Web aplikacija kuharica prema sadržaju hladnjaka

Posavec, Stjepan

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

Web aplikacija kuharica prema sadržaju hladnjaka

Diplomski rad

Stjepan Posavec

Osijek, 2021.

SADRŽAJ

1. UVOD

Tema diplomskog rada je web aplikacija koja je zamišljena kao pomagalo u svakodnevnom životu. Naime, svakog dana ljudi razmišljaju što će pripremiti za svoj obrok, a ponekad nailaze na poteškoće zbog toga što ne mogu pronaći jelo koje mogu napraviti. U tome slučaju aplikacija može uvelike pomoći ljudima koji su naišli na spomenuti problem, dovoljno je samo unijeti namirnice koje trenutno imaju u svome hladnjaku, te im aplikacija u slijedećem koraku nudi sve recepte iz baze koji se mogu pripremiti. Također, na receptima mogu provjeriti pripada li recept u neku od posebnih prehrana kao što su: vegetarijanska, veganska, bez-glutenska, te mnoge druge.

Web aplikacija u ovome diplomskom radu biti će izrađena pomoću React-a, besplatne front-end JavaScript knjižnice za izgradnju korisničkog sučelja uz proširenje React-Redux biblioteke. React-Redux služi nam za upravljanjem stanja aplikacije. Izvor recepata će biti besplatni API (engl. Application programming interface), odnosno aplikacijsko programsko sučelje. Također, korisnici će biti u mogućnosti pojedine recepte označiti sa *„sviđa mi se“* oznakom. Spremanje takvih recepata će se odvijati na Firestore-u, kao i sama autentifikacija korisnika aplikacije. Firestore je noSQL baza podataka koja je u sklopu Firebase-a, platforme koju je razvila tvrtka Google, a služi nam za stvaranje mobilnih i web aplikacija.

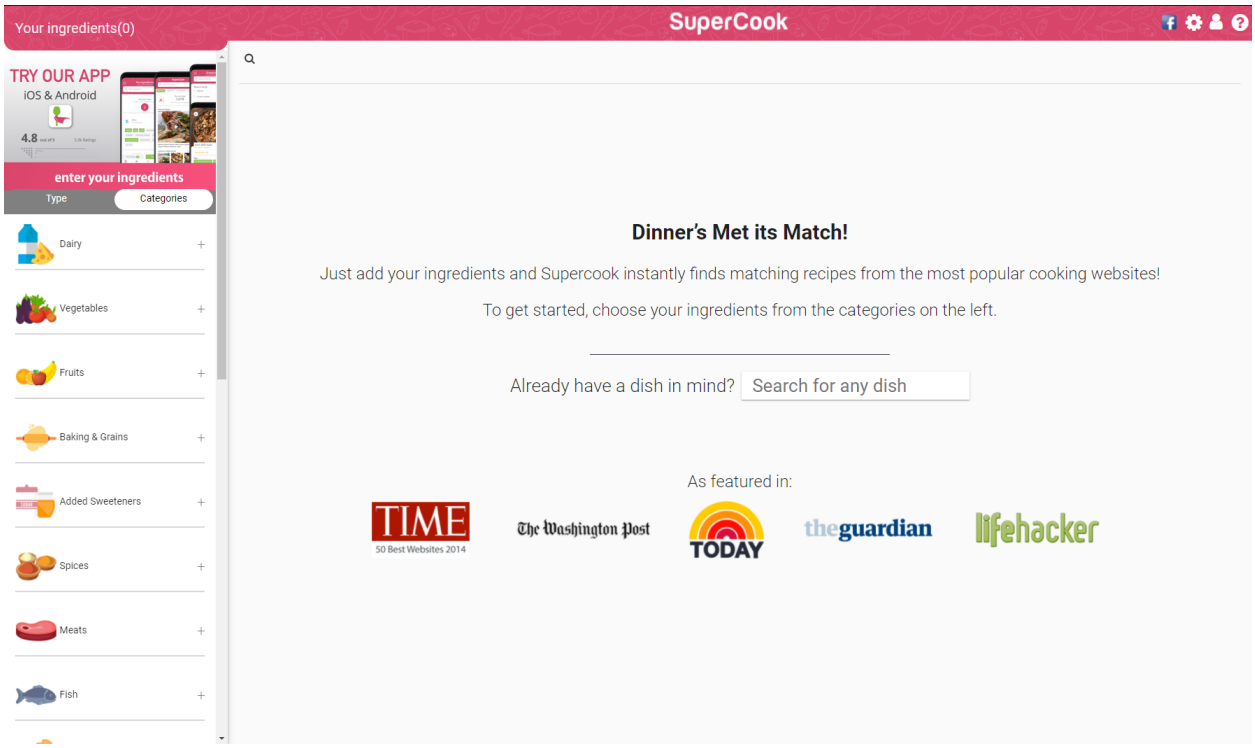
1.1. Zadatak diplomskog rada

Zadatak ovog diplomskog rada je izraditi Internet aplikaciju za personalizirane recepte za prehranu korisnika. Aplikaciju je potrebno napraviti u React okolini te bazom Firebase. Korisnik treba unijeti sadržaj svog hladnjaka, kao što je vrsta i količina pojedine namirnice. U skladu s unesenim količinama i vrstama namirnica, aplikacija će korisniku ponuditi potencijalne recepte. Aplikacija treba imati pretraživanje recepata prema pojedinim namirnicama, vrstama i količinama namirnica te željenom planu ishrane.

2. PRIKAZ POSTOJEĆIH RJEŠENJA

2.1. Supercook

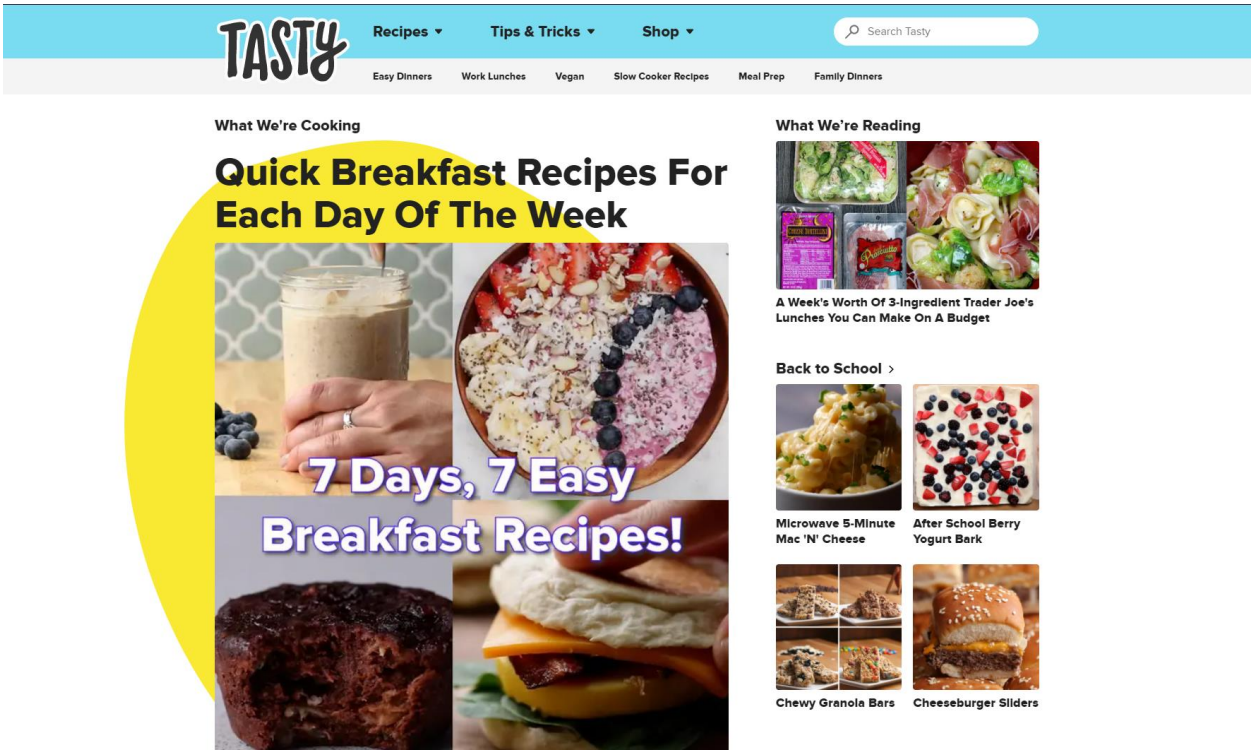
Supercook [1] je aplikacija koja nam omogućava odabiranje namirnica od kojih želimo napraviti jelo. Kada dodamo namirnicu aplikacija nam nudi recepte koje možemo napraviti. Aplikacija je zamišljena kao aplikacija koja nam pruža da unesemo namirnice koje želimo potrošiti i od njih napravimo jelo. Sučelje aplikacije je vidljivo na *Slika 2.1*.



Slika 2.1. Supercook naslovna stranica

2.2. Tasty

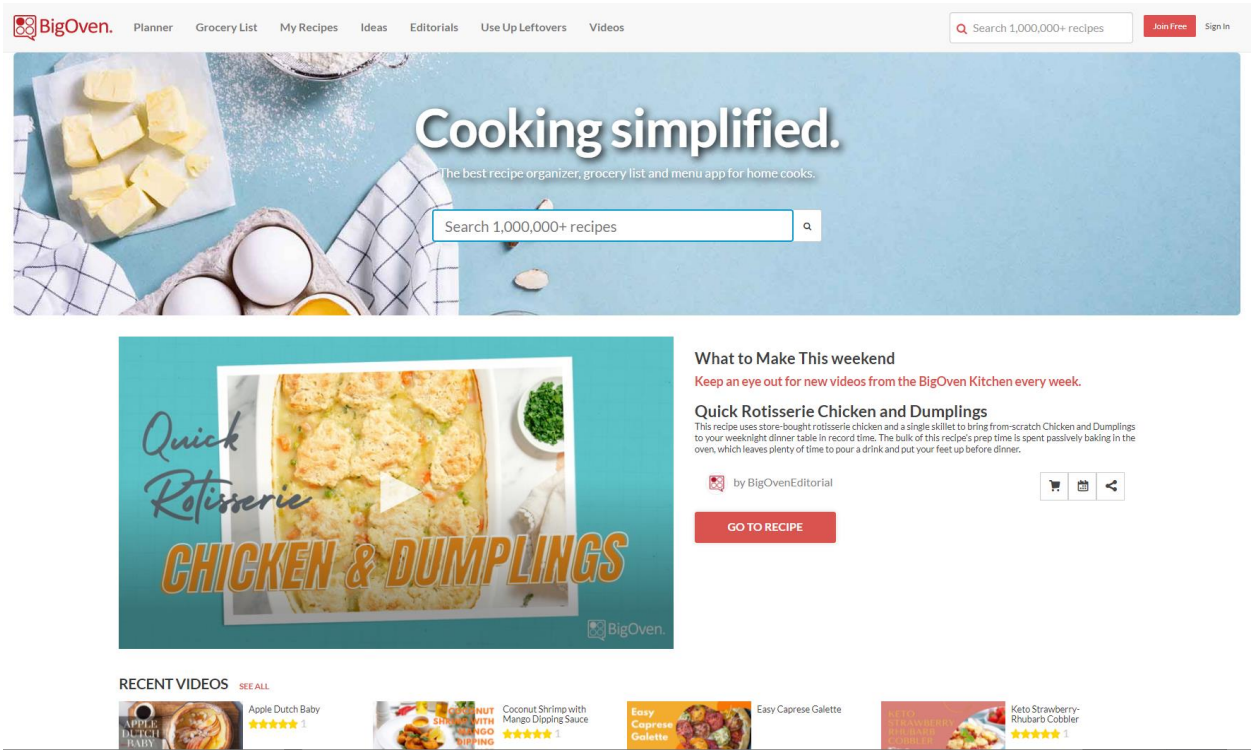
Tasty [2] je aplikacija koja nam omogućava odabiranje namirnica i povezanih recepata. Nakon pretrage omogućava korisniku uvid u tip prehrane kojoj pripada recept. Specifičnost kod aplikacije je ta što prikazuje pripremu recepata preko videa. Sučelje aplikacije je vidljivo na *Slika 2.2*.



Slika 2.2. Tasty naslovna stranica

2.3. BigOven

BigOven [3] je aplikacija u kojoj možemo pretražiti više od milijun recepata što ih čini aplikacijom s najvećom bazom recepata. Aplikacija nam omogućava da pretražimo recepte po namirnicama, ali pretraga je ograničena na tri namirnice po pretrazi. Sučelje aplikacije je vidljivo na *Slika 2.3*.



Slika 2.3. BigOven naslovna stranica

3. KORIŠTENI ALATI I TEHNOLOGIJE

3.1. React

Cijela web aplikacija napisana je koristeći React knjižnici. React je besplatna knjižnica koja se bazira na JavaScript programskog jeziku. Cijela knjižnica je otvorenog koda, te ju svi mogu razvija. Knjižnica je napisana u JavaScript programskom jeziku, a razvija i održava ju Facebook i zajednica pojedinačnih programera.

