Web aplikacija za uvježbavanje uporabe glagolskih vremena na stranom jeziku

Klepač, Franko

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni prijediplomski studij Računarstvo

WEB APLIKACIJA ZA UVJEŽBAVANJE UPORABE GLAGOLSKIH VREMENA NA STRANOM JEZIKU

Završni rad

Franko Klepač

Osijek, 2024.

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

Obrazac Z1P: Obrazac za ocjenu završnog rada na sveučilišnom prijediplomskom studiju

Ocjena završnog rada na sveučilišnom prijediplomskom studiju

Ime i prezime pristupnika:	Franko Klepač
Studij, smjer:	Sveučilišni prijediplomski studij Računarstvo
Mat. br. pristupnika, god.	R4658, 28.07.2021.
JMBAG:	0165091280
Mentor:	doc. dr. sc. Dragana Božić Lenard
Sumentor:	izv. prof. dr. sc. Ivica Lukić
Sumentor iz tvrtke:	
Naslov završnog rada:	Web aplikacija za uvježbavanje uporabe glagolskih vremena na stranom jeziku
Znanstvena grana završnog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada:	Cilj je završnog rada izraditi web aplikaciju za uvježbavanje upotrebe glagolskih vremena. Aplikacija će se izraditi korištenjem HTML jezika i dodatnih tehnologija kao što u CSS, JavaScript, Bootstrap v5.0, jQuery, PHP i AJAX. Na početnoj će stranici korisnik moći odabrati između nekoliko težinskih razina. Također, na istoj će stranici, izrađujući svoj profil, korisnik moći pratiti svoj napredak. Korisnik će trebati sastaviti rečenicu od ponuđenih riječi, a duljina će rečenica i složenost u uporabi glagolskog vremena varirati ovisno na kojoj se težinskoj razini korisnik nalazi. U slučaju netočnog odgovora korisnik neće moći prijeći na

Datum prijedloga ocjene završnog rada od strane mentora:	04.09.2024.
Prijedlog ocjene završnog rada od strane mentora:	Izvrstan (5)
Datum potvrde ocjene završnog rada od strane Odbora:	11.09.2024.
Ocjena završnog rada nakon obrane:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio sveučilišni prijediplomski studij:	11.09.2024.

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

IZJAVA O IZVORNOSTI RADA

Osijek, 11.09.2024.

Ime i prezime Pristupnika:	Franko Klepač
Studij:	Sveučilišni prijediplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R4658, 28.07.2021.
Turnitin podudaranje [%]:	6

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija za uvježbavanje uporabe glagolskih vremena na stranom jeziku**

izrađen pod vodstvom mentora doc. dr. sc. Dragana Božić Lenard

i sumentora izv. prof. dr. sc. Ivica Lukić

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 pristupnika:

SADRŽAJ

1. UVOD

Izreka „Quot linguas calles, tot homines vales“ (lat. Koliko jezika govoriš, toliko ljudi vrijediš) opisuje današnje vrijeme – osoba koja zna više jezika imati će više prilika u životu od osobe koja govori samo jedan jezik. Većina ljudi u hrvatskom obrazovnom sustavu svoj prvi strani jezik počinje učiti u prvom razredu osnovne škole, a drugi strani jezik imaju mogućnost učiti tek od četvrtog razreda osnovne škole. Osim u školskim klupama, učenici i studenti imaju mogućnost učiti strane jezike pomoću raznih alata za samostalno učenje koji su lako dostupni putem interneta. Današnje se tehnologije svakim danom sve više razvijaju i poboljšavaju te je trenutno jedan od popularnijih načina učenja stranog jezika računalno potpomognuto učenje jezika (eng. *computer-assisted language learning*, CALL) za koje se sve više razvijaju aplikacije čiji se raspon kreće od jedne cjeline do programa koji pokrivaju cijele tečajeve. [1]

U sklopu ovog završnog rada izrađena je web aplikacija za računalno potpomognuto učenje glagolskih vremena engleskog jezika. Cilj je aplikacije omogućiti korisnicima lagano i neometano učenje i uvježbavanje glagolskih vremena. Aplikacija pruža teorijsku pozadinu glagolskih vremena koju korisnik nije obavezan proći kako bi započeo rješavanje zadataka. Kako bi započeli učenje, korisnici se prvo moraju registrirati i prijaviti na svoj korisnički račun. Nakon prijave, korisnici imaju na raspolaganju teorijsku pozadinu glagolskih vremena na stranom jeziku te razne zadatke koje mogu rješavati odmah ili nakon proučavanja teorijskog dijela. Napredak riješenih zadataka korisniku je dostupan na njegovom profilu. Aplikacija podržava i ulogu administratora kojem su dostupni alati kreiranja novih zadataka, uređivanja postojećih zadataka i brisanje zadataka.

Rad je podijeljen u 6 poglavlja. U uvodnom se poglavlju objašnjava zadatak završnog rada. Drugo se poglavlje bavi istraživanjem sličnih postojećih rješenja te njihovih prednosti i nedostataka. U trećem se poglavlju objašnjavaju tehnologije korištene prilikom izrade rada. U četvrtom poglavlju se prikazuju isječci koda koji prikazuju funkcionalnost stranice i slike koje prikazuju izgled aplikacije, odnosno mrežnih stranica iz perspektive korisnika i iz perspektive administratora. Na kraju, u petom je poglavlju zaključak.

1.1. Zadatak završnog rada

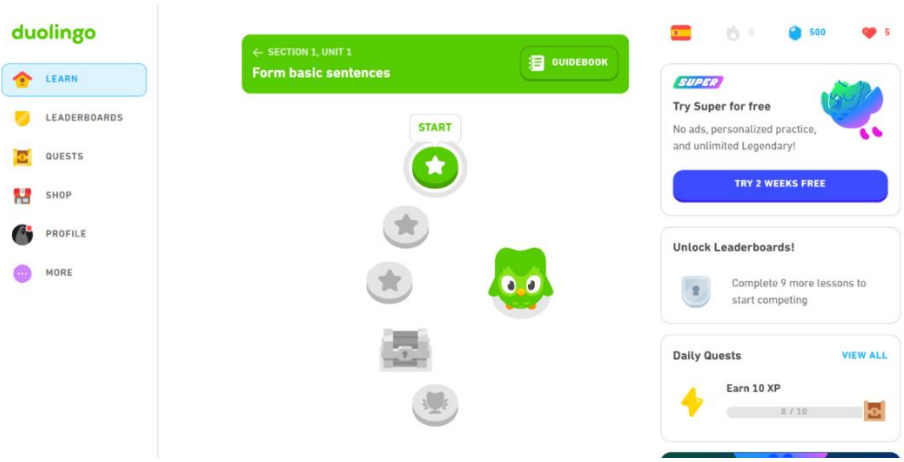
Zadatak je ovog završnog rada izrada web aplikacije za uvježbavanje uporabe glagolskih vremena na stranom jeziku koristeći okvir Django. Korištene su tehnologije HTML i CSS pomoću kojih se definira struktura stranica i njihov stil, JavaScript pomoću kojeg su dodane funkcionalnosti elementima na stranicama te AJAX kojim se ostvaruje komunikacija s poslužiteljem kako bi se obavile određene radnje.

2. ISTRAŽIVANJE POSTOJEĆIH RJEŠENJA

U ovom će poglavlju biti opisane web aplikacije koje su po svojoj funkcionalnosti slične web aplikaciji izrađenoj za ovaj završni rad. Ukratko će biti opisane njihove funkcionalnosti, izgled i dostupnost na platformama.

2.1. Duolingo

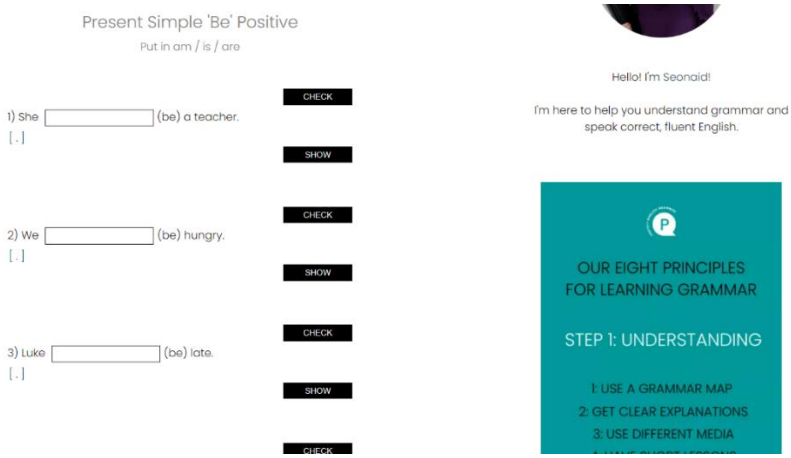
Duolingo je najpoznatija web i mobilna aplikacija za učenje stranih jezika. Dostupna je u obliku web aplikacije te na Android i iOS uređajima. Besplatna je za korištenje, no dostupna je i plaćena verzija na kojoj korisnici koriste aplikaciju bez reklama, imaju neograničen broj srca, odnosno neograničen broj pokušaja prilikom rješavanja zadataka te mogućnost personaliziranih zadataka u kojima će se nalaziti zadatci s kojima korisnik ima najviše problema prilikom rješavanja. Podržava učenje 40 jezika, no trenutno nema podržano učenje hrvatskog jezika. Postoje dva tipa zadataka - audio zadatci gdje se korisniku reproducira audio zapis govora na stranom jeziku, a korisnik mora poredati riječi kako ih je čuo ili ih mora prevesti na materinji jezik. Drugi je tip zadataka slaganje ponuđenih riječi u smislenu rečenicu, što je slično zadatcima iz aplikacije ovog završnog rada. Zadatci su podijeljeni u odjeljke te se u svakom odjeljku nalaze cjeline. Korisnik ima mogućnost preskočiti početni odjeljak te započeti od nekog naprednijeg ukoliko ima prethodno znanje jezika. Ukoliko korisnik napravi grešku prilikom rješavanja zadatka, prikazat će mu se točan odgovor te će pri kraju cjeline imati mogućnost opet riješiti prethodno pogrešno riješen zadatak. Slikom 2.1. prikazana je Duolingo web aplikacija.



Slika 2.1. Prikaz Duolingo web aplikacije [2].

2.2. Perfect English Grammar

Perfect English Grammar je mrežna stranica na kojoj korisnici imaju pristup bazi besplatnih materijala za učenje gramatike engleskog jezika. Osim besplatne opcije, postoji i plaćena verzija pomoću koje korisnici ostvaruju pravo na online predavanja koja sadrže cjelokupnu gramatiku engleskog jezika, svakodnevne zadatke u obliku PDF dokumenta i kviza, a nakon svakog uspješno riješenog poglavlja, korisnik dobije certifikat da je položio poglavlje. Besplatni tipovi zadataka na stranici koncipirani su na način da korisnici imaju prikazan infinitiv traženog glagola te, ovisno o glagolskom vremenu za koje rješavaju zadatak, moraju upotrijebiti glagol u tom glagolskom vremenu. Ne postoje razine zadataka; korisnik može pratiti redoslijed zadataka na stranici, a može ih rješavati po nekom svom redoslijedu. Ukoliko korisnik želi pratiti svoje rezultate, potrebno je registrirati se na stranici i pretplatiti se na mjesečnu članarinu. Ako prilikom rješavanja zadataka korisnik napravi pogrešku, neće dobiti nikakvu povratnu informaciju o točnom odgovoru. Sva teorijska objašnjenja dostupna su samo na engleskom jeziku. Slikom 2.2. prikazan je izgled Perfect English Grammar web aplikacije.

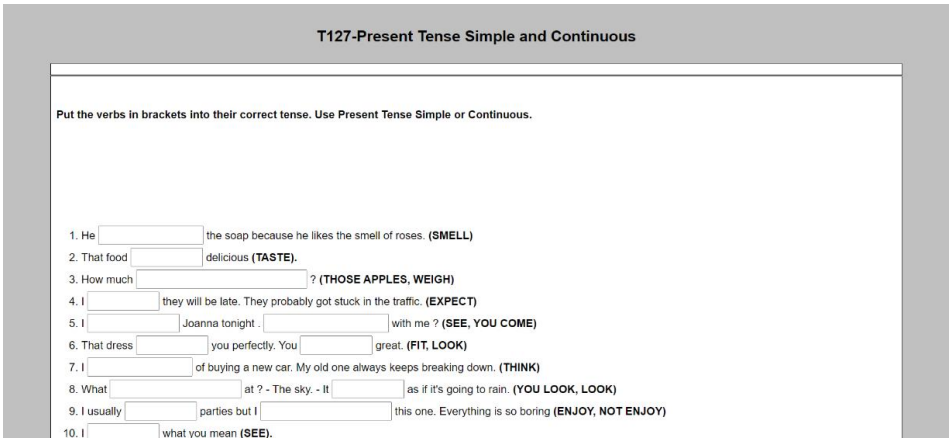


Slika 2.2. Prikaz Perfect English Grammar web aplikacije [3].

2.3. English Grammar

English Grammar je besplatna mrežna stranica na kojoj korisnici imaju pristup raznim vježbama podijeljenima na osnovnu, srednju i naprednu razinu. Korisnici imaju mogućnost preuzimanja PDF dokumenata koji sadrže vježbe, kvizove i testove koji omogućavaju da korisnici samostalno testiraju svoje znanje. Na web aplikaciji zadatci su koncipirani na način da su korisniku predloženi glagoli u obliku infinitiva te, ovisno o glagolskom vremenu koje se rješava, korisnici moraju upotrijebiti glagol u tom obliku. Zadatci nisu zaključani na način da se mora riješiti jedan zadatak kako bi se otključao drugi, što znači da korisnik može rješavati zadatke svojevoljno.

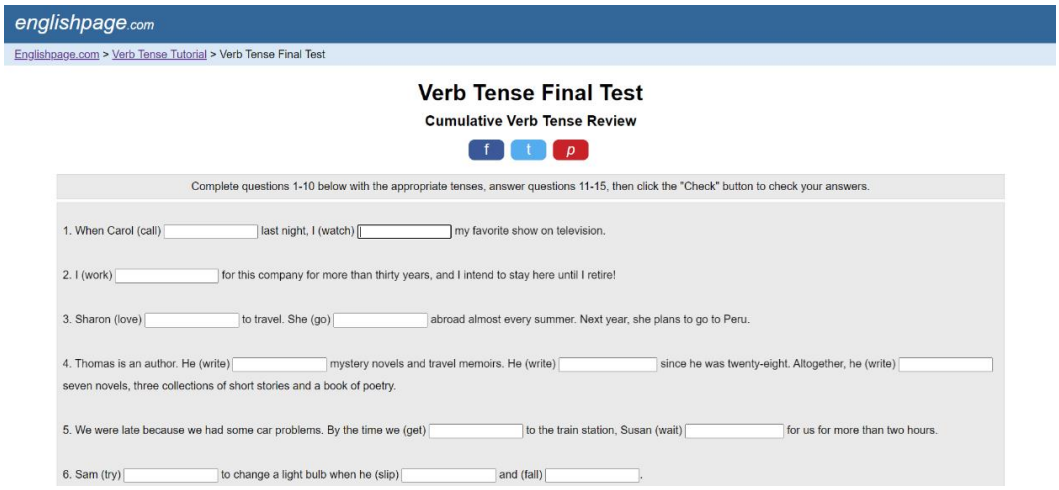
Prilikom rješavanja zadataka, korisnik će kao povratnu informaciju dobiti postotak točno riješenih zadataka cjeline. Točno riješene zadatke ne mora ponovno rješavati, dok pogrešno riješene može probati ponovno riješiti. Sva teorijska objašnjenja dostupna su samo na engleskom jeziku. Slikom 2.3. prikazan je primjer zadataka na English Grammar web aplikaciji.



