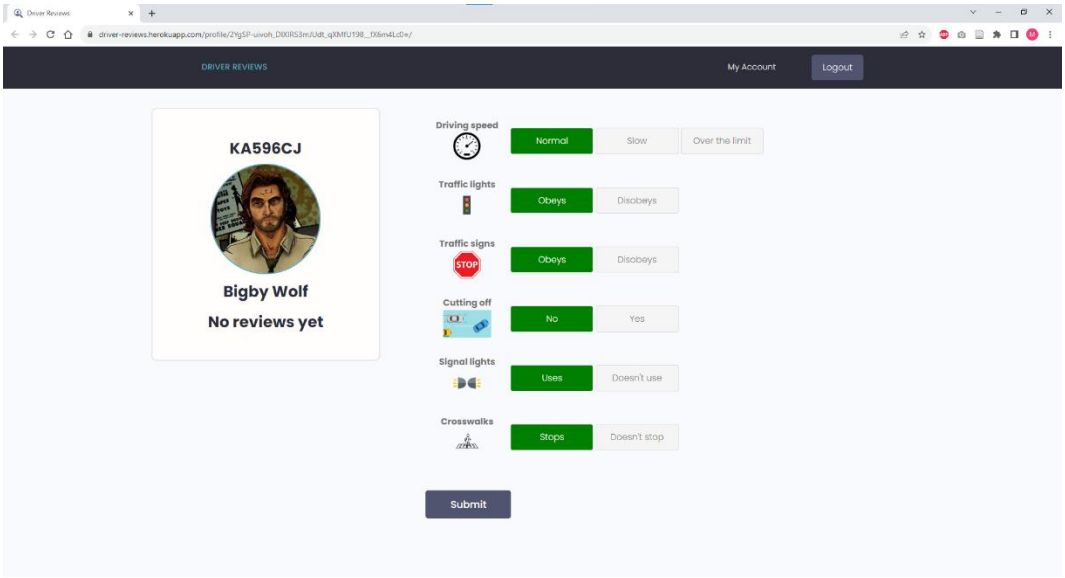
```
(yoloenv) PS C:\Users\Mihovil\Desktop\easy-yolov7>  c:; cd 'c:\Users\Mihovil\Desktop\easy-yolov7'; & 'c:\Users\Mihovil\Desktop\easy-yolov7\yoloenv\Scripts\python.exe'
Files\lib\python\debugpy\adapter/../..\debugpy\launcher' '54118' '--' 'c:\Users\Mihovil\Desktop\easy-yolov7\test_system.py'
test1.png skipped because the label was not recognized

Opened review page for test2.png - ZGORHARIO

Opened review page for test3.png - ZGORMARIO

test4.png skipped because the label is the same as the previous one - ZGORMARIO

test5.png skipped because the label is the same as the previous one - ZGORMARIO
```

Slika 5.38. Rezultat poziva skripte za testiranje ponašanja web aplikacije za pristup profilu vozača pomoću aplikacije za detekciju i prepoznavanje registarskih tablica vozila za pet prethodno navedenih slika za testiranje

Na slici 5.39. prikazan je primjer pristupa profilu vozača pomoću pretraživača pri čemu je onemogućeno recenziranje. Na slici 5.40. prikazan je primjer pristupa profilu vozača pomoću aplikacije za detekciju i prepoznavanje registarskih tablica vozila pri čemu je omogućeno recenziranje.



Slika 5.39. Primjer pristupa profilu vozača pomoću pretraživača pri čemu je onemogućeno recenziranje



Slika 5.40. Primjer pristupa profilu vozača pomoću aplikacije za detekciju i prepoznavanje registarskih tablica vozila pri čemu je omogućeno recenziranje

Tablica 5.5. prikazuje referentne oznake registarskih tablica vozila, načine pristupa profilu vozača, prepoznate oznake registarskih tablica vozila i ponašanja web aplikacije provjerena testiranjem. Očekivano ponašanje web aplikacije za pristup profilu vozača koristeći pretraživač je preusmjeravanje na odgovarajuću stranicu profila vozača uz onemogućeno recenziranje. Očekivano ponašanje web aplikacije za pristup profilu vozača koristeći aplikaciju za detekciju i prepoznavanje registarskih tablica vozila kada je oznaka registarske tablice vozila prepoznata je preusmjeravanje na odgovarajuću stranicu profila vozača uz omogućeno recenziranje. Očekivano ponašanje web aplikacije za pristup profilu vozača koristeći aplikaciju za detekciju i prepoznavanje registarskih tablica vozila kada je registarska tablica vozila detektirana ali je prepoznat prazan *string* je mirovanje, odnosno ne otvara se stranica profila vozača jer registarska tablica vozila nije prepoznata. Prvo moguće odstupanje je kada aplikacija za detekciju i prepoznavanje registarskih tablica vozila pošalje netočnu oznaku registarske tablice vozila, pa se otvori stranica profila netočnog vozača. Drugo moguće odstupanje je kada se zbog prethodno netočno prepoznate oznake registarske tablice vozila te u idućem slučaju točno prepoznate, otvara novi prozor za stranicu profila vozača. Ukoliko je u oba slučaja oznaka registarske tablice vozila bila točno prepoznata, bio bi otvoren samo jedan prozor za stranicu profila vozača, a ne dva.

Tablica 5.5. Referentne oznake registarskih tablica vozila, načine pristupa profilu vozača, prepoznate oznake registarskih tablica vozila i ponašanja web aplikacije provjerena testiranjem

	Referentna oznaka registarske tablice vozila	**Način pristupa profilu vozača**	**Prepoznata oznaka registarske tablice vozila**	**Ponašanje web aplikacije provjereno testiranjem**	**Odstupanje**
Test 1	KA596CJ	Pretraživač	/	Očekivano	Nema
Test 2	ZG2661AJ	Pretraživač	/	Očekivano	Nema
Test 3	OS543FD	Pretraživač	/	Očekivano	Nema
Test 4	ZG2808AA	Pretraživač	/	Očekivano	Nema
Test 5	ZG5715AK	Pretraživač	/	Očekivano	Nema
Test 6	ZGORMARIO	Aplikacija za detekciju i prepoznavanje registarskih tablica vozila		Očekivano	Nema
Test 7	ZGORMARIO	Aplikacija za detekciju i prepoznavanje registarskih tablica vozila	ZGORHARIO	Neočekivano	Prvo moguće odstupanje
Test 8	ZGORMARIO	Aplikacija za detekciju i prepoznavanje registarskih tablica vozila	ZGORMARIO	Neočekivano	Drugo moguće odstupanje
Test 9	ZGORMARIO	Aplikacija za detekciju i prepoznavanje registarskih tablica vozila	ZGORMARIO	Očekivano	Nema
Test 10	ZGORMARIO	Aplikacija za detekciju i prepoznavanje registarskih tablica vozila	ZGORMARIO	Očekivano	Nema

Za slučajeve netočnog prepoznavanja registarske tablice vozila ponašanje web aplikacije bi se moglo unaprijediti implementacijom trajanja sesije za svaku pojedinačnu stranicu koja omogućuje recenziranje, te omogućiti korisniku da odbije stranice za recenziranje koje su otvorene na temelju netočnog prepoznavanja registarske tablice vozila. Za ostvarenje ovog rješenja morao

bi se koristiti kod koji se izvodi paralelno. Trenutni sustav omogućava funkcionalnost trajanja sesija ali samo sekvencijalno, što znači da primjerice iduća sesija mora započeti kako bi se prethodna završila ili iduća sesija mora čekati završetak prethodne. Funkcionalnost koja uz pomoć kružnog spremnika veličine 10 oznaka registarskih tablica vozila otvara stranicu za recenziranje samo ako je od 10 zadnjih uzastopnih sličica barem 5 ili više puta prepoznata ista oznaka registarske tablice vozila bi također bila jedno od rješenja za problem netočnog prepoznavanja registarske tablice vozila.

5.8.6. Testiranje funkcionalnosti recenziranja i izračuna statistike postojećih recenzija

Testiranje funkcionalnosti recenziranja i izračuna statistike postojećih recenzija izvršit će se ručno. Jedan profil vozača će biti ocijenjen od strane deset korisničkih računa. Očekivana statistika će biti ručno izračunata i uspoređena s onom koja je prikazana na web aplikaciji. Kao što je ranije spomenuto, statistika na web aplikaciji se prikazuje tako da se izračuna postotni udio svake od mogućih opcija recenziranja za svaku kategoriju te se pomoću *for* petlje iscrtava svaki udio kao više *div* elemenata širine 1%. Točnost statistike za svaku kategoriju je moguće provjeriti tako da se dohvati *html* kod za svaki od iscrtanih *div* elemenata *html* koda i pomoću *Python* skripte vidljive na slici 5.41. zbroji koliko je kojih *div* elemenata *html* koda.

