REQUIREMENT SPECIFICATION DOCUMENT

Personal Finance Management & Investment Tracker System

Project Title: Personal Finance Management & Investment Tracker System

Prepared By: B.Sc. (Computer Science) Internship 2025

Submission Date: January 8, 2026

Version: 1.0

Table of Contents

---

1. Introduction

1.1 Purpose

This Requirement Specification Document (RSD) provides a comprehensive description of the Personal Finance Management & Investment Tracker System. It outlines the functional and non-functional requirements, system architecture, database design, and implementation details necessary for the development, testing, and deployment of this web application.

1.2 Project Overview

The Personal Finance Management & Investment Tracker System is a web-based application designed to help individuals manage their personal finances effectively. In today's fast-paced world, financial management has become increasingly important, yet many people struggle to track their income, expenses, savings, and investments in an organized manner.

This application addresses these challenges by providing a unified platform that enables users to:

- Track income and expenses with detailed categorization
- Set and monitor monthly budgets
- Create and track savings goals
- Monitor investment portfolios
- Receive bill payment reminders
- Generate financial reports and analytics
- Use financial calculators for planning

1.3 Document Conventions

Term	Description
RSD	Requirement Specification Document
UI	User Interface
UX	User Experience
CRUD	Create, Read, Update, Delete
API	Application Programming Interface
SPA	Single Page Application
EMI	Equated Monthly Installment
SIP	Systematic Investment Plan

1.4 Intended Audience

This document is intended for:

- Project Supervisors and Evaluators
- Development Team Members
- Quality Assurance Personnel
- Future Maintainers of the System

---

2. Project Objectives

2.1 Primary Objectives

1. **Financial Tracking:** Develop a comprehensive system to track all income sources and expenses with proper categorization and tagging.
2. **Budget Management:** Enable users to set monthly budgets for different expense categories and monitor their spending against these limits.
3. **Savings Goals:** Allow users to create financial goals with target amounts and deadlines, tracking progress over time.
4. **Investment Monitoring:** Provide portfolio tracking capabilities with profit/loss calculations and visual representations.
5. **Bill Management:** Implement a reminder system for recurring bills to prevent missed payments.
6. **Financial Analytics:** Generate visual reports and analytics to help users understand their financial patterns.

7. **Financial Calculators:** Provide tools for EMI, SIP, and compound interest calculations.

2.2 Secondary Objectives

1. Ensure a responsive design that works seamlessly across desktop, tablet, and mobile devices.
2. Implement dark/light theme support for better user experience.
3. Provide data export/import functionality for backup purposes.
4. Maintain clean, well-documented, and maintainable code.
5. Ensure data persistence using browser localStorage.

---

3. Scope of the Project

3.1 In Scope

Feature	Description
User Authentication	Login, Registration, Profile Management
Dashboard	Financial overview with charts and summaries
Income Management	Add, edit, delete income entries
Expense Management	Track and categorize expenses
Budget Tracking	Set limits and monitor spending by category
Savings Goals	Create and track progress toward goals
Investment Tracker	Monitor portfolio with P/L calculations
Bill Reminders	Track upcoming and overdue bills
Transaction History	View all transactions with filters
Reports & Analytics	Charts and financial health score
Calculators	EMI, SIP, Compound Interest
Settings	Currency, date format, theme preferences
Data Management	Export/Import JSON backup

3.2 Out of Scope

Feature	Reason
Backend Server	Project uses localStorage for simplicity
Real-time Stock Prices	Would require external API integration
Multi-user Collaboration	Designed for individual use
Mobile Native Apps	Web-only application
Payment Gateway	Not a transactional system

Bank Integration	Security and complexity concerns

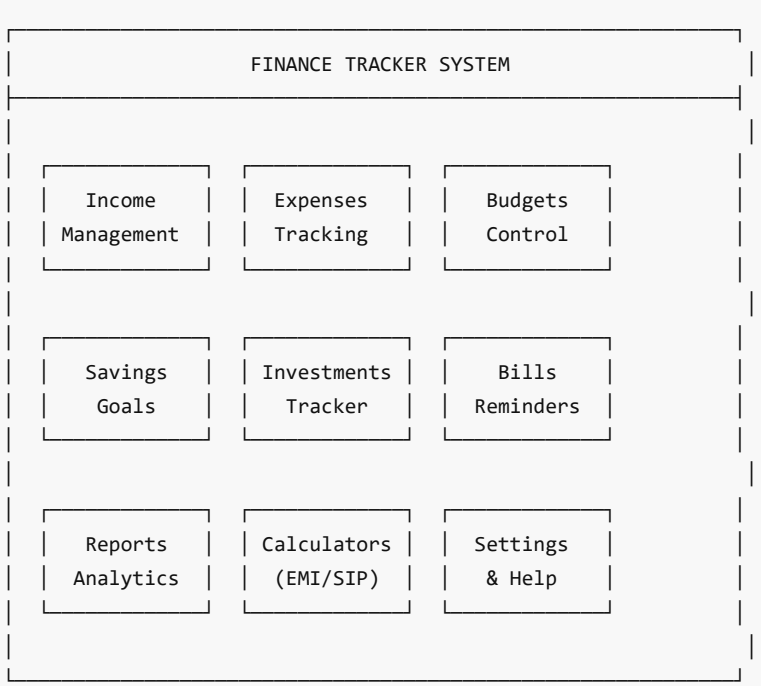
---

4. System Overview

4.1 System Description

The Personal Finance Management & Investment Tracker System is a Single Page Application (SPA) built using modern web technologies. It operates entirely in the browser, storing all user data locally using the browser's localStorage API.

