Software Requirements Specification (SRS)

Project Title: Expense Manager

1. Introduction

The **Expense Manager** is a web-based application designed to help users track, manage, and analyze their personal expenses in an organized way. It provides features like daily and monthly expense tracking, budget alerts, saving goals, and a separate analysis tab for data visualization.The system aims to reduce manual effort in managing expenses and give users clear insights into their financial habits. By using modern web technologies, it ensures a secure, user-friendly, and scalable solution.

2. Purpose

The purpose of this document is to define the requirements of the Expense Manager application. It will guide developers, testers, and stakeholders during the project. The document ensures a clear understanding of system features, behavior, and constraints.

3. Scope

- Users can record and view daily and monthly expenses.
- Budget alerts will notify users when spending exceeds limits.
- A savings tracker will help users monitor their goals.
- An analysis tab will provide visual reports and insights.
- The system will support secure login and user accounts.

4. Definitions, Acronyms, and Abbreviations

User – A person who uses the application to record and track expenses.

Auth – Authentication process for secure login and account access.

UI – User Interface, the design and layout of the application.

DB – Database, where all expense and user information is stored.

5. Overall Description

5.1 Users and Characteristics

User Type	Description
User	Can register, log in, and manage their expenses.
Admin	(Optional) Can view reports or manage system settings.

5.2 Assumptions and Dependencies

- The system will be accessed through modern web browsers.
- Internet connection is required to sync data.
- Supabase will handle database and authentication.
- Email or SMS services may be used for alerts.

6. Functional Requirements

6.1 User Registration/Login

- Users can create an account using email and password.
- Login and password reset options will be provided.

6.2 Add and Manage Expenses

- Users can add, edit, or delete expenses.
- Each expense will include amount, category, date, and notes.

6.3 Budget and Savings Goals

- Users can set monthly budgets and savings targets.
- Alerts will appear when spending exceeds the limit.

6.4 Expense Analysis Tab

- Users can analyze data through charts and visual reports.
- Filters will allow viewing by category, date, or amount range.

6.5 Reports and History

- Monthly and yearly summaries will be generated automatically.
- Reports can be downloaded or viewed in a simple dashboard.

7. Non-Functional Requirements

7.1 Performance

- The system should load and display data within 3 seconds.

7.2 Security

- All user data will be stored securely in the database.
- Authentication and authorization will protect accounts.

7.3 Usability

- The interface will be clean, easy to use, and responsive.