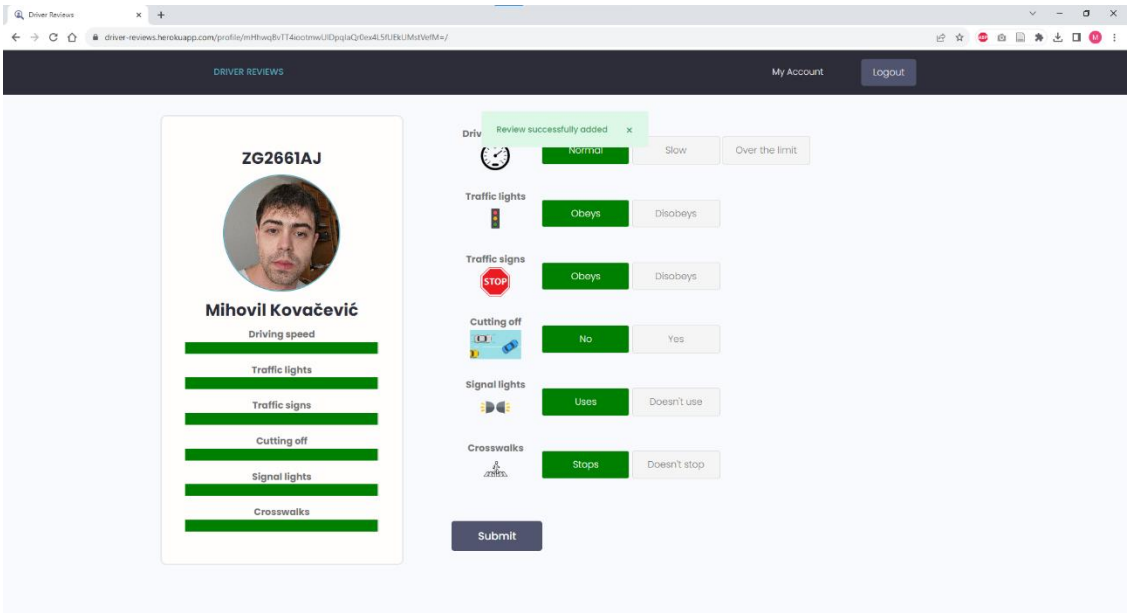
calculate_reviews.py

C: > Users > Mihovil > Desktop > calculate_reviews.py > ...

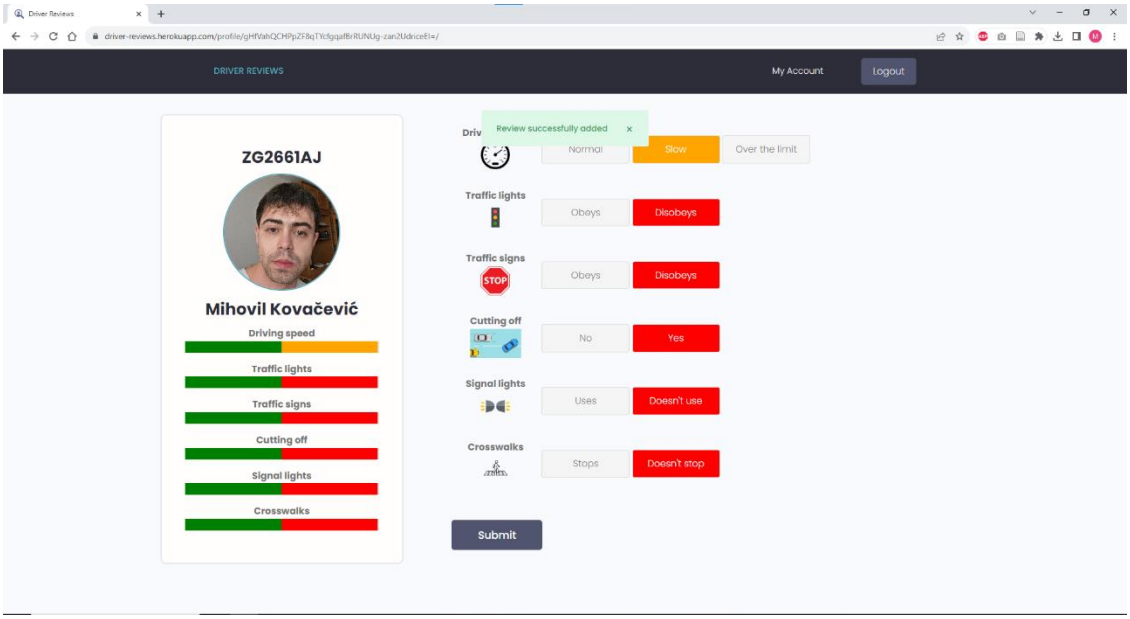
```
from bs4 import BeautifulSoup

html = '''

'''

soup = BeautifulSoup(html, 'html.parser')
outer_div = soup.find('div', class_='bar')

class_counts = {}

div_elements = outer_div.find_all('div')
for div in div_elements:
    class_list = div.get('class', [])
    for class_name in class_list:
        if class_name in class_counts:
            class_counts[class_name] += 1
        else:
            class_counts[class_name] = 1

for class_name, count in class_counts.items():
    print(f"{class_name} = {count}")
```

Slika 5.41. Skripta za računanje broja pojavljivanja svakog pojedinačnog div elementa html koda

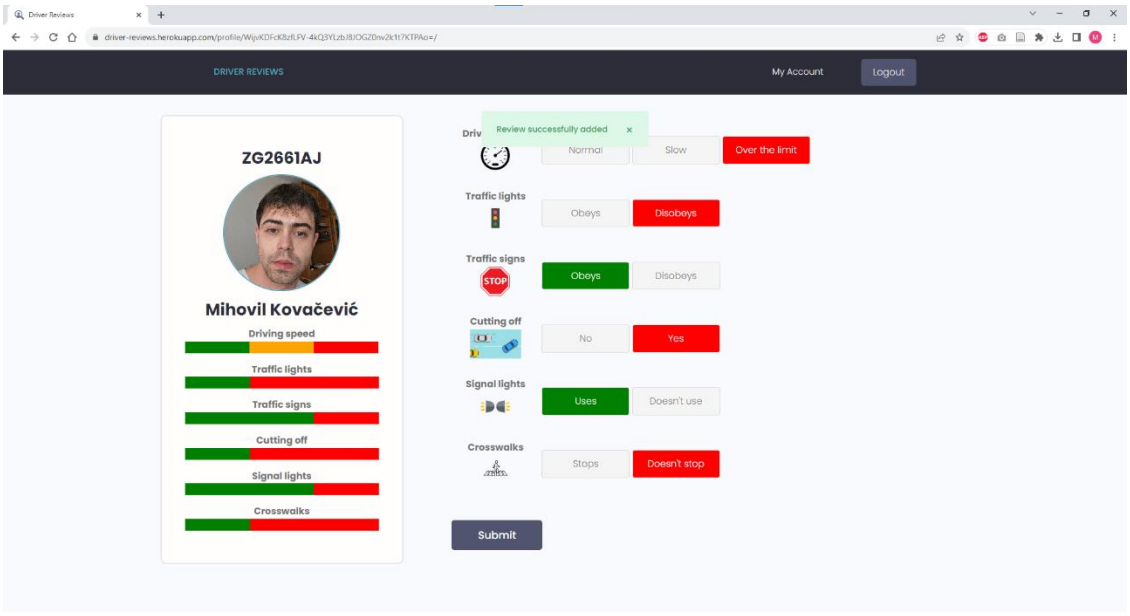
Na slikama 5.42., 5.43., 5.44., 5.45., 5.46., 5.47., 5.48., 5.49., 5.50., 5.51. prikazana je promjena iscrtane statistike recenzija nakon dodavanje recenzije svakog od deset korisničkih računa korištenih za ovo testiranje.



