Funkcionalna specifikacija mobilne aplikacije KupKo – sistem za ustvarjanje jedilnikov na podlagi proračuna, časa in prehranskih omejitev

Ljubljana, januar 2026

Kazalo vsebine

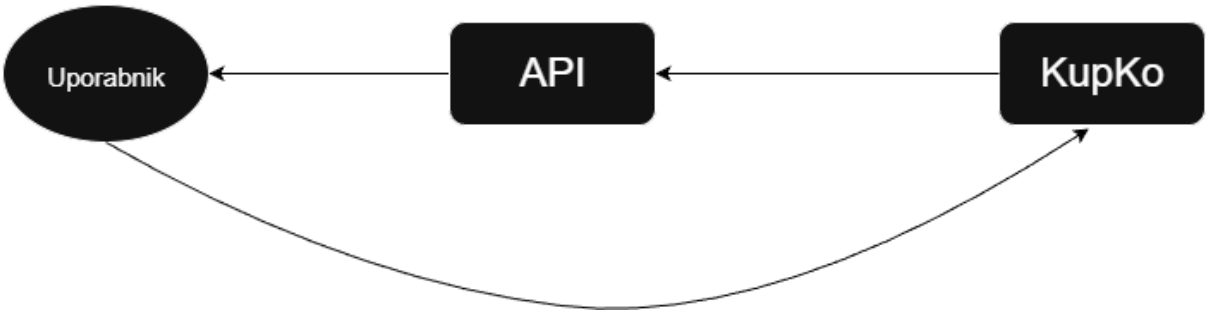
Vsebina:	Funkcionalna specifikacija mobilne aplikacije KupKo za izdelavo prilagojenih jedilnikov na podlagi uporabniških parametrov
Ključne besede:	Mobilna aplikacija, Prilagojeni jedilniki, Načrtovanje prehrane, Proračun, Prehranske omejitve, Alergeni, Čas priprave obrokov
Avtorji:	Blaž Turk, Rok Rihar
Predmet:	Elektronsko in mobilno poslovanje
Datum nastanka:	22.12.2025
Verzija:	1.0

1. Uvod

Dokument podaja funkcionalno specifikacijo aplikacije KupKo, ki natančno opisuje delovanje sistema, njegove glavne funkcionalnosti, podatkovne strukture ter interakcijo med uporabnikom in aplikacijo. Dokument služi kot osnova za razumevanje logike delovanja aplikacije, njeno nadaljnjo nadgradnjo ter morebitno implementacijo.

Aplkacija je namenjena uporabnikom, ki želijo na enostaven način učinkovito načertovati svojo prehrano glede na svoje omejitve in preference.

Uporabniki se pogosto soočajo s pomanjkanjem časa za kuhanje, pri tem pa bi želeli učinkovito razporediti denear za prehrano. Aplikacija pripravi večdnevni jedilnik, nato pa omogoča njegovo urejanje. Uporabnik ga lahko spreminja po želji. Ta aplikacija je uporabna za vse, ki si želijo prihraniti čas, energijo, denar in zdravje pri načrtovanju večdnevnega jedilnika. Aplikacija je povezana z zunanjim API-jem (javno dostopen na naslovu: https://kupko-api.onrender.com/). Ko uporabnik klikne na gumb, se v API-ju pokliče endpoint, ki generira naključen jedilnik, glede na podane parametre.



2. Ozadje delovanja aplikacije

Aplikacija v ozadju uporablja API, ki je sicer javno dostopen, vendar ni namenjen končnim uporabnikom, namenjen je predvsem razvijalcu, ki lahko dodaja poljubne jedi, pri tem pa vnese željene parametre, kot so cena, tip jedi, čas obroka, alergeni in podobno.

Jed se shrani v podatkovno bazo, pri tem se ob klicu endpointa za generiranje jedilnika lahko (ali pa ne) uporabi, saj kot že omenjeno, bo jedilnik vseboval naključne jedi glede na željene parametre uporabnika.

2.1. Delovanje API-ja

API v ozadju uporablja SQLAlchemy (https://www.sqlalchemy.org/) Python knjižnico in Flask, preko katerega so narejeni endpointi.