Za pisanje HTML koristi se JSX, odnosno JavaScript XML. Vrlo je sličan izvornom HTML-u, ali dopušta stvaranje pojedinačnih JavaScript komponenti i njihovo korištenje u daljnjem kodu.

Postoje dvije vrste pisanja Reacta, a to su React klase i React hookovi. Aplikacija u ovome diplomskom radu napisana je pomoću React hookova.

3.1.1. Redux

Redux ili React redux je JavaScript knjižnica otvorenog koda. Služi nam za upravljanje i centralizaciju stanja aplikacije. U ovome diplomskog radu redux je korišten kao lokalna baza podataka. U redux-u su pohranjeni podaci o korisniku, namirnice u hladnjaku i informacije o receptima.

3.2. JavaScript / TypeScript

JavaScript je skriptni programski jezik kojeg podržavaju web preglednici, odnosno izvršava se unutar web preglednika. Vrlo je sličan programskom jeziku Java, no nije objektno orijentirani programski jezik kao što je Java. Izvorno ga je razvila tvrtka pod nazivom Netscape.

TypeScript je proširenje JavaScript programskog jezika. TypeScript razvija i održava Microsoft. Specifično kod TypeScripta je to što je proširio JavaScript na način kako bi JavaScript bio precizniji i definiraniji programski jezik. Odnosno kada se u JavaScriptu definira varijabla npr. *var broj = 20* ta ista varijabla kasnije može poprimiti vrijednost koja nije brojčana i ne predstavlja broj. U TypeScriptu bi za tu istu varijablu rekli da je *var broj: number = 20*. Nakon što smo rekli da je to varijabla tipa *„number“* ta ista varijabla u daljnjem kodu ne može poprimiti druge vrijednosti osim onih koji su tipa *„number“*. Na taj način kod razvoja web aplikacija kada dolazi do komunikacije frontend i backend developera i jedan i drugi developer znaju kojeg će tipa biti podaci.

3.2.1. JavaScript Object Notation – JSON

JSON ili JavaScript Object Notation je tekstualni format koji je čitljiv i stroju i ljudima. Osnovni tipovi podataka koje JSON podržava su: broj (number), tekst (string), logički tip (boolean), polje podataka (array), objekt (object) i null vrijednost – prazna vrijednost. Zbog toga nam je pogodan za pohranu raznih podataka.

Svi podaci koji su slani i dohvaćani s poslužitelja su bili formatirani u JSON-u. Kod JSON-a kada se koristi s TypeScript programskim jezikom možemo zadati tip podatka preko sučelja *Slika 3.1*.

```
export interface RecipeInfoData {
  id: string;
  title: string;
  analyzedInstructions: RecipeSteps[];
  diets: string[];
  extendedIngredients: Ingredient[];
  image: string;
  instructions: string;
  readyInMinutes: number;
  servings: number;
  sourceUrl: string;
  summary: string;
}
```

Slika 3.1. Primjer definicije tipa podatka

Nakon što dohvatimo podatke s poslužitelja (podaci su skraćeni) koji su formatirani u JSON-u, *Slika 3.2*.

```
{
  "id": 716429,
  "title": "Pasta with Garlic, Scallions, Cauliflower & Breadcrumbs",
  "image": "https://spoonacular.com/recipeImages/716429-556x370.jpg",
  "imageType": "jpg",
  "servings": 2,
  "readyInMinutes": 45,
  "license": "CC BY-SA 3.0",
  "sourceName": "Full Belly Sisters",
  .
  .
  .
}
```

Slika 3.2. Primjer JSON formatiranih podataka

Direktno možemo pristupiti podacima, odnosno ime varijable se poveže uz ime taga iz JSON formatiranih podataka. To znači da podataka iz JSON-a *„id“: 716429* se direktno povezuje uz varijablu *id: string*.

3.3. Firebase

Firebase je platforma koju razvija i održava Google. Ona je izvorno nastala 2011. kada ju je razvila neovisna tvrtka, nakon čega Google kupuje platformu 2014. godine. Firebase je platforma koja besplatno pruža opcije kao što su autentifikacija, pohrana podataka (baza podataka – Firestore), skladište datoteka, usluge objavljivanja web aplikacije, te funkcije koje se izvršavaju nad bazom podataka na strani servera, odnosno poslužitelja.

3.3.1. Autentifikacija

Firebase pruža opciju autentifikacije. Na taj način programer može implementirati uslugu u svoju aplikaciju i skladištiti podatke o korisniku preko nekoliko opcija. Korisnik se može prijaviti u aplikaciju pomoću e-mail adrese i lozinke ili preko nekoliko vanjskih poslužitelja, neki od njih su Google i Facebook vanjski poslužitelji autentifikacije korisnika. Ova tri načina prijave, odnosno autentifikacije korisnika korištena su u ovome diplomskom radu.

3.3.2. Baza podataka – Firestore

Firebase pruža opciju pohrane podataka, odnosno bazu podataka. Postoje dvije opcije Firestore baza podataka i Realtime database. U ovome diplomskom radu korištena je Firestore baza podataka koja je temeljena na kolekcija i dokumentima unutar kolekcije. Firestore je noSQL baza podataka. Također, kada se govori o pohrani podataka u Firestore uz kolekcije i dokumente u kolekcijama mogu postojati i složenije strukture kao što su kolekcije u kolekciji što programerima daje slobodu u strukturiranju baze podataka.

3.4. Spoonacular API

Spoonacular API je API korišten u ovome diplomskom radu. API je baziran na Rest API strukturi. Spoonacular je besplatan API za pružanje usluga o davanju informacija o raznim receptima. U ovome diplomskom radu su korištene samo neke od mnogobrojnih opcija koje poslužitelj pruža, neke od njih su dohvaćanje recepata po namirnicama potrebnim za taj recept, dohvaćanje detaljnih podataka o pojedinom receptu pomoću ID-a recepta, dohvaćanje nasumično odabranih recepata, i mnoge druge.

4. MODEL APLIKACIJE

Ovo poglavlje detaljno opisuje model web aplikacije kuharica po sadržaju hladnjaka, od samog početka korištenja aplikacije, odnosno od trenutka kada je korisnik pristupio web aplikaciji. Poglavlje je podijeljeno u više manjih segmenata koji opisuju pojedini dio aplikacije.

Model aplikacije opisan u sljedećim poglavljima može se vidjeti na *Slika 4.1. Prikaz modela aplikacije*.

4.1. Autentifikacija korisnika

Pristupom aplikaciji korisnik mora posjedovati korisnički račun kako bi mogao koristiti sve značajke aplikacije. Autentifikacija korisnika obavlja se preko Firebase autentifikacije (više možete pročitati u poglavlju *3.3.1. Autentifikacija*.

4.1.1. Registracija korisnika

Ako korisnik trenutno ne posjeduje korisnički račun može odabrati opciju *„Don't have account? Create one.“*, odabirom te opcije korisnik će biti preusmjeren na ekran koji posjeduje formu za upis podataka. Upisom podataka koji su naznačeni: e-mail adresa, korisničko ime – *„username“*, lozinka – *„password“*, i ponovljena lozinka – *„repeat password“*, kojom korisnik potvrđuje prvotni unos lozinke, kako bi u potpunosti bio siguran da će i naknadno znati pristupiti svome računu.

Unutar forme u koju korisnik upisuje podatke postoji validacija istih, s ciljem onemogućavanja manipulacije podacima i stvaranja lažnih korisničkih računa od strane samih korisnika aplikacije, čime bi potencijalno mogli narušiti sigurnost same aplikacije. Slijedno, korisnik u predviđeno mjesto za upisivanje e-mail adrese treba unijeti podatak koji je ograničen na: slova, brojke i određene specijalne znakove nakon čega mora uslijediti znak „@“, poslije znaka „@“ slijede isključivo slova, pa znak „.“ – točka, te na kraju slova minimalne duljine od 2 znaka. Također, za lozinku postoji zaštita koja primorava korisnika za unos dovoljno „jake“ lozinke koja se sastoji od: minimalno 8 znakova, od čega minimalno jedan znak treba biti broj i minimalno jedan znak treba biti specijalni znak (npr. +, *, -, /, itd.).

Ako korisnik ne ispuni ili ne ispuni valjano gore navede podatke o tome će biti obaviješten porukama ispod određenog okvira za unos podataka. Kada je korisnik valjano ispunio gore navedene podatke, bit će u mogućnosti pritisnuti gumb za potvrdu registracije. Također, može doći

do „pogreške“ servera, o čemu će korisnik također biti obaviješten porukom iznad zamišljenog gumba potvrde.

4.1.2. Prijava korisnika – prijava preko aplikacije

Ako korisnik posjeduje korisnički račun koji je prethodno kreirao unutar aplikacije može se prijaviti u aplikaciju upisivanjem podataka. Također, unutar forme unosa podataka za prijavu postoje validacije kao i unutar forme za registraciju korisnika.

Tek nakon što je korisnik valjano ispunio formu može potvrditi svoju prijavu klikom na predviđeni gumb. Ako u procesu unosa podataka dođe do pogreške ili se greška dogodi na serveru, korisnik će o njima biti obaviješten porukom unutar već prikazane forme.

Kod prijave korisnik može odabrati opciju *„Remember me“* koja mu pruža mogućnost da aplikacija zapamti prijavu. Ako je korisnik odabrao spomenutu opciju moći će slobodno napuštati stranicu u tijeku jedne sesije, dok u protivnom čim korisnik napusti sesiju morat će se ponovno prijaviti u sustav.

4.1.3. Prijava korisnika – prijava preko vanjskog poslužitelja

Aplikacija pruža mogućnost prijave preko vanjskog poslužitelja, u ovom kontekstu to znači prijavu preko Google i Facebook korisničkih računa. Prijava preko vanjskog poslužitelja je puno brža i jednostavnija za korisnika. Na taj način korisnik izbjegava unošenje podataka u registraciju pri kreiranju korisničkog računa.

Također, kod prijave preko vanjskog poslužitelja postoji opcija *„Remember me“* no ona je automatska, odnosno korisnik ostaje prijavljen jednu sesiju osim ako se ne odjavi prije.

4.1.4. Zaboravljena lozinka – ponovno postavljanje lozinke

Aplikacija pruža podršku za korisnike koji su zaboravili lozinku svojeg korisničkog računa. Klikom na *„Forgot your password?“* korisnik će biti preusmjeren na zaseban ekran u kojemu treba upisati valjanu e-mail adresu koja postoji u sustavu. Ako e-mail adresa nije valjana korisnik će o tome biti obaviješten. Nakon što je korisnik unio valjanu e-mail adresu na svoj e-mail će dobiti poruku s poveznicom za ponovno postavljanje lozinke. Na ekranu ponovnog postavljanja lozinke korisnik treba unijeti novu lozinku i potvrditi unos. Nakon uspješnog postavljanja nove lozinke korisnik je u mogućnosti prijaviti se s novopostavljenom lozinkom.

4.2. Naslovna stranica

Nakon što je korisnik uspješno izvršio autentifikaciju preusmjeren je na ekran s odabirom hoće li unijeti namirnice u hladnjak ili želi prikazati recepte s određenim namirnicama iz hladnjaka. Uz odabir između dvije opcije prikazana su mu tri recepta po nasumičnom odabiru.

4.3. Unos namirnica u hladnjak

Ako korisnik odabere opciju unosa namirnica u hladnjak – *„Add Ingredients to Fridge“*, aplikacija će ga preusmjeriti na ekran gdje se unose namirnice. Na ekranu će imati mogućnost unosa novih namirnica klikom na *„+“*. Gdje će mu se otvoriti novi prozor u kojemu će moći upisati ime namirnice.

Korisnik osim što može dodati namirnice, svaku namirnicu može obrisati s popisa ili je promijeniti, ako je odradio pogrešan unos namirnice. Uz opcije za manipulaciju namirnicama, korisnik može mijenjati koliko namirnica po stranici želi imati prikazano i na kojoj stranici želi biti.

4.4. Odabir namirnica i prikazivanje recepata

Ako korisnik odabere opciju odabira namirnica i prikazivanja recepata – *„Select Ingredients & Fetch Recipes“* aplikacija će otvoriti novi prozor gdje će korisnik moći odabrati od kojih namirnica iz hladnjaka želi napraviti jelo. Nakon što je odabrao željene namirnice potvrđuje svoj odabir. Prilikom potvrde odabira aplikacija korisnika preusmjerava na novi ekran. Na ekranu prikazuje sve recepte koje korisnik potencijalno može napraviti, prikazuje mu ime i sliku recepta, koliko će odabranih namirnica potrošiti u receptu i koliko mu namirnica nedostaje kako bi mogao u potpunosti napraviti prikazani recept.

Kada se korisnik odluči za određeni recept klikom na njega dobiva uvid u detalje pripreme recepta. Na novom ekranu može vidjeti: ime recepta, sliku recepta, kojoj prehrani pripada, za koliko ljudi je recept zamišljen, koliko je vremena potrebno za recept, koje su sve namirnice potrebne i koja će se količina namirnica potrošiti, listu korak po korak uputa za pripremu recepta s mogućnosti označivanja riješenog koraka i sažetak recepta.