Slika 2.3. Prikaz English Grammar web aplikacije [4].

2.4. englishpage.com

Pomoću stranice englishpage.com korisnici mogu samostalno učiti i vježbati gramatiku engleskog jezika besplatno. Korisnik ima opciju proučavanja teorijskog dijela prije nego što krene s rješavanjem zadataka, no to nije obavezno. Na raspolaganju je više vrsta zadataka raspoređenih po razinama. Slično kao ranije navedene web aplikacije, zadatci su koncipirani tako da je korisniku ponuđen glagol u obliku infinitiva te, ovisno o tome koje se glagolsko vrijeme rješava, korisnik mora konjugirati glagol u to glagolsko vrijeme. Slično kao ranije navedeni primjer, zadatci nisu zaključani tako da ih korisnik može rješavati po svom odabiru, bez obzira na njihov redoslijed. Nakon što je korisnik riješio zadatke jedne cjeline kao povratnu će informaciju dobiti postotak točno riješenih zadataka te će imati opciju ponovnog rješavanja netočno riješenih zadataka. Sva su teorijska objašnjenja dostupna samo na engleskom jeziku. Slika 2.4. prikazuje izgled englishpage.com web aplikacije prilikom rješavanja zadataka.



Slika 2.4. Prikaz englishpage.com mrežne stranice [5].

2.5. Prednosti i nedostatci postojećih rješenja

Duolingo se od navedenih postojećih rješenja ističe kao najbolja aplikacija zbog toga što je dostupna na računalima i na mobilnim uređajima, izgledom je privlačna svim uzrastima te, za razliku od ostalih triju stranica, korisnik može pratiti svoj napredak bez dodatnog plaćanja pretplate. Nedostatak je Duolingo aplikacije taj što je potrebno plaćati mjesečnu pretplatu ukoliko korisnik želi dobiti pomoć prilikom netočno riješenog zadatka, želi nesmetano rješavati zadatke bez reklama ili bez čekanja ukoliko je napravio četiri greške tijekom rješavanja zadataka.

Perfect English Grammar ima vrlo veliku bazu zadataka iz područja gramatike te, također kao Duolingo, ima opciju mjesečne pretplate pomoću koje korisnik ostvaruje mogućnost da nakon uspješno riješenog poglavlja dobije certifikat o položenom poglavlju. No, ukoliko korisnik želi pratiti svoj napredak ili želi materijale za učenje gramatike i teorije, mora biti plaćeni korisnik. Također, prilikom netočno riješenog zadatka korisnik ne dobiva povratnu informaciju o načinu ispravljanja ili vrsti pogreške.

English Grammar, za razliku od aplikacija Duolingo i Perfect English Grammar, nema plaćenu verziju, već je sve besplatno. Korisnik prilikom rješavanja zadataka dobije povratnu statističku informaciju o tome koliko je točno riješio zadataka, no nedostatak je ove stranice taj što se zadatci sastoje od jedne rečenice koju korisnik mora nadopuniti tako što upiše jednu do dvije riječi. Zbog svog izgleda stranica nije privlačna predškolcima, učenicima ni studentima.

Stranica englishpage.com je besplatna, isto kao English Grammar, te korisnik na isti način dobije povratnu informaciju prilikom rješavanja zadataka. Osim toga, svaki se tjedan u posebnoj

rubrici nalaze tjedni zadatci koje korisnici mogu rješavati. No, zbog svog izgleda stranica nije privlačna predškolcima, učenicima ni studentima.

Web aplikacija izrađene za potrebe ovog završnog rada, Learn English Tenses, objedinjuje sve prednosti ranije navedenih aplikacija i stranica u jednu cjelinu. Korisnik prije rješavanja zadataka ima dobru teorijsku podlogu koju ne mora proći prije rješavanja zadataka. Prilikom rješavanja zadatka nije ograničen brojem pokušaja, kao što je to slučaj kod aplikacije Duolingo, nema reklama te kada napravi tri puta grešku prilikom rješavanja jednog zadatka, dobije hint u obliku točnog rješenje ili kratkog teorijskog podsjetnika. Korisnik na stranici profila ima uvid u svoj napredak za svako glagolsko vrijeme od kojeg je riješio barem jedan zadatak, a sam izgled stranice privlačan je svim uzrastima.

3. TEHNOLOGIJE KORIŠTENE U IZRADI WEB APLIKACIJE

U ovom su poglavlju opisane tehnologije korištene u izradi web aplikacije.

3.1. HTML

HTML (engl. *Hyper Text Markup Language*) je standardni jezik koji se koristi kako bi se kreirale mrežne stranice. Tim Berners-Lee predstavio ga je 1991. na CERN-u kao jednostavan označni jezik; od tada je napredovao od verzija HTML 2.0 do današnje HTML5 verzije. HTML je kombinacija hipertekstualnog i označnog jezika. Hipertekst definira poveznice između mrežnih stranica, a označni jezik definira tekst dokumenta unutar oznaka. [6] HTML opisuje strukturu mrežne stranice koristeći seriju elemenata koji govore mrežnom pregledniku kako prikazati određeni sadržaj. HTML element je definiran s početnom oznakom, sadržajem i završnom oznakom. [7] Primjer HTML strukture prikazan je na slici 3.1. HTML je u radu korišten u kombinaciji sa CSS-om, JavaScriptom i Djangom. Slika 3.2. prikazuje primjer korištenja HTML-a u kombinaciji s ostalim tehnologijama. Django prvobitno provjerava je li korisnik administrator te, ukoliko je, prikazuje mu formu za odjavljivanje. HTML je korišten za izradu strukture web stranica ovog završnog rada.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>This is a heading!</h1>
  <p>This is a paragraph!</p>
</body>
</html>
```

Slika 3.1. Prikaz strukture HTML dokumenta.

```
65          {% if request.user.is_authenticated %}
66            <form id="logout-form" method="POST" action="{% url 'home:logout' %}">
67              {% csrf_token %}
68            </form>
69            <a href="#" onclick="document.getElementById('logout-form').submit();">
70              <img class="icon" src="{% static 'learntenses/images/logout.png' %}" alt="">
71              Logout
72            </a>
73          {% endif %}
```

Slika 3.2. Primjer korištenja HTML-a.

3.2. CSS

CSS (engl. *Cascading Style Sheets*) opisuje kako se određeni HTML elementi prikazuju na ekranu. Pomoću njega je moguće kontrolirati izgled više mrežnih stranica u isto vrijeme. Moguće ga je implementirati u samom HTML dokumentu ili u odvojenoj .css datoteci. CSS sintaksa sastoji se od selektora na HTML element i deklaracije, a unutar deklaracije navode se parovi svojstava i vrijednosti za to svojstvo. [8] Primjer strukture CSS-a prikazan je slikom 3.3. Pomoću CSS-a stilizirani su HTML elementi tako što su unutar posebne CSS datoteke pojedini HTML elementi pomoću selektora označeni te pomoću parova svojstava i vrijednosti uređeni.

```
.header {
  font-family: 'Times New Roman';
  font-size: 24px;
  color: yellow;
}
```

Slika 3.3. Prikaz strukture CSS-a.

Slika 3.4. prikazuje primjer korištenja CSS-a tako što se na cijeli dokument primjenjuje font 'Roboto', uklanjaju se margine i padding te se pozadinska boja dokumenta stavlja na boju definiranu heksadecimalnim kodom.

```
9  body {
10   font-family: 'Roboto', sans-serif;
11   margin: 0;
12   padding: 0;
13   background-color: #f5f5f5;
14 }
```

Slika 3.4. Primjer korištenja CSS-a.

3.3. JavaScript

JavaScript je programski jezik koji se najčešće koristi za izvođenje dinamičkih skripti na strani korisnika na mrežnim stranicama, no može se koristiti i na strani poslužitelja. JavaScript se primarno koristi u preglednicima. Omogućuje programerima manipuliranje sadržajem mrežnih stranica putem DOM-a (engl. *Document Object Model*), dohvaćanje sadržaja s poslužitelja, pohranjivanje podataka, crtanje grafikona, komuniciranje preglednika s raznim API-jima (engl. *Application Programming Interface*) i više. [9] Koristeći JavaScript, dodane su funkcionalnosti elementima s kojima korisnik ima interakciju. JavaScript omogućuje da se pritiskom na riječ prilikom rješavanja zadataka ista pomakne u tekst zadataka, a dugme s tom riječi oboja drugačijom

bojom. Slika 3.5. prikazuje primjer korištenja JavaScripta. Slika 3.6. prikazuje isječak koda iz rada. Na element kojem je ID 'solve-tasks', dodaje se slušatelj događaja koji se aktivira prilikom klika na element. Dohvaća elemente 'tasks' i 'tenses' te provjerava CSS svojstvo 'display' elementa 'tasks'. Ukoliko je trenutno 'tasks' vidljiv, sakrit će ga tako što će promijeniti vrijednost svojstva 'display' iz 'block' u 'none'. Nakon toga se mijenja vidljivost elementa 'tenses'. Ako je vidljiv, sakrit će ga, a ako je skriven, pokazat će ga.

```
function changeColour() {
    document.getElementById("header").style.color = "red";
}
```

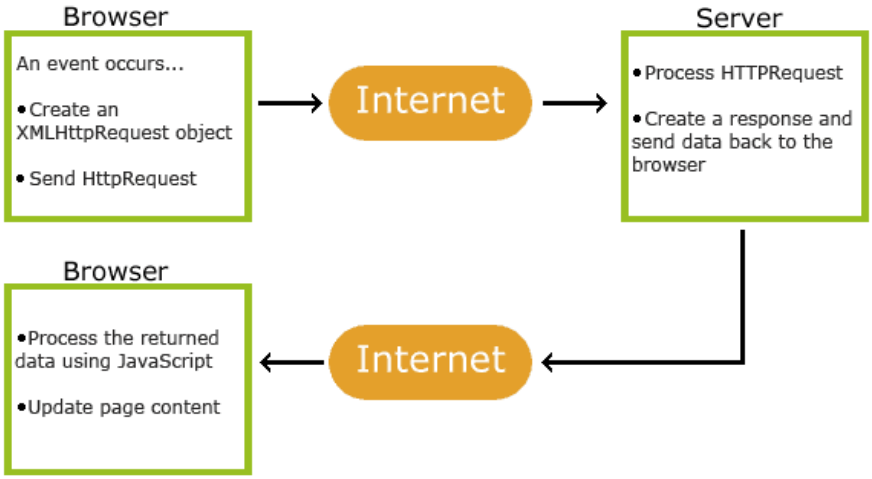
Slika 3.5. Prikaz korištenja JS-a za promjenu boje elementa.

```
document.getElementById('solve-tasks').addEventListener('click', function(event) {
  event.preventDefault();
  var tasks = document.getElementById('tasks');
  var tenses = document.getElementById('tenses');
  if (tenses.style.display === 'block') {
    tenses.style.display = 'none';
  }
  tasks.style.display = tasks.style.display === 'none' ? 'block' : 'none';
});
```

Slika 3.6. Primjer korištenja JavaScripta.

3.4. AJAX

AJAX (engl. *Asynchronous JavaScript and XML*) nije programski jezik, nego kombinacija JavaScripta i HTML DOM-a, koji se koriste za prikaz i korištenje podataka i ugrađenog XMLHttpRequest objekta, koji se koristi za uzimanje podataka s poslužitelja. Koristeći AJAX, moguće je ažurirati mrežnu stranicu bez potrebe za osvježavanjem stranice, tražiti i povlačiti podatke sa servera nakon što je stranica učitana te slati podatke na poslužitelj. [10] Slika 3.7. prikazuje osnovni proces komunikacije između korisnika i poslužitelja koristeći XMLHttpRequest. Na strani korisnika prvo se dogodi neka vrsta događaja kao što je pritisak dugmeta. Kreira se novi XMLHttpRequest i šalje se poslužitelju. Poslužitelj prima i obrađuje XMLHttpRequest, nakon čega kreira odgovor i šalje ga natrag korisniku koji se onda pomoću JavaScripta obrađuje nakon čega se ažuriraju podaci na stranici. Slikom 3.8. prikazan je primjer korištenja AJAX-a. Prvo se šalje zahtjev na URL /home/mark_as_completed/${taskId} gdje {taskId} služi kao identifikator zadatka. To je dinamički dio URL-a i ukazuje na to da je zahtjev povezan s označavanjem zadatka kao riješenog. Metoda koja se koristi za slanje podataka je POST. Zahtjev se sastoji od dvaju zaglavlja - prvo zaglavlje X-CSRFToken služi za sigurnost, a Content-Type označava da je tijelo zahtjeva formatirano kao JSON.



Slika 3.7. Prikaz načina rada AJAX-a.

```
fetch(`/home/mark_as_completed/${taskId}/`, {
    method: 'POST',
    headers: {
        'X-CSRFToken': getCookie('csrftoken'),
        'Content-Type': 'application/json'
    }
})
```

Slika 3.8. Primjer korištenja AJAX-a.

3.5. Python

Python je interpretirani, interaktivni i objektno orijentirani programski jezik. Uključuje module, iznimke, podatke vrlo visoke razine i klase. Podržava programske paradigme izvan objektno orijentiranog programiranja kao što su proceduralno i funkcionalno programiranje. Kombinira nevjerojatnu snagu s vrlo jasnom sintaksom. [11] Posljednja verzija Pythona (3.12.4) izašla je 6. lipnja 2024. godine, a ovaj završni rad je programiran na verziji 3.12.0 koja je bila najnovija u trenutku pisanja rada. Završni je rad pisan u Pythonu u kojemu se definiraju funkcije, obrađuju zahtjevi, upravlja bazom podataka i autentificiraju korisnici. Slikom 3.9. prikazan je primjer funkcije koja se koristi za promjenu lozinke korisnika.

```
def change_password(request):
    if request.method == 'POST':
        form = PasswordChangeForm(request.user, request.POST)
        if form.is_valid():
            user = form.save()
            update_session_auth_hash(request, user)
            return redirect('home:profile')
    else:
        form = PasswordChangeForm(request.user)
    return render(request, 'learntenses/change_password.html', {'form': form})
```

Slika 3.9. Funkcija za promjenu lozinke.

3.6. Django

Django je Python framework koji olakšava kreiranje mrežnih stranica i web aplikacija koristeći Python. Django prati princip DRY, „Don't repeat yourself“ koji ima za cilj smanjiti broj ponavljanja iste informacije. Također, dolazi opremljen značajkama kao što su sustav za registraciju i prijavu, povezanim bazama podataka te CRUD operacijama. Django prati MVT (Model, View, Template) obrazac. *Model* je podatak koji se prikazuje, *view* je zahtjev koji kao odgovor daje template i sadržaj ovisno o zahtjevu koji korisnik šalje, a *template* definira HTML dokument koji sadrži izgled stranice s logikom prikazivanja podataka. [12] Django povezuje modele u Pythonu s bazom podataka, što omogućuje manipuliranje podacima iz baze podataka. Slika 3.10. prikazuje primjer modela UserTask koji se koristi kako bi se povezao korisnik sa zadatkom.