Podatkovna baza se sestoji iz ene same tabele, ki ima naslednja polja:

Polje	Tip	Zahteva	Opis
id	Integer	Primarni ključ	Avtomatski ID
name	String(100)	Da	Ime jedi (unikaten)
price	Float	Da	Tip jedi (npr. "regular", "vegan")
meal_type	String(100)	Da	Cena v evrih
time_of_day	String(100)	Da	Čas dneva ("breakfast", "lunch", "dinner")
prep_time	Integer	Da	Čas priprave v minutah
allergies	String(255)	Da (ali prazno, če ni alergenov)	Alergeni (ločeni z vejicami)

2.1.1 Končne točke (endpoints)

#	Pot (Endpoint)	Metoda	Opis	Parametri	Odgovor	Napake
1	/	GET	Začetna stran, prikazuje vse jedi v HTML obliki	–	HTML stran (index.html) z vsemi jedmi	–
2	/meal	GET	Pridobi vse jedi	–	JSON array vseh jedi	–
3	/meal/<id>	GET	Pridobi posamezno jed	id (v URL) – ID jedi	JSON ene jedi	404, če jed ne obstaja
4	/meal	POST	Ustvari novo jed	–	JSON ustvarjene jedi (ali potrditev)	–
5	/meal/<id>	PUT	Posodobi obstoječo jed	id (v URL) – ID jedi	JSON posodobljene jedi	–
6	/delete/<id>	GET	Izbriše jed	id (v URL) – ID jedi	Preusmeritev na /	–
7	/random_menu	GET	Generira naključni meni	–	JSON ali HTML naključnega menija	–

Query parametri:

Parameter	Privzeto	Opis
n	7	Število dni v meniju
time_of_day	»Breakfast, lunch, dinner«	Časi dneva (ločeni z vejicami)
allergies	/	Alergeni za izključitev (ločeni z vejicami)
max_price	/	Maksimalna cena (≤)
meal_type	/	Tip jedi (case-insensitive)
time	/	Maksimalni čas priprave (≤)

Repozitorij za API je ločen od same aplikacije.

3. Osrednji del delovanja aplikacije in dizajn

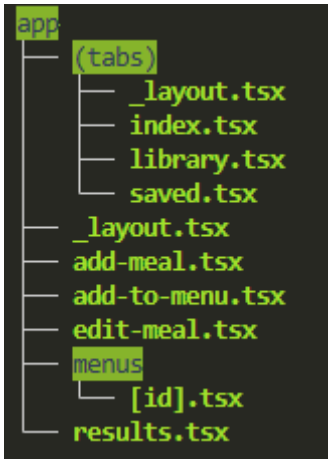
Prvotni prototipi aplikacije so bili zasnovani v Figmi, inspirirani pa so bili po aplikaciji MyFitnessPal.

3.1. Struktura frontend projekta in navigacija (Expo Router)

Mobilna aplikacija je zgrajena z React Native in uporablja Expo Router, kjer je navigacija neposredno povezana s strukturo mape »app/«. Datoteke znotraj mape »app/« predstavljajo zaslone (routes), podmape pa omogočajo organizacijo (npr. »(tabs)« za zavihke). Prehodi med zasloni se izvajajo z »router.push(...)« oziroma »router.back()«, parametri pa se prenašajo prek »params« in berejo z »useLocalSearchParams()«.