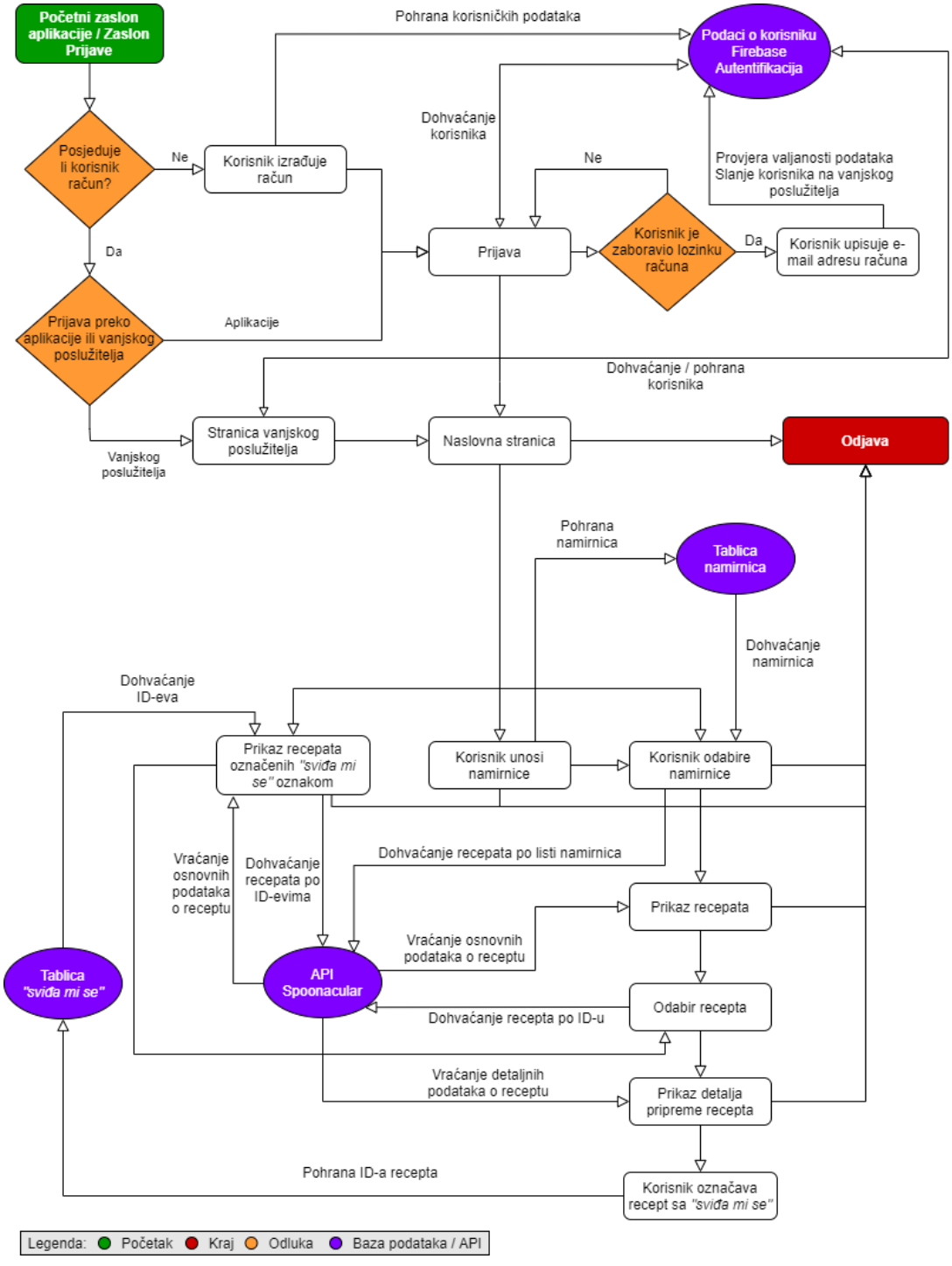
4.5. Oznaka *„sviđa mi se“*

Kada korisnik pristupi detaljnom prikazu recepta na odabir ima opciju *„sviđa mi se“*. Ako korisnik odabere tu opciju ID recepta se sprema unutar baze podataka. U svakome trenutku korisnik može pristupiti označenim receptima klikom na padajući meni koji se nalazi u zaglavlju

aplikacije u kojemu onda može odabrati opciju *„Liked recipes“*. Klikom na spomenutu opciju korisniku se prikazuju svi recepti koje je označio oznakom *„sviđa mi se“*.

4.6. Odjava

Korisnik nakon svakoga korištenja aplikacije može odabrati opciju *„Logout“* pomoću koje briše sve podatke lokalno na uređaju s kojega je pristupio aplikaciji.



Slika 4.1. Prikaz modela aplikacije

5. PROGRAMSKO RJEŠENJE

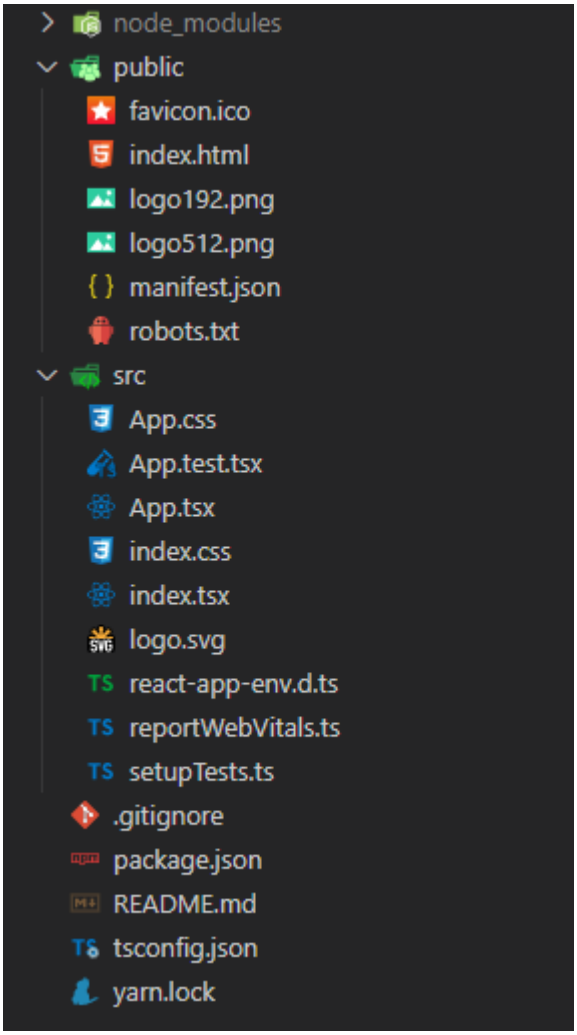
U ovome poglavlju prikazan je programski kod za pojedine segmente aplikacije, počevši od stvaranja aplikacije. Također, uz logiku stvaranja svake pojedine komponente u poglavlju je prikazana i logika komunikacije s Firestore bazom i Spoonacular API-jem.

5.1. Stvaranje React projekta

Stvaranje svakog React projekta započinje naredbom u terminalu koju možemo vidjeti na *Slika 5.1*. Naredba stvara direktorij s datotekama koje su vidljive na *Slika 5.2*. Projekt je stvoren pomoću typescript predloška.

```
yarn create react-app chefs-alone --template=typescript
```

Slika 5.1. Naredba stvaranja projekta



Slika 5.2. Datoteke nakon stvaranja projekta

5.2. Postavljanje okoliša za rad s Firebase bazom i Spoonacular API-jem

Za početak potrebno je postaviti određene parametre unutar aplikacije i sigurnosno ih zaštititi. Praksa korištenja zaštićenih podataka u React aplikacijama je slijedeća. Stvorimo novu datoteku u korijenskom direktoriju koju nazovemo *„.env“*. Budući da projekt pohranjujemo unutar git repozitorija, potrebno je reći git-u da datoteku imena *„.env“* ignorira. To možemo napraviti upisom njenog imena unutar datoteke *„.gitignore“*. Kada smo sigurnosno zaštitili datoteku možemo nastaviti s radom. U datoteku unosimo podatke s kojima se možemo spojiti na Firebase bazu i Spoonacular API. Većinski su to API ključevi, no ponekad trebamo unijeti više podataka. Podaci koje unosimo su vidljivi na *Slika 5.3*. Vrijednosti varijabli su na slici opisani, te nisu stvarnih vrijednosti koje se koriste u aplikaciji.

```
REACT_APP_API_KEY=Firebase_API_Ključ
REACT_APP_AUTH_DOMAIN=Firebase_Domena_Autentifikacije
REACT_APP_PROJECT_ID=chefsalone
REACT_APP_STORAGE_BUCKET=Firebase_poveznica_na_skladište_podataka
REACT_APP_MESSAGING_SENDER_ID=Firebase_ID_Pošitljatelja
REACT_APP_APP_ID=Firebase_ID_Aplikacije

REACT_APP_SPOONACULAR_API_KEY=Spoonacular_API_Ključ
```

Slika 5.3. Varijable okoliša

5.2.1. Firebase

Nakon definiranja varijabli okoliša potrebno je inicijalizirati Firebase aplikaciju, odnosno povezati aplikaciju s poslužiteljem. Način na koji povezujemo aplikaciju vidljiv je na *Slika 5.4*. Na slici možemo vidjeti zbog čega smo trebali zaštiti podatke, u protivnom bi ti podaci bili vidljivi javno i mogli bi se zloupotrijebiti. Pri procesu inicijalizacije aplikacije prvo je potrebno provjeriti postoji li već inicijalizirana aplikacija, ako ona ne postoji potrebno ju je inicijalizirati.

```
if (!firebase.apps.length) {
  firebase.initializeApp({
    apiKey: process.env.REACT_APP_API_KEY,
    authDomain: process.env.REACT_APP_AUTH_DOMAIN,
    projectId: process.env.REACT_APP_PROJECT_ID,
    storageBucket: process.env.REACT_APP_STORAGE_BUCKET,
    messagingSenderId: process.env.REACT_APP_MESSAGING_SENDER_ID,
    appId: process.env.REACT_APP_APP_ID,
  });
}
```

Slika 5.4. Inicijalizacija firebase aplikacije

5.2.2. Spoonacular API – dohvaćanje podataka

Recepte dohvaćamo s udaljene baze podataka pod nazivom Spoonacular API. Poslužitelj koristi Rest API koji je opisan na početku rada. Za pristupanje poslužitelju bilo je potrebno registrirati se. Nakon registracije dobiva se API ključ pomoću kojega se pristupa podacima na API-ju. Na *Slika 5.5* je vidljiv način dohvaćanja recepata za određene namirnice. Za dohvaćanje podataka s udaljenog poslužitelja prvo je bilo potrebno u projekt dodati *„axios“* paket koji nam služi za dohvaćanje podataka.

```
axios
      .get(
        `https://api.spoonacular.com/recipes/findByIngredients?${
          process.env.REACT_APP_SPOONACULAR_API_KEY
        }&ingredients=${ingredients.join(',+')}&number=15`,
      )
```

Slika 5.5. Dohvaćanje recepata preko Spoonacular API-ja

5.3. Autentifikacija

Sveukupna autentifikacija korisnika odvija se preko Firebase autentifikacije. Nakon što je Firebase aplikacija inicijalizirana, za korištenje autentifikacije bilo je potrebno inicijalizirati i nju. Na *Slika 5.6.* je vidljiva inicijalizacija generalne autentifikacije kao i autentifikacije putem vanjskih poslužitelja kao što su Google i Facebook, također inicijalizirana je i *„Remember me“* funkcija pomoću *„Persistence.NONE“* – kada opcija nije uključena (korisnik neće ostati zapamćen nakon što napusti sesiju) i *„Persistence.LOCAL“* – kada je opcija uključena (korisnik će ostati zapamćen i nakon što napusti sesiju, sve dok ne zatvori svoj internet preglednik).

```
export const auth = firebase.app().auth();
export const persistenceNone = firebase.auth.Auth.Persistence.NONE;
export const persistenceLocal = firebase.auth.Auth.Persistence.LOCAL;
export const authProviderGoogle = new firebase.auth.GoogleAuthProvider();
export const authProviderFacebook = new firebase.auth.FacebookAuthProvider();
```

Slika 5.6. Inicijalizacija firebase autentifikacije

5.3.1. Registracija

Za početak rada aplikacije korisnik treba izvršiti registraciju. Registracija je osmišljena kao forma s unosnim poljima. Sva unosna polja imaju zaštitu, te će obavijestiti korisnika da nije ništa unio u polje ili nije unio valjani unos. Sve dok forma nije pravilno ispunjena korisnik neće biti u

mogućnosti potvrditi registraciju, odnosno gumb *„Registration“* bit će onemogućen. Nakon što korisnik valjano ispuni formu gumb će biti omogućen, no u trenutku kada pošalje aplikacija pošalje zahtjev prema serveru može doći do pogreške. Ako dođe do pogreške aplikacija će obavijestiti korisnika o istoj. Na *Slika 5.7.* vidljiva je validacija lozinke i njena implementacija.

```
const validatePassword = () => {
  if (password !== '') {
    if (
      !password.match(
        /^(?=.*\d)(?=.*[0-9])(?=.*[a-zA-Z])(?=.*[^0-9a-zA-Z]).{8,}$/,
      )
    ) {
      setPasswordMessage(
        'Password must contain 8 characters total with number/s and special chara
cter/s.',
      );
    } else {
      setPasswordMessage('');
    }
  } else {
    setPasswordMessage('Please input your password.');
  }
};
// ... Implementacija ...
<FormControl>
  <InputLabel htmlFor="password" error={passwordMessage ? true : false}>
    Password
  </InputLabel>
  <Input
    id="password"
    type={showPassword ? 'text' : 'password'}
    value={password}
    onChange={onChangePassword}
    onBlur={validatePassword}
    endAdornment={
      <InputAdornment position="end">
        <IconButton onClick={handleShowPassword}>
          {showPassword ? <Visibility /> : <VisibilityOff />}
        </IconButton>
      </InputAdornment>
    }
  />
  {passwordMessage ? (
    <FormHelperText id="password" error> {passwordMessage} </FormHelperText>
  ) : (<></>)}
</FormControl>
```