4.2 Key Features Summary



4.3 User Roles

Role	Description	Permissions
Guest User	Unauthenticated visitor	Access login/register only
Registered User	Authenticated user	Full access to all features

---

5. Functional Requirements

5.1 User Authentication Module

Req ID	Requirement	Priority
FR-1.1	System shall allow new users to register with name, email, and password	High
FR-1.2	System shall validate email format during registration	High

FR-1.3	System shall enforce minimum password length of 6 characters	High
FR-1.4	System shall allow registered users to login with email and password	High
FR-1.5	System shall redirect authenticated users to dashboard	High
FR-1.6	System shall allow users to logout and clear session	High
FR-1.7	System shall persist login state across browser sessions	Medium
FR-1.8	System shall allow users to update their profile information	Medium
FR-1.9	System shall allow users to change their password	Medium

5.2 Dashboard Module

Req ID	Requirement	Priority
FR-2.1	System shall display total balance (income - expenses)	High
FR-2.2	System shall display current month's income total	High
FR-2.3	System shall display current month's expense total	High
FR-2.4	System shall display total savings across all goals	High
FR-2.5	System shall show expense distribution as a pie chart	High
FR-2.6	System shall display recent transactions list	Medium
FR-2.7	System shall show budget overview with progress bars	Medium
FR-2.8	System shall provide quick action buttons for common tasks	Low

5.3 Income Management Module

Req ID	Requirement	Priority
FR-3.1	System shall allow users to add new income entries	High
FR-3.2	System shall capture income source name, amount, category, and date	High
FR-3.3	System shall allow users to edit existing income entries	High
FR-3.4	System shall allow users to delete income entries	High
FR-3.5	System shall display list of all income entries	High
FR-3.6	System shall allow filtering income by category	Medium
FR-3.7	System shall allow searching income entries	Medium
FR-3.8	System shall mark income as recurring if applicable	Low

5.4 Expense Management Module

Req ID	Requirement	Priority

FR-4.1	System shall allow users to add new expense entries	High
FR-4.2	System shall capture expense name, amount, category, date, and payment method	High
FR-4.3	System shall allow users to edit existing expense entries	High
FR-4.4	System shall allow users to delete expense entries	High
FR-4.5	System shall display list of all expense entries	High
FR-4.6	System shall allow filtering by category and payment method	Medium
FR-4.7	System shall allow sorting by date or amount	Medium

5.5 Budget Management Module

Req ID	Requirement	Priority
FR-5.1	System shall allow users to set budget limits per category	High
FR-5.2	System shall calculate actual spending per category	High
FR-5.3	System shall display progress bars showing budget utilization	High
FR-5.4	System shall show visual alerts when budget exceeds 70%, 90%, 100%	High
FR-5.5	System shall allow editing budget limits	Medium
FR-5.6	System shall allow deleting budgets	Medium
FR-5.7	System shall prevent duplicate budgets for same category	Medium

5.6 Savings Goals Module

Req ID	Requirement	Priority
FR-6.1	System shall allow creating savings goals with name, target, and deadline	High
FR-6.2	System shall track current amount saved toward each goal	High
FR-6.3	System shall display progress percentage	High
FR-6.4	System shall allow adding money to goals	High
FR-6.5	System shall support priority levels (High, Medium, Low)	Medium
FR-6.6	System shall show days remaining until deadline	Medium
FR-6.7	System shall indicate completed goals with celebration	Low

5.7 Investment Tracker Module

Req ID	Requirement	Priority
FR-7.1	System shall allow adding investments with type, quantity, and prices	High
FR-7.2	System shall calculate profit/loss for each investment	High

FR-7.3	System shall display overall portfolio value	High
FR-7.4	System shall show percentage returns	High
FR-7.5	System shall display portfolio distribution chart	Medium
FR-7.6	System shall support multiple investment types (Stocks, Mutual Funds, Crypto, etc.)	Medium

5.8 Bill Reminders Module

Req ID	Requirement	Priority
FR-8.1	System shall allow adding bill reminders with due date and amount	High
FR-8.2	System shall display upcoming bills	High
FR-8.3	System shall highlight overdue bills	High
FR-8.4	System shall allow marking bills as paid	High
FR-8.5	System shall show notification badge for pending bills	Medium
FR-8.6	System shall support recurring bills (Monthly, Quarterly, Yearly)	Medium

5.9 Reports & Analytics Module

Req ID	Requirement	Priority
FR-9.1	System shall generate income vs expense comparison chart	High
FR-9.2	System shall show expense distribution by category	High
FR-9.3	System shall display spending trend over time	High
FR-9.4	System shall calculate financial health score	Medium
FR-9.5	System shall show top spending categories	Medium
FR-9.6	System shall display income sources breakdown	Low

