Univerzitet Metropolitan

Finance Manager sa D3.js bibliotekom

**Student:** Jova Paunović 5575 **Profesor:** Nikola Dimitrijević

# Sadržaj

* [Uvod](#uvod)
* [Planiranje i dizajn sistema](#planiranje-i-dizajn-sistema)
  + [Definicija zahteva](#definicija-zahteva)
  + [Use-case dijagrami](#use-case-dijagrami)
* [Razvoj korisničkog interfejsa (UI)](#razvoj-korisničkog-interfejsa-ui)
  + [Opis tehnologija za stilizaciju](#opis-tehnologija-za-stilizaciju)
  + [Pregled ključnih stranica](#pregled-ključnih-stranica)
* [Implementacija funkcionalnosti](#implementacija-funkcionalnosti)
  + [Korisničke funkcionalnosti](#korisničke-funkcionalnosti)
  + [Administratorske funkcionalnosti](#administratorske-funkcionalnosti)
* [Simulacija backend-a](#simulacija-backend-a)
  + [Korišćenje json-servera](#korišćenje-json-servera)
  + [Povezivanje frontenda i backend-a](#povezivanje-frontenda-i-backend-a)
* [Zaključak](#zaključak)
* [Prilozi](#prilozi)

# 1. Uvod

## Opis teme projekta

Ovaj projekat se bavi razvojem veb aplikacije za praćenje ličnih finansija. Aplikacija omogućava korisnicima da evidentiraju svoje prihode i rashode, kategorišu troškove, postavljaju budžete i prate svoju finansijsku situaciju kroz vizuelne izveštaje. Cilj je da se korisnicima pruži jednostavan i intuitivan alat za bolje upravljanje novcem.

## Cilj projekta i funkcionalnosti

Osnovni cilj projekta je kreiranje funkcionalne i pregledne aplikacije koja će korisnicima pomoći da steknu uvid u svoje finansijske navike. Ključne funkcionalnosti koje su implementirane uključuju:

* **Autentifikacija korisnika:** Registracija i prijava korisnika sa dva nivoa pristupa (korisnik i administrator).
* **Upravljanje troškovima:** Dodavanje, pregled, izmena i brisanje troškova.
* **Kategorizacija:** Definisanje i upravljanje kategorijama troškova.
* **Vizuelizacija podataka:** Prikazivanje statistike troškova putem grafikona (pite i stubičasti dijagrami).
* **Administrativni panel:** Upravljanje korisnicima i kategorijama.

## Kratak pregled tehnologija i alata

Aplikacija je razvijena korišćenjem modernih veb tehnologija. Za frontend je korišćen **React**, popularna JavaScript biblioteka za izgradnju korisničkih interfejsa. Za rutiranje unutar aplikacije korišćen je **React Router**. Stilovi su definisani pomoću **Tailwind CSS**, “utility-first” CSS frejmvorka koji omogućava brzu i efikasnu stilizaciju. Za vizuelizaciju podataka korišćena je **D3.js** biblioteka. Backend je simuliran pomoću **json-servera**, koji omogućava brzo kreiranje REST API-ja na osnovu JSON fajla.

# 2. Planiranje i dizajn sistema

## 2.1. Definicija zahteva

Funkcionalni zahtevi sistema su definisani tako da omoguće efikasno praćenje finansija. Sistem omogućava dve vrste korisnika: obične korisnike i administratore.

**Korisnici mogu da:**

* Se registruju i prijave na sistem.
* Dodaju nove troškove, unoseći iznos, opis, datum i kategoriju.
* Pregledaju listu svih svojih troškova.
* Menjaju i brišu postojeće troškove.
* Vide vizuelni prikaz svojih troškova na kontrolnoj tabli (dashboard).

**Administratori mogu da:**

* Rade sve što i obični korisnici.
* Pristupe administrativnom panelu.
* Pregledaju listu svih registrovanih korisnika.
* Brišu korisničke naloge.
* Upravljaju kategorijama troškova (dodaju, menjaju, brišu).
* Vide statistiku na nivou cele aplikacije (ukupan broj korisnika, troškova, itd.).

