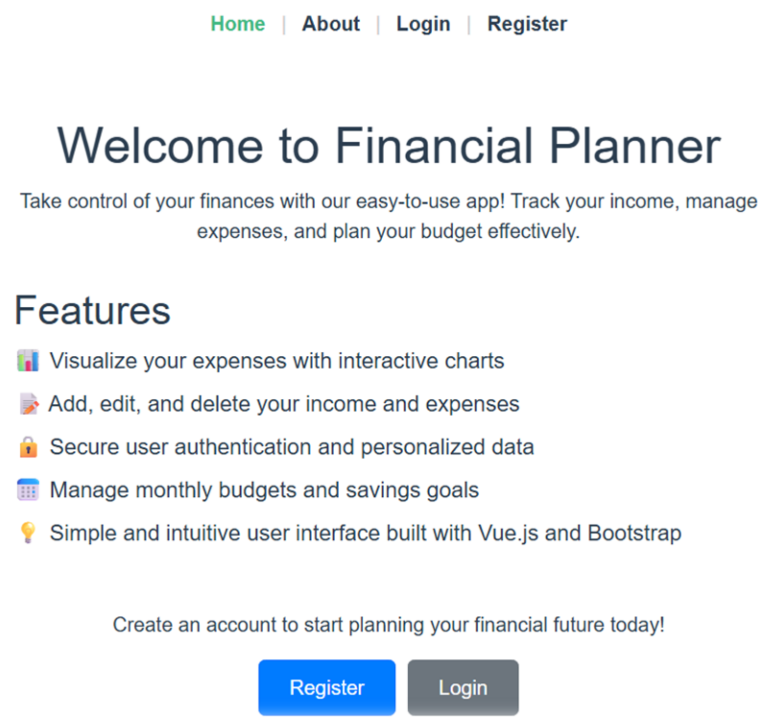
Leonora Siljanovska – 221115

Personal Financial Planner — Project Report

1. Introduction

In today's fast-paced world, managing personal finances effectively is essential for achieving financial stability and planning for the future. This project presents a ***Personal Financial Planner*** — a web application designed to help users track their income, expenses, and savings in a user-friendly environment.
The application leverages modern web development technologies, providing a responsive and intuitive interface where users can easily record expenses, visualize spending trends, and plan budgets monthly. This report details the project's motivation, architecture, technologies, design, implementation, and challenges faced during development.



Welcome to Financial Planner

Take control of your finances with our easy-to-use app! Track your income, manage expenses, and plan your budget effectively.