```
65  class UserTask(models.Model):
66      user = models.ForeignKey(UserProfile, on_delete=models.CASCADE)
67      task = models.ForeignKey(Task, on_delete=models.CASCADE)
68      completed = models.BooleanField(default=False)
69      attempts = models.IntegerField(default=0)
70
71      def __str__ (self):
72          return f"{self.user} - {self.task}"
73
74      def mark_as_completed(self):
75          self.completed = True
76          self.save()
77
78      def check_attempts(self):
79          return self.attempts>=3
80
81      def reset_attempts(self):
82          self.attempts = 0
83          self.save()
```

Slika 3.10. Model UserTask.

Osim za modele, Django je korišten za mapiranje hiperveza za odrađivanje specifične funkcije. Slikom 3.11. prikazan je urls.py dokument unutar kojeg se nalaze definicije za sve hiperveze.

```python
from django.urls import path, include
from . import views

app_name = 'home'
urlpatterns = [
    path('', views.index, name='index'),
    path('register/', views.register, name='register'),
    path('login/', views.login, name='login'),
    path('logout/', views.logout, name='logout'),
    path('landing/', views.landing, name='landing'),
    path('profile/', views.profile, name='profile'),
    path('edit_profile/', views.edit_profile, name='edit_profile'),
    path('change_password/', views.change_password, name='change_password'),
    path('<str:tense_name>/learn', views.learn_tense, name='learn_tense'),
    path('<str:tense_name>/tasks', views.task_list, name='task_list'),
    path('<str:tense>/task/<int:task_id>', views.task_detail, name='task_detail'),
    path('increment_attempts/<int:task_id>/', views.increment_attempts, name='increment_attempts'),
    path('mark_as_completed/<int:task_id>/', views.mark_as_completed, name='mark_as_completed'),
    path('reset_attempts/<int:task_id>/', views.reset_attempts, name='reset_attempts'),
    path('create_task/', views.create_task, name='create_task'),
    path('delete_task/<int:task_id>/', views.delete_task, name='delete_task'),
    path('edit_task/<int:task_id>/', views.edit_task, name='edit_task'),
    path('manage_tasks/', views.manage_tasks, name='manage_tasks'),
]
```

Slika 3.11. Sadržaj urls.py datoteke.

4. PRIMJENA OPISANIH TEHNOLOGIJA I PRIKAZ FUNKCIONALNOSTI WEB APLIKACIJE

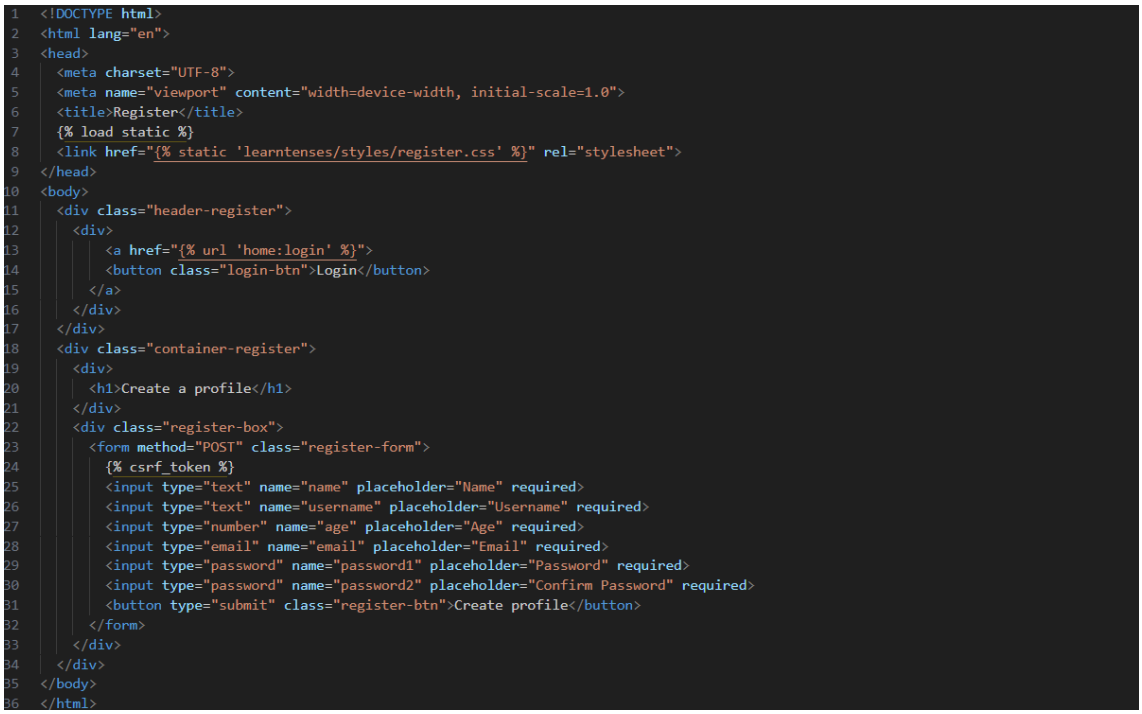
U ovom će se poglavlju objasniti tehnologije korištene u izradi web aplikacije i prikazat će se izgled web aplikacije.

4.1. Korisničko sučelje

Na slikama koje slijede bit će prikazana primjena ranije navedenih tehnologija za izradu završnog rada kao i funkcionalnosti izgleda web aplikacije. Prikazat će se početna stranica prilikom prve posjete, forma za registraciju i prijavu korisnika i forme preko kojih korisnik može uređivati svoje podatke profila. Nadalje, prikazat će se kod koji obavlja spremanje podataka korisničkog profila, početna stranica prilikom prijave na korisnički račun, stranica profila, stranica na kojoj korisnik uči teoriju odabranog glagolskog vremena, stranica na kojoj korisnik odabire zadatke za rješavanje te stranica na kojoj korisnik rješava zadatak.

4.1.1. Registracija i prijava korisnika

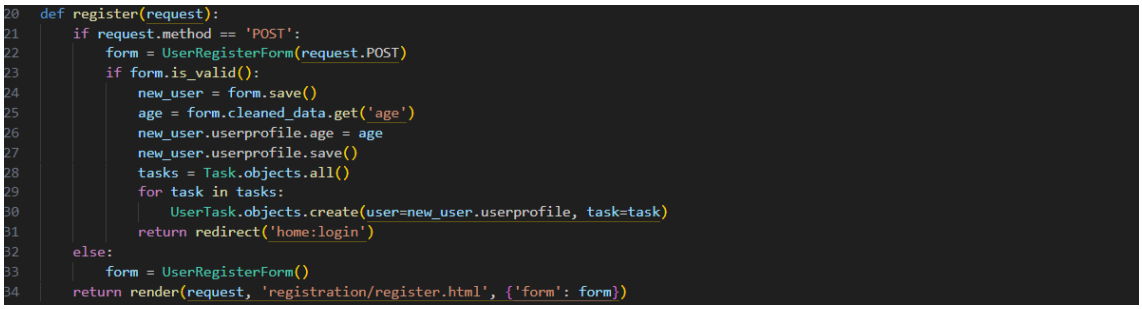
Predložak za prikaz stranice za registraciju korisnika nalazi se na slici 4.1. Kako bi se korisnik uspješno registrirao, potrebno je unijeti podatke kao što su ime, prezime, nadimak, godine, email te lozinku dva puta. Definira se HTML forma s POST metodom koja se koristi za unos podataka za registraciju koje korisnik šalje na poslužitelj. Radi sigurnosti, Django generira CSRF token koji se koristi za zaštitu od CSRF napada.



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Register</title>
  {% load static %}
  <link href="{% static 'learntenses/styles/register.css' %}" rel="stylesheet">
</head>
<body>
  <div class="header-register">
    <div>
      <a href="{% url 'home:login' %}">
      <button class="login-btn">Login</button>
      </a>
    </div>
  </div>
  <div class="container-register">
    <div>
      <h1>Create a profile</h1>
    </div>
    <div class="register-box">
      <form method="POST" class="register-form">
        {% csrf_token %}
        <input type="text" name="name" placeholder="Name" required>
        <input type="text" name="username" placeholder="Username" required>
        <input type="number" name="age" placeholder="Age" required>
        <input type="email" name="email" placeholder="Email" required>
        <input type="password" name="password1" placeholder="Password" required>
        <input type="password" name="password2" placeholder="Confirm Password" required>
        <button type="submit" class="register-btn">Create profile</button>
      </form>
    </div>
  </div>
</body>
</html>
```

Slika 4.1. Predložak za registraciju korisnika.

Na slici 4.2. nalazi se pogled (eng. view) koji upravlja registracijom korisnika. Koristeći UserRegistrationForm(request.POST), instancira se forma UserRegistrationForm s podacima koje je korisnik poslao na poslužitelja. Pomoću is_valid() provjera se format podataka te jesu li sva obavezna polja popunjena. Ukoliko je forma ispravna, podatci se spremaju u bazu podataka, što rezultira stvaranjem novog korisničkog računa. Nakon toga se dohvaćaju svi zadatci unutar baze podataka i povezuju se s korisnikom tako što se stvori novi UserTask objekt. Nakon uspješne registracije, korisnik se preusmjerava na stranicu za prijavu koristeći Django redirect funkciju.



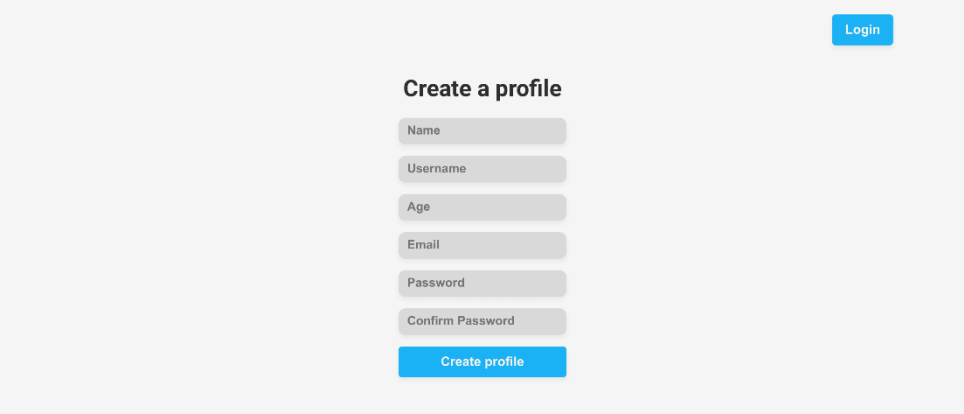
```
def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            new_user = form.save()
            age = form.cleaned_data.get('age')
            new_user.userprofile.age = age
            new_user.userprofile.save()
            tasks = Task.objects.all()
            for task in tasks:
                UserTask.objects.create(user=new_user.userprofile, task=task)
            return redirect('home:login')
    else:
        form = UserRegisterForm()
    return render(request, 'registration/register.html', {'form': form})
```

Slika 4.2. Pogled za registraciju korisnika.

Slika 4.3. prikazuje stranicu koju korisnik vidi prilikom prvog posjeta te svaki put kada se odjavi sa svog računa. Pritiskom na dugme 'GET STARTED' bit će preusmjeren na stranicu

prikazanu na slici 4.4. gdje, prilikom registracije, popunjava polja sa svojim imenom i prezimenom, nadimkom, godinama, emailom te lozinkom koju mora dva puta unijeti.

Slika 4.3. Prikaz početne stranice.



Slika 4.4. Prikaz forme za registraciju.

Na slici 4.5. nalazi se predložak za prikaz stranice na kojoj se korisnik prijavljuje na svoj korisnički račun. Definirana je forma isto kao kod postupka za registraciju koja također sadrži CSRF token za zaštitu.

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login</title>
  {% load static %}
  <link href="{% static 'learntenses/styles/login.css' %}" rel="stylesheet">
</head>
<body>
  <div class="container-login">
    <div>
      <h1>Log in</h1>
    </div>
    <div class="login-box">
      <form method="POST" class="login-form">
        {% csrf_token %}
        <input type="text" name="username" placeholder="Username" class="login-input" required>
        <input type="password" name="password" placeholder="Password" class="login-input" required>
        <button type="submit" class="login-btn">Log in</button>
      </form>
    </div>
  </div>
</body>
</html>
```

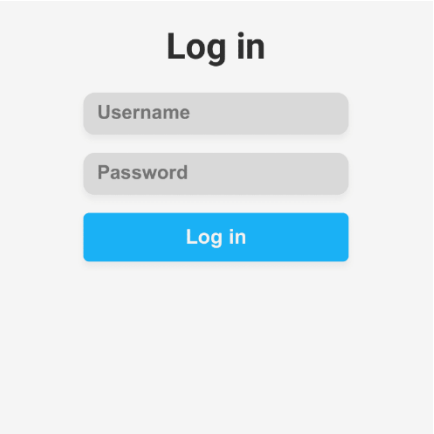
Slika 4.5. Predložak za prijavu.

Na slici 4.6. prikazan je pogled koji upravlja prijavom korisnika. Nakon što je provjerena valjanost unesenih podataka, dohvaćaju se korisničko ime i lozinka iz unesene forme i pokušava se autentificirati korisnik s danim korisničkim imenom i lozinkom. Ukoliko korisnik postoji, prijavljuje ga se u sustav i preusmjerava na početnu stranicu. Ukoliko korisnik ne postoji, preusmjerava se natrag na stranicu za prijavu.

```python
def login(request):
    if request.method == 'POST':
        form = AuthenticationForm(data=request.POST)
        if form.is_valid():
            username = form.cleaned_data.get('username')
            password = form.cleaned_data.get('password')
            user = authenticate(username=username, password=password)
            if user is not None:
                auth_login(request, user)
                return redirect('home:index')
            else:
                return redirect('home:login')
    else:
        form = AuthenticationForm()
    return render(request, 'registration/login.html', {'form': form})
```

Slika 4.6. Pogled za prijavu.

Na slici 4.7. prikazana je forma koju korisnik popunjava prilikom prijave na svoj račun, a potrebno je unijeti nadimak i lozinku.



Slika 4.7. Prikaz forme za prijavu.

4.1.2. Uređivanje podataka profila

Slika 4.8. prikazuje predložak stranice za uređivanje profila unutar web aplikacije. Forma za uređivanje profila koristi POST metodu za slanje podataka na server te se unutar forme dodaje CSRF radi sigurnosti. Na slici 4.9. prikazuje se pogled koji se koristi za uređivanje podataka profila.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    {% load static %}
    <link href="{% static 'learntenses/styles/edit_profile.css' %}" rel="stylesheet">
    <title>Edit your profile</title>
  </head>
  <body>
    <div class="back-button">
      <a href="{% url 'home:profile' %}">
        <button class="back-btn">Back</button>
      </a>
    </div>
    <div class="edit-profile-header">
      <h2>Edit your profile</h2>
    </div>
    <div class="edit-profile-container">
      <form method="POST" action="{% url 'home:edit_profile' %}">
        {% csrf_token %}
        <div class="form-field">
          {{ user_form.username.label_tag }}
          {{ user_form.username }}
        </div>
        <div class="form-field">
          {{ user_form.email.label_tag }}
          {{ user_form.email }}
        </div>
        <div class="form-field">
          {{ profile_form.name.label_tag }}
          {{ profile_form.name }}
        </div>
        <div class="form-field">
          {{ profile_form.age.label_tag }}
          {{ profile_form.age }}
        </div>
        <button type="submit" class="save-btn">Save changes</button>
      </form>
    </div>
    <div class="change-password">
      <h2>Want to change your password?</h2>
      <a href="{% url 'home:change_password' %}">Click here!</a>
    </div>
  </body>
</html>
```

Slika 4.8. Predložak za uređivanje profila.

```python
def edit_profile(request):
    if not hasattr(request.user, 'userprofile'):
        UserProfile.objects.create(user=request.user, age=0)
    if request.method == 'POST':
        user_form = UserEditForm(request.POST, instance=request.user)
        profile_form = UserProfileEditForm(request.POST, instance=request.user.userprofile)
        if user_form.is_valid() and profile_form.is_valid():
            user_form.save()
            profile_form.save()
            return redirect('home:profile')
    else:
        user_form = UserEditForm(instance=request.user)
        profile_form = UserProfileEditForm(instance=request.user.userprofile)
    return render(request, 'learntenses/edit_profile.html', {'user_form': user_form, 'profile_form': profile_form})
```

Slika 4.9. Pogled za uređivanje profila.