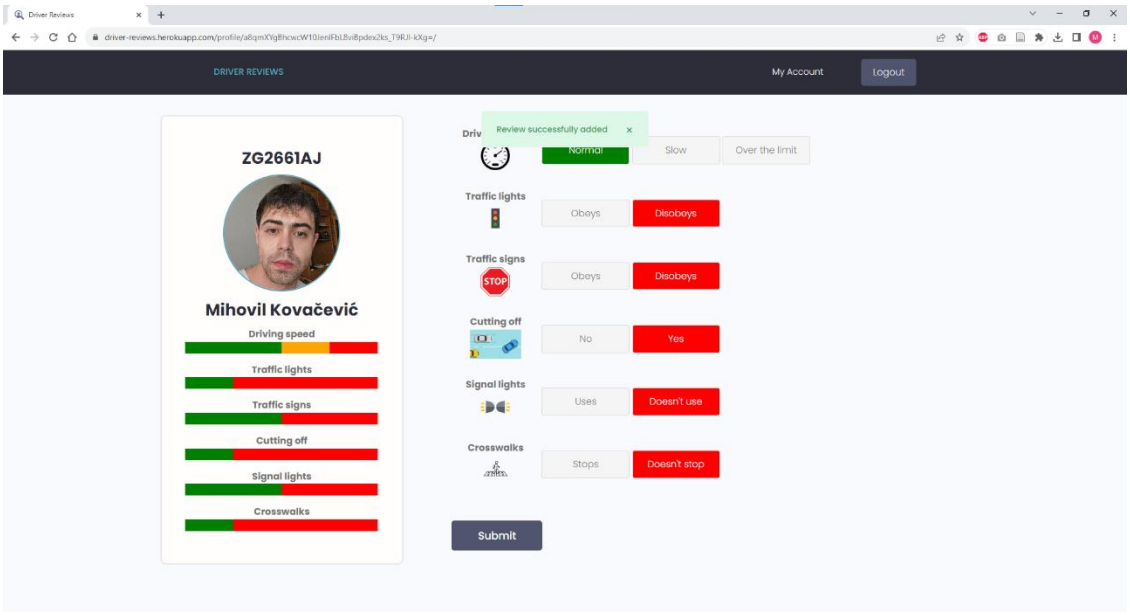
Slika 5.42. Prikaz iscrtane statistike recenzija nakon dodavanja recenzije prvog korisničkog računa



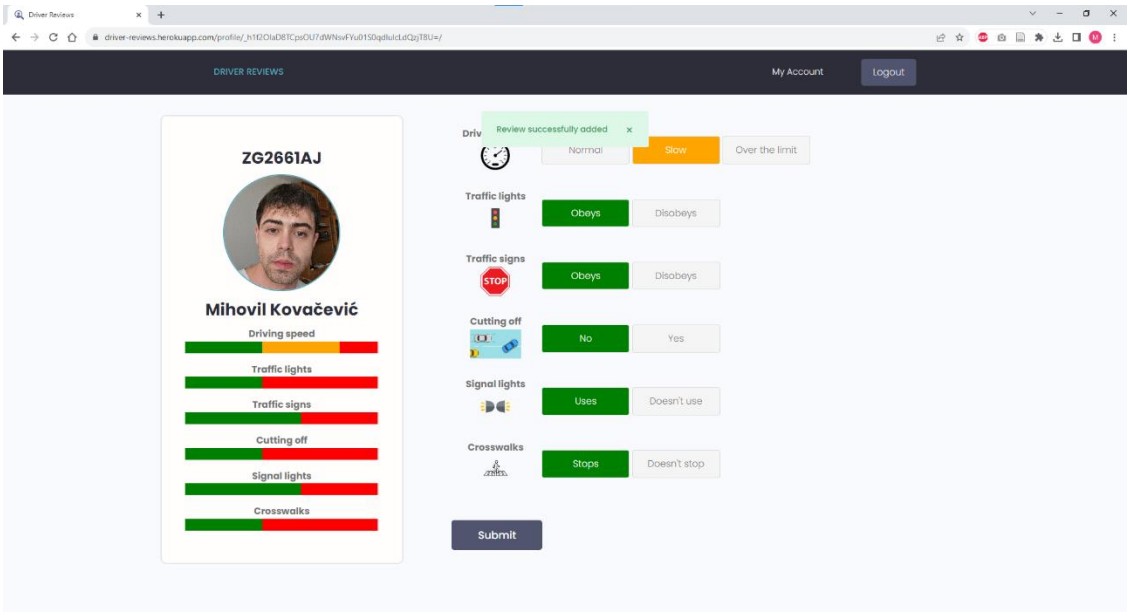
Slika 5.43. Prikaz iscrtane statistike recenzija nakon dodavanja recenzije drugog korisničkog računa



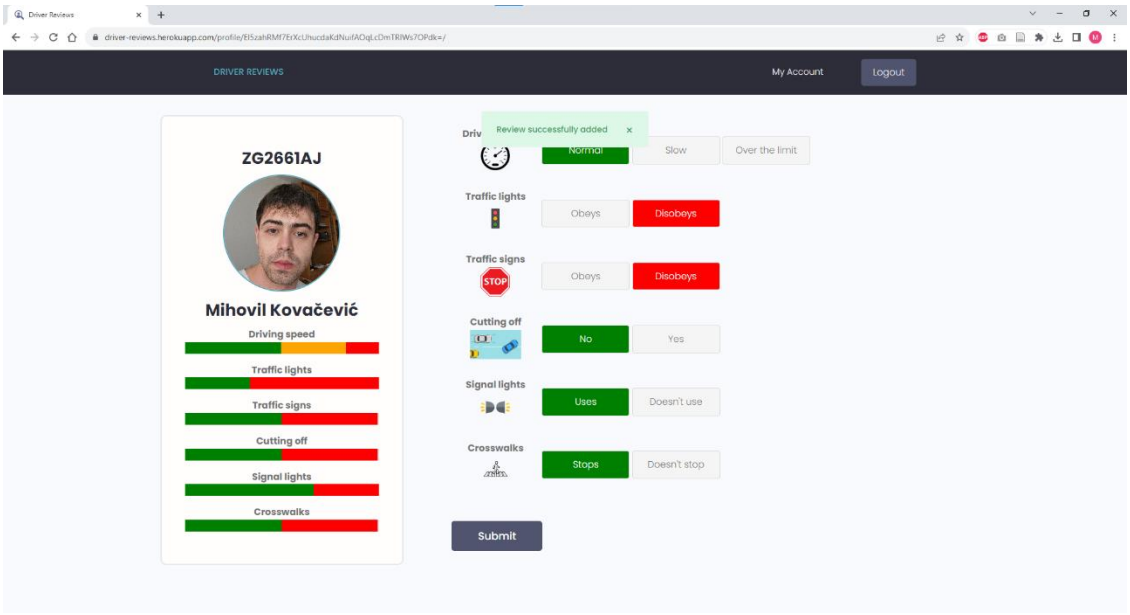
Slika 5.44. Prikaz iscrtane statistike recenzija nakon dodavanja recenzije trećeg korisničkog računa



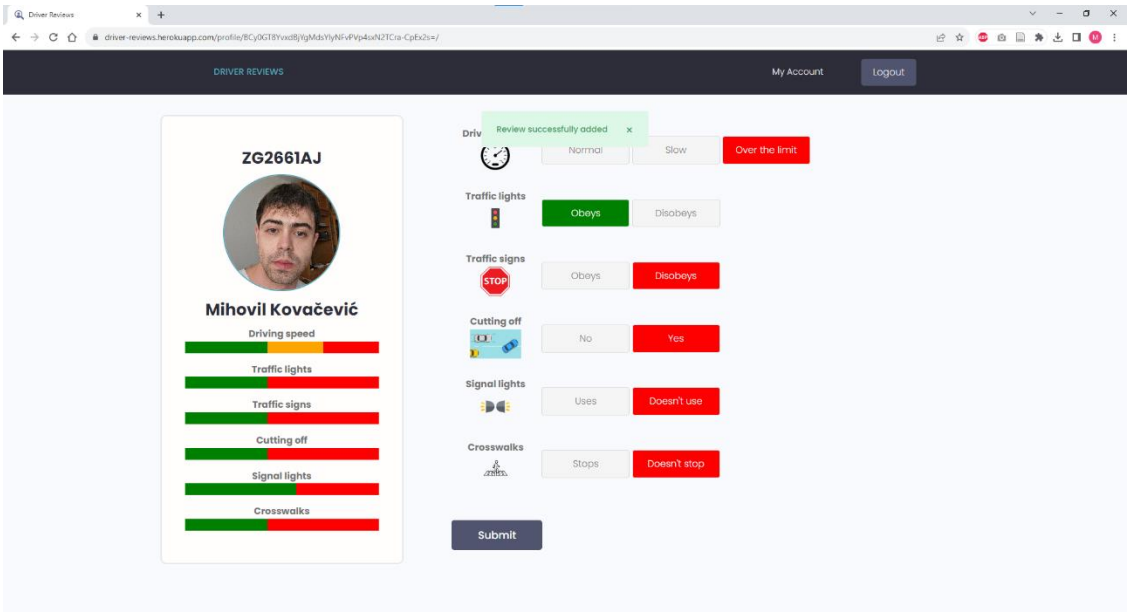
Slika 5.45. Prikaz iscrtane statistike recenzija nakon dodavanja recenzije četvrtog korisničkog računa