5.10 Financial Calculators Module

Req ID	Requirement	Priority
FR-10.1	System shall provide EMI calculator (loan amount, rate, tenure)	High
FR-10.2	System shall provide SIP calculator (monthly investment, rate, years)	High
FR-10.3	System shall provide Compound Interest calculator	High
FR-10.4	System shall display detailed results with breakdown	Medium

5.11 Settings & Data Management Module

Req ID	Requirement	Priority

FR-11.1	System shall allow changing currency symbol	High
FR-11.2	System shall allow changing date format	Medium
FR-11.3	System shall allow toggling dark/light theme	High
FR-11.4	System shall allow exporting all data as JSON	High
FR-11.5	System shall allow importing data from JSON backup	High
FR-11.6	System shall allow clearing all data	Medium
FR-11.7	System shall allow loading sample data	Low

---

6. Non-Functional Requirements

6.1 Performance Requirements

Req ID	Requirement	Metric
NFR-1.1	Application shall load within 3 seconds on standard connection	< 3 seconds
NFR-1.2	UI interactions shall respond within 100ms	< 100ms
NFR-1.3	Charts shall render within 500ms	< 500ms
NFR-1.4	Application shall handle 1000+ transactions without degradation	1000+ records

6.2 Usability Requirements

Req ID	Requirement
NFR-2.1	Interface shall be intuitive and require no training
NFR-2.2	System shall provide visual feedback for all actions
NFR-2.3	Error messages shall be clear and actionable
NFR-2.4	Navigation shall be consistent across all pages
NFR-2.5	Forms shall validate input and show inline errors

6.3 Reliability Requirements

Req ID	Requirement
NFR-3.1	Data shall persist across browser sessions
NFR-3.2	Application shall handle localStorage quota exceeded gracefully
NFR-3.3	System shall not crash on invalid input

6.4 Compatibility Requirements

Req ID	Requirement

NFR-4.1	Application shall support Chrome (latest 2 versions)
NFR-4.2	Application shall support Firefox (latest 2 versions)
NFR-4.3	Application shall support Edge (latest 2 versions)
NFR-4.4	Application shall support Safari (latest 2 versions)

6.5 Responsiveness Requirements

Req ID	Requirement	Breakpoint
NFR-5.1	Application shall be fully functional on mobile devices	320px - 480px
NFR-5.2	Application shall be fully functional on tablets	481px - 768px
NFR-5.3	Application shall be optimized for desktop	769px+

6.6 Accessibility Requirements

Req ID	Requirement
NFR-6.1	Application shall support keyboard navigation
NFR-6.2	Interactive elements shall have visible focus states
NFR-6.3	Color contrast shall meet WCAG 2.1 AA standards

---

7. System Requirements

7.1 Hardware Requirements

Development Environment

Component	Minimum	Recommended
Processor	Intel Core i3 / AMD Ryzen 3	Intel Core i5 / AMD Ryzen 5
RAM	4 GB	8 GB
Storage	500 MB free space	1 GB free space
Display	1366 x 768	1920 x 1080

Client Environment

Component	Minimum
Processor	Any modern processor
RAM	2 GB
Storage	50 MB for localStorage
Display	320px width (mobile)

7.2 Software Requirements

Development Environment

Software	Version
Operating System	Windows 10/11, macOS, or Linux
Node.js	18.x or higher
npm	9.x or higher
Code Editor	VS Code (recommended)
Web Browser	Chrome, Firefox, or Edge (latest)
Git	2.x or higher

Runtime Environment

Software	Version
Web Browser	Chrome 90+, Firefox 88+, Edge 90+, Safari 14+
JavaScript	ES6+ support required
localStorage	5 MB minimum

---

8. Technology Stack

8.1 Frontend Technologies

Technology	Version	Purpose
React	18.x	UI component library
React Router	6.x	Client-side routing
Tailwind CSS	3.x	Utility-first CSS framework
Recharts	2.x	Charting library for data visualization
Lucide React	Latest	Icon library
Vite	5.x	Build tool and development server

8.2 Development Dependencies

Package	Purpose
autoprefixer	CSS vendor prefixing
postcss	CSS processing
eslint	Code linting