Slike 4.10. i 4.11. prikazuju HTML predložak za stranicu koja korisniku omogućava promjenu lozinke i pogled koji se koristi za promjenu lozinke.

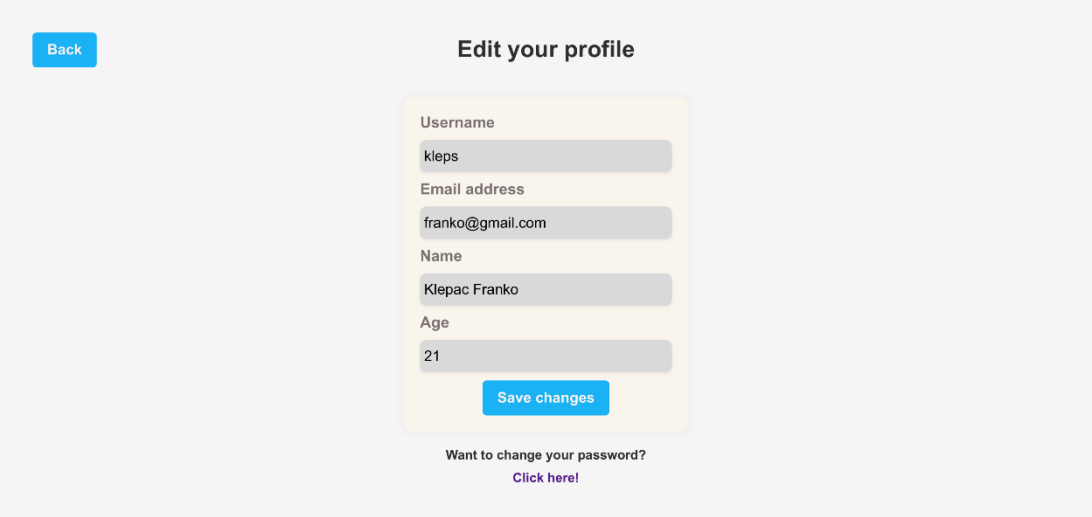
```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Change password</title>
    {% load static %}
    <link href="{% static 'learntenses/styles/change_password.css' %}" rel="stylesheet">
  </head>
  <body>
    <div class="back-button">
      <a href="{% url 'home:edit_profile' %}">
        <button class="back-btn">Back</button>
      </a>
    </div>
    <div class="change-pwd-header">
      <h2>Change your password</h2>
    </div>
    <div class="change-pwd-container">
      <form method="POST" action="{% url 'home:change_password' %}">
        {% csrf_token %}
        <div class="form-field">
          <label for="{{ form.old_password.id_for_label }}">Old password</label>
          {{ form.old_password }}
        </div>
        <div class="form-field">
          <label for="{{ form.new_password1.id_for_label }}">New password</label>
          {{ form.new_password1 }}
        </div>
        <div class="form-field">
          <label for="{{ form.new_password2.id_for_label }}">Confirm new password</label>
          {{ form.new_password2 }}
        </div>
        <button type="submit" class="change-pwd-btn">Change password</button>
      </form>
    </div>
    {% for field in form %}
      {% for error in field.errors %}
        <div class="error">{{ error }}</div>
      {% endfor %}
    {% endfor %}
  </body>
</html>
```

Slika 4.10. Predložak za promjenu lozinke.

```python
def change_password(request):
    if request.method == 'POST':
        form = PasswordChangeForm(request.user, request.POST)
        if form.is_valid():
            user = form.save()
            update_session_auth_hash(request, user)
            return redirect('home:profile')
    else:
        form = PasswordChangeForm(request.user)
    return render(request, 'learntenses/change_password.html', {'form': form})
```

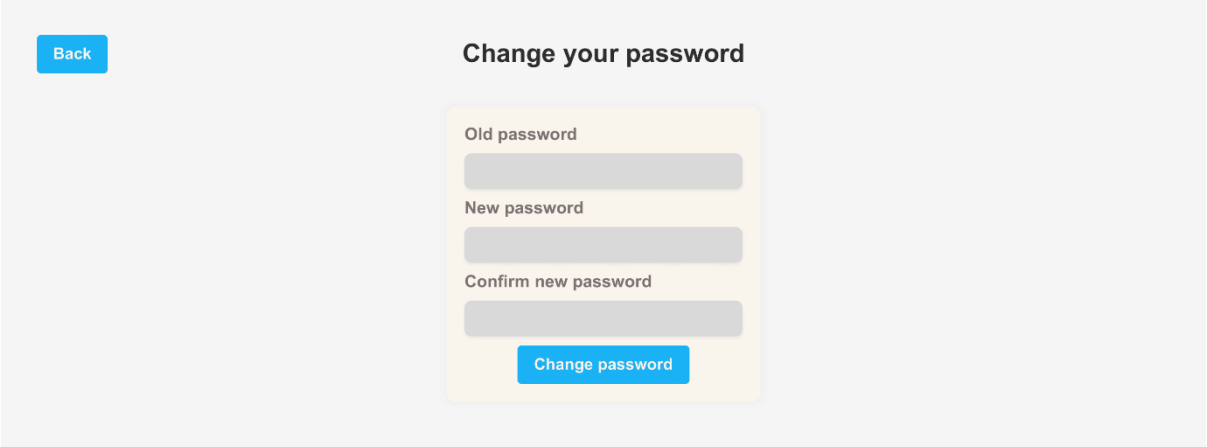
Slika 4.11. Pogled za promjenu lozinke.

Slikom 4.12. prikazana je stranica na kojoj korisnik ima mogućnost urediti svoje osobne podatke.



Slika 4.12. Stranica za uređivanje profila.

Korisnik ima mogućnost uređivanja svog aliasa, adresu e-pošte, ime i godine. Ukoliko želi promijeniti svoju lozinku, mora pritisnuti na 'Click here!' koji se nalazi ispod forme za izmjenu podatka računa. Stranica za promjenu lozinke prikazana je slikom 4.13.



Slika 4.13. Stranica za promjenu lozinke.

Kako bi korisnik promijenio lozinku svog računa, mora prvo unijeti svoju trenutnu lozinku te novu lozinku dva puta radi sigurnosti.

4.1.3. Spremanje podataka korisnika

Slika 4.14. prikazuje model UserProfile koji se kreira prilikom registracije novog korisnika. U sedmoj liniji koda prikazuje se vanjski ključ koji povezuje UserProfile s ugrađenim Django modelom User. Svaki korisnik može imati samo jedan profil te se prilikom brisanja korisnika briše i njegov profil. U idućim linijama koda pohranjuju se ime (name), godine (age) i datum registracije (date_joined). Model UserProfile također sadrži dvije metode. Prva metoda, 'completed_tasks_by_tense' vraća broj završenih zadataka grupiranih po vremenu. Koristi 'self.usertask_set' za pristup svim UserTask objektima povezanih s profilom te ih filtrira po završenim zadatcima i onda ih grupira po vremenu 'task__tense' i broji koliko ih ima po svakoj grupi. Druga metoda, '__str__' koristi se za vraćanje korisničkog imena profila.

```
6  class UserProfile(models.Model):
7      user = models.OneToOneField(User, on_delete=models.CASCADE)
8      name = models.CharField(max_length=100)
9      age = models.IntegerField()
10     date_joined = models.DateTimeField(auto_now_add=True)
11
12     def completed_tasks_by_tense(self):
13         return self.usertask_set.filter(completed=True).values('task__tense').annotate(count=models.Count('task__tense'))
14
15     def __str__(self):
16         return self.user.username
```

Slika 4.14. Model UserProfile.

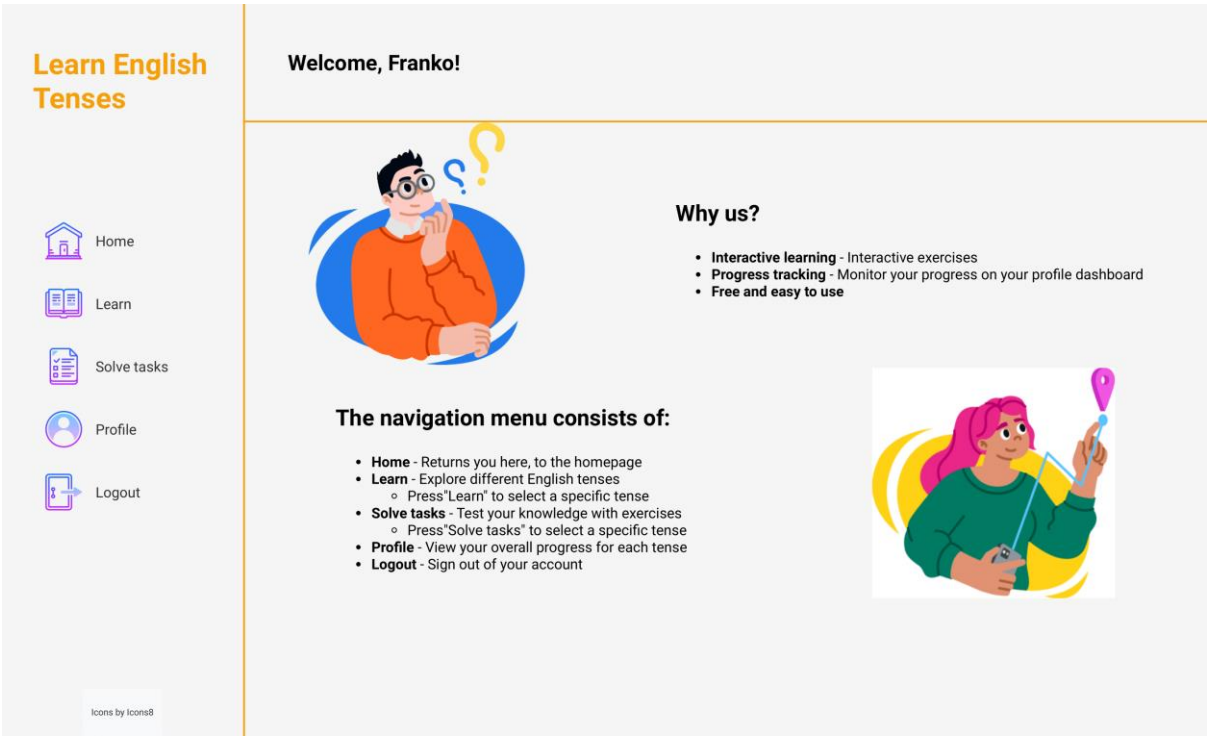
Slika 4.15. prikazuje model UserTask koji predstavlja zadatak korisnika. U liniji 66 pomoću vanjskog ključa povezuje se UserTask s modelom UserProfile, a linija nakon povezuje UserTask s modelom Task. Ukoliko dolazi do brisanja korisničkog profila ili zadatka, svi povezani UserTask objekti također će se obrisati. Pomoću polja 'completed' označava se je li zadatak završen ili ne, a početna je vrijednost 'False'. Metoda 'mark_as_completed' označava zadatak kao završen (completed=True) i sprema promjene u bazu podataka sa 'self.save()'. Metode check_attempts i reset_attempts koriste se kako bi se prikazivali hintovi korisniku prilikom rješavanja zadataka.

```python
class UserTask(models.Model):
    user = models.ForeignKey(UserProfile, on_delete=models.CASCADE)
    task = models.ForeignKey(Task, on_delete=models.CASCADE)
    completed = models.BooleanField(default=False)
    attempts = models.IntegerField(default=0)

    def __str__ (self):
        return f"{self.user} - {self.task}"

    def mark_as_completed(self):
        self.completed = True
        self.save()

    def check_attempts(self):
        return self.attempts>=3

    def reset_attempts(self):
        self.attempts = 0
        self.save()
```

Slika 4.15. Model UserTask.

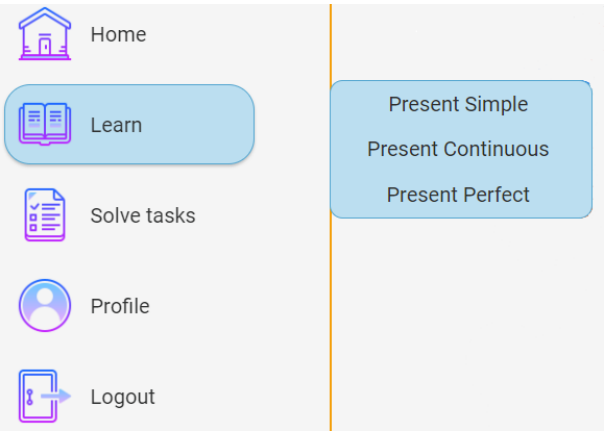
4.1.4. Početna stranica

Na početnoj stranici koja je prikazana slikom 4.16. korisnik se upoznaje sa sadržajem navigacije, gdje mu je svaka opcija objašnjena. Navigacijska se traka nalazi s lijeve strane, a od ponuđenih opcija ima povratak na početnu stranicu, odabir glagolskog vremena za učenje, odabir zadataka iz pojedinog glagolskog vremena, stranicu za pregled profila i odjavu sa svog računa.



Slika 4.16. Prikaz početne stranice.

Na slici 4.17. nalazi se izbornik koji se korisniku prikaže prilikom pritiska na opciju 'Learn' iz kojeg onda može odabrati glagolsko vrijeme koje želi učiti. Isti će se izbornik prikazati i pritiskom na opciju 'Solve tasks'.



Slika 4.17. Pregled glagolskih vremena za učenje.

4.1.5. Stranica profila

Slika 4.18. prikazuje predložak za stranicu profila korisnika na kojoj korisnik ima uvid u statistiku riješenih zadataka. U zaglavlju se korisniku prikazuje pozdravna poruka s datumom pridruživanja te opcija uređivanja podataka korisničkog računa. Tijelo stranice sadrži statistiku o riješenim zadatcima koja se prikazuje koristeći 'progress-bar' koji se širi proporcionalno broju riješenih zadataka. Pored tog 'progress-bar-a' nalazi se prozor u kojem će se prikazati kvačica kada korisnik uspješno riješi sve zadatke iz određenog glagolskog vremena. Slikom 4.19. prikazan je pogled koji obavlja posao prikazivanja stranice profila.