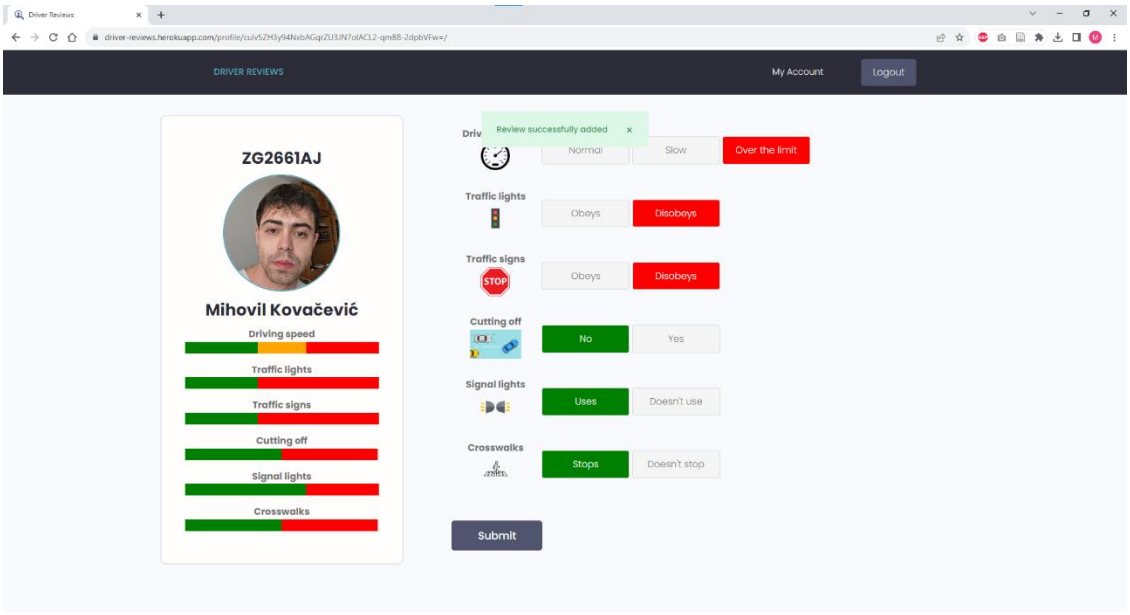
Slika 5.46. Prikaz iscrtane statistike recenzija nakon dodavanja recenzije petog korisničkog računa



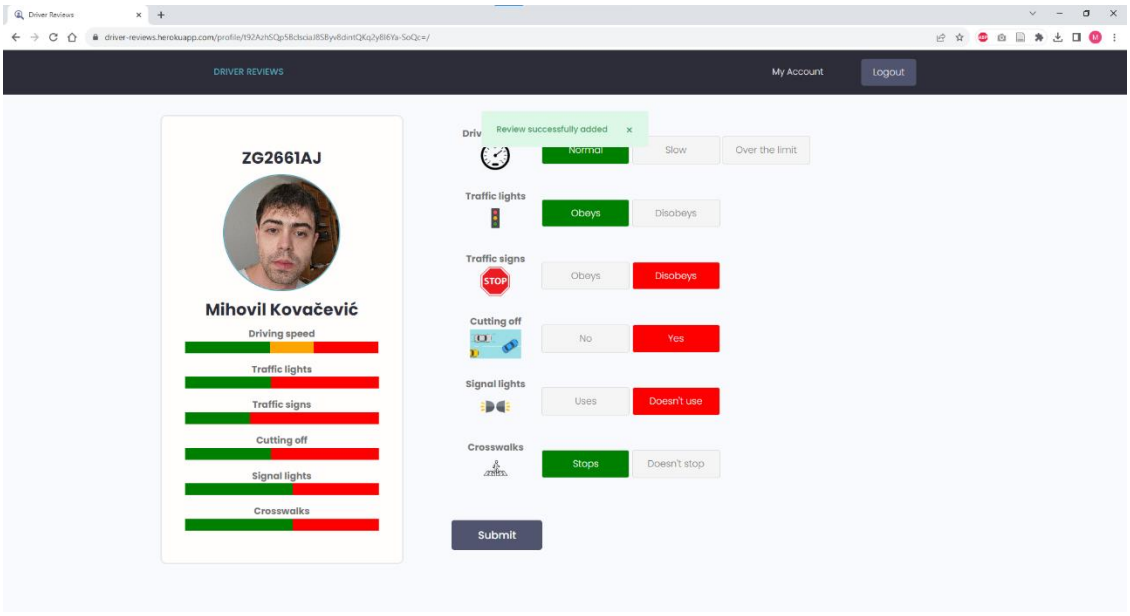
Slika 5.47. Prikaz iscrtane statistike recenzija nakon dodavanja recenzije šestog korisničkog računa



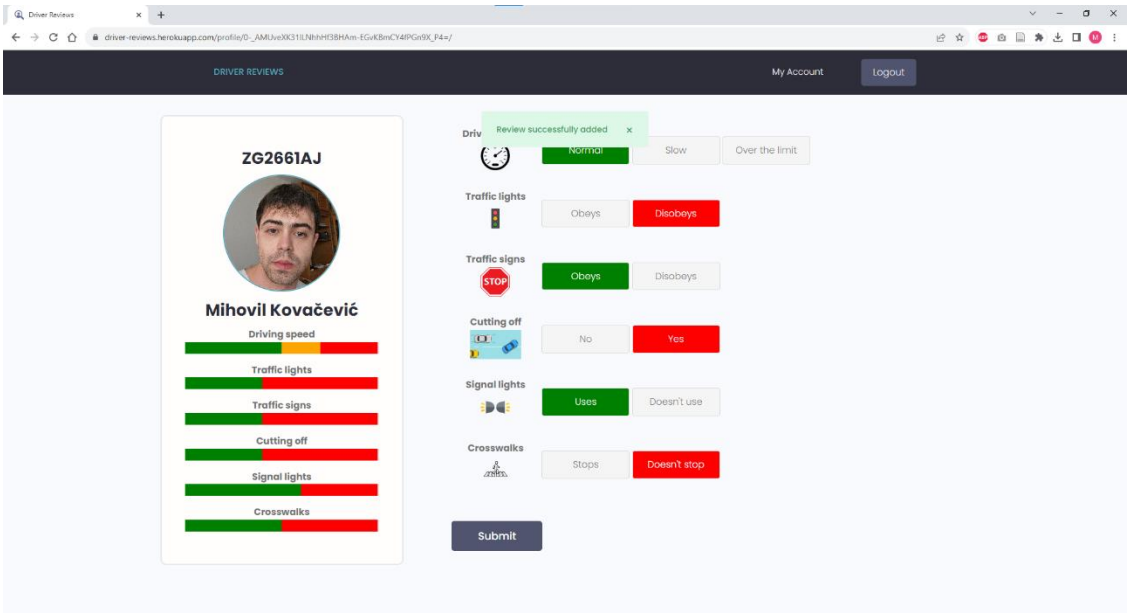
Slika 5.48. Prikaz iscrtane statistike recenzija nakon dodavanja recenzije sedmog korisničkog računa



Slika 5.49. Prikaz iscrtane statistike recenzija nakon dodavanja recenzije osmog korisničkog računa