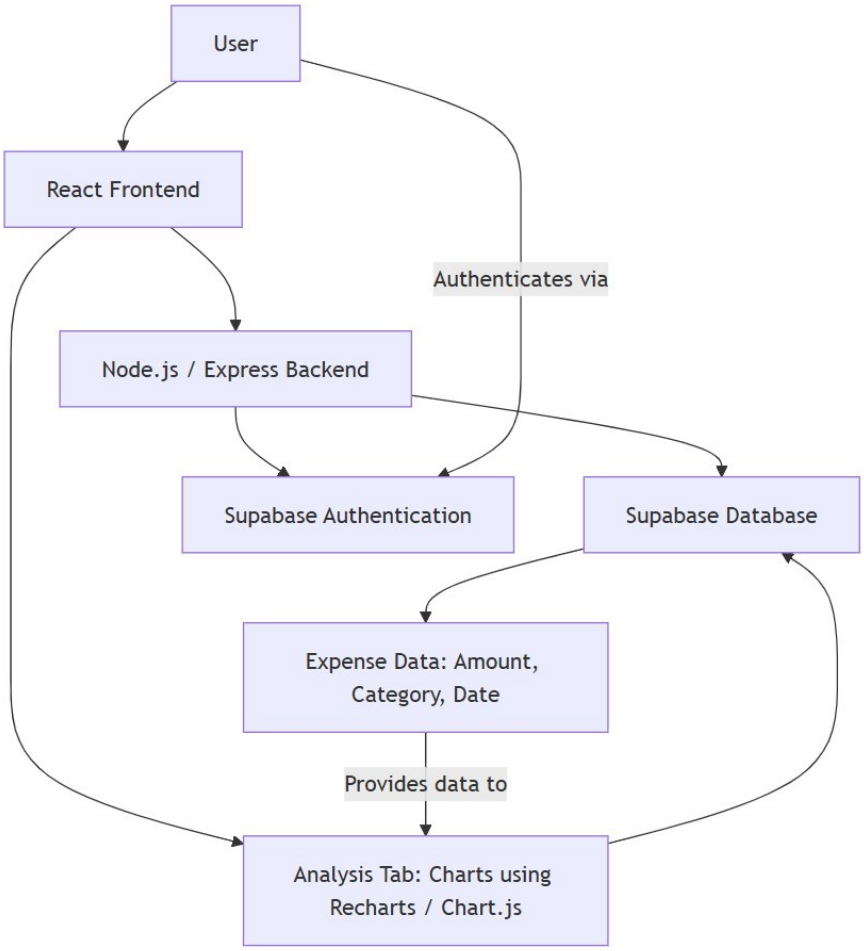
7.4 Scalability

- The system should handle many users and growing data smoothly.

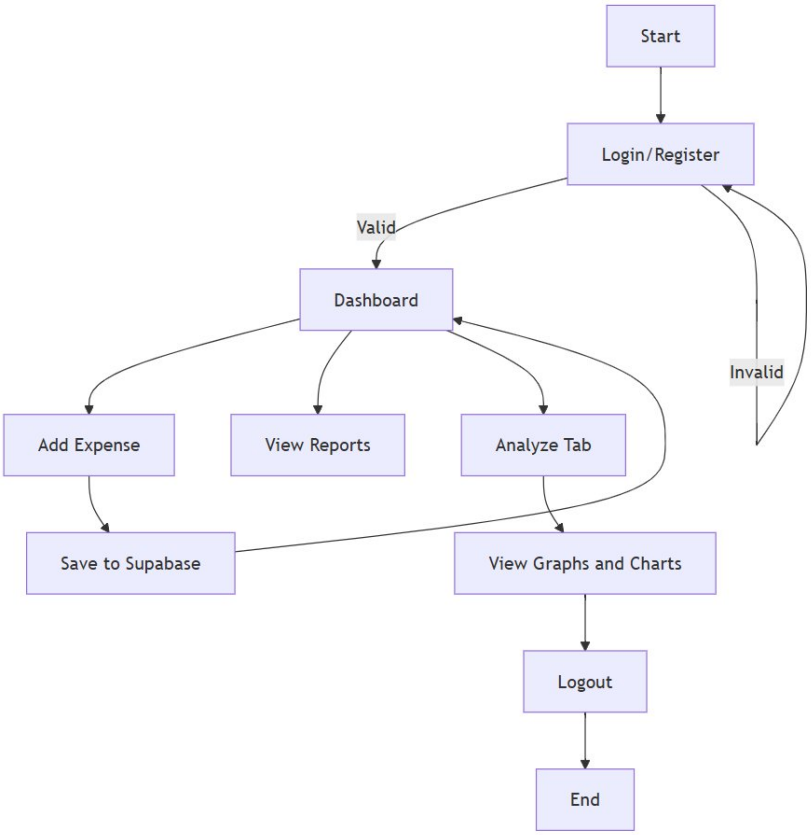
8. System Architecture

- **Frontend:** React for building interactive user interfaces.
- **Backend:** Node.js with Express for API handling.
- **Database:** Supabase (PostgreSQL-based).
- **Authentication:** Supabase Auth for secure login.
- **Hosting:** Deployed on cloud platforms like Vercel or Render.

9. System Flow Diagram



10.User Flow Diagram



11. Constraints

- Requires internet connection for full functionality.
- Must comply with data privacy and security standards.
- Limited to supported browsers (Chrome, Edge, Firefox).

12. Future Enhancements

- Integration with AI-based spending suggestions.
- Mobile app version for Android and iOS.
- Support for shared or family accounts.
- Automatic bill tracking through bank SMS or email.

13. Feasible Solution

Technical Approach

- Frontend: React for dynamic and responsive UI.
- Backend: Express.js for API logic and routes.
- Database: Supabase for data storage and authentication.
- Data Visualization: Chart.js or Recharts for analysis tab.

Implementation Plan

1. Requirement gathering and planning.
2. UI/UX design and database schema creation.
3. Develop core modules – authentication, expense tracking, alerts.
4. Add analysis and reporting features.
5. Testing, bug fixing, and deployment.

Tools and Technologies

- Frontend: React, HTML, CSS, JavaScript.
- Backend: Node.js, Express.js.
- Database: Supabase.
- Visualization: Chart.js / Recharts.
- Version Control: GitHub.
- Hosting: Vercel / Render.