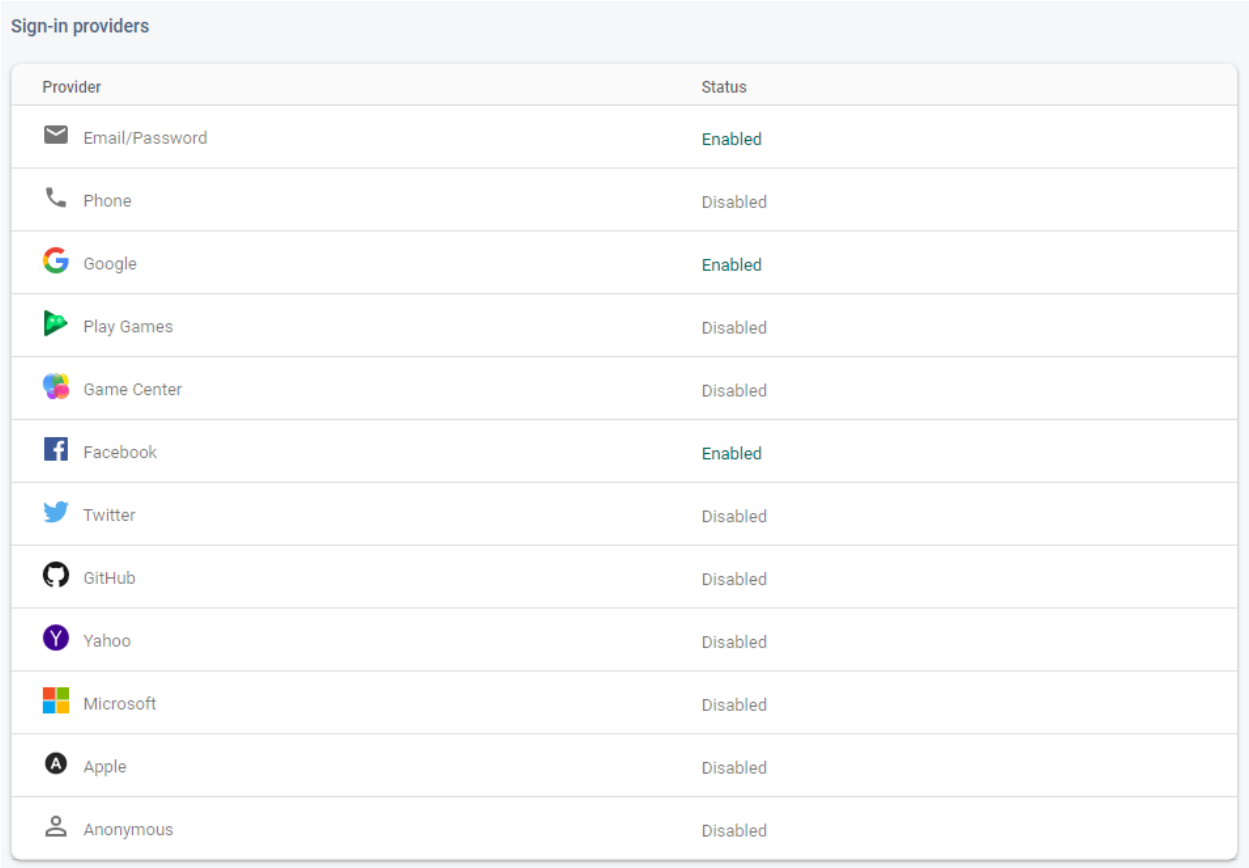
Slika 5.7. Validacija lozinke i implementacija

Nakon što korisnik unese podatke i potvrdi unos podataka izvršava se slanje podataka na server gdje se oni obrađuju. Na *Slika 5.8.* je vidljivo da, ako korisnik nije valjano unio sve podatke neće moći izvršiti registraciju.

```
const onClickRegister = () => {
  if (email && password && repeatPassword && username) {
    auth
      .createUserWithEmailAndPassword(email, password)
      .then((response) => {
        firestore
          .collection('users')
          .doc(response.user?.uid)
          .set({
            username: username,
          })
          .then(() => {
            history.push('/home');
          })
          .catch((error) => {
            console.log(error);
          });
      })
      .catch((err: firebaseError) => {
        setMessage(err.message);
      });
  }
};
// ... Implementacija ...
<Button
  variant="contained"
  color="primary"
  onClick={onClickRegister}
  disabled={disableButton}
>
  Register
</Button>
```

Slika 5.8. Potvrda registracije

Podaci se zatim šalju na server pomoću *„auth.createUserWithEmail AndPassword“* no prije samog korištenja funkcije za stvaranje novog korisnika bilo je potrebno omogućiti sve načine autentifikacije na Firebase poslužitelju .



Slika 5.9. Metode prijave

5.3.2. Prijava – pomoću e-maila

Korisnik se u aplikaciju može prijaviti na dva načina, pomoću podataka koje je unio u postupku registracije ili pomoću vanjskih poslužitelja. Prijava pomoću podataka s kojima se registrirao se vrši putem unosa istih podataka u formu i potvrde njihovog valjanog unosa.

```
const onClickLogIn = () => {
  auth
    .setPersistence(saveUser ? persistenceLocal : persistenceNone)
    .then(() => {
      auth
        .signInWithEmailAndPassword(email, password)
        .then(() => {
          history.push('/home');
        })
        .catch((err: firebaseError) => {
          setMessage(err.message);
        });
    });
};
```

Slika 5.10. Prijava pomoću e-mail adrese

Kada se korisnik prijavljuje pomoću e-maila, može odabrati opciju *„Remember me“*. Na *Slika 5.10* je vidljivo ako korisnik odabere ovu opciju prijava se vrši s prije definiranim *„persistenceLocal“* koji čuva korisničke podatke sve dok se internet preglednik ne zatvori.

5.3.3. Prijava – pomoću vanjskog poslužitelja

Ako korisnik odabere prijavu preko vanjskog poslužitelja, to može učiniti klikom na jednu od dvije ikone (Google ili Facebook) na ekranu za prijavu. Nakon klika na ikonu otvara se skočni prozor u kojemu korisnik unosi svoje podatke za odabranog poslužitelja. I ovaj način prijave također koristi Firebase autentifikaciju.

```
const onClickLoginWithProvider = () => {
  auth
    .signInWithPopup(provider)
    .then((result) => {
      firestore
        .collection('users')
        .doc(result.user?.uid)
        .set({
          username: result.user?.displayName,
        })
        .then(() => {
          history.push('/home');
        })
        .catch((error) => {
          console.log(error);
        });
    })
    .catch((error) => {
      console.log(error);
    });
};
```

Slika 5.11. Prijava pomoću vanjskog poslužitelja

Nakon uspješne registracije/prijave u aplikaciju korisnik je preusmjeren na naslovnu stranicu. To se može vidjeti na *Slika 5.8.*, *Slika 5.10.* i *Slika 5.11.* Preusmjeravanje korisnika se vrši primjenom *„history.push('[putanja]')“* naredbe.

5.3.4. Odjava

Korisnik nakon svakog korištenja aplikacije može odlučiti i odjaviti se s iste. Nakon odjave korisnički podaci unutar aplikacije se brišu, kao i određeni podaci o receptima i namirnicama. Proces odjave korisnika vidljiv je na *Slika 5.12*.

```
const onClickLogout = () => {
  auth.signOut().then(() => {
    dispatch(logoutCurrentUser());
    dispatch(clearFridgeIngredients());
    dispatch(clearLikedRecipes());
    history.push('/login');
  });
};
```

Slika 5.12. Odjava korisnika

5.4. Lokalna baza – Redux

U ovome poglavlju bit će objašnjeno kako se koristi Redux, svojevrsna lokalna baza podataka. Naime, na *Slika 5.12.* vidljivo je korištenje funkcije *„dispatch“* ona nam služi za obavještavanje Redux-a da se poziva funkcija iz njega. U slijedećem tekstu bit će objašnjena uporaba Redux-a na primjeru pohrane i dohvaćanja namirnica u hladnjaku.

5.4.1. Akcije

Funkciju koju možemo vidjeti unutar *„dispatch()“* stvaramo unutar redux akcija. Prvo je potrebno definirati tipove redux akcija, kao što je vidljivo na *Slika 5.13*. Postoje dvije vrste akcije: akcija s payloadom (podacima koje predajemo reduceru) i akcija bez payloada (podaci se ne predaju, već se samo odrađuje nekakva akcija).

```
interface SetFridgeIngredientsAction {
  type: typeof SET_FRIDGE_INGREDIENTS;
  payload: FridgeIngredient;
  page: number;
  rowsPerPage: number;
}

interface GetFridgeIngredientsAction {
  type: typeof GET_FRIDGE_INGREDIENTS;
  payload: FirebaseFridgeIngredientsState;
}

interface ClearFridgeIngredientsAction {
  type: typeof CLEAR_FRIDGE_INGREDIENTS;
}

interface DeleteFridgeIngredientAction {
  type: typeof DELETE_FRIDGE_INGREDIENT;
  payload: string;
}
```

```
export type FridgeIngredientsActions =
  | SetFridgeIngredientsAction
  | GetFridgeIngredientsAction
  | ClearFridgeIngredientsAction
  | DeleteFridgeIngredientAction;
```

Slika 5.13. Definiranje tipova akcija

Nadalje, nakon što su tipovi akcija definirani potrebno je uz svaki tip akcije definirati i funkciju, odnosno što će točno ta akcija napraviti. Na *Slika 5.14.* se vide dvije akcije, jedna je akcija postavljanja namirnica u hladnjak, dok druga akcija izvršava dohvaćanje namirnica koje smo spremili u hladnjak. Dohvaćanje se izvršava putem Firestore baze podataka. Kod postavljanja namirnica u hladnjak akcija se izvršava sinkrono, jer u trenutku kada želimo izvršiti akciju podatak postoji u memoriji. No, kada izvršavamo akciju dohvaćanja namirnica iz baze podataka akciju moramo izvršavati asinkrono, odnosno nakon što pozovemo funkciju dohvaćanja podataka moramo pričekati da se podaci obrade i pošalju s poslužitelja.

```
export const setFridgeIngredients = (
  id: string,
  ingredient: string,
  rowsPerPage: number,
) => {
  return {
    type: SET_FRIDGE_INGREDIENTS,
    payload: { id: id, ingredient: ingredient },
    rowsPerPage: rowsPerPage,
  };
};

export const getFridgeIngredients = (
  page: number,
  rowsPerPage: number,
): ThunkAction<
  void,
  typeof fridgeIngredientsReducer,
  null,
  FridgeIngredientsActions
> => {
  return async (dispatch) => {
    firestore
      .collection('fridges')
      .doc(auth.currentUser?.uid)
      .get()
      .then((doc) => {
```

```
        const total = (doc.data() as FirebaseFridgeIngredientsState).total;
        const ingredients = (
          doc.data() as FirebaseFridgeIngredientsState
        ).ingredients
          .reverse()
          .slice(page * rowsPerPage, page * rowsPerPage + rowsPerPage);

        dispatch({
          type: GET_FRIDGE_INGREDIENTS,
          payload: {
            total: total,
            ingredients: ingredients,
          },
        });
      });
  };
};
```

Slika 5.14. Prikaz sinkrone i asinkrone akcije

Kao što je prije spomenuto postoje dvije vrste akcija: akcija koja predaje podatke i ona koja ne predaje podatke. Na *Slika 5.14.* su vidljive dvije akcije od kojih obje predaju podatke u reducer. Dok na *Slika 5.15.* možemo vidjeti akciju koja ne predaje podatke u reducer već samo govori reduceru koju akciju s već postojećim podacima treba obaviti.

```
export const clearFridgeIngredients = () => {
  return { type: CLEAR_FRIDGE_INGREDIENTS };
};
```

Slika 5.15. Prikaz akcije bez payloada

5.4.2. Reducer

Reducer se bavi upravljanjem podataka, odnosno *„state-a“* aplikacije. Na istome primjeru možemo vidjeti kako reducer upravlja podacima. Na *Slika 5.16.* je vidljivo da u početku mora biti definirana neka početka vrijednost podatka.

```
const INITIAL_STATE: FridgeIngredientsState = {
  total: 0,
  data: [],
};
```