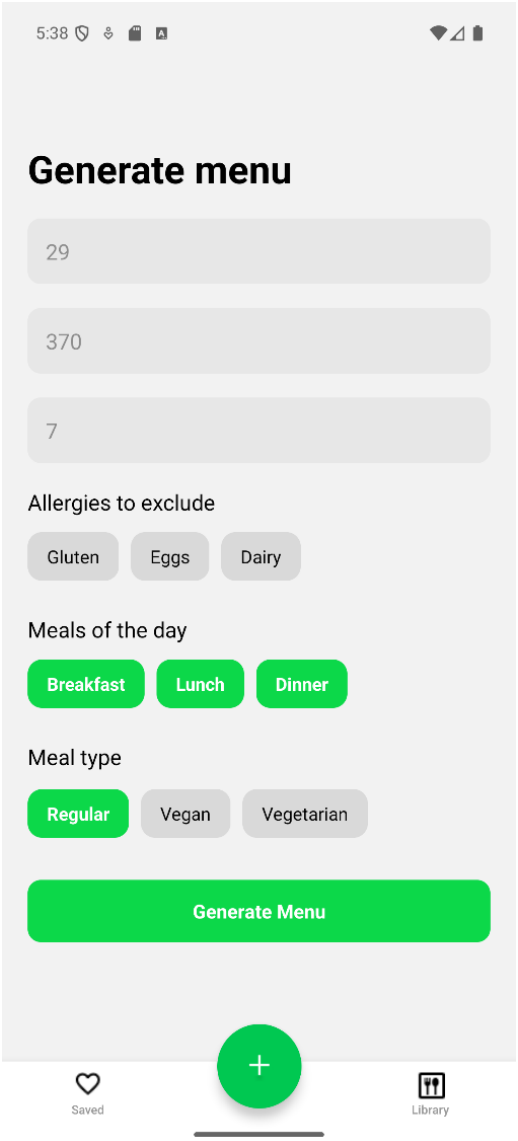
Glavni zasloni v mapi »app/«:

- app/(tabs)/index.tsx – vnos parametrov in generiranje jedilnika prek API-ja.
- app/results.tsx – prikaz generiranih rezultatov in shranjevanje v lokalno shrambo.
- app/(tabs)/saved.tsx – pregled shranjenih menijev po dnevih in urejanje (brisanje/dodajanje).
- app/(tabs)/library.tsx – knjižnica obrokov (dodajanje, urejanje, brisanje, dodajanje v meni).
- app/add-meal.tsx – dodajanje novega obroka v knjižnico.
- app/edit-meal.tsx – urejanje obstoječega obroka v knjižnici.
- app/add-to-menu.tsx – dodajanje izbranega obroka v meni za izbran dan in del dneva.
- app/menus/[id].tsx – dinamična pot (route) za prikaz menija po identifikatorju (če je uporabljena v aplikaciji).



3.2 Zaslon »Generate menu« – /(tabs)/index.tsx

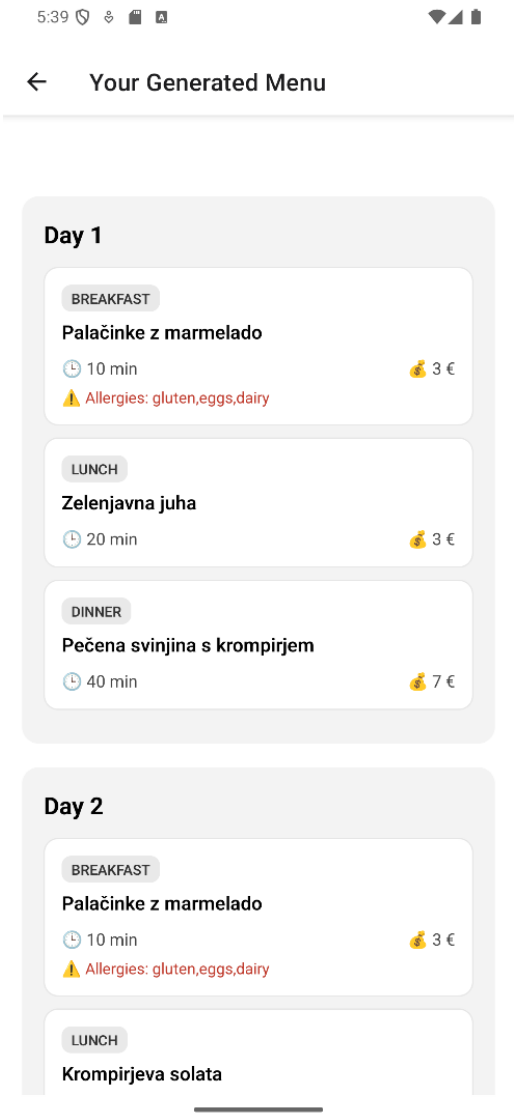
Zaslon omogoča uporabniku, da vnese ključne parametre za generiranje jedilnika: proračun, maksimalni čas priprave, število dni, izločene alergene, želene obroke dneva (npr. breakfast/lunch/dinner) ter tip prehrane (regular/vegan/vegetarian). Parametri se pretvorijo v nizovne predstavitve (npr. »mealsStr«, »allergiesStr«) in se uporabijo pri klicu funkcije »fetchMenu(...)«. Ob kliku gumba »Generate Menu« aplikacija izvede klic API-ja ter ob uspehu navigira na zaslon »/results«. Generirani podatki se prenesejo kot URL parameter »data« v obliki JSON niza, ker Expo Router parametre prenaša kot nize.