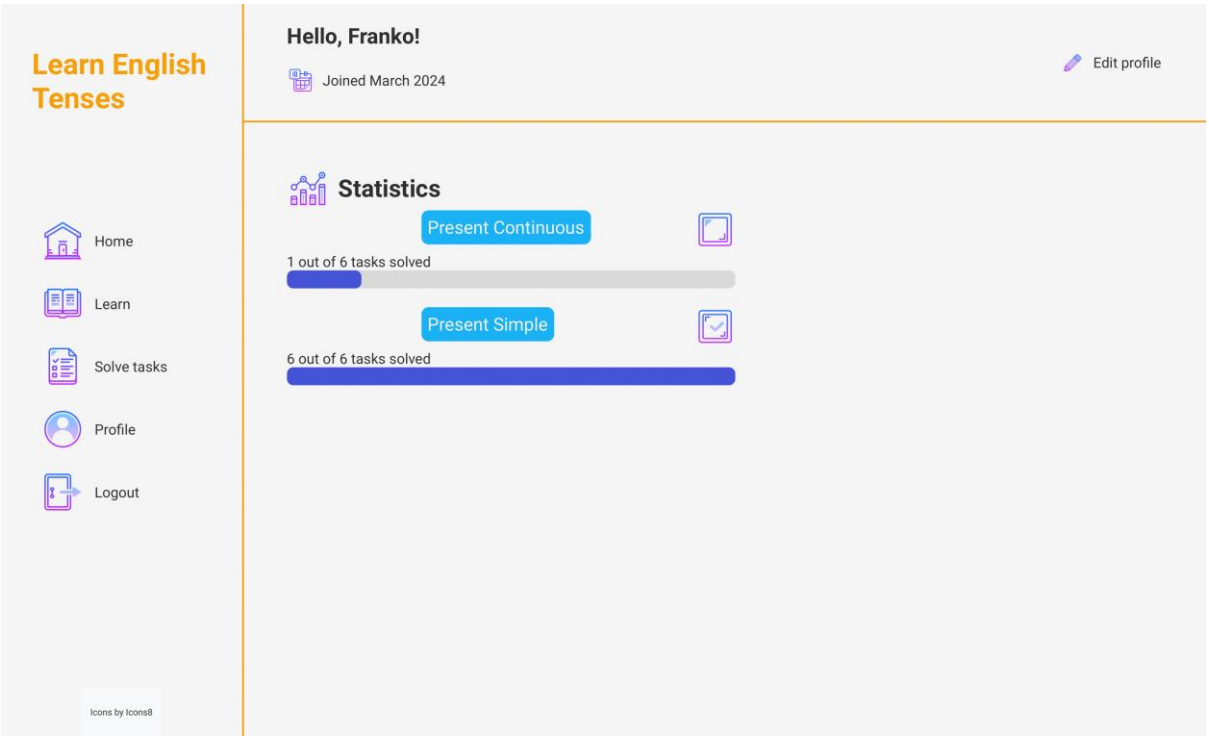
```
<div class="profile-container">
  <div class="profile-header">
    <div>
      <h2>Hello, {{request.user.userprofile.name.split.1}}!</h2>
      <div class="joined-info">
        <img class="icon-joined" src="{% static 'learntenses/images/calendar.png' %}" alt="">
        <p>Joined {{ request.user.date_joined|date:"F Y" }}</p>
      </div>
    </div>
    <div>
      <div class="edit-profile">
        <a href="{% url 'home:edit_profile' %}">
          <div class="edit-info">
            <img class="edit-icon" src="{% static 'learntenses/images/edit.png' %}" alt="">
            <p>Edit profile</p>
          </div>
        </a>
      </div>
    </div>
  </div>
  <div class="profile-body">
    <div class="stats">
     <img class="stats-icon" src="{% static 'learntenses/images/statistics.png' %}" alt="">
     Statistics
    </div>
  {% for tense in user.userprofile.completed_tasks_by_tense %}
    <div class="tense-stats">
      <div class="tense-header">
        <div class="tense" data-tense="{{ tense.task__tense }}">
          <div class="tense-name">
            <p>{{ tense.task__tense }}</p>
          </div>
        </div>
        <div>
          {% if tense.count == 6 %}
            <img src="{% static 'learntenses/images/checked-checkbox.png' %}" class="checkbox-icon" alt="All tasks completed">
          {% else %}
            <img src="{% static 'learntenses/images/unchecked-checkbox.png' %}" class="checkbox-icon" alt="Tasks not completed">
          {% endif %}
        </div>
      </div>
      <div class="tense-progress">
        <div class="progress-num">
          {{ tense.count }} out of 6 tasks solved
        </div>
        <div class="progress-bar">
          {% if tense.count == 1 %}
            <div class="progress-bar-fill" style="width: 16.67%;"></div>
          {% elif tense.count == 2 %}
            <div class="progress-bar-fill" style="width: 33.33%;"></div>
          {% elif tense.count == 3 %}
            <div class="progress-bar-fill" style="width: 50%;"></div>
          {% elif tense.count == 4 %}
            <div class="progress-bar-fill" style="width: 66.67%;"></div>
          {% elif tense.count == 5 %}
            <div class="progress-bar-fill" style="width: 83.33%;"></div>
          {% elif tense.count == 6 %}
            <div class="progress-bar-fill" style="width: 100%;"></div>
          {% endif %}
        </div>
      </div>
    </div>
  {% endfor %}
  </div>
```

Slika 4.18. Predložak za stranicu profila.

```
def profile(request):
    return render(request, 'learntenses/profile.html')
```

Slika 4.19. Pogled za prikaz stranice profila.

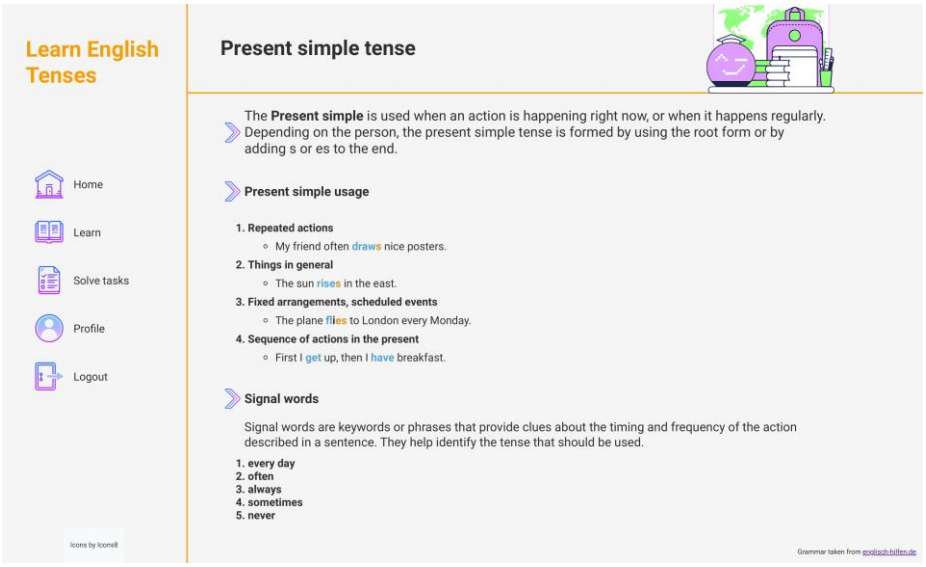
Na slici 4.20. prikazana je stranica na koju će korisnik biti preusmjeren prilikom pritiska na opciju 'Profile' koja se nalazi na navigacijskoj traci.



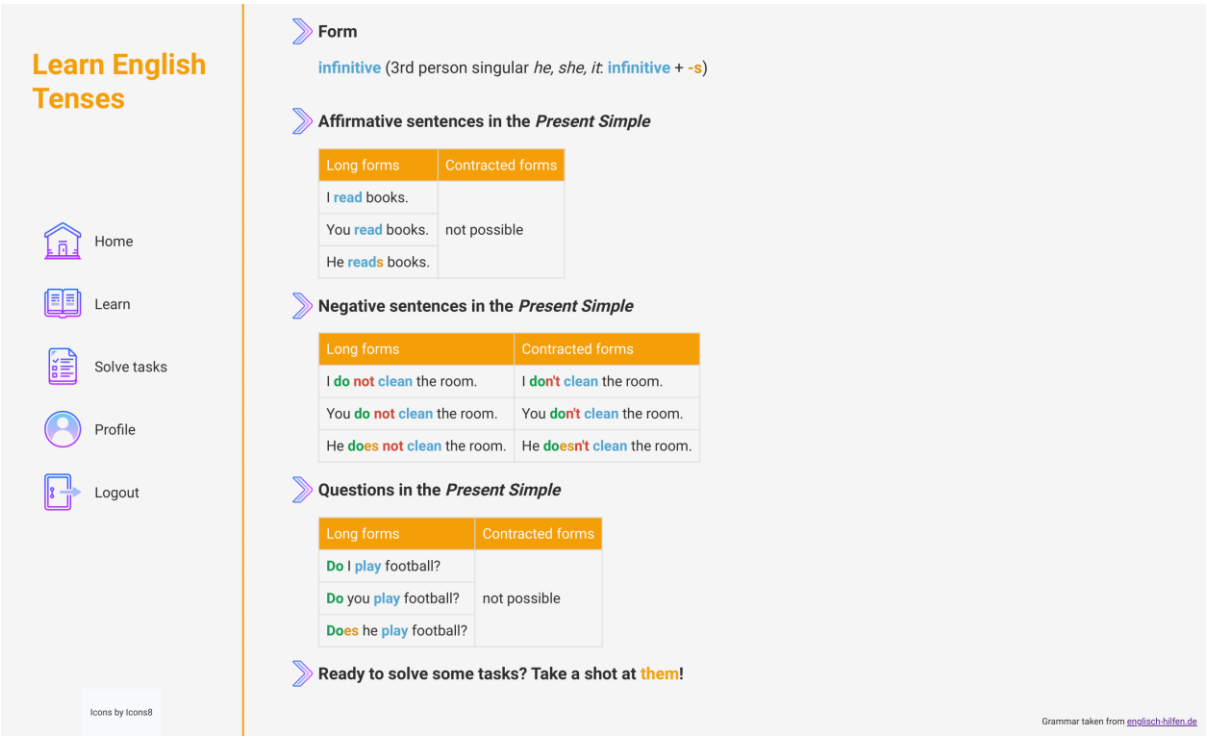
Slika 4.20. Stranica profila.

4.1.6. Stranica s teorijom

Odabirom opcije 'Learn' i pritiskom na jedno od ponuđenih glagolskih vremena, korisnik je preusmjeren na stranicu prikazanu slikama 4.21. i 4.22. Na ovoj stranici korisnik ima teorijsku pozadinu pojedinog glagolskog vremena s primjerima o tvorbi izjavnih, negacijskih i upitnih rečenica kao i pravilima za i načinima upotrebe tog vremena u kontekstu.



Slika 4.21. Prikaz stranice za učenje glagolskog vremena.



Slika 4.22. Prikaz stranice za učenje glagolskog vremena.

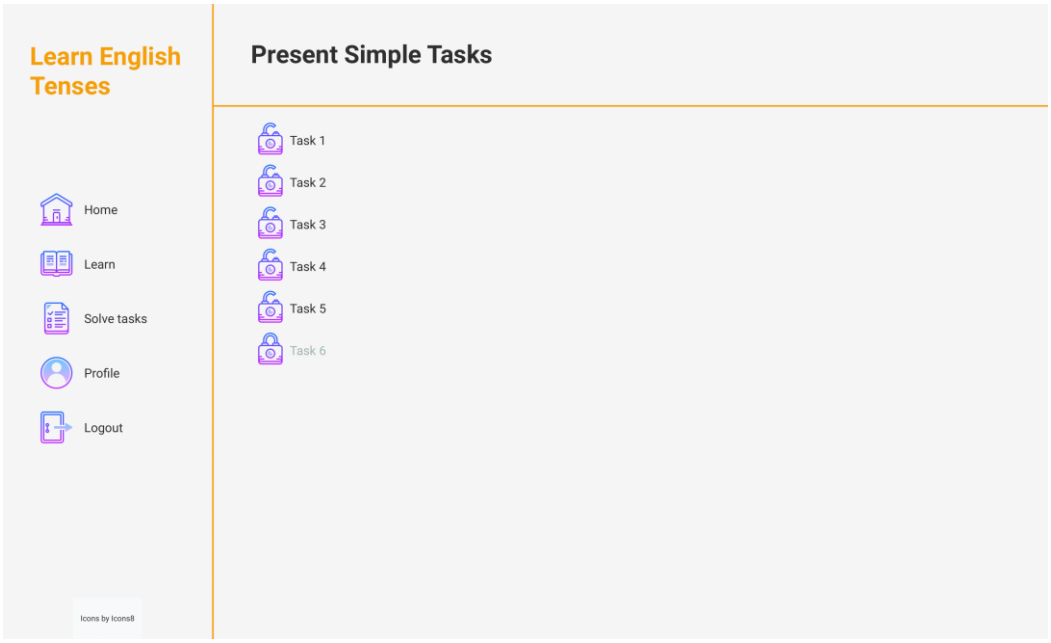
4.1.7. Popis zadataka

Kako bi korisnik došao na stranicu s zadatcima glagolskog vremena, treba odabrati opciju 'Solve tasks' i pritisnuti na jedno od ponuđenih glagolskih vremena. Slikom 4.23. prikazan je predložak stranice na kojoj korisnik ima pregled svih zadataka, zadataka koje je riješio te zadataka kojima još uvijek nema pristup. Zadatci kojima korisnik nema pristup imaju sličicu zaključanog lokota 'locked.png', dok zadatci koji su mu dostupni za rješavanje imaju sličicu otključanog lokota 'unlocked.png'. Kako bi dobio pristup zadatku, korisnik prvo mora točno riješiti prethodni zadatak sa 100 % točnošću. Je li korisnik riješio prijašnji zadatak provjerava se koristeći 'user_task.locked', te se pomoću filtera 'yesno' ime zadatka boja bojom ovisno o tome je li zadatak dostupan za rješavanje ili ne. Ako je dostupan za rješavanje, ime zadatka pretvara se u hipervezu, a kada korisnik pritisne na ime zadatka, bit će preusmjeren na stranicu tog zadatka.

```
{% extends 'learntenses/base.html' %}

{% block extra_css %}
  {% load static %}
  <link rel="stylesheet" href="{% static 'learntenses/styles/task_list.css' %}">
{% endblock %}

{% block content %}
  <div class="tasks-container">
    <div class="tasks-header">
      <h1>{{ tense_name }} Tasks</h1>
    </div>
    <div class="task-list">
      <ul>
        {% for user_task in user_tasks %}
          <li>
            <div class="task-item">
              <div class="task-icon">
                {% if not user_task.locked %}
                  <img src="{% static 'learntenses/images/unlocked.png' %}" class="task-icon">
                {% else %}
                  <img src="{% static 'learntenses/images/locked.png' %}" class="task-icon">
                {% endif %}
              </div>
              <div class="task-name" style="color: {{ user_task.locked|yesno:'#A1B5B5,#2E2F2F' }};">
                {% if not user_task.locked %}
                  <a href="{% url 'home:task_detail' user_task.task.tense user_task.task.id %}" style="color: inherit;">
                    Task {{ user_task.task.name }}
                  </a>
                {% else %}
                  Task {{ user_task.task.name }}
                {% endif %}
              </div>
            </div>
          </li>
        {% endfor %}
      </ul>
    </div>
  </div>
{% endblock %}
```

Slika 4.23. Predložak stranice s popisom zadataka.