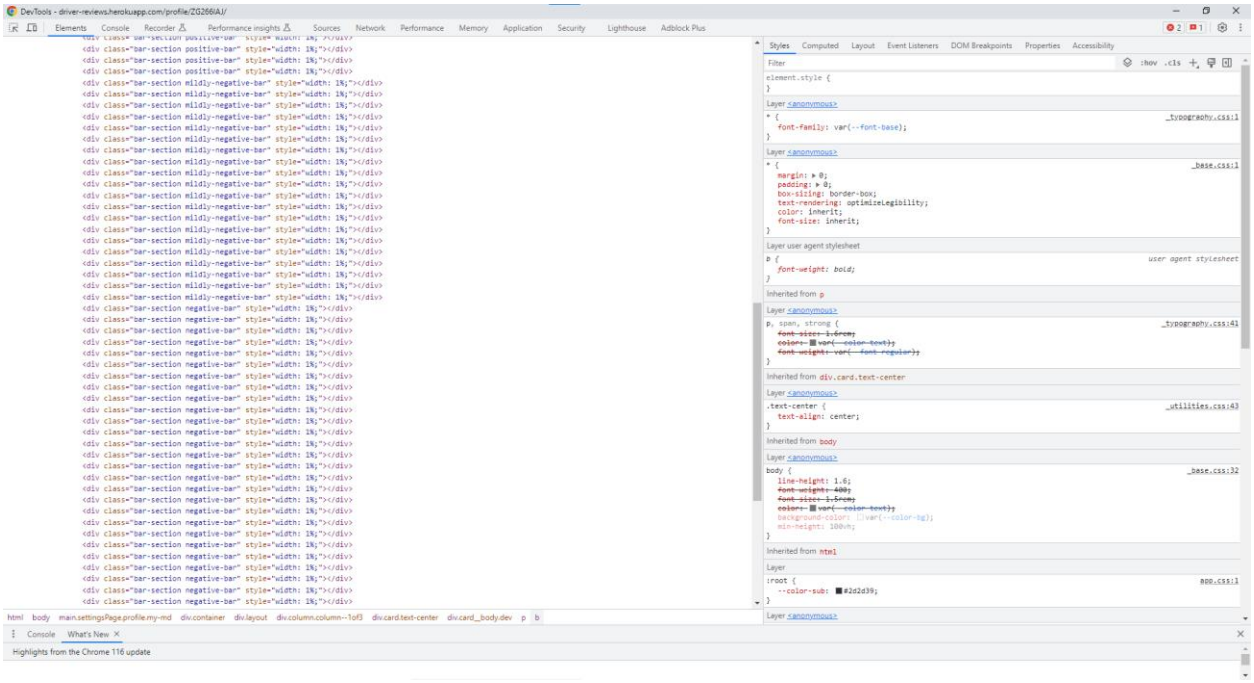


Slika 5.50. Prikaz iscrtane statistike recenzija nakon dodavanja recenzije devetog korisničkog računa



Slika 5.51. Prikaz iscrtane statistike recenzija nakon dodavanja recenzije desetog korisničkog računa

Na slici 5.52. prikazan je primjer *html* koda koji će se kopirati u *Python* skriptu sa slike 5.41. za računanje broja pojavljivanja svakog pojedinačnog *div* elementa *html* koda, a na slici 5.53. prikazani su rezultati izvođenja iste *Python* skripte.



Slika 5.52. Primjer html koda iscrtane statistike recenzija za jednu kategoriju

```
PS C:\Users\Mihovil\Desktop>  & 'C:\Users\Mihovil\AppData\Local\Programs\Python\Python39\python.exe' 'c:\Users\Mihovil\.vscode\extensions\ms-python.python-2023.14.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '53688' '--' 'c:\Users\Mihovil\Desktop\calculate_reviews.py'
bar-section = 100
positive-bar = 50
mildly-negative-bar = 20
negative-bar = 30
PS C:\Users\Mihovil\Desktop>  c:; cd 'c:\Users\Mihovil\Desktop'; & 'C:\Users\Mihovil\AppData\Local\Programs\Python\Python39\python.exe' 'c:\Users\Mihovil\.vscode\extensions\ms-python.python-2023.14.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '53708' '--' 'c:\Users\Mihovil\Desktop\calculate_reviews.py'
bar-section = 100
positive-bar = 40
negative-bar = 60
PS C:\Users\Mihovil\Desktop>  c:; cd 'c:\Users\Mihovil\Desktop'; & 'C:\Users\Mihovil\AppData\Local\Programs\Python\Python39\python.exe' 'c:\Users\Mihovil\.vscode\extensions\ms-python.python-2023.14.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '53725' '--' 'c:\Users\Mihovil\Desktop\calculate_reviews.py'
bar-section = 100
positive-bar = 40
negative-bar = 60
PS C:\Users\Mihovil\Desktop>  c:; cd 'c:\Users\Mihovil\Desktop'; & 'C:\Users\Mihovil\AppData\Local\Programs\Python\Python39\python.exe' 'c:\Users\Mihovil\.vscode\extensions\ms-python.python-2023.14.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '56460' '--' 'c:\Users\Mihovil\Desktop\calculate_reviews.py'
bar-section = 100
positive-bar = 40
negative-bar = 60
PS C:\Users\Mihovil\Desktop>  c:; cd 'c:\Users\Mihovil\Desktop'; & 'C:\Users\Mihovil\AppData\Local\Programs\Python\Python39\python.exe' 'c:\Users\Mihovil\.vscode\extensions\ms-python.python-2023.14.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '56470' '--' 'c:\Users\Mihovil\Desktop\calculate_reviews.py'
bar-section = 100
positive-bar = 60
negative-bar = 40
PS C:\Users\Mihovil\Desktop>  c:; cd 'c:\Users\Mihovil\Desktop'; & 'C:\Users\Mihovil\AppData\Local\Programs\Python\Python39\python.exe' 'c:\Users\Mihovil\.vscode\extensions\ms-python.python-2023.14.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '56493' '--' 'c:\Users\Mihovil\Desktop\calculate_reviews.py'
bar-section = 100
positive-bar = 50
negative-bar = 50
PS C:\Users\Mihovil\Desktop>
```

Slika 5.53. Rezultati izvođenja Python skripte za računanje broja pojavljivanja svakog pojedinačnog div elementa html koda

U tablici 5.6. prikazane su odabrane opcije recenziranja svake kategorije za svaki korisnički račun, referentna ručno izračunata statistika recenzija i statistika recenzija prikazana na web aplikaciji. Iz priloženog je vidljivo da nema odstupanja.

Tablica 5.6. Odabrane opcije recenziranja svake kategorije za svaki korisnički račun i usporedba ručno izračunate statistike recenzija i statistike recenzija prikazane na web aplikaciji

	Driving speed	***Traffic lights***	***Traffic signs***	***Cutting off***	***Signal lights***	***Crosswalks***
Korisnik 1	*Normal*	*Obeys*	*Obeys*	*No*	*Uses*	*Stops*
Korisnik 2	*Slow*	*Disobeys*	*Disobeys*	*Yes*	*Doesn't use*	*Doesn't stop*
Korisnik 3	*Over the limit*	*Disobeys*	*Obeys*	*Yes*	*Uses*	*Doesn't stop*
Korisnik 4	*Normal*	*Disobeys*	*Disobeys*	*Yes*	*Doesn't use*	*Doesn't stop*
Korisnik 5	*Slow*	*Obeys*	*Obeys*	*No*	*Uses*	*Stops*
Korisnik 6	*Normal*	*Disobeys*	*Disobeys*	*No*	*Uses*	*Stops*
Korisnik 7	*Over the limit*	*Obeys*	*Disobeys*	*Yes*	*Doesn't use*	*Doesn't stop*
Korisnik 8	*Over the limit*	*Disobeys*	*Disobeys*	*No*	*Uses*	*Stops*
Korisnik 9	*Normal*	*Obeys*	*Disobeys*	*Yes*	*Doesn't use*	*Stops*
Korisnik 10	*Normal*	*Disobeys*	*Obeys*	*Yes*	*Uses*	*Doesn't stop*
Referentna statistika recenzija (ručno izračunata)	*Normal* : *Slow* : *Over the limit* = 5 : 2 : 3	*Obeys* : *Disobeys* = 4 : 6	*Obeys* : *Disobeys* = 4 : 6	*No* : *Yes* = 4 : 6	*Uses* : *Doesn't use* = 6 : 4	*Stops* : *Doesn't stop* = 5 : 5
Statistika recenzija prikazana na web aplikaciji	50 : 20 : 30 = 5 : 2 : 3	40 : 60 = 4 : 6	40 : 60 = 4 : 6	40 : 60 = 4 : 6	60 : 40 = 6 : 4	50 : 50 = 5 : 5
Odstupanje	Nema	Nema	Nema	Nema	Nema	Nema

5.8.7. Komentar na cjelokupno testiranje web aplikacije

Rezultati testiranja registracije korisnika, prijave i odjave korisnika, uređivanja informacija vlastitog korisničkog računa, pretraživanja profila vozača i funkcionalnost recenziranja te izračun statistike postojećih recenzija pokazali su da ponašanje web aplikacije ne odstupa od očekivanog ponašanja definiranog funkcionalnim zahtjevima web aplikacije. Uz to, prilikom testiranja funkcionalnosti omogućavanja recenziranja samo prijavljenim korisnicima koji koriste aplikaciju za detekciju i prepoznavanje registarskih tablica vozila utvrđeno je da ukoliko aplikacija za detekciju i prepoznavanje registarskih tablica vozila netočno prepozna oznaku registarske tablice vozila, ponašanje web aplikacije odstupa od očekivanog ponašanja definiranog funkcionalnim zahtjevima web aplikacije. Za pronađeno odstupanje navedena su moguća rješenja koja bi smanjila učestalost odstupanja ili ga potpuno uklonila.