3.3 Zaslon »Results« – /results.tsx

Zaslon »Results« iz »useLocalSearchParams()« prejme parameter »data« in ga pretvori v strukturo tipa »DayMenu[]«. Uporabniški vmesnik prikaže meni po dnevih (npr. Day 1, Day 2 …) in znotraj posameznega dne seznam obrokov z metapodatki (čas dneva, ime, čas priprave, cena, alergeni).

Gumb »OK« sproži funkcijo »saveAndGo()«, ki rezultate trajno shrani v lokalno shrambo AsyncStorage: 1) menije doda pod ključ »STORAGE_KEYS.MENUS«, 2) iz vseh dni izlušči obroke in jih shrani v knjižnico pod ključ »STORAGE_KEYS.MEALS«. Pri shranjevanju knjižnice se izvede deduplikacija po »id« (Map), da se isti obrok ne podvaja. Po shranjevanju aplikacija preusmeri uporabnika na zavihek »Saved«.

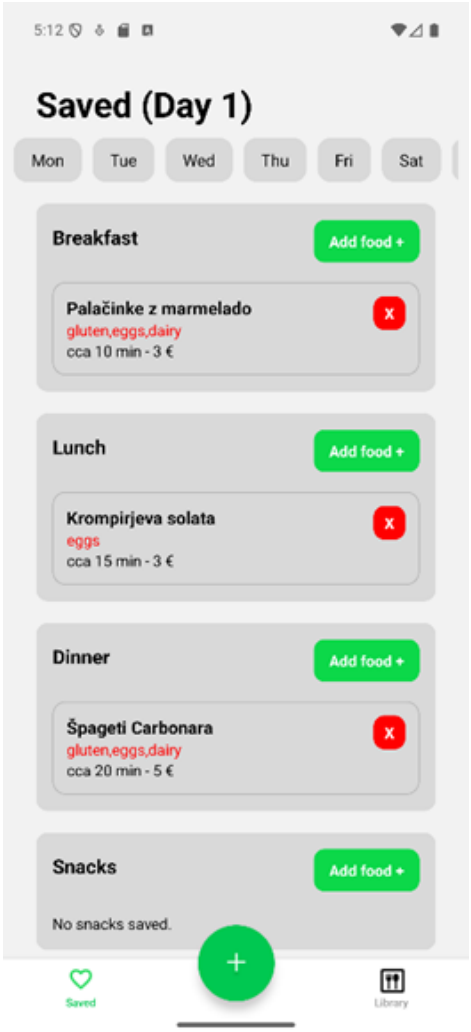


3.4 Zaslon »Saved« – /(tabs)/saved.tsx

Zavihek »Saved« prikazuje shranjen jedilnik po dnevih. Privzeto je izbran današnji dan (izračunan z »getTodayDayNumber()«), uporabnik pa lahko dan spremeni preko horizontalnega seznama (Mon–Sun). Podatki se ob vsakem prihodu na zaslon osvežijo z »useFocusEffect«, kar zagotavlja, da so spremembe po dodajanju ali brisanju takoj vidne.

Jedilnik je razdeljen na sekcije po času dneva (Breakfast/Lunch/Dinner/Snacks). V vsaki sekciji se izrišejo obroki filtrirani po »time_of_day«. Uporabnik lahko posamezen obrok odstrani iz menija (»deleteMeal(selectedDay, mealId)«) ali pa doda novega preko gumba »Add food +«, ki ga preusmeri na zaslon »Library« skupaj s parametri izbranega dneva in časa dneva.