Slikom 4.24. prikazan je izgled stranice na kojoj korisnik vidi popis zadataka odabranog glagolskog vremena.



Slika 4.24. Prikaz stranice za odabir zadataka.

4.1.8. Rješavanje zadataka

Predložak za rješavanje zadataka prikazan je slikom 4.25. Rečenica trenutnog zadatka prikazuje se pomoću task.sentence|safe. Django template filter safe koristi se kako bi se HTML unutar rečenice ispravno prikazao. Riječi koje se korisniku prikazuju kao moguća rješenja prikazuju se na stranici koristeći kod na liniji 31. Pomoću for petlje, dobiva se svaka riječ tako što se razdvoji string task.words te se na liniji 32, za svaku riječ iz petlje, kreira gumb s klasom word-button, atributom data-word koji sadrži tu riječ, i prikazuje ju kao tekst gumba. Nakon toga, kreira se gumb za provjeru s klasom check-button koji sadrži podatke o zadatku kao što su ID zadatka, razina, glagolsko vrijeme i ispravne riječi tog zadatka. Ukoliko je zadatak točno riješen, prikazuje se dugme koje vodi korisnika na idući zadatak.

```
{% extends 'learntenses/base.html' %}
{% load replace_filter %}
{% block extra_css %}
  {% load static %}
  <link rel="stylesheet" href="{% static 'learntenses/styles/task_detail.css' %}">
{% endblock %}

{% block content %}
    {% load static %}

    <div class="solve-task-container">
        <div class="solve-task-header">
            <h1>{{ task.get_tense_display }} Tense</h1>
        </div>
        <div class="solve-task-body">
            <div class="task-name">
                <div>
                    <img src="{% static 'learntenses/images/task.png' %}" class="icon" alt="">
                </div>
                <div>
                    <h1>Task {{ task.name }}</h1>
                </div>
            </div>
            <div class="task-result">
                <p id ="result"></p>
            </div>
            <div class="task-text">
                <p>{{ task.sentence|safe }}</p>
            </div>
            <div class="task-options">
                {% for word in task.words.split %}
                <button class="word-button" data-word="{{ word }}">{{ word }}</button>
                {% endfor %}
            </div>
            <br>
            <div class="task-result-buttons">
                <button class="check-button"
                data-task-id="{{task.id}}"
                data-level="{{task.level}}"
                data-tense="{{task.tense}}"
                style="display:block;"
                data-correct-words="{{task.correct_words}}">Check</button>
                <button class="next-task-btn" id="next-task-button" style="display: none;"></button>
            </div>
        </div>
    </div>
```

Slika 4.25. Predložak za rješavanje zadataka.

Slika 4.26. prikazuje pogled koji upravlja detaljima zadatka na temelju zadanog vremena i ID-a zadatka. Na osnovu ID zadatka koji je odabran, dohvaća se zadatak i profil trenutno prijavljenog korisnika te se na osnovu zadatka i korisnika pokušava dohvatiti instancu UserTask koja povezuje trenutnog korisnika sa zadatkom. Ukoliko ju nije moguće dohvatiti, dobiva se HTTP 404 odgovor.

```python
@csrf_exempt
@require_http_methods(['GET','POST'])
def task_detail(request, tense, task_id):
    task = get_object_or_404(Task, id=task_id, tense=tense)
    user_profile = UserProfile.objects.get(user=request.user)
    user_task = get_object_or_404(UserTask, user=user_profile, task=task)
    task.sentence = replace_with_span(task.sentence)
    previous_task = Task.objects.filter(id__lt=task_id, tense=tense).order_by('-id').first()

    if not user_task.completed and (previous_task is not None and not UserTask.objects.filter(user=user_profile, task=previous_task, completed=True).exists()):
        return HttpResponseForbidden()

    return render(request, 'learntenses/task_detail.html', {'task': task})
```

Slika 4.26. Pogled za rješavanje zadataka.

Slika 4.27. prikazuje JavaScript kod koji dodaje funkcionalnost gumba za provjeru odgovora. Na dugme s klasom .check-button dodaje se click event listener koji, kada se dugme pritisne, izvršava funkciju definiranu unutar listenera. Izvlače se točne riječi iz data-correct-words atributa gumba, razdvajaju se u niz, uspoređuju s riječima koje je korisnik odabrao i pohranjuju u niz clickedWordsText. Ukoliko je odgovor točan, prikazuje se poruka „Correct!“ i onemogućuju se svi gumbi s mogućim odabirom riječi. Ukoliko je odgovor netočan, prikazuje se poruka „Incorrect! Try again.“ i korisniku se omogućuje ponovno pokušavanje rješavanja zadatka. Ako je odgovor točan, šalje se POST zahtjev na /home/mark_as_completed/${taskId}/ kako bi se zadatak označio kao dovršen. Ovisno o tome je li korisnik na posljednjoj razini (razina 6), prikazuje se odgovarajuća poruka na gumbu za sljedeći zadatak. Kada korisnik točno riješi zadnji zadatak, umjesto 'Next Task' teksta na gumbu će se prikazati tekst 'Congratulations' i prilikom pritiska na gumb korisnik će biti preusmjeren na stranicu s popisom zadataka.

```javascript
document.querySelectorAll('.check-button').forEach(checkButton => {
    checkButton.addEventListener('click', () => {
        let correctWords = checkButton.dataset.correctWords.split(' ');
        const resultElement = document.getElementById('result');
        const currentLevel = parseInt(checkButton.getAttribute('data-level'));
        let clickedWordsText = clickedWords.map(button => button.textContent);
        if (JSON.stringify(correctWords) === JSON.stringify(clickedWordsText)) {
            resultElement.textContent = 'Correct!';
            resultElement.classList.add('correct');
            resultElement.classList.remove('incorrect');

            document.querySelectorAll('.word-button').forEach(button => {…

            const taskId = checkButton.getAttribute('data-task-id');

            if (currentLevel === 6) {…
            else {…

            fetch(`/home/mark_as_completed/${taskId}/`, {…
            .then(response => response.json())
            .then(data => {…
        }
        else {
            resultElement.textContent = 'Incorrect! Try again';
            resultElement.classList.add('incorrect');
            resultElement.classList.remove('correct');
            selectedWordsElement.textContent = '';
            clickedWords = [];
            document.querySelectorAll('.filled').forEach(element => {
                element.textContent = '_____';
                element.classList.remove('filled');
                element.classList.add('blank');
            });

            document.querySelectorAll('.selected').forEach(element => {
                element.classList.remove('selected');
            });

        }
    });
});
```

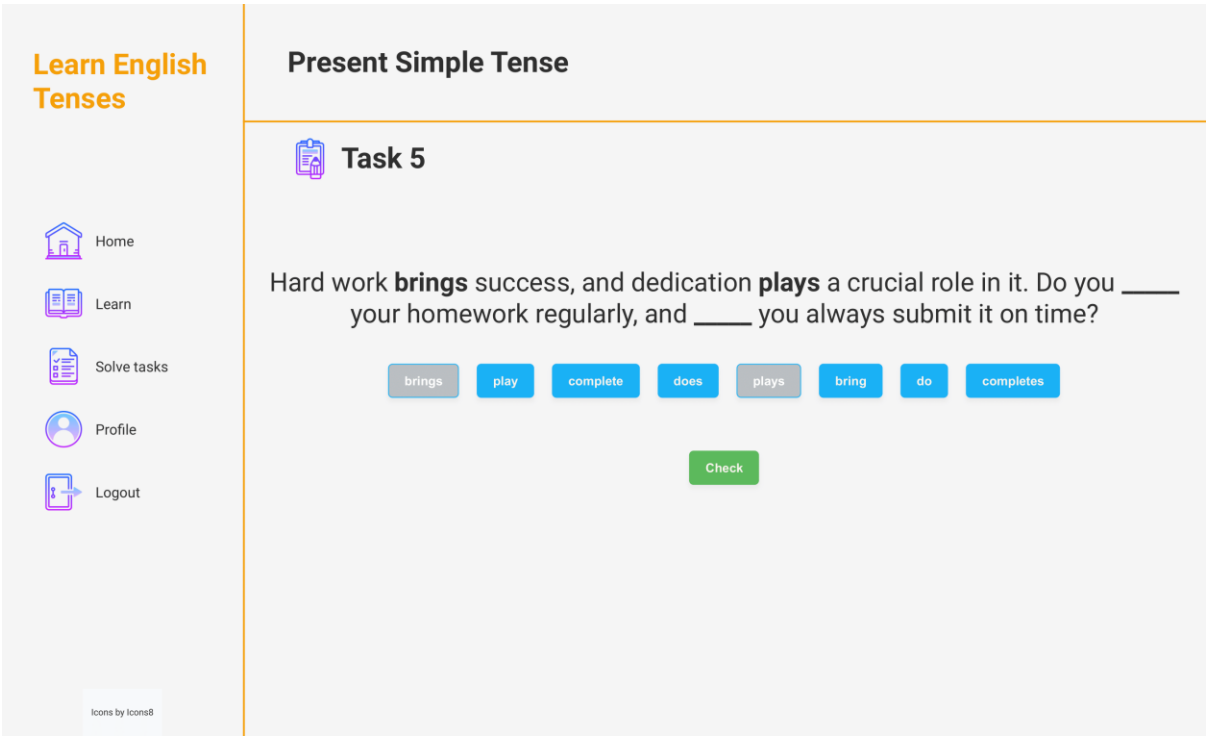
Slika 4.27. JavaScript kod za provjeru rješenja zadatka.

Na slici 4.28. nalazi se pogled koji se poziva unutar JavaScript koda na slici 4.27. Pogled prima identifikator zadatka koji je točno riješen i kojega je potrebno označiti kao dovršenog. To se radi na način da pozove metodu mark_as_completed() na dohvaćenom objektu UserTask.

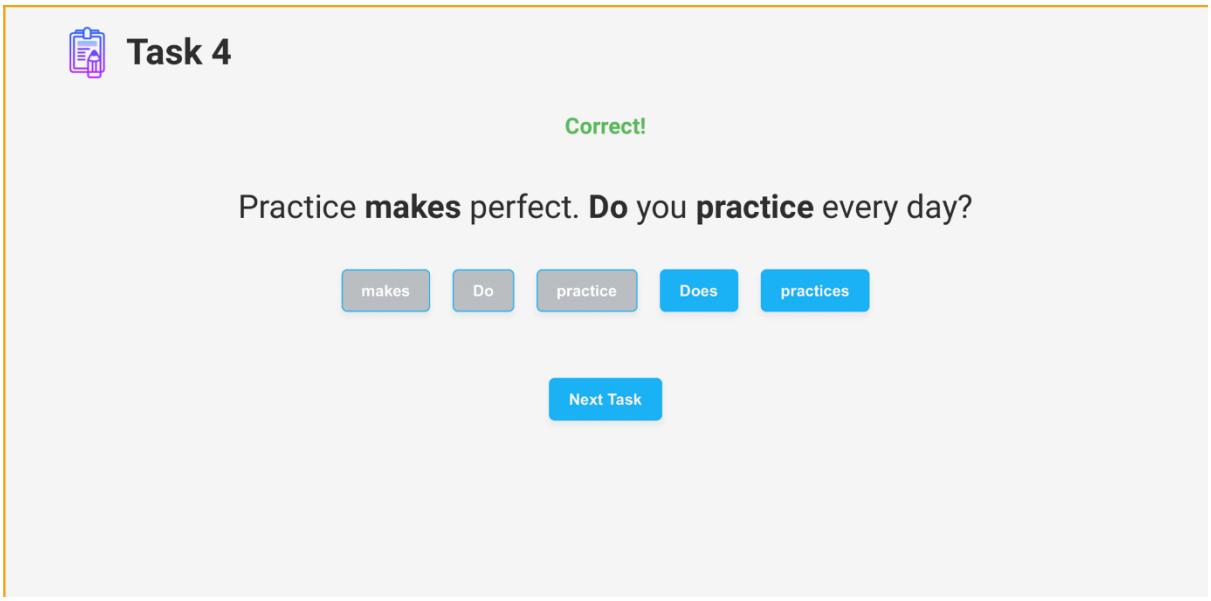
```
153 @csrf_exempt
154 @require_POST
155 def mark_as_completed(request, task_id):
156     user_task = get_object_or_404(UserTask, user=request.user.userprofile, task_id=task_id)
157     user_task.mark_as_completed()
158     return JsonResponse({'status': 'success'})
```

Slika 4.28. Pogled za označavanje zadatka kao riješenog.

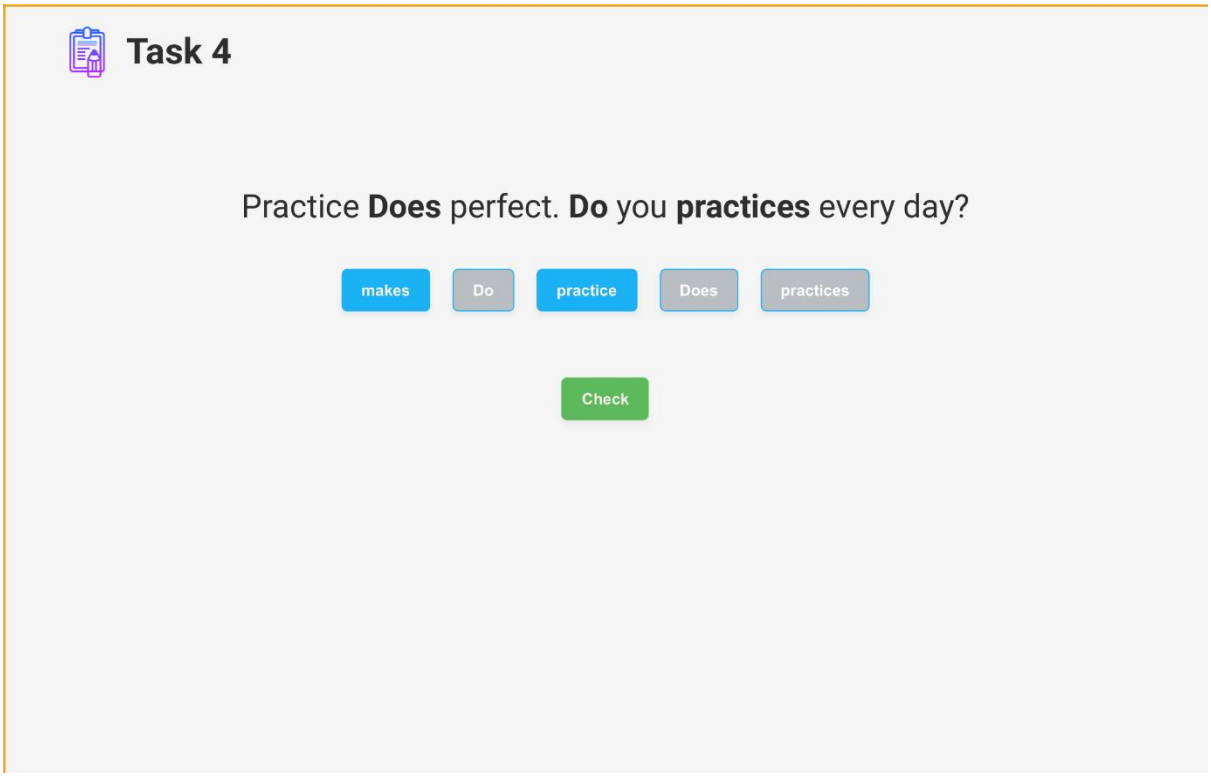
Slikom 4.29. prikazana je stranica na kojoj korisnik rješava zadatke. Pritiskom na dugme koje predstavlja riječ, pritisnuta će se riječ pojaviti na prvom slobodnom mjestu u rečenici, a slobodna su mjesta predstavljena praznim crtama. Nakon što je riječ odabrana, dugme koje predstavlja tu riječ mijenja boju. Pritiskom dugmeta već odabrane riječi uklonit će se ta riječ iz rečenice. Nakon što je korisnik popunio prazna mjesta, pritiskom na dugme 'Check' provjerava se njegovo rješenje. Ukoliko je rješenje točno, korisniku se prikazuje poruka prikazana slikom 4.30. Ukoliko je pogrešno riješio zadatak, kao što je prikazan odgovor koji će korisnik predati na zadatak na slici 4.31., prikazuje mu se poruka prikazana slikom 4.32. i rješenje koje je korisnik predao se briše.