Slika 5.16. Reducer - početna vrijednost

Nakon što je definirana početna vrijednost ovisno o tome koja je akcija izvršena reducer vrši akcije nad podacima – *Slika 5.17*. Na primjer ako je pozvana akcija *„setFridgeIngredients“* reducer će na već postojeće podatke dodati novu poslanu namirnicu na prvo mjesto, a ostale u podacima o već prije zadanim namirnicama izbaciti zadnju namirnicu, ako ih je više od onoliko koliko ih se može prikazati na jednoj stranici u tablici *„itemsPerPage“*.

```
export const fridgeIngredientsReducer = (
  state: FridgeIngredientsState = INITIAL_STATE,
  action: FridgeIngredientsActions,
) => {
  switch (action.type) {
    case SET_FRIDGE_INGREDIENTS: {
      return {
        ...state,
        data: [
          { id: action.payload.id, ingredient: action.payload.ingredient },
          ...state.data.slice(0, action.rowsPerPage - 1),
        ],
      };
    }
    case GET_FRIDGE_INGREDIENTS: {
      return {
        ...state,
        total: action.payload.total,
        data: action.payload.ingredients,
      };
    }
    case CLEAR_FRIDGE_INGREDIENTS: {
      return {
        data: [],
      };
    }
    case DELETE_FRIDGE_INGREDIENT: {
      return {
        ...state,
        data: state.data.filter((item) => item.id !== action.payload),
      };
    }
    default: {
      return state;
    }
  }
};
```

Slika 5.17. Reducer - manipulacija podacima

5.4.3. Store – Skladište

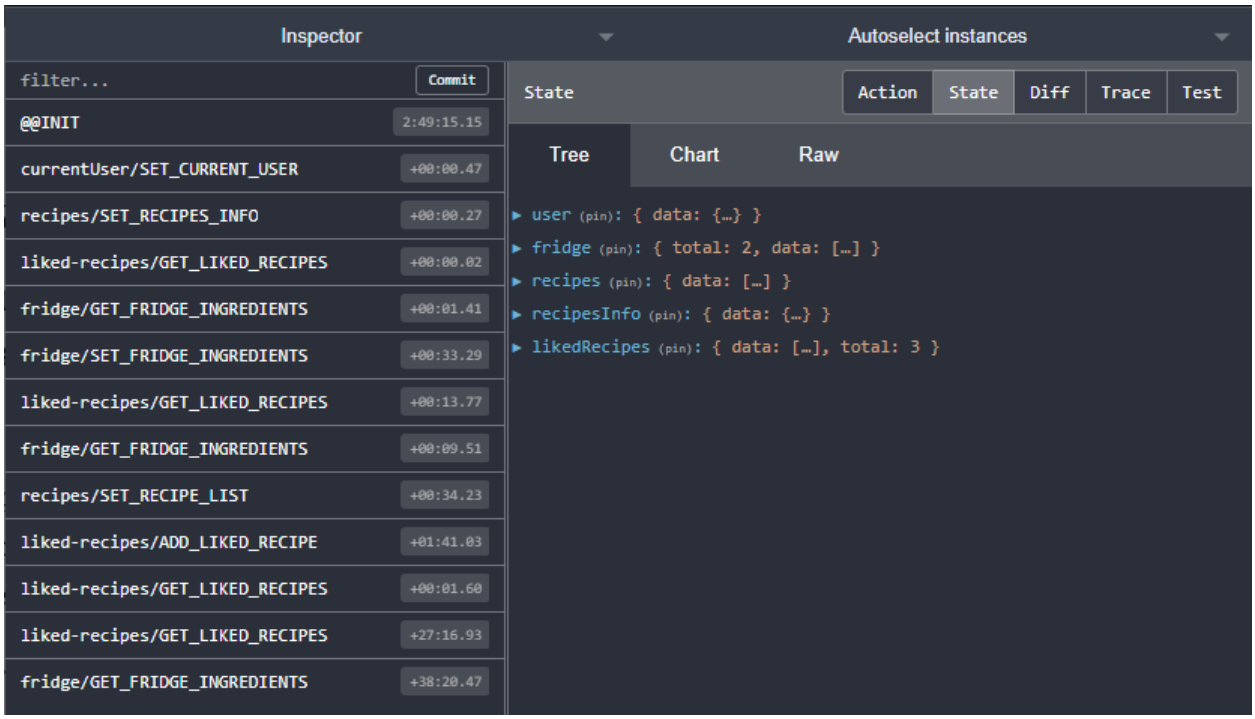
Za pristpanje podacima kojima reducer manipulira uz pomoć definiranih akcija potrebno je definirati skladište podacima na cijelu aplikaciju.

```
export const store = createStore(
  combineReducers({
    user: currentUserReducer,
    fridge: fridgeIngredientsReducer,
    recipes: recipesListReducer,
    recipesInfo: infoRecipeReducer,
    likedRecipes: likedRecipesReducer,
  }),
  composeWithDevTools(applyMiddleware(thunk)),
);
```

Slika 5.18. Definicija skladišta

5.4.4. Redux – DevTools (Programerski alati)

Tijekom korištenja redux-a i manipulacije i pohrane podacima, te iste podatke možemo nadzirati pomoću programerskog alata za redux. Alat je od iznimne koristi, jer daje konstantan uvid u podatke unutar aplikacije.



Slika 5.19. Redux DevTools

5.5. Baza podataka – Firestore

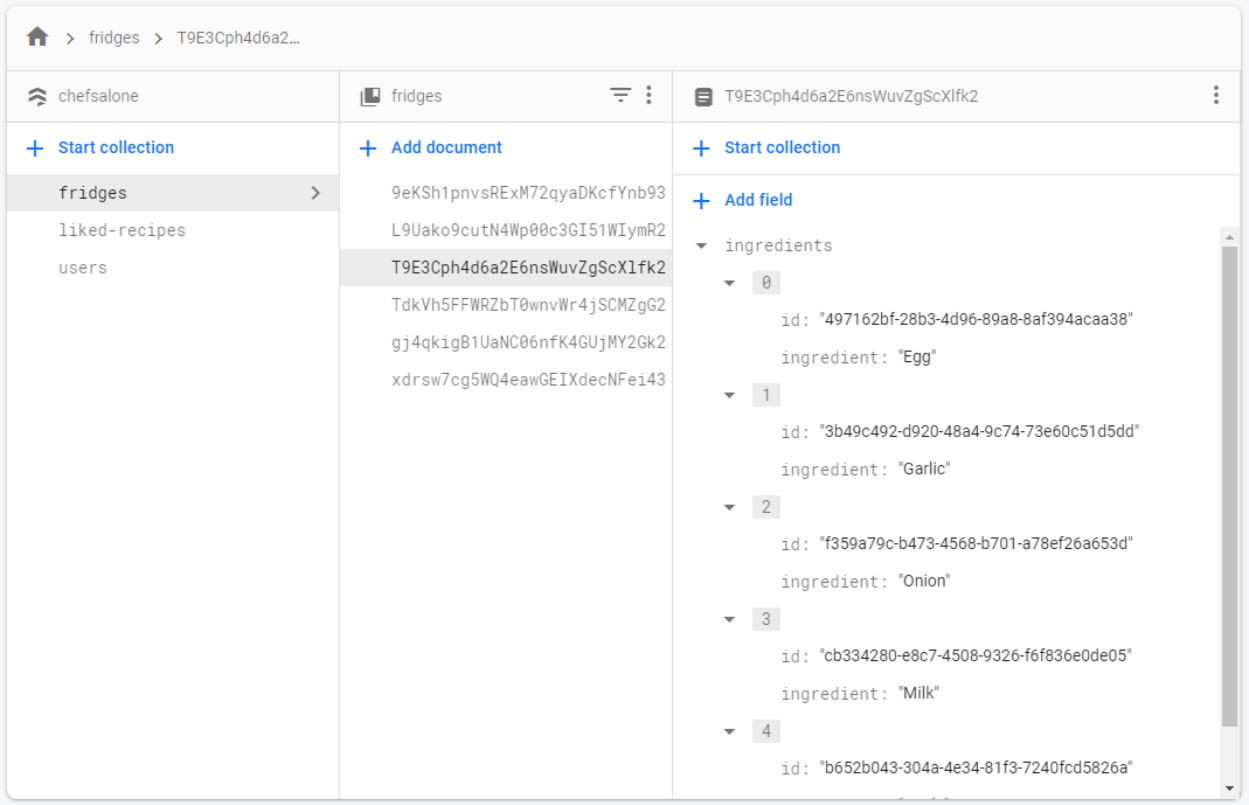
U ovom poglavlju će biti prikazan način pohrane podataka i njihovo dohvaćanje s Firestore-a. Bazu podataka se koristi za pohranu i čuvanje podataka o korisniku. Potrebno je sačuvati podatke o njegovim namirnicama u hladnjaku, podatke o registraciji i podatke o receptima označenim sa *„sviđa mi se“* oznakom.

Korisnik kada pristupi ekranu hladnjaka, potrebno je dohvatiti sve namirnice koje je do tada unio, ako one postoje. Dohvaćanje se izvodi na slijedeći način, *Slika 5.20. Firestore - dohvaćanje namirnica*.

```
firestore
  .collection('fridges')
  .doc(auth.currentUser?.uid)
  .get()
  .then((doc) => {
    dispatch({
      type: GET_FRIDGE_INGREDIENTS,
      payload: {
        total: (doc.data() as FirebaseFridgeIngredientsState).total,
        ingredients: (doc.data() as FirebaseFridgeIngredientsState)
          .ingredients,
      },
    });
  });
```

Slika 5.20. Firestore - dohvaćanje namirnica

Za dohvaćanje podataka prvo treba unijeti putanju u kojoj se oni nalaze. Podaci o namirnicama se u bazi nalaze u kolekciji nazvanoj *„fridges“*. Firestore baza se sastoji od kolekcija i dokumenata, *Slika 5.21. Firestore - prikaz baze*. Tako kolekcija *„fridges“* sadrži u sebi više dokumenata od kojih su svi nazvani po ID-evima svojih korisnika. ID korisnika dohvaćamo pomoću ugrađene firebase funkcije. Nakon što je putanja zadana podatke možemo dohvatiti i obraditi.



Slika 5.21. Firestore - prikaz baze

Svaka manipulacija podataka odrađuje se lokalno, te na kraju kada se podatak više neće mijenjati šalje se na pohranu u Firestore bazu. Gdje će se podatak spremiti isto treba odrediti putanjom kolekcija i dokumenata *Slika 5.22. Firestore - dodavanje vrijednosti*.

```
firestore
  .collection('fridges')
  .doc(auth.currentUser?.uid)
  .update({
    ingredients: firestoreRef.FieldValue.arrayUnion({
      id: id,
      ingredient: ingredient,
    }),
  });
```

Slika 5.22. Firestore - dodavanje vrijednosti

5.6. Spoonacular API – baza podataka recepata

Nakon što korisnik unese namirnice potrebno mu je omogućiti dohvaćanje recepata. Recepti se dohvaćaju s API-ja. Način dohvaćanja recepata s API-ja vidljiv je na *Slika 5.23. Spoonacular API - dohvaćanje recepata*.

```
axios
  .get(
    `https://api.spoonacular.com/recipes/findByIngredients?${
      process.env.REACT_APP_SPOONACULAR_API_KEY
    }&ingredients=${ingredients.join(',+')}&number=15`,
  )
```

Slika 5.23. Spoonacular API - dohvaćanje recepata

Za dohvaćanje recepata sa API-ja potrebno je koristiti *„axios"* pomoću kojega dohvaćamo podatke. Spoonacular API uz dohvaćanje recepata po namirnicama, omogućava i dohvaćanje recepta po ID-u recepta čiju opciju koristimo kada se dogodi klik na određeni recept. Omogućava dohvaćanje više recepata preko ID-a, ta opcija je korištena kada se dohvaćaju recepti koji su označeni s oznakom sviđa mi se.