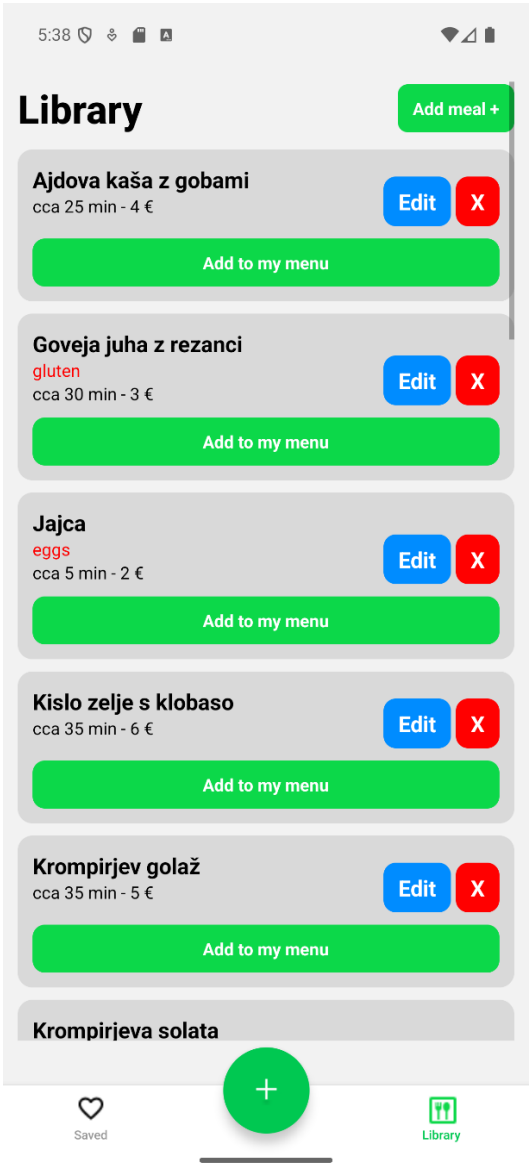
Gumb (se ne vidi) »Delete All« izbriše vse shranjene menije (»deleteAll()«).



3.5 Zaslon »Library« – /(tabs)/library.tsx

Zavihek »Library« predstavlja knjižnico obrokov, ki so shranjeni lokalno (AsyncStorage pod ključem »STORAGE_KEYS.MEALS«). Ob fokusu zaslona se podatki preberejo iz shrambe, filtrirajo na unikatne (po »id«) in se opcijsko sortirajo po imenu.