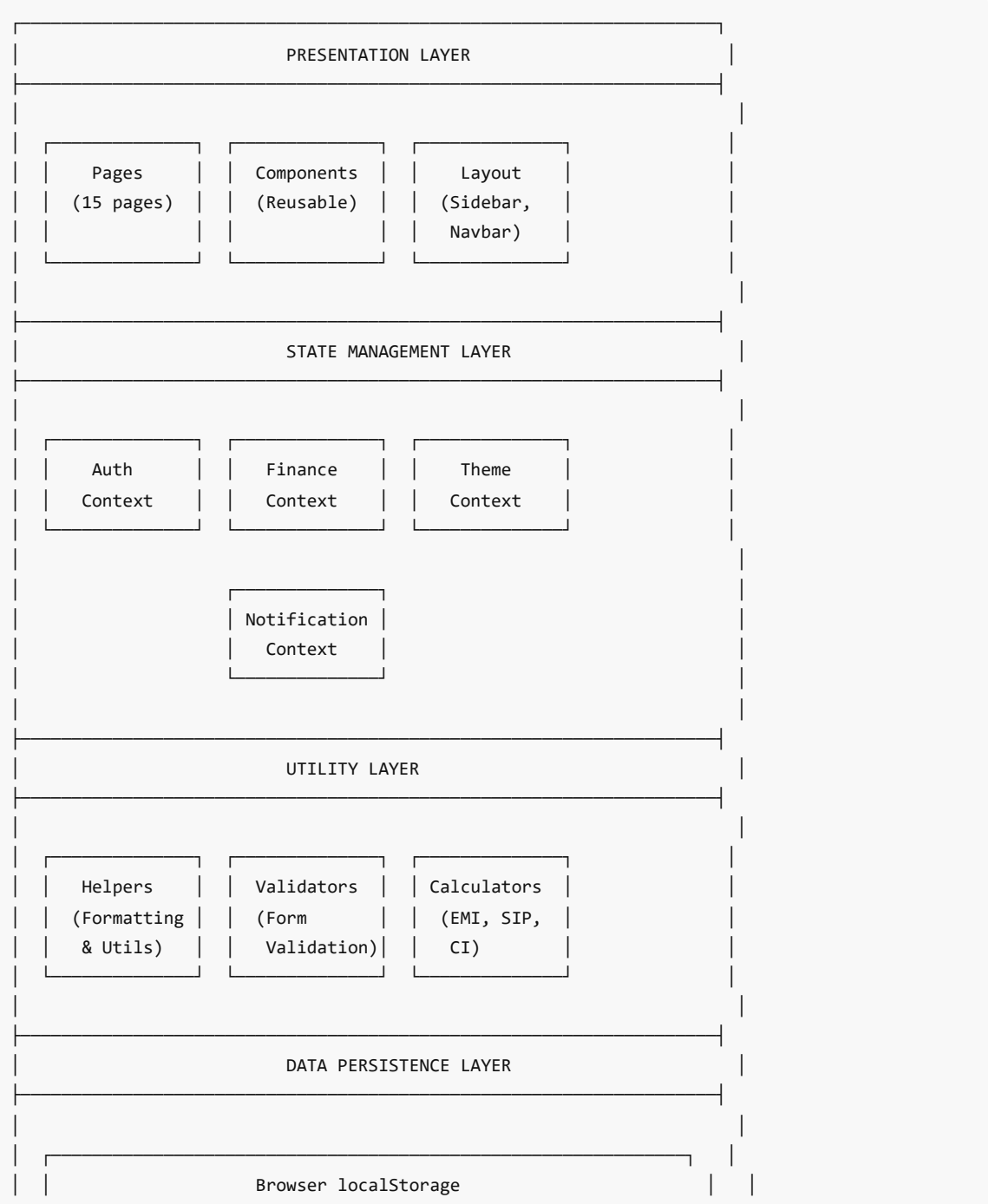
8.3 Architecture Pattern

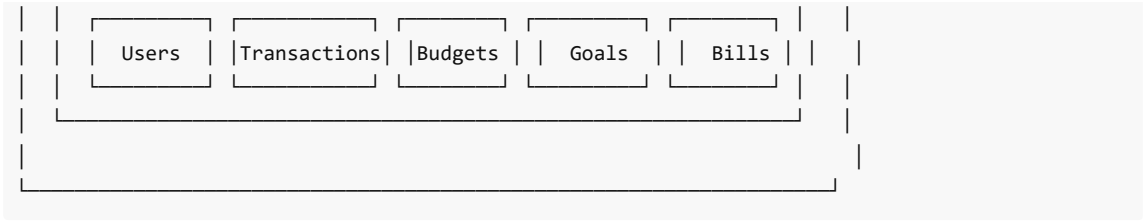
- **Component-Based Architecture:** Using React functional components
- **Context API:** For global state management
- **Custom Hooks:** For reusable logic
- **Module CSS:** Using Tailwind utility classes

---

9. System Architecture

9.1 High-Level Architecture Diagram

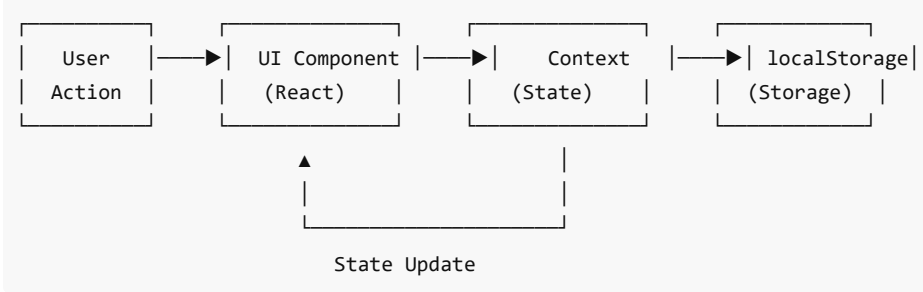




9.2 Component Hierarchy

```
App
├── ThemeProvider
│   └── NotificationProvider
│       └── AuthProvider
│           └── FinanceProvider
│               └── BrowserRouter
│                   └── Routes
│                       ├── /login → Login
│                       ├── /register → Register
│                       └── /* → ProtectedRoute
│                           └── Layout
│                               ├── Sidebar
│                               ├── Navbar
│                               ├── Toast
│                               └── [Page Content]
│                                   ├── Dashboard
│                                   ├── Income
│                                   ├── Expenses
│                                   ├── Budgets
│                                   ├── Goals
│                                   ├── Transactions
│                                   ├── Reports
│                                   ├── Investments
│                                   ├── Bills
│                                   ├── Calculators
│                                   ├── Categories
│                                   ├── Settings
│                                   ├── Profile
│                                   └── Help
```

9.3 Data Flow Diagram



---

10. Database Design

10.1 Storage Strategy

Since this is a client-side application, data is stored in the browser's localStorage as JSON strings. Each data type has its own storage key.