Features

📊 Visualize your expenses with interactive charts

📝 Add, edit, and delete your income and expenses

🔒 Secure user authentication and personalized data

📅 Manage monthly budgets and savings goals

💡 Simple and intuitive user interface built with Vue.js and Bootstrap

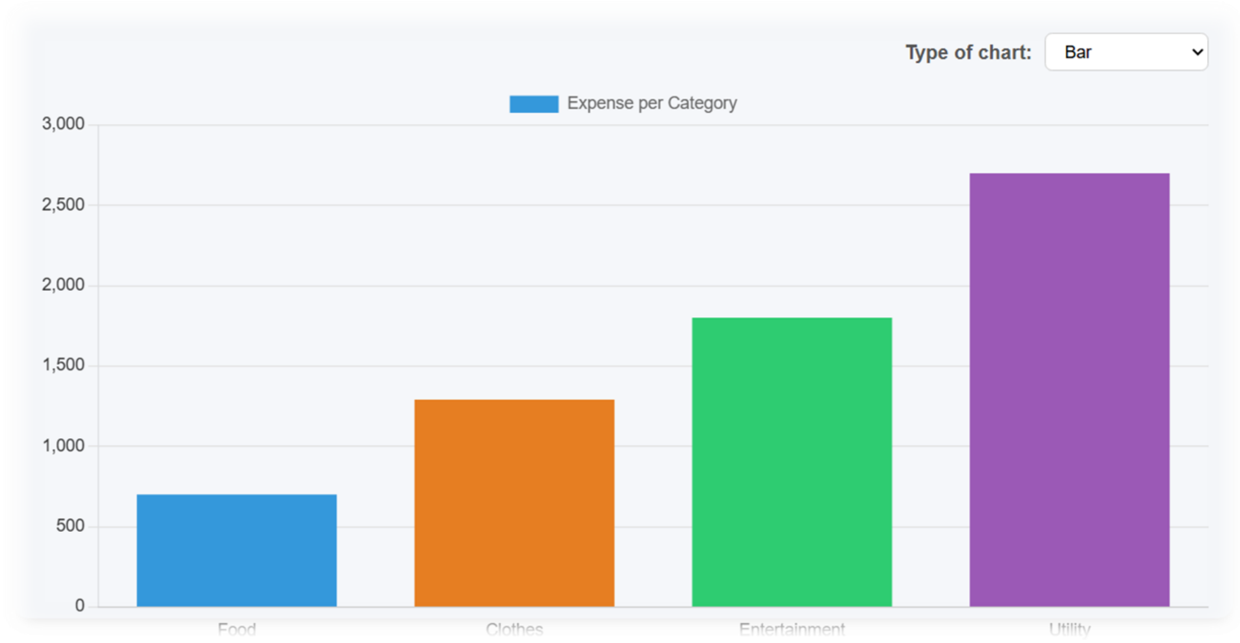
Create an account to start planning your financial future today!

2. Project Purpose and Objectives

The main objective of this project is to develop a digital tool that enables users to:

- Log income and categorize expenses (e.g., food, clothing, entertainment, utilities).
- Visualize financial data through interactive charts.
- Maintain monthly budgets and monitor savings.
- Provide secure user authentication for personalized data management.
- Export financial data for offline use.

By meeting these goals, the app helps users gain better control over their finances, identify spending patterns, and make informed financial decisions.