6. PRIKAZ RADA APLIKACIJE

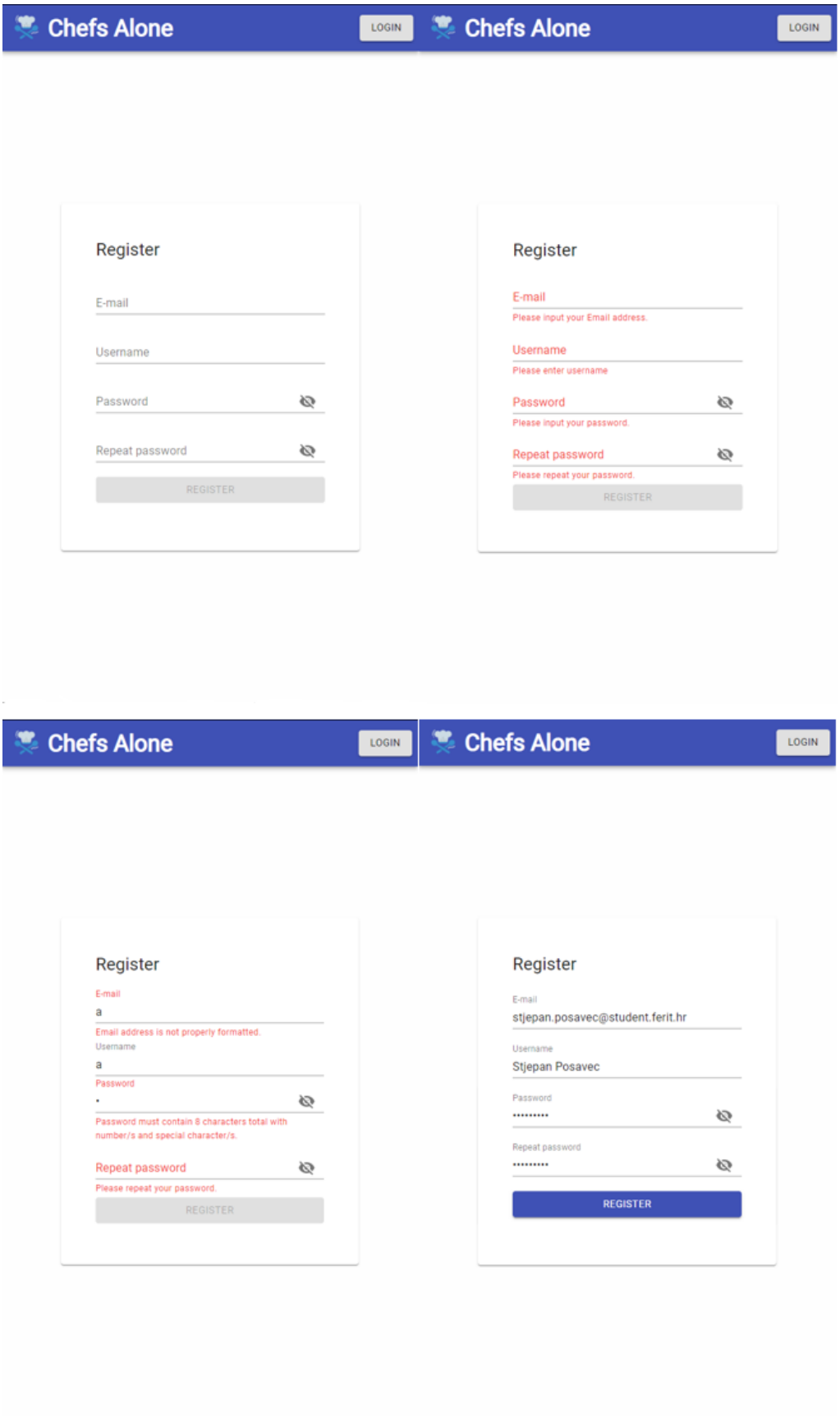
Ovo poglavlje detaljno opisuje sam rad aplikacije, njene funkcionalnosti i način korištenja određenih segmenata. Korisnik slijedeći ove korake iskorištava aplikaciju u potpunosti.

6.1. Slijedni prikaz korištenja aplikacije

U ovom prikazu korištenja aplikacije proći ćemo kroz aplikaciju od samog početka korištenja, odnosno od pravljenja novog korisničkog računa (koji ćemo napraviti na više ponuđenih načina), pa sve do pretrage recepta i označivanja istog s oznakom *„sviđa mi se“*.

6.1.1. Registracija

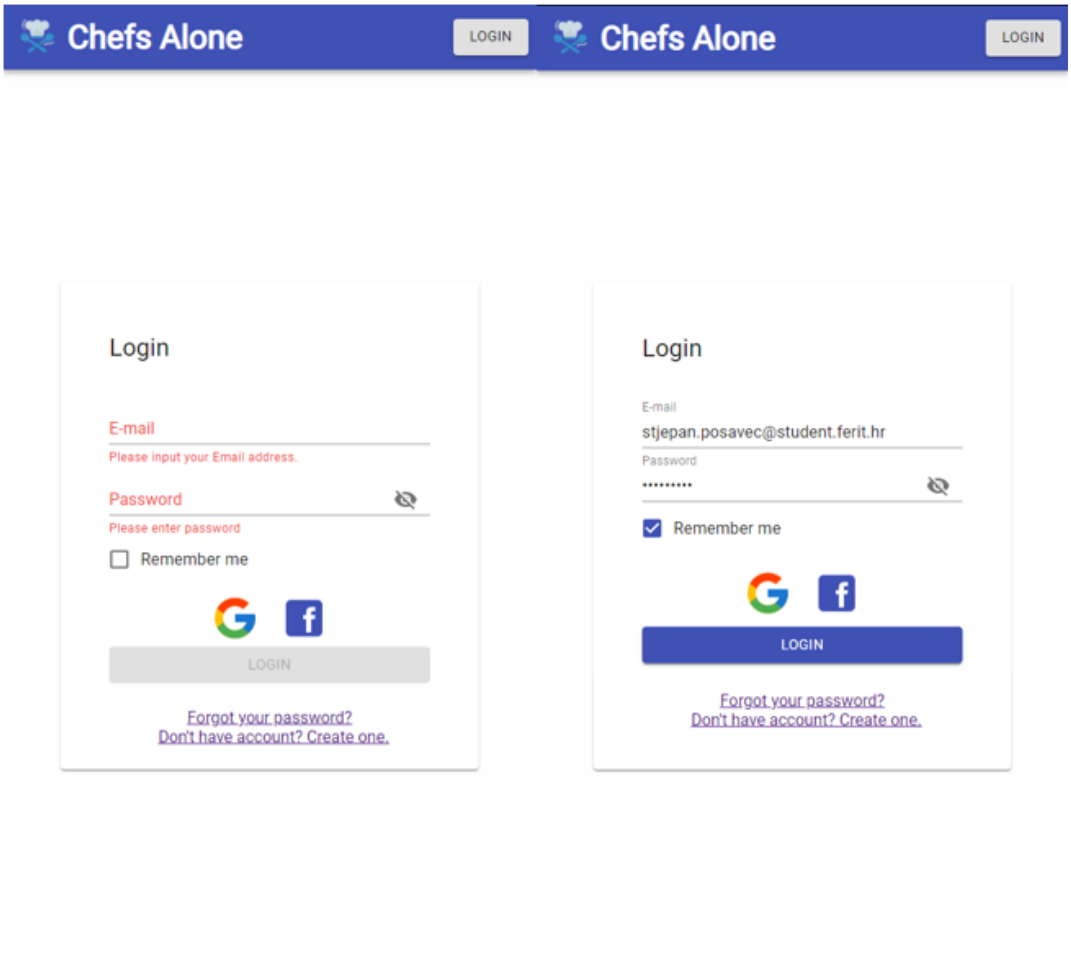
Za početak potrebno je napraviti korisnički račun, njega možemo napraviti tako što upišemo valjane podatke u formu za registraciju. Ako podaci nisu pravilno formatirani aplikacija će javiti grešku i neće dopustiti registraciju. Nakon što uspješno unesemo valjane podatke aplikacija će nam omogućiti registraciju, te klikom na gumb *„Register“* izvršavamo registaciju, ako je registracija uspješna korisnik će biti preusmjeren na naslovnu stranicu aplikacije. Koraci registracije vidljivi su na *Slika 6.1*.



Slika 6.1. Prikaz ekrana registracije

6.1.2. Prijava

Korisnik se u aplikaciju može prijaviti na dva načina: unosom podataka s kojima je izvršio registraciju korisničkog računa ili pomoću vanjskog poslužitelja (Google ili Facebook računi). Prvo ćemo proći kroz prijavu s korisničkim podacima s kojima je izvršena registracija kao što je vidljivo na *Slika 6.2*.

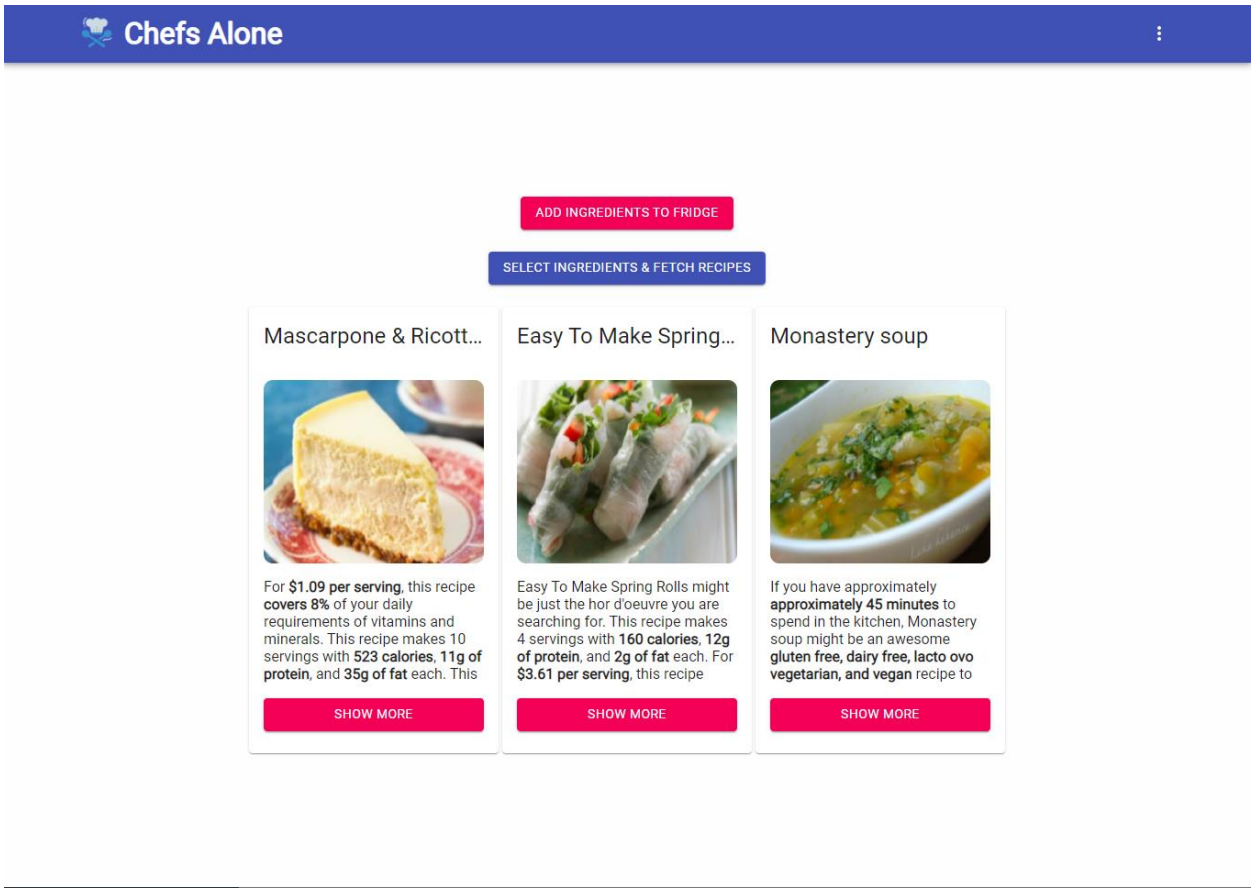


Slika 6.2. Prikaz ekrana prijave

Također, prijavu u aplikaciju možemo izvršiti i pomoću vanjskog poslužitelja. Klikom na jednu od dvije ikone otvara se skočni prozor u kojemu imamo opciju odabira, odnosno unosa podataka za prijavu na Google ili Facebook korisnički račun. Uspješnom prijavom aplikacija korisnika preusmjerava na naslovnu stranicu aplikacije.

6.1.3. Naslovna stranica

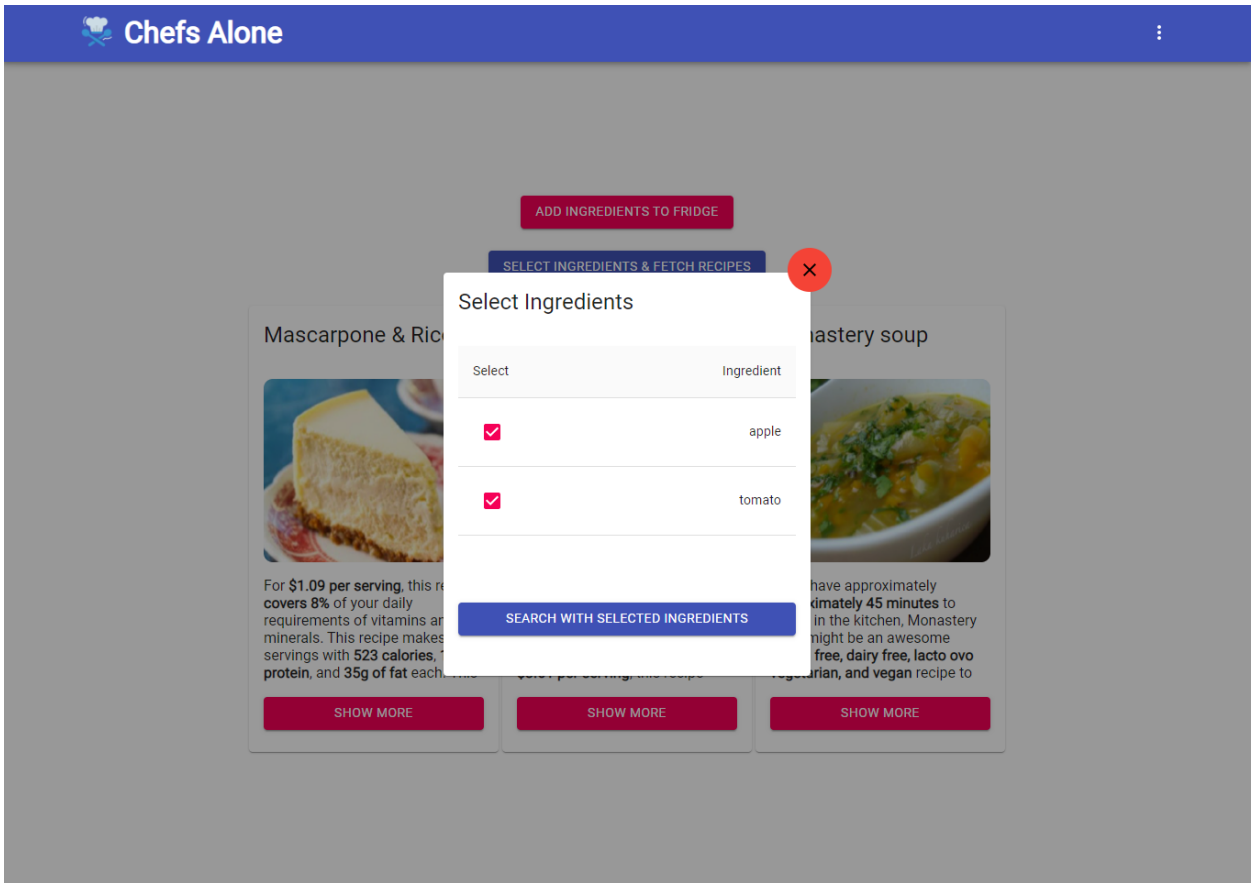
Uspješnom prijavom u aplikaciju korisnik se sada nalazi na naslovnoj stranici aplikacije. U ovome trenutku korisniku se na odabir daje nekoliko opcija. Korisnik može pristupiti hladnjaku kako bi dodao namirnice klikom na *„Add Ingredients to Fridge“*, može odabrati opciju odabira unesenih namirnica i pretrage recepta klikom na *„Select Ingredients & Fetch Recipes“*, na odabir još ima i tri nasumično odabrana recepta, te klikom na njih može dobiti detaljan uvid u recept.