10.2 localStorage Keys

Key	Description	Data Type
`finance_users`	User account information	Array of User objects
`finance_currentUser`	Currently logged in user	User object
`finance_transactions`	All income and expense records	Array of Transaction objects
`finance_budgets`	Budget configurations	Array of Budget objects
`finance_goals`	Savings goals	Array of Goal objects
`finance_investments`	Investment portfolio	Array of Investment objects
`finance_bills`	Bill reminders	Array of Bill objects
`finance_categories`	Custom categories	Object with income/expense arrays
`finance_settings`	App settings	Settings object
`finance_theme`	Theme preference	String ('light' or 'dark')
`finance_initialized`	First-run flag	Boolean

10.3 Data Schemas

User Schema

```
{
  id: "user_123456789",           // Unique identifier
  name: "John Doe",               // Full name
  email: "john@example.com",      // Email (unique)
  password: "hashedPassword",     // Password (hashed)
  phone: "9876543210",            // Phone number (optional)
  occupation: "Software Engineer", // Occupation (optional)
  createdAt: "2026-01-01T00:00:00" // Registration date
}
```

Transaction Schema

```
{
  id: "txn_123456789",            // Unique identifier
  userId: "user_123456789",       // Owner's user ID
  type: "income" | "expense",     // Transaction type
```

```
  name: "Monthly Salary",          // Description
  amount: 50000,                   // Amount in currency units
  category: "Salary",              // Category name
  date: "2026-01-08",              // Transaction date
  paymentMethod: "Bank Transfer", // Payment method (expenses)
  description: "January salary",   // Optional notes
  recurring: false                 // Is recurring
}
```

Budget Schema

```
{
  id: "bgt_123456789",             // Unique identifier
  userId: "user_123456789",        // Owner's user ID
  category: "Food",                // Category name
  limit: 15000,                    // Monthly limit
  spent: 8500,                     // Amount spent (calculated)
  month: "2026-01"                 // Budget month
}
```

Goal Schema

```
{
  id: "goal_123456789",            // Unique identifier
  userId: "user_123456789",        // Owner's user ID
  name: "Emergency Fund",          // Goal name
  targetAmount: 100000,            // Target amount
  currentAmount: 45000,            // Current savings
  deadline: "2026-12-31",          // Target date
  priority: "High",                // Priority level
  description: "6 months expenses" // Optional description
}
```

Investment Schema

```
{
  id: "inv_123456789",             // Unique identifier
  userId: "user_123456789",        // Owner's user ID
  name: "Reliance Industries",     // Investment name
  type: "Stocks",                  // Investment type
  purchasePrice: 2500,             // Buy price per unit
  currentValue: 2750,              // Current price per unit
  quantity: 10,                    // Number of units
  purchaseDate: "2025-06-15"       // Purchase date
}
```

Bill Schema

```
{
  id: "bill_123456789",           // Unique identifier
  userId: "user_123456789",       // Owner's user ID
  name: "Electricity Bill",       // Bill name
  amount: 2500,                   // Bill amount
  dueDate: "2026-01-15",          // Due date
  category: "Utilities",          // Category
  recurring: "Monthly",           // Recurrence type
  isPaid: false,                  // Payment status
  paidDate: null                  // Date when paid
}
```

---

11. User Interface Design

11.1 Design Principles

1. **Clean & Modern:** Minimalist design with ample whitespace
2. **Consistent:** Uniform styling, colors, and interactions throughout
3. **Responsive:** Adapts seamlessly to all screen sizes
4. **Accessible:** High contrast, keyboard navigable
5. **Intuitive:** Self-explanatory UI elements

11.2 Color Palette

Color	Light Mode	Dark Mode	Usage
Primary	#3b82f6	#60a5fa	Main actions, links
Success	#10b981	#34d399	Income, positive values
Danger	#ef4444	#f87171	Expenses, errors, alerts
Warning	#f59e0b	#fbbf24	Warnings, pending items
Background	#f9fafb	#121212	Page background
Surface	#ffffff	#1e1e1e	Cards, modals
Text	#111827	#f9fafb	Primary text
Muted	#6b7280	#9ca3af	Secondary text

11.3 Typography

Element	Font	Size	Weight
Headings (H1)	Inter	24px	Bold (700)
Headings (H2)	Inter	20px	Semibold (600)
Headings (H3)	Inter	18px	Semibold (600)
Body Text	Inter	14px	Regular (400)

Button Text	Inter	14px	Medium (500)
Small Text	Inter	12px	Regular (400)

11.4 Page Layouts

Login/Register Pages

- Centered card layout
- Gradient background
- Logo and branding prominent
- Form with validation

Dashboard

- Summary cards at top (4 columns)
- Quick actions section
- Two-column chart layout
- Recent transactions list

List Pages (Income, Expenses, etc.)