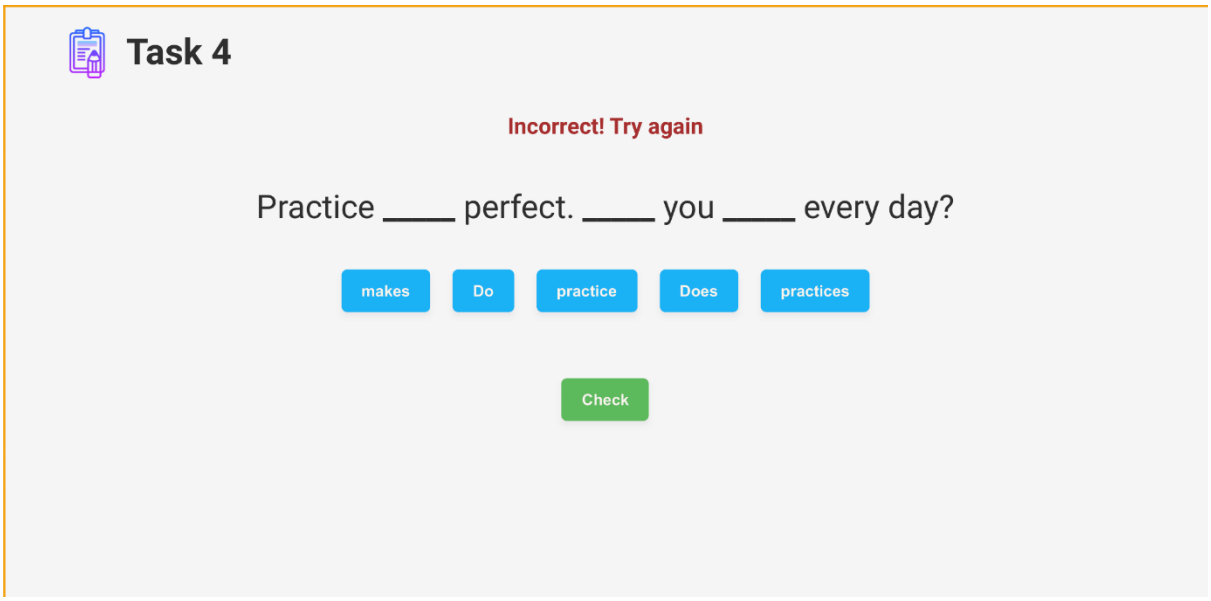
Slika 4.29. Prikaz stranice za rješavanje zadataka.



Slika 4.30. Točno riješen zadatak.



Slika 4.31. Netočno riješen zadatak.



Slika 4.32. Poruka nakon netočno riješenog zadataka.

4.1.9. Prikazivanje hinta

Slika 4.33. prikazuje JavaScript kod koji povećava broj pokušaja korisnika te nakon 3 pokušaja prikazuje hint korisniku. Kada korisnik napravi grešku, u kombinaciji s AJAX-om, šalje se request na '/home/increment_attempts/${taskId}/ kako bi se povećao broj pokušaja. Kada broj pokušaja dođe do tri, prikaže se hint ovisno o razini na kojoj se korisnik nalazi. Na prvoj, trećoj i petoj se dobije prva točna riječ, a na drugoj, četvrtoj i šestoj druga točna riječ.

```
192                 if (attempts >= 3) {
193                     document.getElementById('attempts-message').style.display = 'block';
194                     let hintWord='';
195                     if ([1, 3, 5].includes(currentLevel)) {
196                         hintWord = `The first word is ${correctWords[0]}`;
197                     } else {
198                         hintWord = `The second word is ${correctWords[1]}`;
199                     }
200                     document.getElementById('hint').textContent = hintWord;
201                 }
202                 document.querySelectorAll('.selected').forEach(element => {
203                     element.classList.remove('selected');
204                 });
205                 const taskId = checkButton.getAttribute('data-task-id');
206                 fetch(`/home/increment_attempts/${taskId}/`, {
207                     method: 'POST',
208                     headers: {
209                         'Content-Type': 'application/json',
210                         'X-CSRFToken': document.querySelector('[name=csrfmiddlewaretoken]').value
211                     }
212                 })
213                 .then(response => response.json())
214                 .then(data => {
215                     if (data.status === 'success') {
216                         if (data.attempts_reached) {
217                             document.querySelector('.attempts-message').style.display = 'block';
218 
219                         }
220                     }
221                 });
```

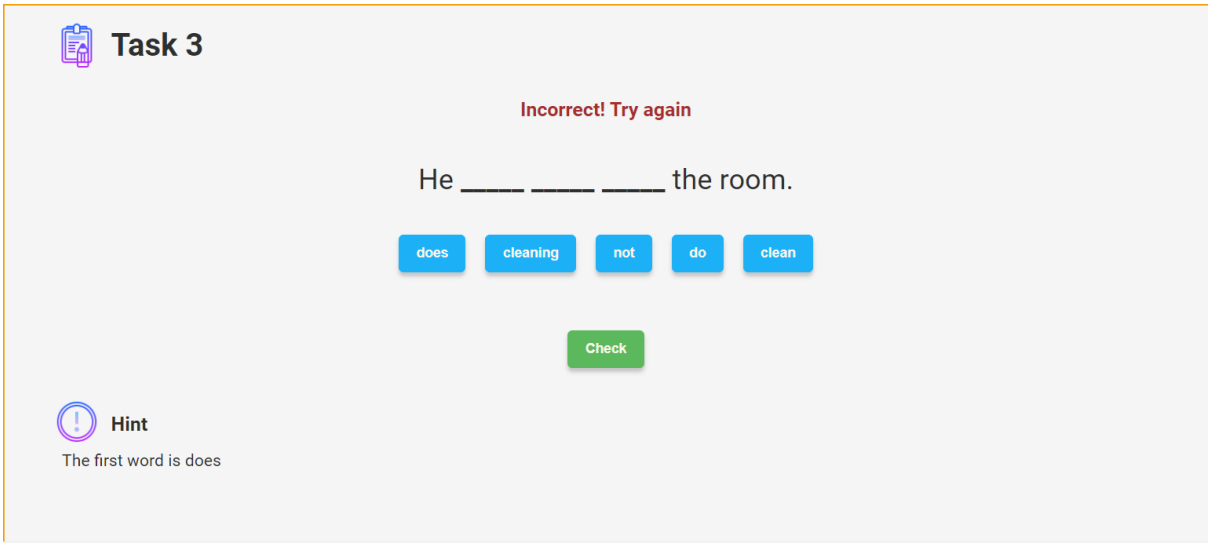
Slika 4.33. JavaScript kod za upravljanje pokušajima zadatka.

Na slici 4.34. prikazani su pogledi increment_attempts i reset_attempts koji se koriste za povećavanje broja pokušaja prilikom rješavanja zadatka te nakon točno riješenog zadatka resetiranje samog broja pokušaja. Unutar pogleda increment_attempts dohvaćaju se trenutni korisnik, zadatak te objekt user_task koji spaja korisnika sa zadatkom koji se trenutno rješava. Prilikom poziva ovog pogleda broj se pokušaja povećava za jedan te se provjerava je li korisnik dostigao 3 pokušaja koristeći user_task.check_attempts(). Pogled reset_attempts poziva se nakon što je korisnik točno riješio zadatak, dohvaćaju se podaci o korisniku i ID zadataka te objekt user_task nad kojim onda vrši metodu reset_attempts().

```python
@csrf_exempt
def increment_attempts(request, task_id):
    user_profile = UserProfile.objects.get(user=request.user)
    task = get_object_or_404(Task, id=task_id)
    user_task = get_object_or_404(UserTask, user=user_profile, task=task)

    user_task.attempts += 1
    user_task.save()

    attempts_reached = user_task.check_attempts()

    return JsonResponse({'status': 'success', 'attempts_reached': attempts_reached})

@csrf_exempt
def reset_attempts(request, task_id):
    if request.method == 'POST':
        try:
            user_profile = UserProfile.objects.get(user=request.user)
            user_task = UserTask.objects.get(user=user_profile, task_id=task_id)
            user_task.reset_attempts()
            return JsonResponse({'status': 'success'})
        except UserTask.DoesNotExist:
            return JsonResponse({'status': 'error', 'message': 'UserTask not found'})
    return JsonResponse({'status': 'error', 'message': 'Invalid request method'})
```

Slika 4.34. Pogled za povećavanje i resetiranje broja pokušaja.

Slika 4.35. prikazuje tekst hinta koji se prikaže korisniku nakon što je pogrešno riješio zadatak barem 3 puta.



Slika 4.35. Prikaz izgleda hinta.

4.2. Administratorsko sučelje

Na slikama koje slijede bit će prikazane mogućnosti administratora koje uključuju stvaranje novih zadataka, pregled svih postojećih zadataka te uređivanje i brisanje postojećih zadataka.

4.2.1. Kreiranje novih zadataka

Predložak za kreiranje novog zadatka prikazan je slikom 4.36. Administratoru je prikazana forma za kreiranje zadatka koju treba popuniti tako da prvo odabere za koje glagolsko vrijeme kreira zadatak, nakon toga odabire razinu zadatka, unosi tekst zadatka te moguće i točne odgovore. Ispod svakog prozora za unos teksta nalazi se paragraf koji govori administratoru kako točno mora unositi tekst zadatka te na koji način mora unijeti moguće i točne odgovore. Kada su podatci o zadatku uneseni, pritiskom na dugme 'Create' zadatak će biti kreiran.

```
{% extends 'learntenses/base.html' %}

{% block extra_css %}
  {% load static %}
  <link rel="stylesheet" href="{% static 'learntenses/styles/create_task.css' %}">
{% endblock %}

{% block content %}

  <div class = "create-task-card">
    <div>
      <h1>Create a new task</h1>
    </div>
    {% if form.non_field_errors %}
      <div class="alert alert-danger">
        {{ form.non_field_errors }}
      </div>
    {% endif %}
    <div class= "create-task-form">
      <form method="POST">
        {% csrf_token %}
        <div class="form-container">
          <div class="form-group">
            {{ form.tense.label_tag }}
            {{ form.tense }}
          </div>
          <div class="form-group">
            <div class="unselectable-label">Task level</div>
            {{ form.name }}
          </div>
          <div class="form-group">
            {{ form.sentence.label_tag }}
            <textarea id="id_form-sentence" name="sentence">{{ form.sentence.value|default_if_none:'' }}</textarea>
            <small class="form-text">Input the sentence. Use '<>' where the missing word should be.</small>
          </div>
          <div class="form-group">
            <div class="unselectable-label">Possible choices</div>
            <textarea id="id_form-words" name="words">{{ form.words.value|default_if_none:'' }}</textarea>
            <small class="form-text">Input words, each separated by a single space</small>
          </div>
          <div class="form-group">
            <div class="unselectable-label">Correct words</div>
            <textarea id="id_form-correct_words" name="correct_words">{{ form.correct_words.value|default_if_none:'' }}</textarea>
            <small class="form-text">Input words, each separated by a single space</small>
          </div>
          <button type="submit" class="create-btn">Create</button>
        </form>
        </div>
    </div>
  </div>
{% endblock %}
```

Slika 4.36. Predložak za stvaranje novih zadataka.

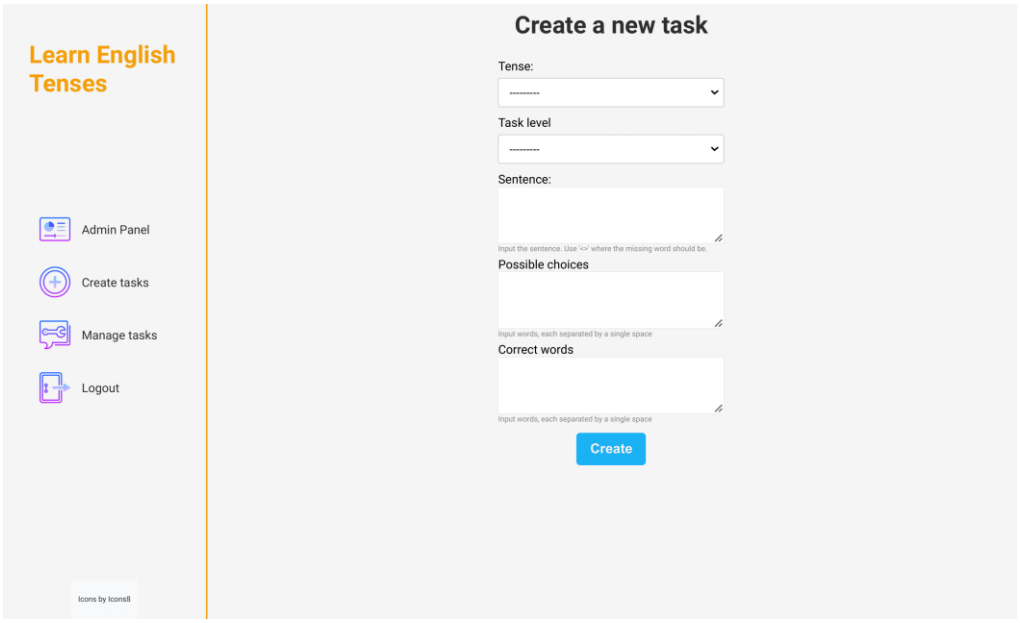
Na slici 4.37. prikazan je pogled za kreiranje novog zadatka. Svi postojeći zadatci spremaju se u varijablu 'tasks' te se organiziraju u riječnik imenovan 'tenses' gdje su ključevi vremena, a

vrijednosti su liste zadataka koje pripadaju tim vremenima. Nakon toga se provjerava je li metoda zahtjeva POST. Ako je, kreira se forma s podatcima iz zahtjeva. Ukoliko je forma valjana, provjerava se postoji li već zadatak s istim imenom za odgovarajuće vrijeme. Ukoliko postoji, u formu se dodaje greška i administratoru će se prikazati poruka da zadatak koji pokušava kreirati već postoji.

```
160 def create_task(request):
161     tasks = Task.objects.all()
162     tenses = {choice[0]: [] for choice in Task.TENSE_CHOICES}
163     for task in tasks:
164         tenses[task.tense].append(task)
165     if request.method == 'POST':
166         form = TaskForm(request.POST)
167         if form.is_valid():
168             try:
169                 form.save()
170                 return redirect('home:create_task')
171             except IntegrityError:
172                 form.add_error(None, 'A task with this name already exists for this tense')
173 
174     else:
175         form = TaskForm()
176     return render(request, 'learntenses/create_task.html', {'form': form, 'tenses':tenses})
```

Slika 4.37. Pogled za kreiranje novih zadataka.