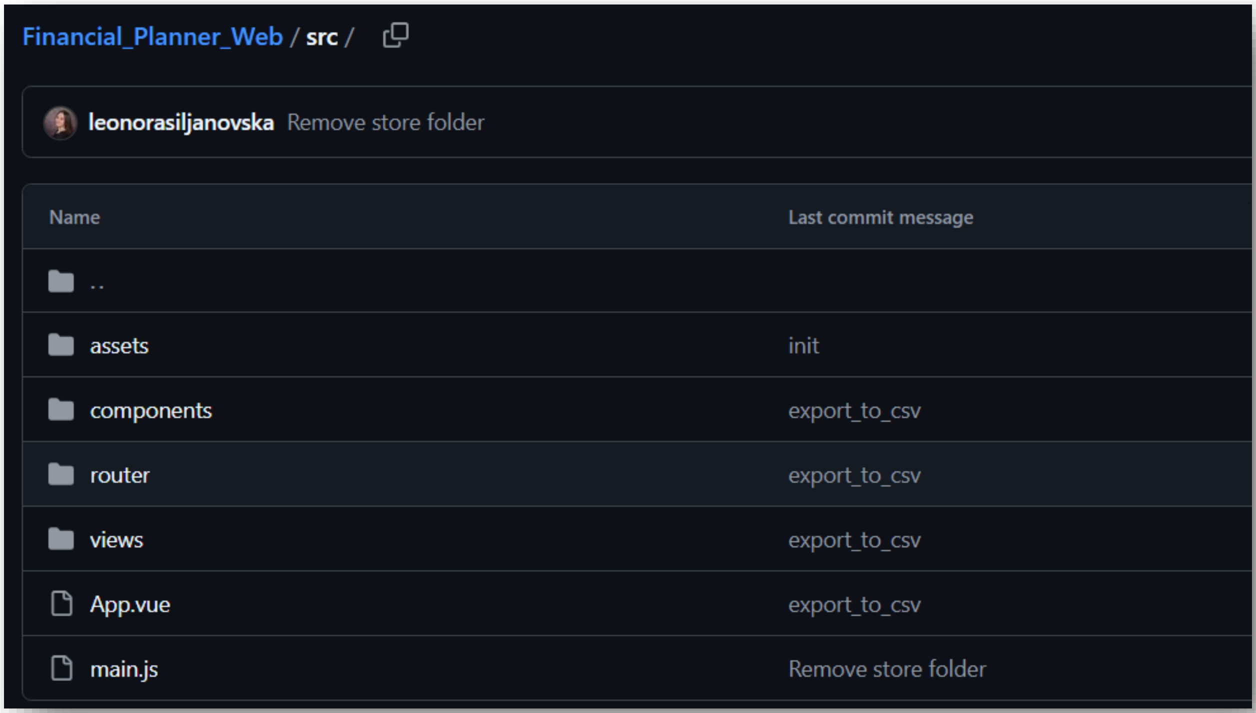
---

3. Technologies Used

The project is built as a Single Page Application (SPA) utilizing the following key technologies:

- Vue.js (v3): The JavaScript framework powering the frontend with component-based architecture.
- Vite: A fast build tool and development server that optimizes Vue project setup and development experience.
- CSS: For styling and responsive layouts ensuring usability on desktop and mobile devices.
- Vue Router: Handles client-side routing to manage navigation without full page reloads.
- Chart.js: For rendering dynamic charts that visualize financial data.
- Git and GitLab/GitHub: Version control and code repository hosting for collaboration and source code management.

The backend is simulated or minimal, focusing primarily on frontend user experience and local storage management, though the architecture allows easy extension to connect to APIs or databases.



Financial_Planner_Web / src /

leonorasiljanovska Remove store folder

Name	Last commit message
..	
assets	init
components	export_to_csv
router	export_to_csv
views	export_to_csv
App.vue	export_to_csv
main.js	Remove store folder

---

4. System Architecture

The application follows a modular, component-based architecture typical of modern SPAs. It separates concerns clearly between views, components, routing, and state management.

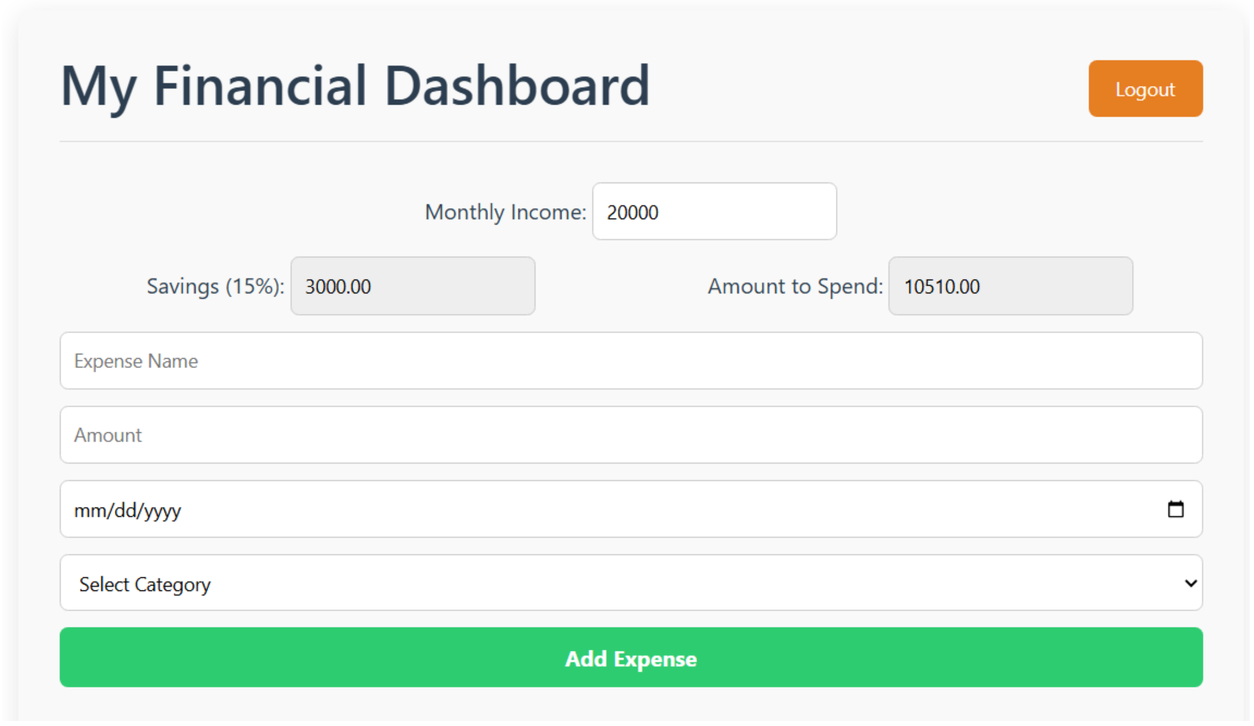
- Views serve as pages or main screens, such as DashboardView, ExpensesView, LoginView, and RegisterView.
- Components are reusable UI parts embedded inside views, including expense forms, charts, tables, and authentication forms.
- The Router controls navigation between views.