- Header with title and Add button
- Summary card
- Filter/search section
- Data table with actions

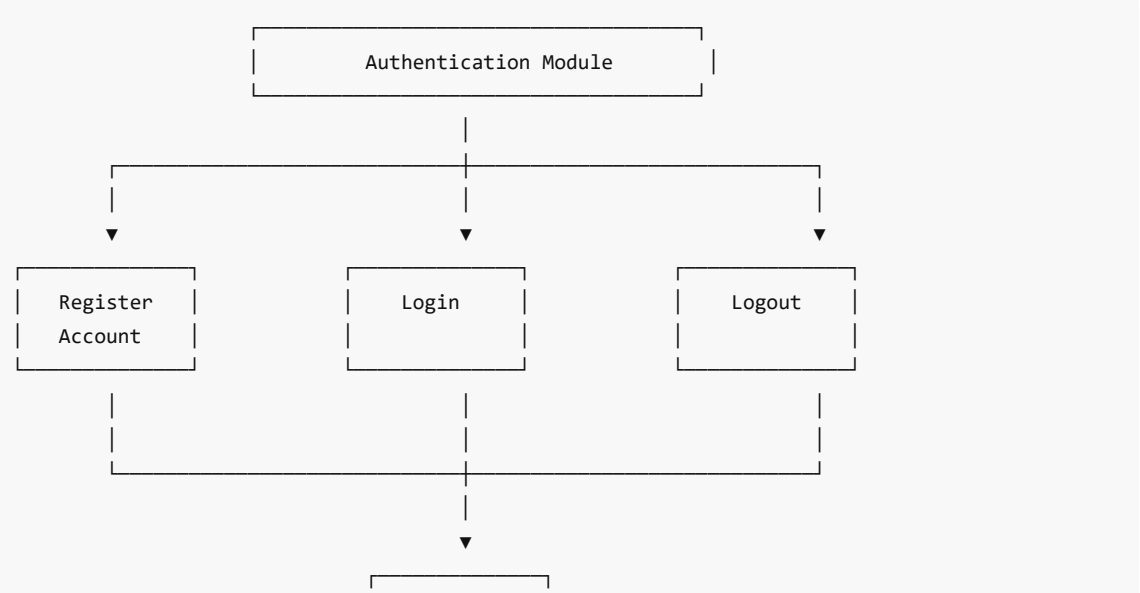
Form Modals

- Overlay with backdrop
- Centered modal
- Form fields with labels
- Cancel and Submit buttons

