Software Development Project Proposal

Personal Finance Management Application

Prepared by: John Doe (Project Manager)

Team Members:

John Doe, Jane Smith, Alex Johnson, Sarah Lee

Course: Software Development Project

Submission Date: July 08, 2025

Submitted to: [Instructor’s Name]

[Institution/Department Name]

1 Team Members

- John Doe (Project Manager)
- Jane Smith (Lead Developer)
- Alex Johnson (UI/UX Designer)
- Sarah Lee (QA Engineer)

2 Project Overview

2.1 Objective

Develop a user-friendly web-based application to help individuals track income, expenses, and savings goals, promoting better financial decision-making.

2.2 Problem Statement

Many individuals struggle to manage their finances due to lack of accessible, intuitive tools that provide real-time insights into their spending habits and financial goals.

2.3 Scope

The project will deliver a web application with core features for budget tracking, expense categorization, and savings goal visualization. It will support single-user accounts with secure data storage.

2.4 Technologies

- **Frontend**: React.js, Tailwind CSS
- **Backend**: Node.js, Express.js
- **Database**: MongoDB
- **Tools**: Git for version control, Figma for design prototyping

3 Related Projects

1. **Mint**: A popular personal finance tool offering budget tracking, bill reminders, and credit score monitoring. Our project will focus on a simpler interface and local data storage for privacy.
2. **YNAB (You Need A Budget)**: Emphasizes zero-based budgeting. Our project will adopt a similar goal-oriented approach but target users seeking a free, open-source alternative.
3. **PocketGuard**: Focuses on real-time spending limits. Our project will incorporate similar real-time□□ but with customizable categories for flexibility.

4 Tentative Features

1. **User Authentication**: Secure login and registration system to protect user data.

2. **Expense Tracking**: Add, edit, and categorize expenses (e.g., groceries, utilities).
3. **Budget Planning**: Set monthly budgets for different categories with alerts for overspending.
4. **Savings Goals**: Create and track savings goals with visual progress bars.
5. **Dashboard**: Display summary of income, expenses, and savings in charts.
6. **Data Export**: Export financial data as CSV for external use.
7. **Future Enhancements (Post-MVP)**:
    - Mobile app integration
    - Automated transaction imports from bank accounts
    - Multi-user support for shared budgets

5 System Design

5.1 Requirement Gathering

Requirements will be collected through:

- **User Surveys**: Conduct surveys targeting college students and young professionals to identify key financial management needs.
- **Stakeholder Interviews**: Engage with potential users to understand pain points in existing tools.
- **Competitive Analysis**: Review features of tools like Mint and YNAB to identify gaps and opportunities.

Key functional requirements include secure user authentication, real-time expense tracking, and customizable budget categories. Non-functional requirements include scalability, responsiveness, and data privacy.

5.2 Feasibility Analysis

- **Technical Feasibility**: The chosen technologies (React.js, Node.js, MongoDB) are well-documented, open-source, and within the team's skill set. Hardware requirements are minimal, as the application will be cloud-hosted.
- **Economic Feasibility**: Development costs are low due to open-source tools and free hosting platforms (e.g., Heroku for MVP). No external funding is required.
- **Operational Feasibility**: The application aligns with user needs for simple financial management and can be maintained with minimal resources post-deployment.
- **Schedule Feasibility**: The 12-week timeline is sufficient for an MVP, with tasks distributed evenly across team members.

5.3 System Architecture

The application will follow a client-server architecture:

- **Client Layer**: React.js frontend with Tailwind CSS for responsive UI, handling user interactions and data visualization.

- **Server Layer**: Node.js with Express.js for API endpoints, managing business logic and user requests.
- **Data Layer**: MongoDB for storing user profiles, transactions, and budget data, with RESTful API integration.

5.4 Data Flow

- User inputs (e.g., expenses, budgets) are captured via the frontend and sent to the backend via API calls.
- The backend validates and processes requests, storing data in MongoDB.
- Data retrieval requests (e.g., dashboard charts) fetch data from MongoDB, processed by the backend, and displayed on the frontend.

6 Project Timeline (Gantt Chart)

The project will span 12 weeks, with the proposal due in Week 4. Below is a Gantt chart outlining key tasks and milestones.

Task	**W1**	**W2**	**W3**	**W4**	**W5**	**W6**	**W7**	**W8**	**W9**	**W10**	**W11**	**W**
Requirement Analysis	X	X										
Research & Related Projects	X	X										
Proposal Drafting		X	X									
Proposal Submission				X								
System Design					X	X						
Frontend Development						X	X	X				
Backend Development						X	X	X				
Database Setup							X	X				
Integration & Testing								X	X	X		
User Testing & Feedback									X	X		
Final Adjustments										X	X	
Project Submission											X	

Note: X represents task duration in a given week.

7 Deliverables

- **Week 4**: Project proposal document (this document)
- **Week 8**: Minimum Viable Product (MVP) with core features (authentication, expense tracking, budget planning)
- **Week 12**: Final application with all features, documentation, and user manual