This separation ensures maintainability, scalability, and testability.

5. Design and Implementation

5.1 Components Overview

- **ExpenseChart.vue**:
  Utilizes Chart.js to graphically represent spending trends by categories and months. It offers visual insights to help users understand their financial habits.
- **ExpenseForm.vue**:
  A form interface for adding new expenses. It includes fields for selecting expense category, name, amount, and date. Form validation ensures data integrity.
- **ExpenseList.vue and ExpenseTable.vue**:
  Display recorded expenses either as a simple list or organized table. They support quick review and potential future editing/deleting features.
- **LoginForm.vue and RegisterForm.vue**:
  Handle user authentication workflows with input validation and feedback mechanisms, enabling secure access to personal financial data.
- **HelloWorld.vue**:
  Initial scaffold component, customizable or removable as needed.

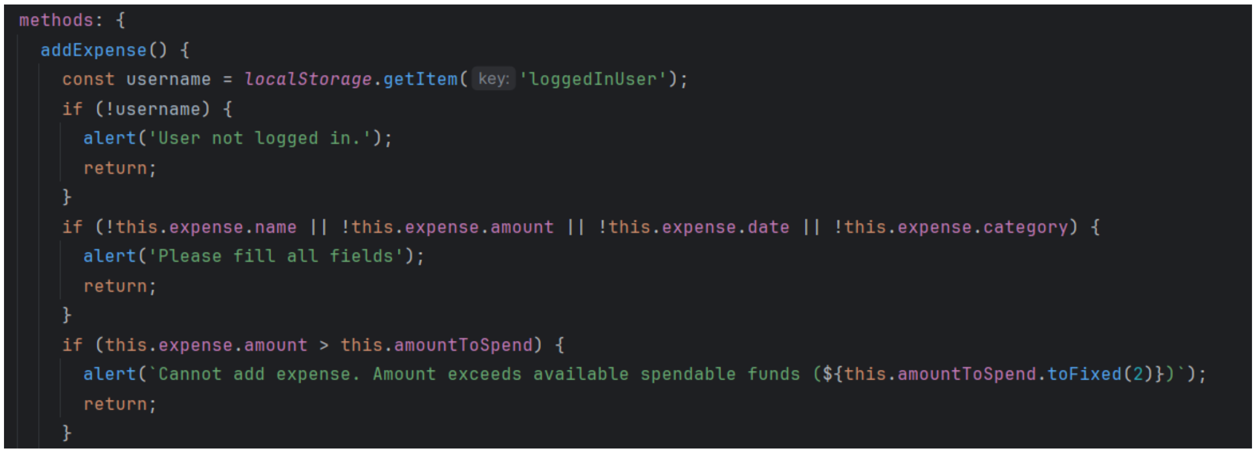

My Financial Dashboard

Monthly Income: 20000

Savings (15%): 3000.00

Amount to Spend: 10510.00

[ExpensesForm UI]