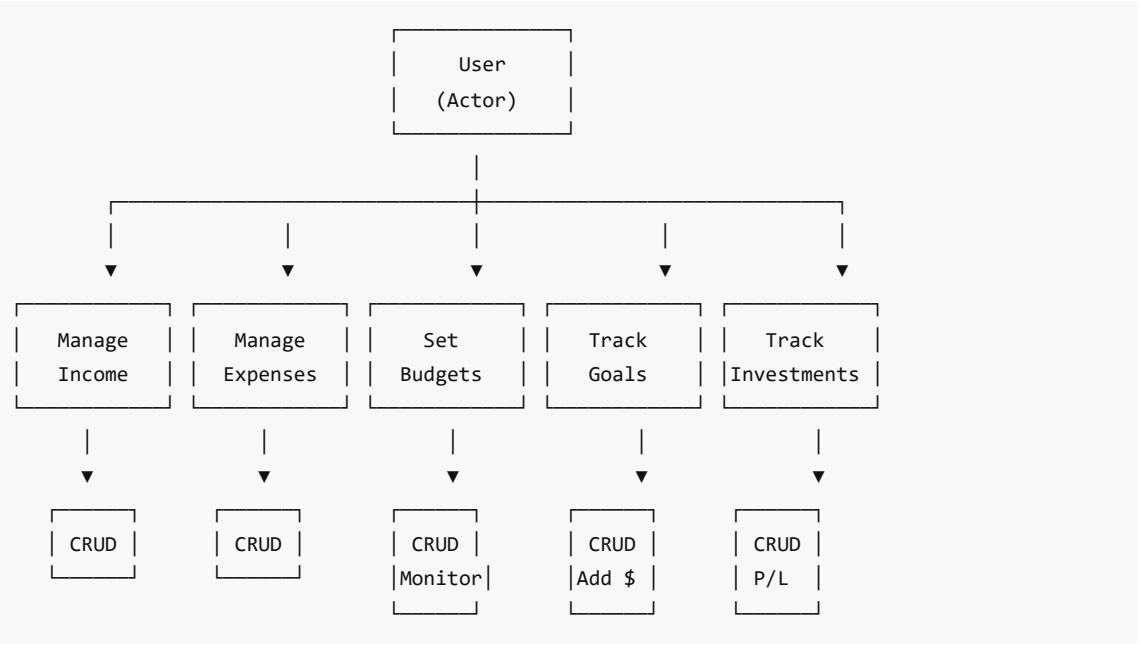
---

12. Use Case Diagrams

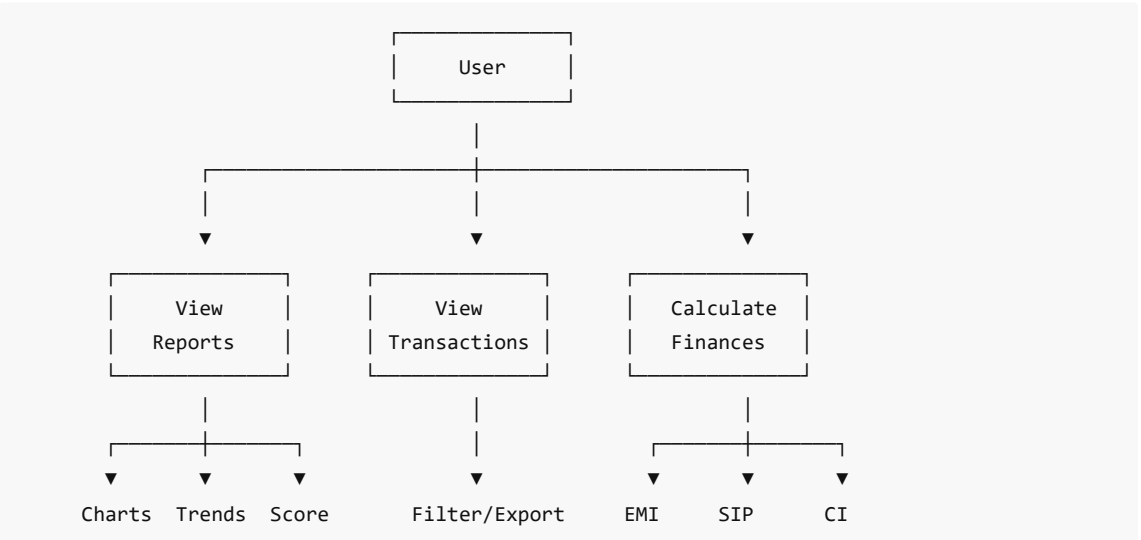
12.1 Authentication Use Cases



12.2 Financial Management Use Cases



12.3 Reporting Use Cases



---

13. Module Descriptions

13.1 Authentication Module

Purpose: Handle user registration, login, logout, and session management.

Components:

- `Login.jsx` - Login form with validation
- `Register.jsx` - Registration form with validation
- `AuthContext.jsx` - Authentication state management
- `ProtectedRoute.jsx` - Route guard component

Key Functions:

- `login(email, password)` - Authenticate user
- `register(userData)` - Create new user account
- `logout()` - Clear session
- `updateProfile(data)` - Update user information
- `changePassword(oldPass, newPass)` - Change password

13.2 Dashboard Module

Purpose: Provide financial overview and quick access to key features.

Components:

- `Dashboard.jsx` - Main dashboard page
- `SummaryCard.jsx` - Metric display card
- `RecentTransactions.jsx` - Transaction list widget
- `QuickActions.jsx` - Action buttons
- `BudgetOverview.jsx` - Budget progress widget

13.3 Transaction Management Module

Purpose: Handle all income and expense CRUD operations.

Components:

- `Income.jsx` - Income management page
- `Expenses.jsx` - Expense management page
- `Transactions.jsx` - Combined transaction view

Key Functions:

- `addTransaction(data)` - Add new transaction
- `updateTransaction(id, data)` - Edit transaction
- `deleteTransaction(id)` - Remove transaction

13.4 Budget Module

Purpose: Allow users to set spending limits and track compliance.

Components:

- `Budgets.jsx` - Budget management page
- `ProgressBar.jsx` - Visual progress indicator

Key Functions:

- `addBudget(data)` - Create budget
- `updateBudget(id, data)` - Modify budget
- `deleteBudget(id)` - Remove budget

13.5 Goals Module

Purpose: Enable users to create and track savings goals.

Components:

- `Goals.jsx` - Goals management page

Key Functions:

- `addGoal(data)` - Create goal
- `updateGoal(id, data)` - Modify goal
- `deleteGoal(id)` - Remove goal
- `addToGoal(id, amount)` - Add savings to goal

13.6 Investment Module

Purpose: Track investment portfolio with profit/loss calculations.

Components:

- `Investments.jsx` - Investment tracking page
- `PortfolioChart.jsx` - Portfolio distribution chart

Key Functions:

- `addInvestment(data)` - Add investment
- `updateInvestment(id, data)` - Update investment details
- `deleteInvestment(id)` - Remove investment

13.7 Bills Module

Purpose: Manage bill reminders and payment tracking.

Components:

- `Bills.jsx` - Bill management page

Key Functions:

- `addBill(data)` - Add bill reminder
- `updateBill(id, data)` - Modify bill
- `deleteBill(id)` - Remove bill
- `markBillPaid(id)` - Mark as paid

13.8 Reports Module

Purpose: Generate visual analytics and reports.

Components:

- `Reports.jsx` - Analytics page
- `ExpensePieChart.jsx` - Expense distribution
- `IncomeExpenseChart.jsx` - Income vs Expense comparison
- `TrendLineChart.jsx` - Spending trends

13.9 Calculators Module

Purpose: Provide financial calculation tools.

Components:

- `Calculators.jsx` - Calculator page

Calculation Functions:

- `calculateEMI(principal, rate, tenure)` - EMI calculation
- `calculateSIP(monthly, rate, years)` - SIP returns
- `calculateCompoundInterest(principal, rate, time, frequency)` - CI calculation

13.10 Settings Module

Purpose: Allow users to customize app preferences and manage data.