## 2.2. Use-case dijagrami

Sistem ima dva glavna aktera: **Korisnika** i **Administratora**.

A diagram of a company

AI-generated content may be incorrect.**Korisnik** može da izvršava sledeće akcije: \* Registracija \* Prijava \* Odjava \* Pregled kontrolne table \* Pregled liste troškova \* Dodavanje novog troška \* Izmena postojećeg troška \* Brisanje troška

**Administrator** nasleđuje sve akcije korisnika i dodatno može da: \* Pristupi administrativnom panelu \* Upravlja korisnicima (pregled, brisanje) \* Upravlja kategorijama (dodavanje, izmena, brisanje)

# 3. Razvoj korisničkog interfejsa (UI)

## 3.1. Opis tehnologija za stilizaciju

Za stilizaciju korisničkog interfejsa izabran je **Tailwind CSS**. Ovaj frejmvork je odabran zbog svoje “utility-first” prirode, koja omogućava brzo prototipiranje i razvoj bez napuštanja HTML koda. Umesto pisanja posebnih CSS fajlova, stilovi se primenjuju direktno na elemente kroz predefinisane klase. Ovo dovodi do konzistentnog dizajna, lakšeg održavanja i manje duplikacije koda. Konfiguracija se vrši kroz tailwind.config.js fajl.

// tailwind.config.js  
/\*\* @type {import('tailwindcss').Config} \*/  
export default {  
 content: [  
 "./index.html",  
 "./src/\*\*/\*.{js,ts,jsx,tsx}",  
 ],  
 theme: {  
 extend: {},  
 },  
 plugins: [],  
}

## 3.2. Pregled ključnih stranica

### Početna stranica (Login)

Početna stranica je stranica za prijavu. Korisnik unosi svoju email adresu i lozinku kako bi pristupio sistemu. Ukoliko nema nalog, postoji link ka stranici za registraciju. A screenshot of a computer

AI-generated content may be incorrect.

### Stranice za registraciju i prijavu

Stranica za registraciju omogućava novim korisnicima da kreiraju nalog unosom imena, email adrese i lozinke. Stranica za prijavu je već opisana kao početna stranica. A screen shot of a computer

AI-generated content may be incorrect.

### Stranice za pregled podataka

Glavna stranica za korisnika je **Dashboard**, gde može videti sumarni pregled svojih finansija, uključujući ukupnu potrošnju i grafikone koji prikazuju troškove po kategorijama i po danima. Stranica **My Expenses** prikazuje tabelarni pregled svih unetih troškova, sa opcijama za izmenu i brisanje. Forma za dodavanje i izmenu troškova je posebna stranica. A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a graph

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

### Administrativni panel

Administrator ima pristup posebnom panelu koji mu daje pregled statistike cele aplikacije. Sa ovog panela može da pristupi stranicama za upravljanje korisnicima i kategorijama. A screenshot of a computer

AI-generated content may be incorrect.

# 4. Implementacija funkcionalnosti

## 4.1. Korisničke funkcionalnosti

### Registracija i prijava korisnika

Registracija i prijava su implementirane kroz Register.jsx i Login.jsx komponente. Za upravljanje stanjem autentifikacije koristi se AuthContext, koji pruža informacije o trenutno prijavljenom korisniku celoj aplikaciji.

**Login.jsx**

import React from 'react'  
import { useState } from 'react';  
import { useNavigate, Link } from 'react-router-dom';  
import { useAuth } from '../components/AuthContext';  
  
export default function Login() {  
 const [email, setEmail] = useState('');  
 const [password, setPassword] = useState('');  
 const [error, setError] = useState('');  
 const [loading, setLoading] = useState(false);  
   
 const { login } = useAuth();  
 const navigate = useNavigate();  
  
 async function handleSubmit(e) {  
 e.preventDefault();  
   
 try {  
 setError('');  
 setLoading(true);  
 await login(email, password);  
 navigate('/dashboard');  
 } catch (err) {  
 setError('Failed to login. Check your credentials.');  
 } finally {  
 setLoading(false);  
 }  
 }  
  
  
}

### Pregled podataka i interakcija sa njima

Korisnik pregleda svoje troškove na stranici ExpenseList.jsx. Podaci se dobavljaju iz expenseService-a. Komponenta prikazuje troškove u tabeli.

