Software Requirements Specification (SRS) for PocketPulse Expense Tracker

Project By:-
Siddharth Pillai
Sparsh Jain

Mentored By:-
Dr. Shaurya Gupta

Index

Software Requirements Specification (SRS) for Pocket Pulse Expense Tracker

1. Introduction

1.1 Purpose

This document specifies the functional and non-functional requirements for a modern, AI-ready expense tracker built using a full-stack JavaScript framework. The goal is to help users manage personal or small business finances effectively by logging, categorizing, visualizing, and receiving insights on their spending habits.

1.2 Scope

The application will:

- Allow users to record income and expenses
- Enable categorization and tagging of transactions
- Provide visual insights into spending
- Notify users of approaching budget limits via **email alerts**
- Provide scheduled reports and anomaly detection (future scope)

1.3 Intended Audience

- Developers and contributors to the project
- Project stakeholders
- End-users and testers
- Technical auditors or security reviewers

1.4 Definitions, Acronyms, and Abbreviations

- **SRS**: Software Requirements Specification
- **UI**: User Interface

- **API**: Application Programming Interface
- **JWT**: JSON Web Token
- **ORM**: Object Relational Mapper

2. Overall Description

2.1 Product Perspective

This is a standalone web-based application, hosted via Vercel, and powered by Supabase as the backend. It integrates Inngest for job scheduling (email alerts, monthly summaries) and ArcJet for API-level security.

2.2 Product Features

- Transaction Logging and Categorization
- Dashboard for Visualization
- Budget Limit Setting and Monitoring
- Scheduled Reports via Email (monthly/weekly)
- Instant Email Alerts on Budget Breach (via Inngest)
- Secure Authentication
- Responsive UI across devices
- Activity Logging (Admin View)

2.3 User Characteristics

Users are assumed to be familiar with basic web applications and financial management. No technical expertise is required.

2.4 Constraints

- Real-time features depend on Supabase's tier limits
- Email alerting depends on integration with SMTP or services like Resend/Mailgun

- AI recommendations require future ML model integration

3. Specific Requirements

3.1 Functional Requirements

3.1.1 User Authentication

- Sign up, login, and logout using Supabase Auth
- JWT-based session management

3.1.2 Transaction Management

- Create, update, and delete transactions
- Fields: date, amount, type, category, optional tags
- Option to attach receipts (Supabase Storage)

3.1.3 Dashboard

- View expenses over time via graphs (daily, weekly, monthly)
- Breakdown by categories (e.g., Rent, Food, Shopping)

3.1.4 Budget Management

- Set monthly budget caps for each category
- Real-time budget utilization tracking

3.1.5 Email Alerts & Reports

- Automated monthly reports sent via email (Inngest + SMTP)
- Alert email when a budget threshold is crossed (e.g., 80%)
- Daily summary option for high-spenders
- Transaction summaries in PDF format (future scope)

3.1.6 Background Job Scheduling

- Inngest schedules report generation and email alert dispatch

- Retry failed jobs and log email delivery success/failure

3.1.7 API Protection

- ArcJet to enforce rate limits and block suspicious activity
- API usage analytics dashboard (admin)

3.1.8 Responsive and Accessible UI

- Built using Tailwind CSS and Shadcn UI
- Light/dark mode support
- Keyboard and screen reader friendly

3.2 Non-Functional Requirements

3.2.1 Performance

- Page load time < 1.5s on average
- Background jobs complete under 5s per task

3.2.2 Scalability

- Built to handle 10,000+ users
- Database indexed for fast querying with Prisma

3.2.3 Security

- Supabase Auth for secure login
- API endpoints protected with ArcJet
- Input validation and sanitization for all user fields

3.2.4 Usability

- Intuitive layout
- Tooltips and onboarding for new users
- Mobile-first responsive design

3.2.5 Maintainability

- Modular file structure in Next.js App Router
- Reusable components with Shadcn UI
- Code linted and tested with unit and E2E tests

4. External Interface Requirements

4.1 User Interfaces

- **Login/Register Screens**
- **Dashboard**: Line charts, bar graphs, pie charts
- **Add/Edit Transaction Forms**
- **Email Preferences Page**: Toggle alerts and reports

4.2 Hardware Interfaces

- Not applicable (web app)