6. ZAKLJUČAK

U posljednjem desetljeću, ubrzan napredak računalne tehnologije značajno je utjecao na različite industrije, a osobito na automobilsku industriju. Ovaj diplomski rad imao je za cilj razviti programsko rješenje sustava za prikaz podataka o profilu vozača vozila ispred vlastitog vozila, temeljenog na detekciji i prepoznavanju registarskih tablica vozila. Sustav se sastoji od dvije aplikacije – aplikacija za detekciju registarskih tablica vozila i prepoznavanje znakova na njima, te web aplikacija za prikaz podataka o profilu i recenziranje vozača.

Istražene su postojeće primjene rješenja za detekciju registarskih tablica vozila i prepoznavanje znakova na njima. Takve primjene uključuju automatizirane sustave za naplatu cestarine, upravljanje parkiralištima, provođenje zakona i mnoge druge. Proces prepoznavanja znakova na registarskim tablicama vozila uključuje ključne zadatke detekcije tablice i prepoznavanja znakova, koji se obavljaju korištenjem tehnika i algoritama strojnog učenja, posebno konvolucijskih i rekurentnih neuronskih mreža. Utvrđeno je da je YOLO algoritam najprikladniji za slučaj upotrebe ovog rada. Opisan je razvoj modela za detekciju registarskih tablica vozila te odabir i podešavanje algoritma za prepoznavanje znakova na detektiranim tablicama. Prema rezultatima testiranja, odabran je YOLOv5 *small* model uz prag pouzdanosti 0.5 za detekciju, s ostvarenim rezultatima preciznosti 0.9905, odziva 0.9811 i F1 mjere 0.9858. Prilikom testiranja za brzinu izvođenja na procesoru Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz omogućava obradu 2.33 sličice u sekundi u čemu ga nadmašuje od testiranih jedino YOLOv7 *tiny* model uz prag pouzdanosti 0.4 s rezultatom obrade 2.69 sličice u sekundi. Od najmodernijih rješenja otvorenog koda za prepoznavanje znakova odabran je *PaddleOCR* kao najtočniji. Za detektirane granične okvire registarskih tablica vozila širine veće od i uključujući 120 piksela procijenjene maksimalne udaljenosti približno 5.42 metara pri rezoluciji slike 1920x1080 piksela ostvaruje točnost 80.33%. Za maksimalnu udaljenost na kojoj će konačni sustav prepoznavati registarske tablice vozila odabrana je širina detektiranih graničnih okvira registarskih tablica vozila veća od i uključujući 80 piksela procijenjene maksimalne udaljenosti približno 8.13 metara pri rezoluciji slike 1920x1080 piksela, gdje je nakon testiranja *PaddleOCR* ostvario točnost 77.08%. Opisan je i razvoj web aplikacije koja omogućuje prikaz podataka o profilu i recenziranje vozača. Ova web aplikacija izgrađena je pomoću *Django* okvira za razvoj pozadinskih funkcionalnosti i implementirana je na javni server pomoću *Amazon* web usluga i *Heroku* platforme. Testiranjem funkcionalnosti web aplikacije registracije korisnika, prijave i

odjave korisnika, uređivanja informacija vlastitog korisničkog računa, pretraživanja profila vozača i recenziranja te izračuna statistike postojećih recenzija utvrđeno je da ponašanja web aplikacije ne odstupaju od očekivanih ponašanja, odnosno ponašanja navedenih unutar funkcionalnih zahtjeva web aplikacije. Nakon testiranja omogućavanja recenziranja samo prijavljenim korisnicima koji koriste aplikaciju za detekciju i prepoznavanje registarskih tablica vozila, utvrđeno je da ukoliko je prepoznavanje registarske tablice netočno, otvara se stranica profila netočnog vozača.

U sklopu ovog diplomskog rada uspješno je razvijeno programsko rješenje sustava za prikaz podataka o profilu vozača vozila temeljenog na detekciji i prepoznavanju registarskih tablica vozila. Odabir najpogodnijih algoritama za detekciju i prepoznavanje znakova ključan je za učinkovitost sustava, a web aplikacija omogućuje korisnicima pregled podataka o profilu i recenziranje vozača. Napredak u korištenju tehnologije za detekciju i prepoznavanje, ili pronalazak novih metoda, otvara mogućnosti za poboljšanje sustava koji je predstavljen u ovom radu. Uz to, prepoznavanje znakova na slikama veće rezolucije bi rezolutiralo većom točnošću prepoznavanja za trenutni sustav. Primjerice, za rezoluciju 3840x2160 piksela ili veću, moguća su točna prepoznavanja registarskih tablica vozila na većim udaljenostima nego za rezoluciju 1920x1080 piksela. Nemogućnost prepoznavanja hrvatskih dijakritičkih znakova moglo bi biti riješeno dodatnim treniranjem postojećih modela algoritama za prepoznavanje znakova ili razvitkom vlastitog algoritma za prepoznavanje znakova. Idealni skup podataka za testiranje algoritma za prepoznavanje znakova pri različitim udaljenostima registarskih tablica vozila bi bio onaj koji sadrži za svaku registarsku tablicu vozila onoliko slika na različitim udaljenostima za koliko se udaljenosti testira. Za slučajeve netočnog prepoznavanja registarske tablice vozila ponašanje web aplikacije bi se moglo unaprijediti implementacijom trajanja sesije za svaku pojedinačnu stranicu koja omogućuje recenziranje, te omogućiti korisniku da odbije stranice za recenziranje koje su otvorene na temelju netočnog prepoznavanja registarske tablice vozila. Za ostvarenje ovog rješenja morao bi se koristiti kod koji se izvodi paralelno. Funkcionalnost koja uz pomoć kružnog spremnika veličine 10 oznaka registarskih tablica vozila otvara stranicu za recenziranje samo ako je na 10 uzastopnih sličica barem 5 ili više puta prepoznata ista oznaka registarske tablice vozila bi također bila jedno od rješenja za problem netočnog prepoznavanja registarske tablice vozila.

LITERATURA

[1] C. Henry, S. Y. Ahn, and S.-W. Lee, "Multinational License Plate Recognition Using Generalized Character Sequence Detection," *IEEE Access*, vol. 8, pp. 35185–35199, 2020, doi: 10.1109/ACCESS.2020.2974973.

[2] "Gachon University." http://pr.gachon.ac.kr/ALPR.html (accessed Aug. 12, 2023).

[3] "Papers with Code - AOLP Dataset." https://paperswithcode.com/dataset/aolp (accessed Aug. 12, 2023).

[4] M. Weber and P. Perona, "Caltech Cars 1999." CaltechDATA, Apr. 06, 2022. doi: 10.22002/D1.20084.

[5] "Medialab LPR database." http://www.medialab.ntua.gr/research/LPRdatabase.html (accessed Aug. 12, 2023).

[6] "Projekt 'License Plates.'" http://www.zemris.fer.hr/projects/LicensePlates/english/results.shtml (accessed Aug. 12, 2023).

[7] "OpenALPR - Automatic License Plate Recognition." https://www.openalpr.com/ (accessed Aug. 12, 2023).

[8] "Automatic License Plate Recognition Software," *Sighthound*. https://www.sighthound.com/products/alpr (accessed Aug. 12, 2023).

[9] X. Wang, Z. Man, M. You, and C. Shen, "Adversarial Generation of Training Examples: Applications to Moving Vehicle License Plate Recognition." arXiv, Nov. 10, 2017. Accessed: Aug. 12, 2023. [Online]. Available: http://arxiv.org/abs/1707.03124

[10] S. M. Silva and C. R. Jung, "License Plate Detection and Recognition in Unconstrained Scenarios," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., in Lecture Notes in Computer Science, vol. 11216. Cham: Springer International Publishing, 2018, pp. 593–609. doi: 10.1007/978-3-030-01258-8_36.

[11] R. Laroca *et al.*, "A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector," in *2018 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2018, pp. 1–10. doi: 10.1109/IJCNN.2018.8489629.

[12] S. Luo and J. Liu, "Research on Car License Plate Recognition Based on Improved YOLOv5m and LPRNet," *IEEE Access*, vol. 10, pp. 93692–93700, 2022, doi: 10.1109/ACCESS.2022.3203388.

[13] G. Jocher, "YOLOv5 by Ultralytics." May 2020. doi: 10.5281/zenodo.3908559.