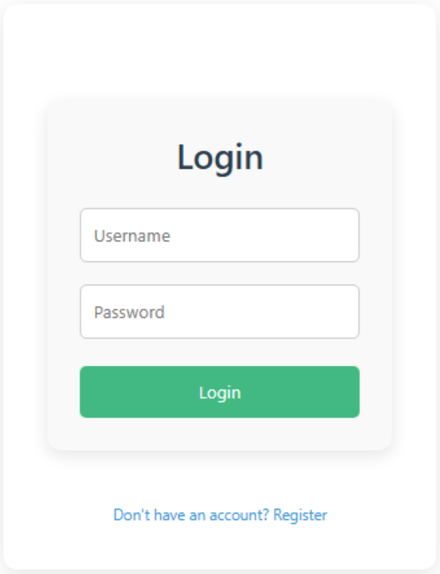


```
methods: {
  addExpense() {
    const username = localStorage.getItem( key: 'loggedInUser');
    if (!username) {
      alert('User not logged in.');
      return;
    }
    if (!this.expense.name || !this.expense.amount || !this.expense.date || !this.expense.category) {
      alert('Please fill all fields');
      return;
    }
    if (this.expense.amount > this.amountToSpend) {
      alert(`Cannot add expense. Amount exceeds available spendable funds (${this.amountToSpend.toFixed(2)})`);
      return;
    }
```

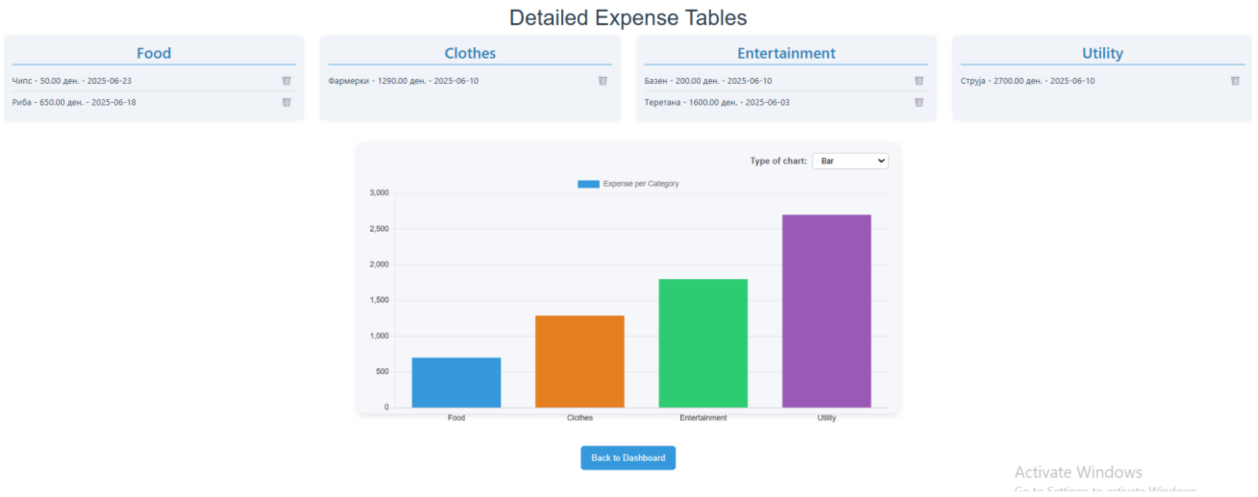
[form validation]

5.2 Views Overview

- **HomeView.vue**
  Serves as the application's landing page. It introduces users to the platform, outlines key features, and encourages new users to register or log in.
- **DashboardView.vue –** ***Summary focused***
  The main interface displayed after a successful login. It presents a financial summary including monthly income, categorized expenses, spendable amount, savings, and options to export data. Users can also navigate to detailed tables and charts from here.
- **ExpensesView.vue –** ***Detail focused***
  A detailed view for managing expenses. It includes functionality to add new expenses, view a breakdown of all entries, visualize expenses by category through charts, and delete specific entries if needed.
- **LoginView.vue & RegisterView.vue**
  Interfaces dedicated to user authentication. These views handle secure login and registration workflows and ensure that users are properly directed to their dashboard upon successful authentication.