**ExpenseList.jsx**

import React from 'react';  
import { useState, useEffect } from 'react';  
import { Link } from 'react-router-dom';  
import { useAuth } from '../components/AuthContext';  
import { expenseService } from '../services/expenseService';  
  
export default function ExpenseList() {  
 const { currentUser } = useAuth();  
 const [expenses, setExpenses] = useState([]);  
 const [loading, setLoading] = useState(true);  
  
 useEffect(() => {  
 loadExpenses();  
 }, []);  
  
 async function loadExpenses() {  
 try {  
 const data = await expenseService.getByUserId(currentUser.id);  
 setExpenses(data);  
 } catch (error) {  
 console.error('Failed to load expenses:', error);  
 } finally {  
 setLoading(false);  
 }  
 }  
  
}

### Dodavanje, brisanje i/ili ažuriranje podataka

Dodavanje i ažuriranje troškova se vrši preko forme u ExpenseForm.jsx. Komponenta proverava da li se radi o dodavanju novog ili izmeni postojećeg troška na osnovu id parametra iz URL-a.

**ExpenseForm.jsx**

import React from 'react';  
import { useState, useEffect } from 'react';  
import { useNavigate, useParams } from 'react-router-dom';  
import { useAuth } from '../components/AuthContext';  
import { expenseService } from '../services/expenseService';  
import { categoryService } from '../services/categoryService';  
  
export default function ExpenseForm() {  
 const { id } = useParams();  
 const isEditing = !!id;  
   
 const { currentUser } = useAuth();  
 const navigate = useNavigate();  
  
 async function handleSubmit(e) {  
 e.preventDefault();  
  
 if (!validate()) return;  
  
 try {  
 setLoading(true);  
   
 const expenseData = {  
 userId: currentUser.id,  
 amount: parseFloat(formData.amount),  
 category: formData.category,  
 description: formData.description,  
 date: formData.date  
 };  
  
 if (isEditing) {  
 await expenseService.update(id, expenseData);  
 } else {  
 await expenseService.create(expenseData);  
 }  
  
 navigate('/expenses');  
 } catch (error) {  
 console.error('Failed to save expense:', error);  
 } finally {  
 setLoading(false);  
 }  
 }  
  
}

## 4.2. Administratorske funkcionalnosti

Administratori imaju pristup stranicama za upravljanje korisnicima (UserManagement.jsx) i kategorijama (CategoryManagement.jsx). Ove komponente omogućavaju pregled, dodavanje, izmenu i brisanje podataka.

**UserManagement.jsx**

import React from 'react'  
import { useState, useEffect } from 'react';  
import { useNavigate } from 'react-router-dom';  
import { userService } from '../services/userService';  
import { useAuth } from '../components/AuthContext';  
  
export default function UserManagement() {  
 const [users, setUsers] = useState([]);  
 const [loading, setLoading] = useState(true);  
 const { currentUser } = useAuth();  
 const navigate = useNavigate();  
  
 useEffect(() => {  
 loadUsers();  
 }, []);  
  
 async function loadUsers() {  
 // ... loads users  
 }  
  
 async function handleDelete(id) {  
 if (id === currentUser.id) {  
 alert("You cannot delete your own account!");  
 return;  
 }  
  
 if (window.confirm('Are you sure you want to delete this user?')) {  
 try {  
 await userService.delete(id);  
 setUsers(users.filter(user => user.id !== id));  
 } catch (error) {  
 console.error('Failed to delete user:', error);  
 }  
 }  
 }  
  
}

# 5. Simulacija backend-a

## 5.1. Korišćenje json-servera

Za potrebe ovog projekta, backend je simuliran pomoću json-server alata. On omogućava kreiranje RESTful API-ja na osnovu jednostavnog JSON fajla. db.json fajl definiše strukturu podataka, uključujući korisnike, troškove i kategorije.