Slika 6.3. Prikaz naslovne stranice

Ukoliko korisnik odabere opciju dodavanja namirnica u hladnjak *„Add Ingredients to Fridge“* preusmjerava ga se na ekran hladnjaka – *6.1.4. Hladnjak.*

Ukoliko korisnik odabere opciju odabira namirnica i pretrage recepta *„Select Ingredients & Fetch Recipes“* otvara se prozor u kojemu korisnik može odabrati namirnice i pretražiti recepte – *Slika 6.4.*

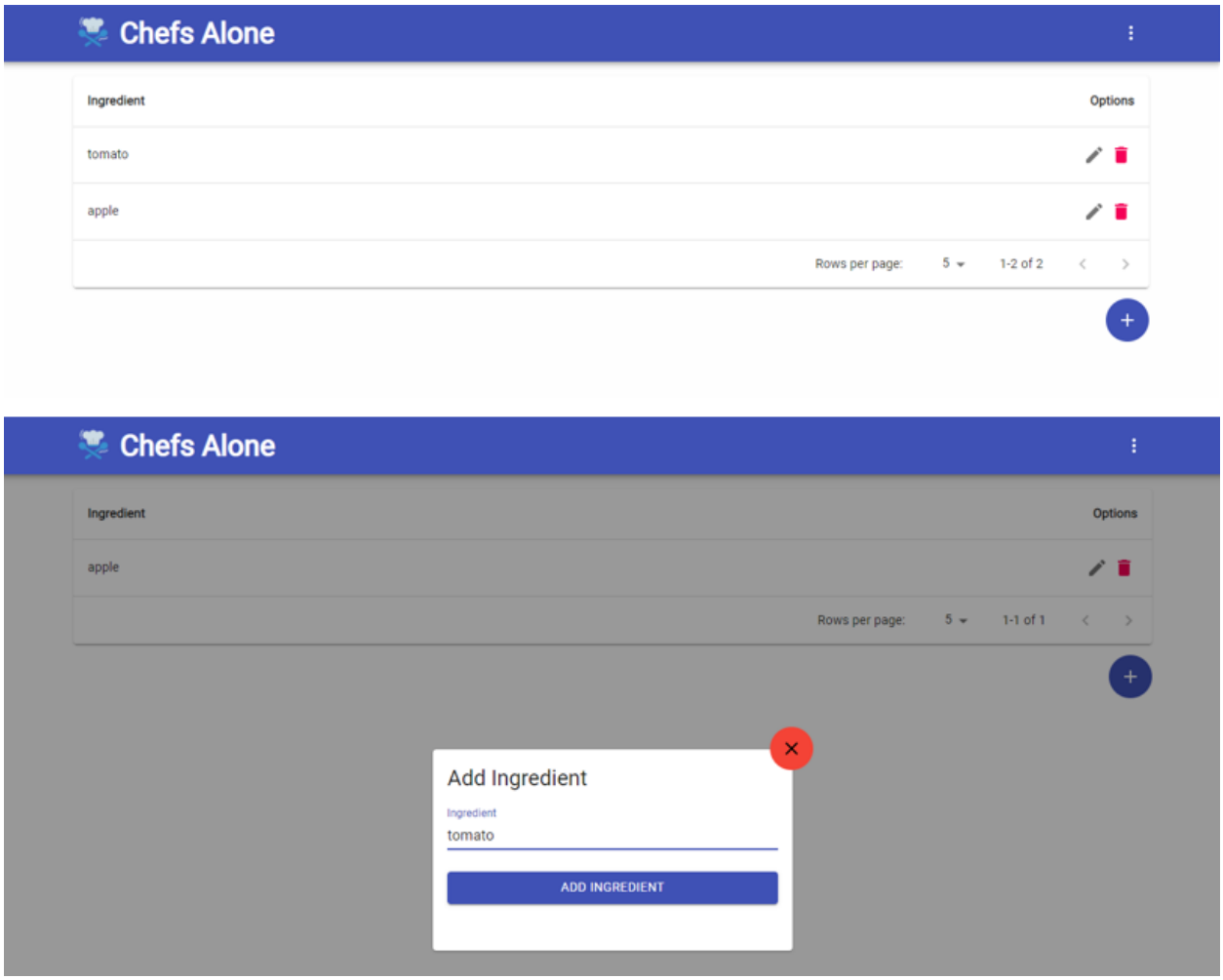


Slika 6.4. Odabir namirnica

6.1.4. Hladnjak

Kada je korisnik odabrao opciju *„Add Ingredients to Fridge“* preusmjeren je na ekran koji je vidljiv na *Slika 6.5. Prikaz tablice hladnjaka i dodavanja namirnice*. Na ekranu je vidljiva tablica u koju korisnik unosi namirnice u svome hladnjaku. Klikom na „+“ gumb otvara se prozor u kojemu korisnik upisuje ime namirnice koju želi dodati u tablicu, odnosno u svoj hladnjak. Svaku dodanu namirnicu korisnik može obrisati ili urediti. Namirnicu uređuje klikom na sivu ikonu olovke, dok namirnicu može obrisati klikom na crvenu ikonu kante za smeće.

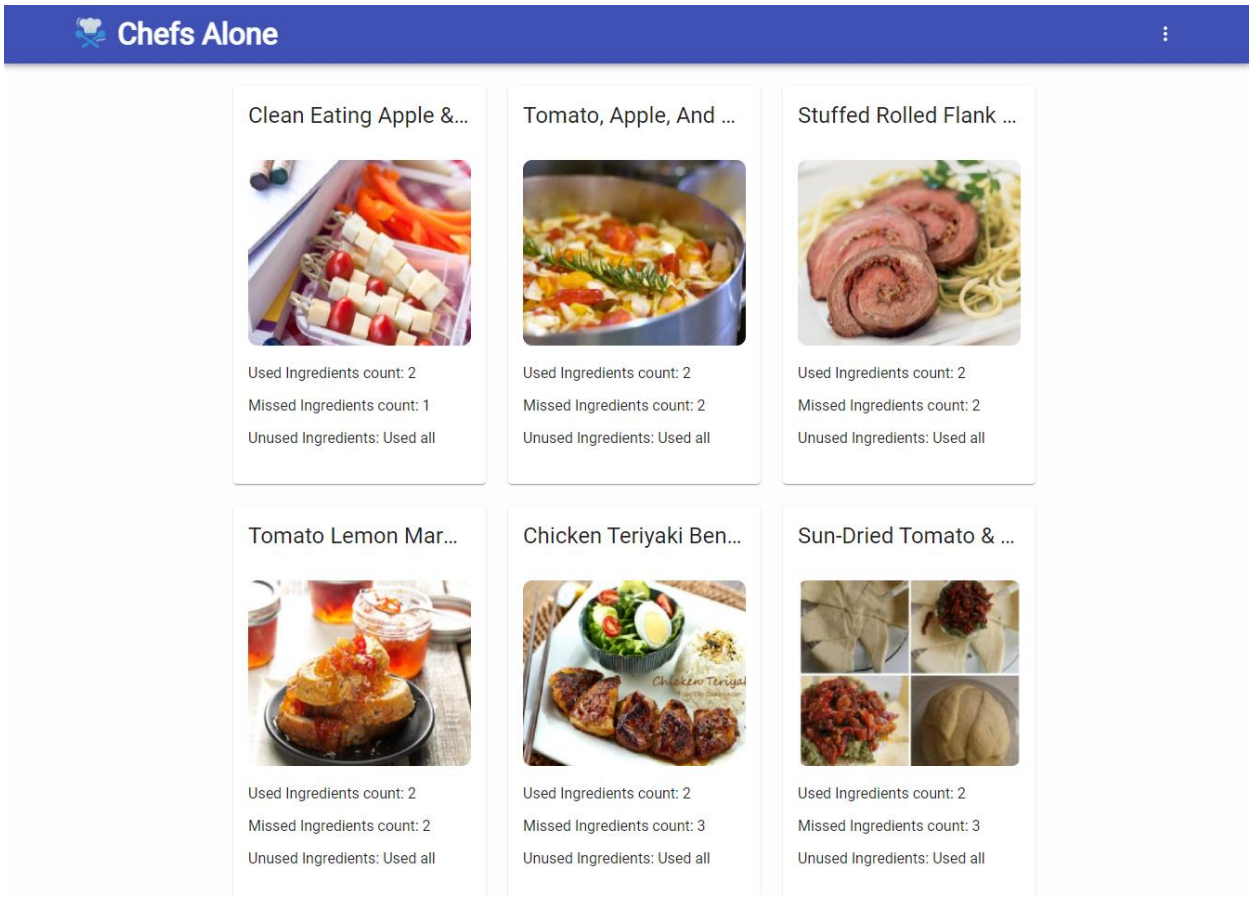
Također, korisniku je dan izbor na koji način želi prikazati svoje namirnice u tablici. Na odabir mu je ponuđeno želi li prikazati 5, 10 ili 25 namirnica po stranici. Slijedno tome korisnik može prebacivati stranice klikom na strelice pored kojih može vidjeti koliko je ukupno namirnicu u hladnjaku i koliko ih je trenutno prikazano.



Slika 6.5. Prikaz tablice hladnjaka i dodavanja namirnice

6.1.5. Odabir namirnice i prikaz recepata

Kada je korisnik na naslovnoj stranici odabrao opciju *„Select Ingredients & Fetch Recipes“* aplikacija otvora prozor u kojemu korisniku daje na odabir namirnice iz hladnjake koje želi upotrijebiti u receptu, kao što je vidljivo na *Slika 6.4*. Korisnik nakon što je odabrao namirnice koje želi upotrijebiti i nakon što je potvrdio odabir klikom na gumb *„Search with Selected Ingredients“* aplikacija ga preusmjeri na stranicu s prikazanim receptima *Slika 6.6. Prikaz liste recepata*.



Slika 6.6. Prikaz liste recepata

Korisnik na listi recepata može vidjeti određene podatke o receptu. Podatke koje vidi su: naziv recepta, slika (potencijalni izgled jela) recepta, koliko je namirnica od njih odabranih iskoristio, koliko mu namirnica nedostaje kako bi mogao jelo napraviti u cijelosti i koliko namirnica od njih odabranih nije iskoristio.