[Login View]



Detailed Expense Tables

Food
- Чипс - 50.00 ден. - 2025-06-23
- Риба - 650.00 ден. - 2025-06-18

Clothes
- Фармерки - 1290.00 ден. - 2025-06-10

Entertainment
- Базен - 200.00 ден. - 2025-06-10
- Теретана - 1600.00 ден. - 2025-06-03

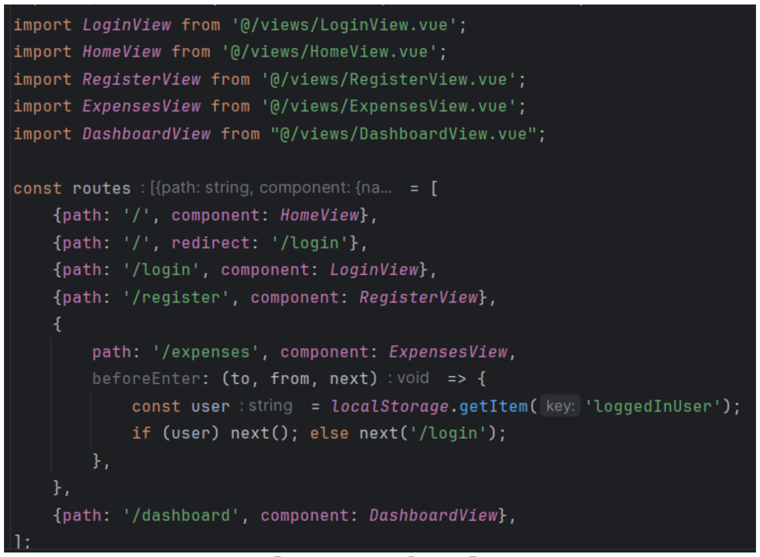
Utility
- Струја - 2700.00 ден. - 2025-06-10

Type of chart: Bar

[Expenses View]

5.3 Routing and State Management

The app employs Vue Router to manage navigation between pages smoothly without reloads, improving user experience.



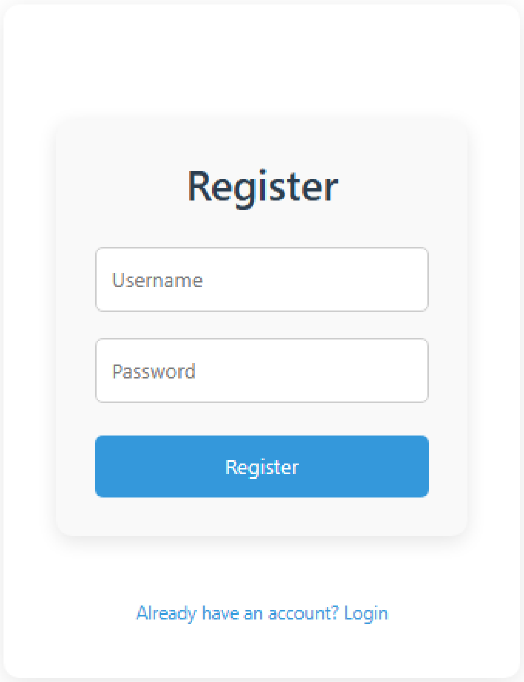
```
import LoginView from '@/views/LoginView.vue';
import HomeView from '@/views/HomeView.vue';
import RegisterView from '@/views/RegisterView.vue';
import ExpensesView from '@/views/ExpensesView.vue';
import DashboardView from "@/views/DashboardView.vue";

const routes : [{path: string, component: {na...  = [
    {path: '/', component: HomeView},
    {path: '/', redirect: '/login'},
    {path: '/login', component: LoginView},
    {path: '/register', component: RegisterView},
    {
        path: '/expenses', component: ExpensesView,
        beforeEnter: (to, from, next) : void  => {
            const user : string  = localStorage.getItem( key: 'loggedInUser');
            if (user) next(); else next('/login');
        },
    },
    {path: '/dashboard', component: DashboardView},
];
```