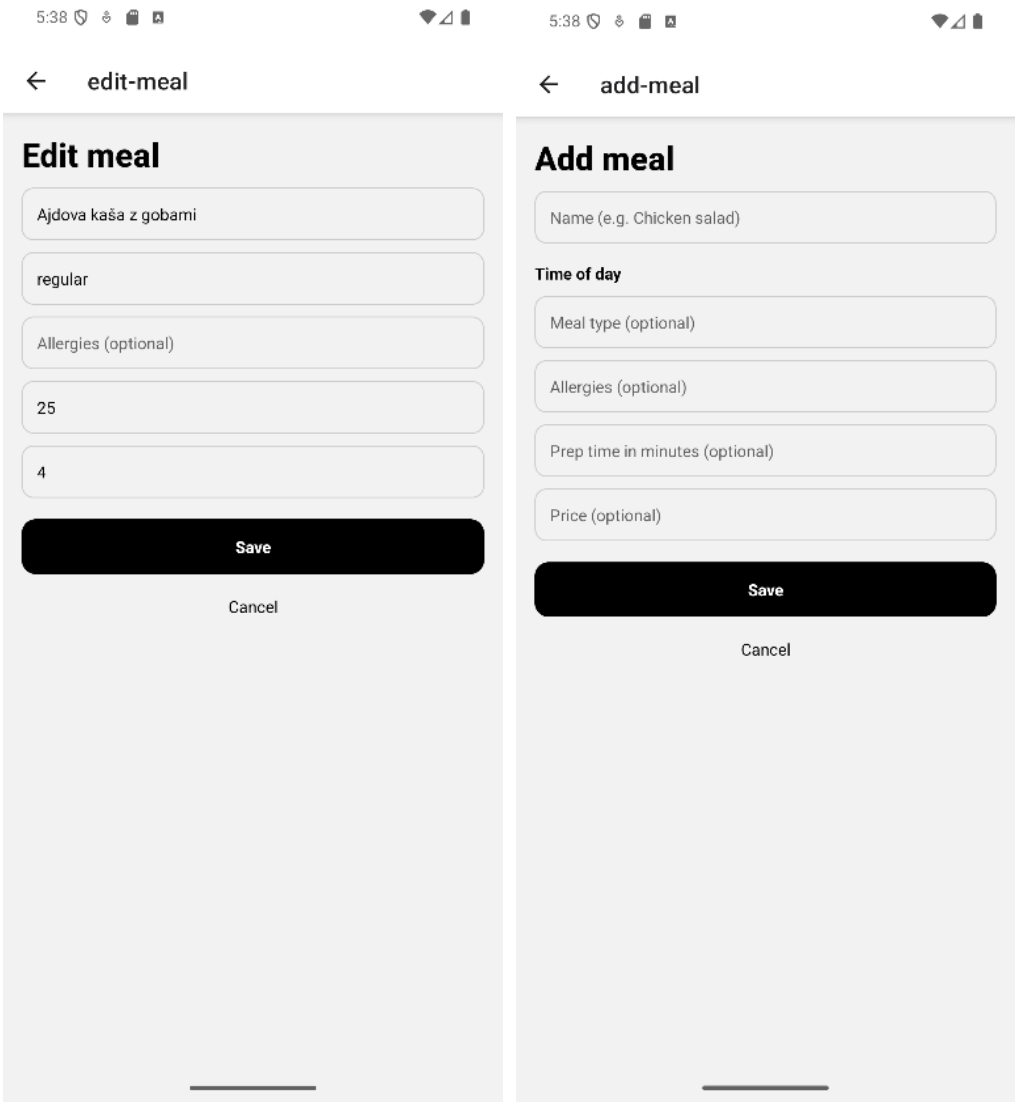
Uporabniške akcije na zaslonu: dodajanje novega obroka (»Add meal +« → /add-meal), urejanje (»Edit« → /edit-meal?mealId=...), brisanje posameznega obroka (»deleteMeal(id)«) ter brisanje vseh obrokov (»deleteAllMeals()«). Funkcija »Add to my menu« uporabnika vodi na /add-to-menu, kjer izbere dan in čas dneva ter obrok doda v shranjen meni.



3.6 Dodajanje in urejanje obrokov

Zaslona »/add-meal« in »/edit-meal« uporabljata skupen podatkovni model (tipi v »lib/types/mealTypes.ts«). Pri vnosu se izvajajo osnovne validacije (npr. obvezno ime, numerični vnos za čas priprave in ceno). Urejanje obroka vnaprej napolni polja z »loadSingleMeal(id)«, shranjevanje pa izvede »updateMeal(meal)«.

Zaslon »/add-to-menu« naloži izbrani obrok iz knjižnice (»loadSingleMeal«), uporabnik izbere dan (Mon–Sun) in čas dneva (breakfast/lunch/dinner/snacks), nato pa se obrok doda v shranjen meni preko »addMeal(day, menuMeal)«.



3.7 Lokalno shranjevanje podatkov (AsyncStorage) in podatkovne strukture

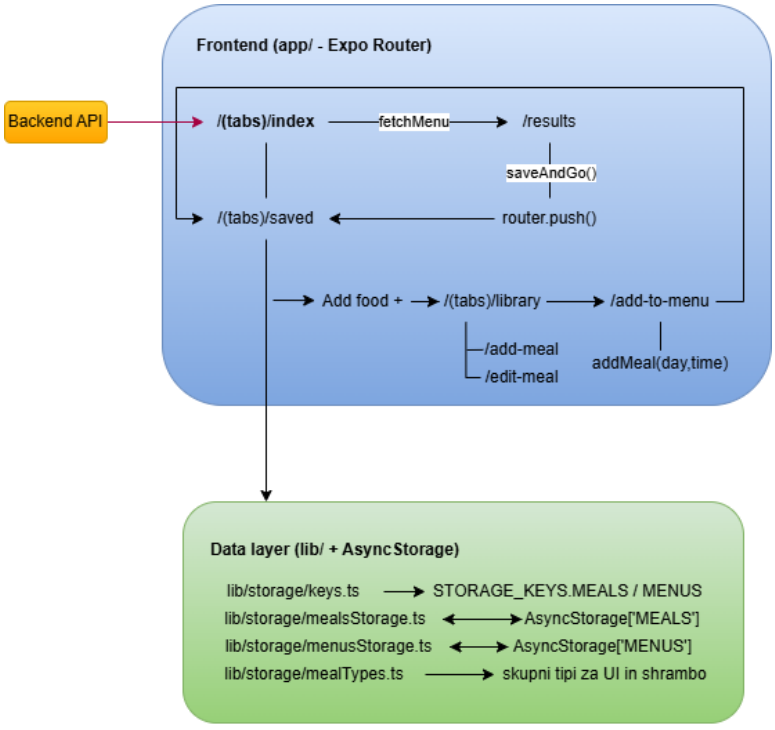
Lokalna shramba je realizirana z AsyncStorage (key–value shramba), pri čemer aplikacija uporablja dodatno plast funkcij v mapi »lib/storage/«, da je logika branja/pisanja ločena od uporabniškega vmesnika.