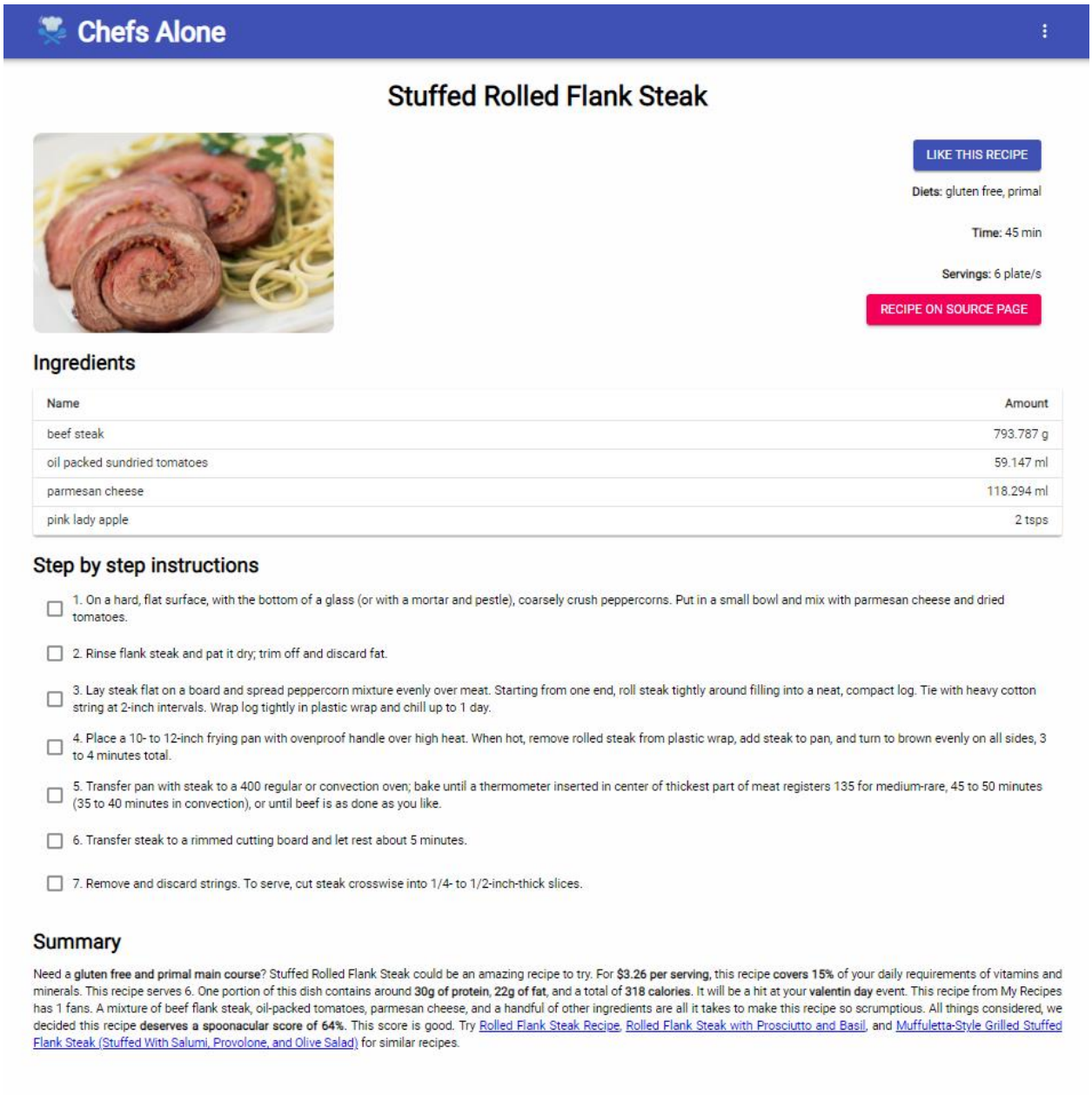
6.1.6. Prikaz detalja recepta

Korisnik nakon što je odabrao namirnice i pretražio koje sve recepte može pripremiti, klikom na jedan od recepata aplikacija ga preusmjerava na ekran koji prikazuje više detalja o receptu, kao što je vidljivo na *Slika 6.7. Prikaz detalja recepta*. Također, korisnik je u mogućnosti recept označiti s oznakom *„sviđa mi se“* nakon što pristupu detaljima recepta.

Korisnik na ekranu s detaljima recepta može pronaći informacije kao što su: naziv recepta, slika recepta koja prikazuje potencijalni izgled jela, vidljivo je kojoj prehrani recept pripada, koliko je vremena potrebno za cjelokupnu pripremu recepta, za koliko porcija, odnosno ljudi je recept zamišljen, koje su sve namirnice potrebne za izradu jela i njihova količina, korak po korak

upute kako recept pripremiti (moguće je pojedini korak označiti kao gotov, kako bi se korisnik lakše snalazio u koracima) i na kraju je moguće pročitat autorov sažetak recepta u kojem se nalaze neki osnovni podaci o receptu, odnosno jelu.

Za korisnika postoji i opcija odlaska na izvorno mjesto gdje se recept nalazi, to može učiniti klikom na gumb *„Recipe on Source Page“*, te će ga aplikacija preusmjeriti na web stranicu recepta.



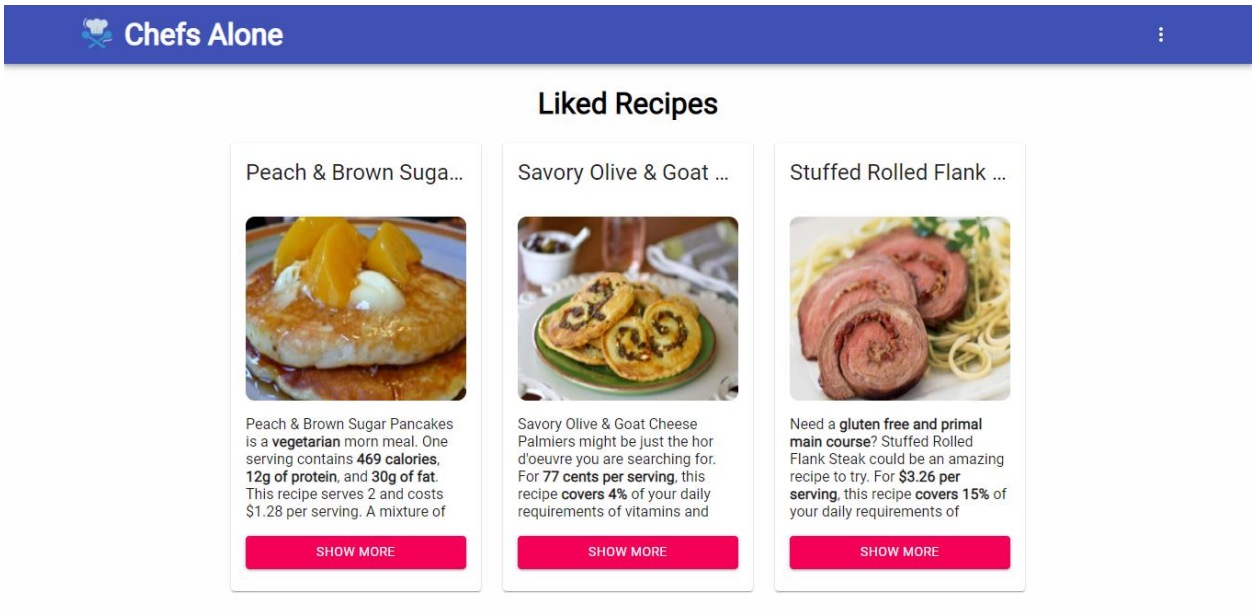
Slika 6.7. Prikaz detalja recepta

6.1.7. Oznaka *„sviđa mi se“*

Korisniku je pružena opcija oznake *„sviđa mi se“* kojom može spremiti recept na posebnu listu unutar aplikacije za brži pristup receptima. Na *Slika 6.7.* vidljiv je gumb kojim korisnik može označiti recept – *„Like this Recipe“*. Nakon što je kliknuo na gumb recept je spremljen, ako želi ukloniti recept s liste to može učiniti klikom na gumb na istoj poziciji koji glasi *„Unlike this Recipe“*. Nakon što je recept označio s oznakom *„sviđa mi se“* receptu može pristupiti na slijedeći način – klikom na opcije u desnome kutu zaglavlja aplikacije i odabirom opcije *„Liked recipes“* kao što je vidljivo na *Slika 6.8. Zaglavlje s aktiviranim opcijama*.

Slika 6.8. Zaglavlje s aktiviranim opcijama

Nakon što je korisnik odabrao opciju *„Liked recipes“* aplikacija ga je preusmjerila na ekran s listom recepata koje je označio sa *„sviđa mi se“*. Na listi recepata za svaki pojedini recept vidljiv je naziv recepta, slika recepta i sažetak recepta . Također, klikom na pojedini recept otvaraju se detalji recepta koji su vidljivi na *Slika 6.7.*



Slika 6.9. Prikaz recepata označenih sa "sviđa mi se"

6.1.8. Odjava

Nakon svakog korištenja aplikacije korisnik se može odjaviti iz aplikacije klikom na gumb *„Logout“* vidljiv na *Slika 6.8*. Odjavom briše sve lokalno spremljene podatke o korisniku.

7. ZAKLJUČAK

U ovome diplomskom radu izrađena je web aplikacija koja korisniku pruža spremanje namirnica iz hladnjake u tablicu povezanu s korisničkim računom kako bi korisnik uvijek imao pristup popisu namirnica. Glavni cilj aplikacije je korisniku olakšati odabir jela koje može pripremiti. Korisniku se na raspolaganje daju mnogobrojni podaci od kojih su, po mome osobnom mišljenju, najvažniji koliko namirnica za pojedini recept je iskoristio i koliko mu njih nedostaje kako bi mogao recept napraviti u cijelosti. Također, aplikacija osim svog glavnog cilja može biti korištena i kao podsjetnik tokom kupovine, jer je popis namirnica u hladnjaku u svakome trenutku dostupan korisniku aplikacije.

Aplikacija je razvijena pomoću React knjižnice s mnogim proširenjima od kojih su najvažniji Firebase i Redux. Aplikaciji je omogućeno korištenje na osobnim računalima i mobilnim telefonima, odnosno izgled aplikacije je prilagođen za korištenje na pametnim uređajima. Aplikacija je testirana na više web preglednika neki od njih su: Google Chrome (primarni razvoj aplikacije se odvijao na ovome pregledniku), Safari i Firefox, te na mobilnim uređajima koje nam Google Chrome programerski alati omogućuju neki od njih su: iPhone 5S, Moto G4, Galaxy Fold i mnogi drugi.

Aplikacija se može proširiti, neki od planiranih proširenja su omogućiti korisniku pohranu i stvaranje vlastitih recepata koji će se pohranjivati unutar Firestore-a. Te u određenom trenutku može doći do napuštanja API-a kao glavnog izvora dohvaćanja podataka o receptu.

SLIKE

LITERATURA

[1] "Supercook," [Online]. Available: https://www.supercook.com/. [Accessed 17 7 2021].

[2] "Tasty," [Online]. Available: https://tasty.co/. [Accessed 17 7 2021].

[3] "BigOven," [Online]. Available: https://www.bigoven.com/. [Accessed 17 7 2021].

[4] "React," [Online]. Available: https://reactjs.org/. [Accessed 21 7 2021].

[5] "React Redux," [Online]. Available: https://react-redux.js.org/. [Accessed 21 7 2021].

[6] "JavaScript," [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Accessed 1 8 2021].

[7] "TypeScript," [Online]. Available: https://www.typescriptlang.org/. [Accessed 1 8 2021].

[8] "JSON," [Online]. Available: https://www.json.org/json-en.html. [Accessed 1 8 2021].

[9] "Firebase," [Online]. Available: https://firebase.google.com/. [Accessed 15 8 2021].

[10] "Spoonacular," [Online]. Available: https://spoonacular.com/food-api. [Accessed 20 8 2021].

SAŽETAK

U ovome diplomskom radu razvijena je web aplikacija za pohranu namirnica iz hladnjaka i pretragu recepata nakon odabira namirnica koje korisnik želi potrošiti. Korisnik nakon registracije i prijave u web aplikaciju ima opciju pohrane namirnica iz hladnjaka, nakon što je unio namirnice može pretražiti koje sve recepte s tim namirnicama može pripremiti. Ako se korisniku neki određeni recept sviđa može ga pohraniti na posebnu listu nazvanu *„sviđa mi se“*.

Ključne riječi: recepti, web aplikacija, React, Firebase, pretraga po namirnicama, pohrana namirnica

ABSTRACT

Title: *Web application cookbook by ingredients in your fridge*

This final work evolves an application that is used to store ingredients and searches recipes by ingredients that the user wants to consume. After registration and logging in, the user has the option to store ingredients from the fridge, after the user stores ingredients. Users can search for recipes that they can make with these ingredients. If the user wants to save some recipes for later, it can be done with the option "like this recipe".

Key words: recipes, web application, React, Firebase, search by ingredients, store ingredients

ŽIVOTOPIS

Stjepan Posavec rođen je u Osijeku, 03.09.1997. godine. Prva četiri razreda osnovne škole pohađao je u OŠ Frana Krste Frankopana, a slijedeća četiri razreda OŠ Ivana Filipovića Osijek. U tom razdoblju je aktivno trenirao vaterpolo, te sudjelovao na brojnim športskim natjecanjima vezano uz vodene sportove. Nakon osnovne škole 2012. godine upisuje željenu Prirodoslovno matematičku gimnaziju u Osijeku. Tijekom srednjoškolskog obrazovanja pokazuje sve više interesa za programiranje, te sudjeluje u natjecanjima na području Informatike. Nakon završetka srednje škole upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, preddiplomski smjer Računarstvo. Nakon završenog preddiplomskog i stečenog zvanja prvostupnika računarstva (Bachelor degree of computer science), odlučuje upisati diplomski studij na istome fakultetu. Upisuje diplomski sveučilišni studij Računarstvo, izborni blok Programsko inženjerstvo gdje se najviše odlučuje specijalizirati kao „frontend developer“. Trenutno radi u firmi PROTOTYP d.o.o. na poziciji Angular developera kao student.

_____

Potpis autora