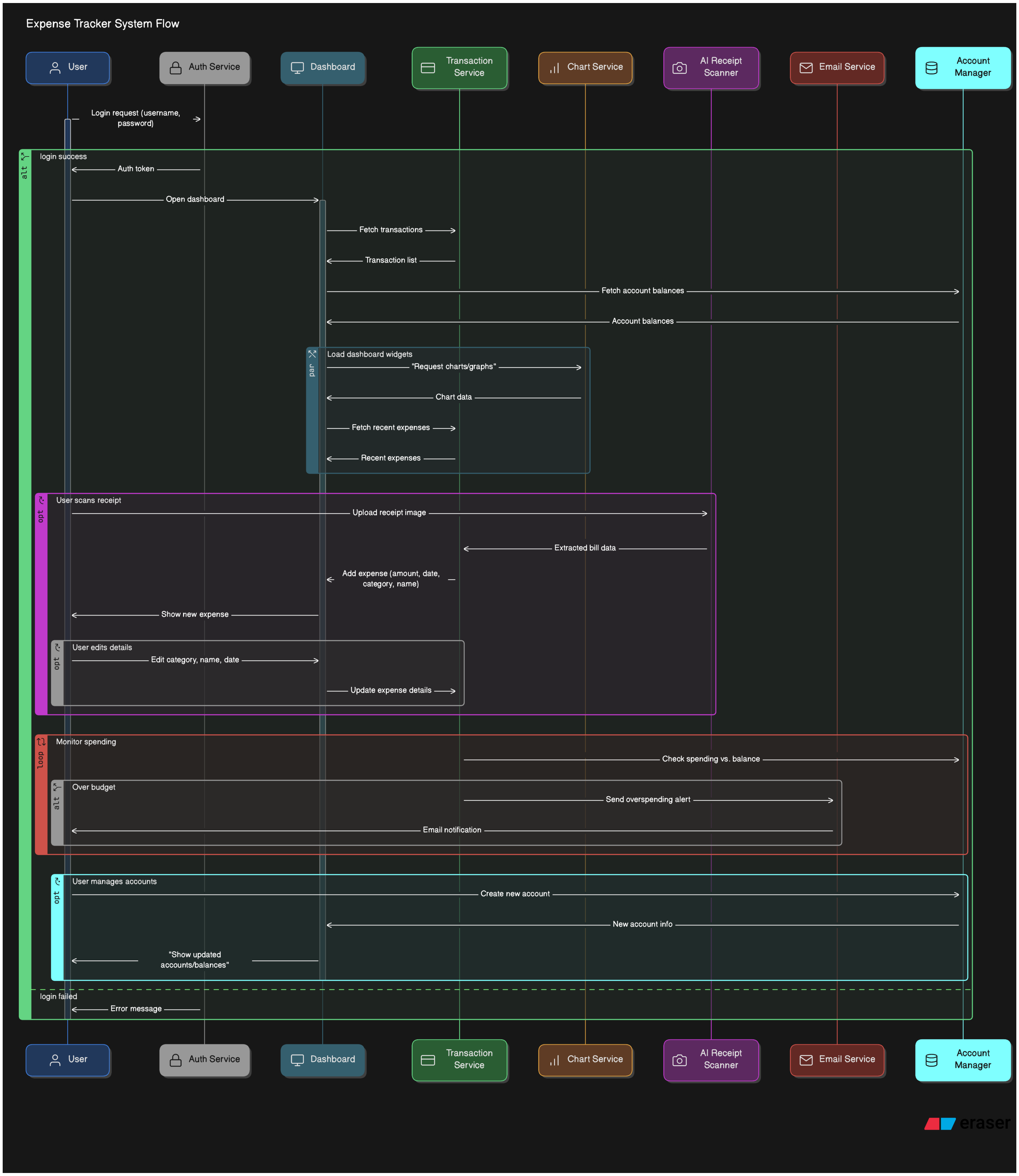
4.3 Software Interfaces

- **Supabase Auth + DB**
- **Inngest API for workflows**
- **SMTP/Mail Provider API for emails**
- **ArcJet API for security layer**

4.4 Communications Interfaces

- HTTPS for secure data transmission
- WebSocket (optional future use) for real-time updates

5. System Architecture Overview



6. Future Enhancements

- AI model to suggest budget changes and detect anomalies
- Integration with bank APIs (Plaid, Salt Edge)
- Multi-user shared wallets
- Export reports to Excel/PDF
- Voice-command input for logging expenses

7. Conclusion

The AI-Powered Expense Tracker aims to provide users with a seamless and intelligent way to manage their finances. By leveraging modern technologies such as **Next.js, Supabase, Prisma, Inngest, ArcJet, Tailwind CSS**, and **Shadcn UI**, the platform offers a secure, scalable, and user-friendly experience. Key features like real-time tracking, budget insights, and automated **email alerts** make the application not only practical but also proactive in helping users develop better financial habits.