[14] "LPRNet," *NVIDIA Docs*. https://docs.nvidia.com/tao/tao-toolkit/text/character_recognition/lprnet.html (accessed Aug. 12, 2023).

[15] "CCPD - V7 Open Datasets." https://www.v7labs.com/open-datasets/ccpd (accessed Aug. 12, 2023).

[16] M.-X. He and P. Hao, "Robust Automatic Recognition of Chinese License Plates in Natural Scenes," *IEEE Access*, vol. 8, pp. 173804–173814, 2020, doi: 10.1109/ACCESS.2020.3026181.

[17] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection." arXiv, Jul. 27, 2020. Accessed: Aug. 12, 2023. [Online]. Available: http://arxiv.org/abs/1911.09070

[18] Y. Liu, Y. Li, G. Chen, and H. Gao, "An Edge-end Based Fast Car License Plate Recogniton Method," in *2020 International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE)*, Oct. 2020, pp. 394–397. doi: 10.1109/ICBASE51474.2020.00090.

[19] J.-Y. Sung, S.-B. Yu, and S. P. Korea, “Real-time Automatic License Plate Recognition System using YOLOv4,” in *2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*, Nov. 2020, pp. 1–3. doi: 10.1109/ICCE-Asia49877.2020.9277050.

[20] M. Li, F. Ren, Z. Zhang, Y. Long, and J. Zeng, “Chinese License Plate Recognition Algorithm Based On UNet3+,” in *2022 3rd International Conference on Computing, Networks and Internet of Things (CNIOT)*, May 2022, pp. 60–64. doi: 10.1109/CNIOT55862.2022.00018.

[21] X. Tian, L. Wang, and R. Zhang, “License Plate Recognition Based on CNN,” in *2022 14th International Conference on Computer Research and Development (ICCRD)*, Jan. 2022, pp. 244–249. doi: 10.1109/ICCRD54409.2022.9730272.

[22] N. Kharina and S. Chernyadyev, “Software for Car License Plates Recognition with Minimal Computing Resources,” in *2022 24th International Conference on Digital Signal Processing and its Applications (DSPA)*, Mar. 2022, pp. 1–4. doi: 10.1109/DSPA53304.2022.9790745.

[23] J. Kim, J.-Y. Sung, and S. Park, “Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition,” in *2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*, Nov. 2020, pp. 1–4. doi: 10.1109/ICCE-Asia49877.2020.9277040.

[24] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” arXiv, Jan. 06, 2016. Accessed: Aug. 16, 2023. [Online]. Available: http://arxiv.org/abs/1506.01497

[25] W. Liu *et al.*, “SSD: Single Shot MultiBox Detector,” 2016, pp. 21–37. doi: 10.1007/978-3-319-46448-0_2.

[26] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors.” arXiv, Jul. 06, 2022. Accessed: Jul. 06, 2023. [Online]. Available: http://arxiv.org/abs/2207.02696

[27] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “Scaled-YOLOv4: Scaling Cross Stage Partial Network.” arXiv, Feb. 21, 2021. Accessed: Aug. 24, 2023. [Online]. Available: http://arxiv.org/abs/2011.08036

[28] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, “YOLOX: Exceeding YOLO Series in 2021.” arXiv, Aug. 05, 2021. Accessed: Aug. 24, 2023. [Online]. Available: http://arxiv.org/abs/2107.08430

[29] S. Xu *et al.*, “PP-YOLOE: An evolved version of YOLO.” arXiv, Dec. 11, 2022. Accessed: Aug. 24, 2023. [Online]. Available: http://arxiv.org/abs/2203.16250

[30] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, “You Only Learn One Representation: Unified Network for Multiple Tasks.” arXiv, May 10, 2021. Accessed: Aug. 24, 2023. [Online]. Available: http://arxiv.org/abs/2105.04206

[31] “COCO - Common Objects in Context.” https://cocodataset.org/#home (accessed Aug. 16, 2023).

[32] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation.” arXiv, Oct. 22, 2014. Accessed: Aug. 24, 2023. [Online]. Available: http://arxiv.org/abs/1311.2524

[33] R. Girshick, “Fast R-CNN.” arXiv, Sep. 27, 2015. Accessed: Aug. 24, 2023. [Online]. Available: http://arxiv.org/abs/1504.08083

[34] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection." arXiv, May 09, 2016. Accessed: Jul. 17, 2023. [Online]. Available: http://arxiv.org/abs/1506.02640
[35] "Regression vs. Classification in Machine Learning: What's the Difference?," *Springboard Blog*, Oct. 06, 2021. https://www.springboard.com/blog/data-science/regression-vs-classification/ (accessed Jul. 23, 2023).
[36] "What is a Neural Network? - Artificial Neural Network Explained - AWS," *Amazon Web Services, Inc.* https://aws.amazon.com/what-is/neural-network/ (accessed Jul. 17, 2023).
[37] "Step-by-step instructions for training YOLOv7 on a Custom Dataset," *Paperspace Blog*, Oct. 20, 2022. https://blog.paperspace.com/train-yolov7-custom-data/ (accessed Jul. 25, 2023).
[38] *What is YOLO algorithm? | Deep Learning Tutorial 31 (Tensorflow, Keras & Python)*, (2020). Accessed: Jul. 23, 2023. [Online Video]. Available: https://www.youtube.com/watch?v=ag3DLKsl2vk
[39] O. Chernytska, "Training YOLO? Select Anchor Boxes Like This," *Medium*, Aug. 18, 2022. https://towardsdatascience.com/training-yolo-select-anchor-boxes-like-this-3226cb8d7f0b (accessed Jul. 22, 2023).
[40] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger." arXiv, Dec. 25, 2016. Accessed: Jul. 23, 2023. [Online]. Available: http://arxiv.org/abs/1612.08242
[41] D. Cochard, "mAP : Evaluation metric for object detection models," *axinc-ai*, Oct. 06, 2021. https://medium.com/axinc-ai/map-evaluation-metric-of-object-detection-model-dd20e2dc2472 (accessed Jul. 23, 2023).
[42] "Theos AI - The artificial visual cortex." https://theos.ai/ (accessed Jul. 24, 2023).
[43] "Google Colaboratory." https://colab.research.google.com/?utm_source=scs-index (accessed Jul. 24, 2023).
[44] "City Drive - YouTube." https://www.youtube.com/ (accessed Jul. 24, 2023).
[45] "👊 Easy YOLOv7 ⚡." Theos AI, Jun. 11, 2023. Accessed: Jul. 24, 2023. [Online]. Available: https://github.com/theos-ai/easy-yolov7
[46] "PyTorch." https://pytorch.org/ (accessed Jul. 24, 2023).
[47] "Make Sense." https://www.makesense.ai/ (accessed Jul. 25, 2023).
[48] *YOLO v7 Object Detection Models: FPS & Object Detection Comparison (All 7 Models)*, (2022). Accessed: Jul. 26, 2023. [Online Video]. Available: https://www.youtube.com/watch?v=t3mTGXF_85M
[49] "keras-ocr — keras_ocr documentation." https://keras-ocr.readthedocs.io/en/latest/ (accessed Aug. 19, 2023).
[50] "EasyOCR." Jaided AI, Aug. 19, 2023. Accessed: Aug. 19, 2023. [Online]. Available: https://github.com/JaidedAI/EasyOCR
[51] S. Hoffstaetter, "pytesseract: Python-tesseract is a python wrapper for Google's Tesseract-OCR." Accessed: Aug. 19, 2023. [Online]. Available: https://github.com/madmaze/pytesseract
[52] "PaddlePaddle/PaddleOCR." PaddlePaddle, Jul. 31, 2023. Accessed: Jul. 31, 2023. [Online]. Available: https://github.com/PaddlePaddle/PaddleOCR
[53] "RNN vs CNN vs Transformer," *Zheyuan BAI's Blog*, Jun. 21, 2020. http://baiblanc.github.io/2020/06/21/RNN-vs-CNN-vs-Transformer/index.html (accessed Jul. 31, 2023).

[54] VIDI, “Nove hrvatske registarske pločice,” *Vidiauto*. https://www.vidiauto.com/Zanimljivosti/INFOZONA/Nove-hrvatske-registarske-plocice (accessed Sep. 11, 2023).
[55] “HERO9 Black: Digital Lenses FOV Information.” https://community.gopro.com/s/article/HERO9-Black-Digital-Lenses-FOV-Information?language=en_US (accessed Sep. 11, 2023).
[56] “Django,” *Django Project*. https://www.djangoproject.com/ (accessed Jul. 29, 2023).
[57] “Django Beginners Course.” https://dennisivy.teachable.com/p/django-beginners-course (accessed Jul. 29, 2023).
[58] “🎨 Mumble Ui Kit.” http://mumble-lp.s3-website-us-west-2.amazonaws.com/ (accessed Aug. 21, 2023).