Na slici 4.38. prikazana je stranica na kojoj administrator stvara nove zadatke. Prilikom stvaranja novog zadatka prvo se odabire glagolsko vrijeme zadatka i razina zadatka pomoću opcija na padajućem izborniku. U polje 'Sentence' administrator unosi tekst zadatka tako da pritom prazna mjesta zadatka označava s kombinacijom znakova manje od '<' i veće od '>'. Nakon toga, u polju 'Possible choices' navode se riječi koje će korisniku biti prikazane prilikom rješavanja zadatka, a u polju 'Correct words' navode se točna rješenja.



Slika 4.38. Stranica za stvaranje novih zadataka.

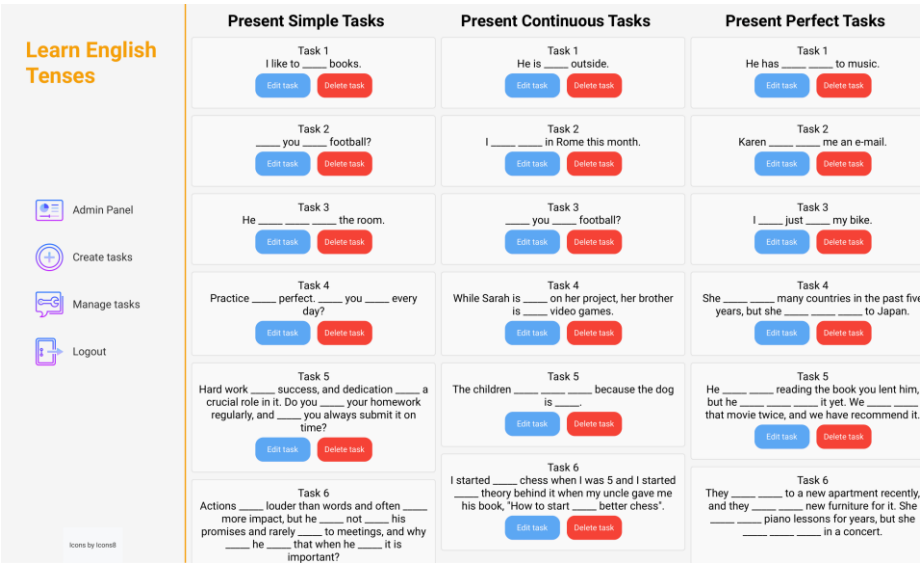
4.2.2. Upravljanje zadatcima

Slika 4.39. prikazuje predložak za upravljanje zadatcima. Administrator na ovoj stranici ima pregled svih zadataka te mogućnosti uređivanja i brisanja zadataka. Zadatci se prikazuju tako što se pomoću 'for' petlje prolazi kroz sve zadatke u bazi podataka, a prikazuju se grupirani po glagolskim vremenima. Ispod svakog zadataka nalaze se dva dugmeta; jedno za uređivanje zadatka, a drugo za brisanje zadatka iz baze podataka.

```
{% extends 'learntenses/base.html' %}

{% block extra_css %}
  {% load static %}
  <link rel="stylesheet" href="{% static 'learntenses/styles/manage_tasks.css' %}">
{% endblock %}

{% block content %}

<div class="tasks-container">
  <div class="tasks-row">
    {% for tense_code, tasks in tenses.items %}
      {% if tense_code in present_tenses %}
        <div class="tasks-column">
          <h2>
            {% if tense_code == 'PS' %}
              Present Simple Tasks
            {% elif tense_code == 'PC' %}
              Present Continuous Tasks
            {% elif tense_code == 'PP' %}
              Present Perfect Tasks
            {% endif %}
          </h2>
          <ul>
            {% for task in tasks %}
              <li>
                <div class="task-level">Task {{ task.level }}</div>
                <div class="task-sentence">{{ task.sentence }}</div>
                <div class = "task-actions">
                  <a href="{% url 'home:edit_task' task.id %}"><button class="edit-task" data-task-id="{{ task.id }}">Edit task</button></a>
                  <button class="delete-task" data-task-id="{{ task.id }}">Delete task</button>
                </div>
              </li>
            {% endfor %}
          </ul>
        </div>
      {% endif %}
    {% endfor %}
  </div>
</div>
```

Slika 4.39. Predložak za upravljanje zadatcima.

Slika 4.40. prikazuje stranicu upravljanja zadatcima.



Slika 4.40. Stranica za upravljanje zadatcima.

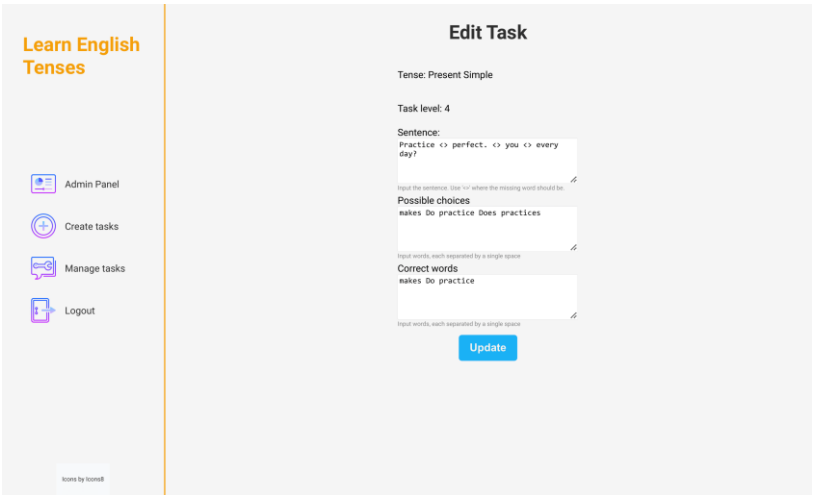
4.2.3. Uređivanje zadatka

Slika 4.41. prikazuje predložak stranice za uređivanje zadatka koji funkcionira na isti način kao kreiranje novog zadatka, samo što nije moguće promijeniti glagolskog vrijeme i razinu zadatka.

```
{% extends 'learntenses/base.html' %}

{% block extra_css %}
  {% load static %}
  <link rel="stylesheet" href="{% static 'learntenses/styles/edit_task.css' %}">
{% endblock %}

{% block content %}

  <div class = "create-task-card">
    <div>
      <h1>Edit Task</h1>
    </div>
    {% if form.non_field_errors %}
      <div class="alert alert-danger">
        {{ form.non_field_errors }}
      </div>
    {% endif %}
    <div class= "create-task-form">
      <form method="POST">
        {% csrf_token %}
        <div class="form-container">
          <div class="form-group">
            <p>Tense: {{ full_tense }}</p>
          </div>
          <div class="form-group">
            <p>Task level: {{ form.name.value }}</p>
          </div>
          <div class="form-group">
            {{ form.sentence.label_tag }}
            <textarea id="id_form-sentence" name="sentence">{{ form.sentence.value|default_if_none:'' }}</textarea>
            <small class="form-text">Input the sentence. Use '<>' where the missing word should be.</small>
          </div>
          <div class="form-group">
            <div class="unselectable-label">Possible choices</div>
            <textarea id="id_form-words" name="words">{{ form.words.value|default_if_none:'' }}</textarea>
            <small class="form-text">Input words, each separated by a single space</small>
          </div>
          <div class="form-group">
            <div class="unselectable-label">Correct words</div>
            <textarea id="id_form-correct_words" name="correct_words">{{ form.correct_words.value|default_if_none:'' }}</textarea>
            <small class="form-text">Input words, each separated by a single space</small>
          </div>
          <button type="submit" class="update-btn">Update</button>
        </form>
        </div>
    </div>

{% endblock %}
```

Slika 4.41. Predložak za uređivanje zadatka.

Pritiskom na dugme 'Edit task', ispod teksta zadatka na stranici za upravljanje zadatcima, administrator će biti preusmjeren na stranicu prikazanu slikom 4.42.



Slika 4.42. Stranica za uređivanje zadatka.

Moguće je urediti tekst zadatka i odabrati točne riječi, no nije moguće promijeniti glagolsko vrijeme zadatka ili razinu zadatka.

4.2.4. Brisanje zadataka

Na slici 4.43. nalazi se pogled koji obavlja funkcionalnost brisanja zadataka. Administrator pritiskom na dugme 'Delete', koje se nalazi na stranici za pregled svih zadataka koja je prikazana slikom 4.40., briše zadatak. Pomoću 'task_id' dohvaća se zadatak te se u idućoj liniji briše iz baze podataka.

```
190  @require_POST
191  def delete_task(request, task_id):
192      task = get_object_or_404(Task, id=task_id)
193      task.delete()
194      return redirect('home:create_task')
```

Slika 4.43. Pogled za brisanje zadatka.

5. ZAKLJUČAK

Danas sve više ljudi uči nešto koristeći mobilne ili web aplikacije. Učenje stranog jezika nije iznimka, stoga postoji potreba za razvijanjem i poboljšavanjem aplikacija za računalno potpomognuto učenje kako bi se udovoljilo rastućim zahtjevima korisnika. Za potrebe je ovog završnog rada izrađena web aplikacija koja korisnicima omogućuje samoučenje glagolskih vremena i vježbanje usvojenih struktura.

Nakon proučavanja trenutno dostupnih rješenja, uočeno je da aplikacije nude mogućnost vježbanja uporabe glagolskih vremena bez mogućnosti praćenja vlastitoga napretka. Ukoliko su besplatne za korištenje, aplikacije sadrže reklamni sadržaj koji ometa korisnika prilikom učenja. Korisnik može ukloniti reklamni sadržaj tako da plati mjesečnu pretplatu na pojedinu aplikaciju. Navedeni su nedostaci otklonjeni u ovoj aplikaciji koja omogućava besplatno učenje uz mogućnost praćenja napretka.

Izrađena se aplikacija sastoji od teorijskoga dijela glagolskih vremena, no ukoliko korisnik smatra da je usvojio teorijske osnove, može odmah krenuti s praktičnim dijelom rješavanja zadataka. Zadatci su podijeljeni na šest razina tako da je svaki zadatak teži od prethodnog. Korisnik mora riješiti zadatak točno kako bi otključao mogućnost rješavanja idućeg, težeg zadatka. Vlastiti napredak korisnik može vidjeti u bilo kojem trenutku tako što će otići na svoju stranicu profila gdje će imati vizualnu reprezentaciju o broju riješenih zadataka svakog glagolskog vremena.

Prilikom izrade aplikacije korištene su tehnologije HTML, CSS i JavaScript kako bi se kreiralo korisničko sučelje koje je pregledno i jednostavno za korištenje. Pomoću AJAX-a se vrše izmjene podataka u bazi podataka, a Django je korišten za poslužiteljski dio (eng. *backend*) aplikacije kao što je definiranje modela, spajanje hiperveza s funkcijama i uvođenje dinamičkih elemenata u HTML.

Aplikacija je trenutno ograničena na samoučenje i vježbanje glagolskih vremena na engleskom jeziku, no moguće ju je proširiti i na druge strane jezike i dodati druge dijelove gramatike engleskoga jezika.

Literatura

[1] A. Abusa'aleek, - A Review of Emerging Technologies: Mobile Assisted Language Learning (MALL), Asian Journal of Education and e-Learning, Volume 6, Issue 6, prosinac 2014.

[2] Duolingo, [online] dostupno na: https://www.duolingo.com [10.4.2024.]

[3] Perfect English Grammar, [online] dostupno na: https://www.perfect-english-grammar.com/grammar-exercises.html [10.4.2024.]

[4] English Grammar, [online] dostupno na: https://www.english-grammar.at/online_exercises/tenses/tenses_index.htm [10.4.2024.]

[5] englishpage.com, [online] dostupno na: https://www.englishpage.com/ [12.4.2024.]

[6] HTML Tutorial, GeeksForGeeks, [online] dostupno na: https://www.geeksforgeeks.org/html-tutorial/ [13.3.2024.]

[7] HTML Introduction, W3Schools, [online] dostupno na: https://www.w3schools.com/html/html_intro.asp [13.3.2024.]

[8] CSS Tutorial, W3Schools, [online] dostupno na: https://www.w3schools.com/css/default.asp [13.3.2024.]

[9] JavaScript, MDN Web Docs, [online] dostupno na: https://developer.mozilla.org/en-US/docs/Glossary/JavaScript [14.3.2024.]

[10] AJAX Introduction, W3Schools, [online] dostupno na: https://www.w3schools.com/xml/ajax_intro.asp [15.3.2024.]

[11] Python FAQ, Python, [online] dostupno na: https://docs.python.org/3/faq/general.html#what-is-python [13.3.2024.]

[12] Django Introduction, W3Schools, [online] dostupno na: https://www.w3schools.com/django/django_intro.php [16.3.2024.]

Sažetak

Neovisno o području/predmetu učenja, računalno je potpomognuto učenje sve raširenije jer pruža korisnicima mogućnost samostalnoga učenja vlastitom brzinom bez vršnjačkoga pritiska. Cilj je ovog rada bio izraditi web aplikaciju za računalno potpomognuto samoučenje glagolskih vremena engleskoga jezika. Aplikacija je razvijena korištenjem tehnologija HTML-a, CSS-a, JavaScripta, AJAX-a i Djanga zato što omogućuju izradu jednostavnog, interaktivnog i responzivnog sučelja te jednostavno upravljanje sadržajem stranice, njenim funkcionalnostima i komunikaciju s bazom podataka. Osim praktičnog dijela sa zadatcima, aplikacija sadrži i teorijski dio s objašnjenjima pojedinog glagolskog vremena koji nije obavezan. Zadatci su podijeljeni na šest razina s postupnim povećavanje složenosti. Korisnik može pratiti svoj napredak na stranici profila. Aplikacija je namijenjena učenicima i studentima koji mogu usavršiti upotrebu glagolskih vremena na engleskom jeziku. Prednosti su ove aplikacije izostanak reklamnoga sadržaja i besplatno korištenje. Aplikacija je dizajnirana s naglaskom na jednostavnost i preglednost, a može se nadograditi proširivanjem sadržaja ili dodavanjem novih jezika.

Ključne riječi: engleski jezik, glagolska vremena, računalno potpomognuto učenje, web aplikacija

Abstract

Web application for practicing the use of verb tenses in a foreign language

Computer-assisted learning has become increasingly widespread regardless of the field or subject of study because it allows users to learn independently at their speed without any peer pressure. This paper aimed to create a web application for computer-assisted self-learning of English verb tenses. The application was developed using HTML, CSS, JavaScript, and Django because they enable the creation of a simple, interactive, and responsive interface, an easy-to-use app, its functionalities, and communication with the database. In addition to the practical part, the application also has theoretical explanations of tenses, which are optional to take part in. The tasks are divided into six gradually more difficult levels. The user can check the progress on his profile page. The application is intended for high school and university students who can perfect verb tenses in English. The advantages of this application are that it is free of charge and has no advertising policy. It emphasizes simplicity and transparency and can be improved by expanding the content or adding new languages.

Keywords: English language, tenses, computer-assisted language learning, web application

Prilozi

1. Izvorni kod programskog rješenja, [online] dostupno na: https://github.com/frankoklepac/zavrsni-rad