[router / index.js]

5.4 User Authentication

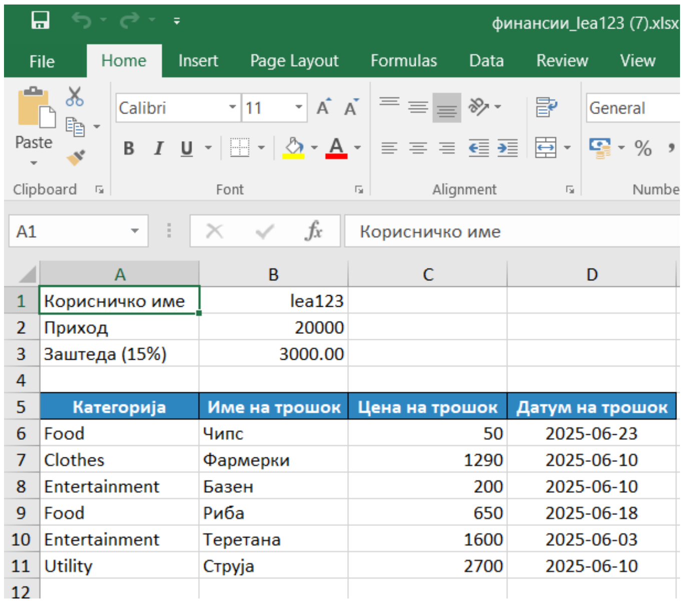
Users can register and log in securely through dedicated forms. Although backend integration is minimal or simulated, the architecture supports token-based authentication or API integration for future scalability.



[Register View]

5.5 Data Persistence

Financial data is stored locally using browser storage (localStorage) for immediate usability without backend dependencies. Export functionality allows users to save their data externally for backup or offline review.



	A	B	C	D
1	Корисничко име	lea123		
2	Приход	20000		
3	Заштеда (15%)	3000.00		
4				
5	Категорија	Име на трошок	Цена на трошок	Датум на трошок
6	Food	Чипс	50	2025-06-23
7	Clothes	Фармерки	1290	2025-06-10
8	Entertainment	Базен	200	2025-06-10
9	Food	Риба	650	2025-06-18
10	Entertainment	Теретана	1600	2025-06-03
11	Utility	Струја	2700	2025-06-10
12				

[Exported data to .xlsx]

6. Conclusion

The Personal Financial Planner project successfully delivers a practical and user-friendly tool to help individuals manage their finances. Leveraging Vue.js and modern web technologies, the app provides features essential for tracking income and expenses, visualizing spending habits, and managing budgets effectively.
This project lays a solid foundation for future enhancements, such as backend integration for multi-device synchronization, advanced reporting, and user collaboration.

---

7. Future Work

Since this course is primarily focused on frontend development, data was stored locally using localStorage to meet the requirements without introducing backend complexity. However, for a more complete and scalable application, several improvements can be made in the future:

- **Integrating a Database:**
  Moving from localStorage to a backend database (e.g., Firebase, MongoDB, or SQL) would allow for persistent data storage, multi-user support, and improved data reliability.
- **Adding Backend Functionality:**
  Implementing an API for managing users, expenses, income, and settings would enhance security and allow for role-based access and user authentication.
- **More Detailed Reporting:**
  Currently, the app provides basic summaries. Future improvements could include generating detailed monthly, quarterly, and yearly reports with trends, charts, and downloadable summaries.
- **Advanced Export Options:**
  In addition to Excel, offering exports in PDF and CSV formats with customizable filters would be beneficial for users who want professional or printable reports.

---

8. References

- Vue.js Documentation — https://vuejs.org
- Vite — https://vitejs.dev
- Chart.js Documentation — https://www.chartjs.org
- Git and GitLab Guides — https://docs.gitlab.com