**Primer db.json datoteke:**

{  
 "users": [  
 {  
 "id": "1",  
 "email": "admin@test.com",  
 "password": "admin123",  
 "name": "Admin",  
 "role": "admin"  
 }  
 ],  
 "expenses": [  
 {  
 "userId": "2",  
 "amount": 55,  
 "category": "Food",  
 "description": "Groceries from supermarket",  
 "date": "2025-09-20",  
 "id": "1"  
 }  
 ],  
 "categories": [  
 {  
 "id": "1",  
 "name": "Food",  
 "color": "#ef4444",  
 "budgetLimit": 500  
 }  
 ]  
}

## 5.2. Povezivanje frontenda i backend-a

Povezivanje sa json-server API-jem se vrši kroz servisne fajlove (authService.js, expenseService.js, itd.). Ovi servisi sadrže funkcije za slanje HTTP zahteva (GET, POST, PUT, DELETE) na odgovarajuće endpoint-e.

**Primer koda iz expenseService.js:**

const API\_URL = 'http://localhost:3000';  
  
export const expenseService = {  
 getByUserId: async (userId) => {  
 const response = await fetch(`${API\_URL}/expenses?userId=${userId}`);  
 const data = await response.json();  
 return data;  
 },  
  
 create: async (expense) => {  
 const response = await fetch(`${API\_URL}/expenses`, {  
 method: 'POST',  
 headers: {  
 'Content-Type': 'application/json'  
 },  
 body: JSON.stringify(expense)  
 });  
 const data = await response.json();  
 return data;  
 },  
  
 update: async (id, expense) => {  
 const response = await fetch(`${API\_URL}/expenses/${id}`, {  
 method: 'PUT',  
 headers: {  
 'Content-Type': 'application/json'  
 },  
 body: JSON.stringify(expense)  
 });  
 const data = await response.json();  
 return data;  
 },  
  
 delete: async (id) => {  
 await fetch(`${API\_URL}/expenses/${id}`, {  
 method: 'DELETE'  
 });  
 }  
};

# 6. Zaključak

## Pregled postignutih rezultata

Projekat je uspešno realizovan i sve planirane funkcionalnosti su implementirane. Kreirana je kompletna veb aplikacija za praćenje ličnih finansija koja omogućava korisnicima da efikasno upravljaju svojim troškovima. Aplikacija ima jasan i intuitivan korisnički interfejs, kao i odvojene uloge za korisnike i administratore.

## Izazovi i prepreke tokom rada na projektu

Najveći izazov tokom razvoja bio je efikasno upravljanje stanjem aplikacije, posebno u pogledu autentifikacije i deljenja podataka između komponenti. Ovaj problem je rešen korišćenjem React Context API-ja. Takođe, implementacija vizuelizacije podataka sa D3.js bibliotekom zahtevala je dodatno vreme za učenje i prilagođavanje.

## Mogućnosti za buduća unapređenja aplikacije

Aplikacija ima potencijal za dalji razvoj. Neke od mogućnosti za unapređenje su: \* Implementacija pravog backend-a (npr. Node.js sa Express.js i MongoDB bazom podataka). \* Dodavanje funkcionalnosti za praćenje prihoda. \* Kreiranje naprednijih izveštaja i filtera. \* Implementacija notifikacija za prekoračenje budžeta. \* Razvoj mobilne aplikacije.

# 7. Literatura

React Zvanična Dokumentacija - https://react.dev/

React Router Zvanična Dokumentacija - https://reactrouter.com/

Tailwind CSS Zvanična Dokumentacija - https://tailwindcss.com/docs/

D3.js Zvanična Dokumentacija - https://d3js.org/

json-server NPM Paket - https://www.npmjs.com/package/json-server

MDN Web Docs (JavaScript) - https://developer.mozilla.org/en-US/docs/Web/JavaScript