SAŽETAK

U ovom diplomskom radu razvijeno je programsko rješenje sustava za prikaz podataka o profilu vozača vozila ispred vlastitog vozila, temeljenog na detekciji i prepoznavanju registarskih tablica vozila. Sustav se sastoji od dvije aplikacije – aplikacija za detekciju registarskih tablica vozila i prepoznavanje znakova na njima, te web aplikacija za prikaz podataka o profilu i recenziranje vozača. Istražene su različite primjene rješenja za detekciju registarskih tablica vozila i prepoznavanje znakova na njima, a YOLO algoritam odabran je kao najprikladniji za detekciju. YOLOv5 *small* model za detekciju uz prag pouzdanosti 0.5 je odabran kao najprikladniji za ovaj slučaj upotrebe, s ostvarenim rezultatima preciznosti 0.9905, odziva 0.9811, F1 mjere 0.9858 i 2.69 obrađenih sličica u sekundi prilikom testiranja na procesoru Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz. Za prepoznavanje znakova najtočnijim se pokazao *PaddleOCR* algoritam. Za maksimalnu udaljenost na kojoj će konačni sustav prepoznavati registarske tablice vozila odabrana je širina detektiranih graničnih okvira registarskih tablica vozila veća od i uključujući 80 piksela procijenjene maksimalne udaljenosti približno 8.13 metara pri rezoluciji slike 1920x1080 piksela, gdje je nakon testiranja *PaddleOCR* ostvario točnost 77.08%. Razvijena je *Django* web aplikacija koja omogućuje prikaz podataka o profilu i recenziranje vozača. Testiranjem funkcionalnosti web aplikacije utvrđeno je da ponašanja web aplikacije ne odstupaju od očekivanih ponašanja, osim ako je prepoznavanje registarske tablice vozila netočno nakon čega se otvara stranica profila netočnog vozača. Napredak korištene tehnologije za detekciju i prepoznavanje ili izum nove otvara mogućnosti za poboljšanje sustava predstavljenog u ovom radu. Uz to, prepoznavanje znakova na slikama veće rezolucije bi rezolutiralo većom točnošću prepoznavanja za trenutni sustav. Nemogućnost prepoznavanja hrvatskih dijakritičkih znakova moglo bi biti riješeno dodatnim treniranjem postojećih modela algoritama za prepoznavanje znakova ili razvitkom vlastitog algoritma za prepoznavanje znakova. Idealni skup podataka za testiranje algoritma za prepoznavanje znakova pri različitim udaljenostima registarskih tablica vozila bi bio onaj koji sadrži za svaku registarsku tablicu vozila onoliko slika na različitim udaljenostima za koliko se udaljenosti testira. Funkcionalnost odbijanja stranice za recenziranje koja je otvorena na temelju netočnog prepoznavanja registarske tablice vozila ili funkcionalnost koja otvara stranicu za recenziranje samo ako je uz pomoć kružnog spremnika na 5 ili više sličica prepoznata ista oznaka su moguća poboljšanja sustava za problem netočnog prepoznavanja registarske tablice vozila.

Ključne riječi: *prepoznavanje registarske tablice vozila, recenziranje vozača, YOLO, PaddleOCR, web aplikacija*

SYSTEM FOR DISPLAYING INFORMATION ABOUT THE PROFILE OF THE DRIVER OF THE VEHICLE IN FRONT OF THE EGO-VEHICLE BASED ON THE VEHICLE LICENSE PLATE DETECTION AND RECOGNITION

ABSTRACT

In this master's thesis, a software solution for a system for displaying driver profile data of vehicles in front of the driver's own vehicle, based on the detection and recognition of vehicle license plates, is developed. The system consists of two applications – an application for detecting vehicle license plates and recognizing characters on them, and a web application for displaying profile data and reviewing drivers. Various implementations of solutions for detecting license plates and recognizing characters on them have been investigated, and the YOLO algorithm was selected as the most suitable for detection. The YOLOv5 small model for detection with a confidence threshold of 0.5 was selected as the most suitable for this use case, with achieved results of precision 0.9905, recall 0.9811, F1 measure 0.9858, and 2.69 processed images per second when testing on an Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz processor. The PaddleOCR algorithm proved to be the most accurate for character recognition. For the maximum distance at which the final system will recognize license plates, the width of the detected bounding boxes of license plates greater than or equal to 80 pixels of the estimated maximum distance of approximately 8.13 meters at a resolution of 1920x1080 pixels was selected, where after testing PaddleOCR achieved an accuracy of 77.08%. A Django web application has been developed that enables the display of profile data and driver reviews. By testing the functionality of the web application, it was found that the behavior of the web application does not deviate from the expected behavior, except if the recognition of the license plate is incorrect, after which the page of the incorrect driver is opened. Progress in the technology used for detection and recognition or the invention of a new one opens up possibilities for improving the system presented in this paper. In addition, recognizing characters on images with higher resolution would result in higher recognition accuracy for the current system. The inability to recognize Croatian diacritics could be solved by additional training of existing models of character recognition algorithms or by developing a proprietary character recognition algorithm. The ideal dataset for testing a license plate recognition algorithm at various distances from vehicle license plates would be one that

contains as many images as there are distances being tested for each license plate. The functionality of rejecting the review page that is opened based on the incorrect recognition of the license plate or the functionality that opens the review page only if the same tag is recognized with the help of a circular buffer on 5 or more images are possible improvements to the system for the problem of incorrect recognition of the license plate.

Keywords: *recognition of vehicle license plates, driver reviews, YOLO, PaddleOCR, web application*

ŽIVOTOPIS

Mihovil Kovačević rođen je 11. rujna 1998. godine u Slavonskom Brodu. Odrastao je u Županji gdje je i pohađao osnovnu školu. Prirodoslovno matematički smjer Gimnazije Županja upisuje 2013. godine te završava 2017. godine. Iste godine upisuje preddiplomski sveučilišni studij Računarstvo na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. Godine 2021. stječe akademski naziv sveučilišni prvostupnik (baccalaureus) inženjer računarstva. Iste godine upisuje diplomski sveučilišni studij računarstva, smjer programsko inženjerstvo na istom fakultetu i uz to postaje stipendist Instituta RT-RK u Osijeku što se nastavilo i sljedeće godine.

Potpis:

_____

PRILOZI

- P.3.1. Skup slika korišten za treniranje YOLO modela (priloženo na DVD-u uz rad)
- P.3.2. Slike ostvarenih rezultata u obliku grafa metrika preciznosti, odziva, mAP@0.5, mAP@0.5:95 i *fitness*-a uz epohe za trenirane modele YOLOv5 *small*, YOLOv5 *medium*, YOLOv5 *extra large*, YOLOv7 *tiny*, YOLOv7 *default* i YOLOv7 *W6* (priloženo na DVD-u uz rad)
- P.3.3. Skup slika korišten za testiranje vlastito treniranih modela YOLOv5 *small*, YOLOv5 *medium*, YOLOv5 *extra large*, YOLOv7 *tiny*, YOLOv7 *default* i YOLOv7 *W6* za metrike preciznosti, odziva i F1 mjere (priloženo na DVD-u uz rad)
- P.3.4. Videozapis korišten za testiranje vlastito treniranih modela YOLOv5 *small*, YOLOv5 *medium*, YOLOv5 *extra large*, YOLOv7 *tiny*, YOLOv7 *default* i YOLOv7 *W6* za brzinu izvođenja (priloženo na DVD-u uz rad)
- P.4.1. Skup slika i oznaka korištenih za testiranje OCR algoritama *KerasOCR*, *EasyOCR*, *Pytesseract* i *PaddleOCR* (priloženo na DVD-u uz rad)
- P.4.2. Videozapis rezolucije 1920x1080 piksela korišten za određivanje udaljenosti prema širini detektiranog graničnog okvira registarske tablice vozila (priloženo na DVD-u uz rad)
- P.4.3. Videozapis rezolucije 3840x2160 piksela korišten za određivanje udaljenosti prema širini detektiranog graničnog okvira registarske tablice vozila (priloženo na DVD-u uz rad)
- P4.4. Konačni repozitorij aplikacije za detekciju i prepoznavanje znakova registarskih tablica vozila (priloženo na DVD-u uz rad)
- P5.1. Konačni repozitorij web aplikacije za prikaz podataka o profilu i recenziranje vozača ispred vlastitog vozila (priloženo na DVD-u uz rad)