Vloge posameznih modulov:

- lib/storage/keys.ts – definicije ključev (npr. STORAGE_KEYS.MEALS, STORAGE_KEYS.MENUS).
- lib/storage/mealsStorage.ts – CRUD funkcije za knjižnico obrokov (dodaj, posodobi, izbriši, naloži).
- lib/storage/menusStorage.ts – funkcije za upravljanje menijev po dnevih (dodaj obrok v dan/čas, briši, izbriši vse).
- lib/types/mealTypes.ts – TypeScript tipi (Meal, LibraryMeal, MenuMeal, DayMenu, TimeOfDay), ki poenotijo podatkovni model.

3.8 Diagram arhitekture: frontend in lokalno shranjevanje

Spodnji diagram prikazuje povezavo med zasloni (frontend) in lokalno shrambo. Zasloni ne dostopajo neposredno do JSON struktur v AsyncStorage, temveč uporabljajo funkcije iz »lib/storage/*«, ki skrbijo za branje/pisanje in osnovne operacije (npr. deduplikacija, dodajanje/brisanje).



3.9 Pregled funkcionalnosti po zaslonih

Zaslon (route)	Namen	Ključne akcije	Lokalna shramba
/(tabs)/index	Vnos parametrov za generiranje	fetchMenu(), router.push('/results')	–
/results	Prikaz generiranega menija	parse data, saveAndGo()	MENUS + MEALS (dedupe)
/(tabs)/saved	Pregled/urejanje menija	izbira dneva, brisanje, Add food +	branje/posodobitev MENUS
/(tabs)/library	Knjižnica obrokov	dodaj/uredi/izbriši, Add to my menu	branje/posodobitev MEALS
/add-meal	Dodaj obrok	validacija, addMealToLibrary()	MEALS
/edit-meal	Uredi obrok	loadSingleMeal(), updateMeal()	MEALS
/add-to-menu	Dodaj obrok v meni	izbira dneva/časa, addMealToMenu()	MENUS

4. Zaključek

V tem delu je bila predstavljena funkcionalna specifikacija mobilne aplikacije KupKo, namenjene enostavnemu načrtovanju večdnevnih jedilnikov glede na proračun, čas priprave in prehranske omejitve uporabnika. Dokument opisuje strukturo aplikacije, delovanje uporabniškega vmesnika, povezavo z zunanjim API-jem ter uporabo lokalne shrambe za trajno hranjenje podatkov.

Aplikacija je razvita z uporabo React Native in Expo Routerja, pri čemer je navigacija vezana na strukturo projekta, kar omogoča pregledno organizacijo zaslonov. Podatkovna logika je jasno ločena od uporabniškega vmesnika, saj se dostop do lokalne shrambe izvaja prek abstrakcijskih funkcij v mapi lib/storage, kar izboljšuje vzdrževanje in razširljivost aplikacije.

KupKo uporabniku ne omogoča zgolj generiranja jedilnika, temveč tudi njegovo prilagajanje, shranjevanje ter upravljanje lastne knjižnice obrokov. Aplikacija tako predstavlja dobro osnovo za nadaljnje nadgradnje, kot so sinhronizacija podatkov, naprednejša analiza prehrane ali dodatne uporabniške funkcionalnosti.