Components:

- `Settings.jsx` - Settings page

Key Functions:

- `updateSettings(data)` - Save preferences
- `exportAllData()` - Export JSON backup
- `importData(data)` - Import from backup
- `clearAllStorage()` - Reset all data

---

14. Security Considerations

14.1 Data Security

Aspect	Implementation
Password Storage	Passwords are stored (note: in production, would use bcrypt hashing)
Session Management	User session stored in localStorage
Data Isolation	Each user can only access their own data (filtered by userId)
Input Validation	All forms validate input before processing
XSS Prevention	React's built-in protection against XSS attacks

14.2 Security Limitations

> ***Note:*** *As this is a client-side only application meant for educational purposes, the following security aspects are simplified:*

1. Passwords are not encrypted (production would use bcrypt)
2. No server-side validation (all validation is client-side)
3. localStorage can be accessed via browser developer tools
4. No HTTPS enforcement (handled by hosting platform)

14.3 Recommendations for Production

1. Implement backend server with proper authentication
2. Use JWT tokens for session management
3. Hash passwords with bcrypt
4. Implement rate limiting
5. Use HTTPS for all communications

6. Add CSRF protection
7. Implement proper data encryption

---

15. Testing Strategy

15.1 Testing Levels

Level	Description	Tools
Unit Testing	Test individual functions	Jest
Component Testing	Test React components	React Testing Library
Integration Testing	Test module interactions	Jest + RTL
End-to-End Testing	Test complete user flows	Cypress
Manual Testing	User acceptance testing	Browser

15.2 Test Cases

Authentication Test Cases

TC ID	Test Case	Expected Result
TC-001	Login with valid credentials	Redirect to dashboard
TC-002	Login with invalid email	Show error message
TC-003	Login with wrong password	Show error message
TC-004	Register with valid data	Create account, redirect to dashboard
TC-005	Register with existing email	Show duplicate error
TC-006	Logout	Clear session, redirect to login

Transaction Test Cases

TC ID	Test Case	Expected Result
TC-010	Add income entry	Entry appears in list, balance updates
TC-011	Add expense entry	Entry appears in list, balance updates
TC-012	Edit transaction	Changes reflected immediately
TC-013	Delete transaction	Entry removed, balance updates
TC-014	Filter by category	Only matching entries shown

Budget Test Cases

TC ID	Test Case	Expected Result
TC-020	Create budget	Budget appears with 0% progress

TC-021	Add expense in category	Progress bar updates
TC-022	Exceed budget 70%	Warning indicator shown
TC-023	Exceed budget 100%	Over budget indicator shown

15.3 Browser Compatibility Testing

Browser	Version	Status
Chrome	120+	✅ Tested
Firefox	121+	✅ Tested
Edge	120+	✅ Tested
Safari	17+	✅ Tested

15.4 Responsive Testing

Device	Screen Size	Status
Mobile (Small)	320px - 375px	✅ Tested
Mobile (Large)	376px - 480px	✅ Tested
Tablet	481px - 768px	✅ Tested
Laptop	769px - 1024px	✅ Tested
Desktop	1025px+	✅ Tested

---

16. Future Enhancements

16.1 Short-Term Enhancements

Enhancement	Description	Priority
Data Sync	Cloud backup using Firebase	High
PWA Support	Installable progressive web app	High
Notifications	Push notifications for bill reminders	Medium
Categories	Custom icon and color for categories	Medium
Recurring Transactions	Auto-create recurring entries	Medium

16.2 Long-Term Enhancements

Enhancement	Description	Priority
Backend API	Node.js/Express backend server	High
Database	MongoDB or PostgreSQL integration	High

Multi-currency	Support for multiple currencies	Medium
Bank Integration	Read-only bank account linking	Medium
AI Insights	ML-based spending predictions	Low
Social Features	Family finance sharing	Low
Mobile Apps	React Native iOS/Android apps	Low

---

17. Conclusion

The Personal Finance Management & Investment Tracker System is a comprehensive web application that addresses the growing need for personal financial management tools. This document has outlined the complete requirements, architecture, and implementation details for the system.

Key Achievements

1. **Complete Feature Set:** All planned features including income/expense tracking, budgeting, goals, investments, and bills have been implemented.
2. **Modern Technology Stack:** Built using React 18, Tailwind CSS, and modern JavaScript practices.
3. **User-Friendly Interface:** Clean, responsive design with dark/light mode support.
4. **Data Persistence:** Reliable localStorage-based data storage with export/import functionality.
5. **Extensible Architecture:** Modular design allows for easy future enhancements.

Learning Outcomes

Through this project, the following skills were developed:

- React component architecture and hooks
- State management using Context API
- Responsive web design with Tailwind CSS
- Data visualization with Recharts
- Client-side data persistence
- Form validation and error handling
- Project documentation

---

18. References

1. **React Documentation** - https://react.dev/
2. **Tailwind CSS Documentation** - https://tailwindcss.com/docs
3. **React Router Documentation** - https://reactrouter.com/
4. **Recharts Documentation** - https://recharts.org/
5. **Lucide Icons** - https://lucide.dev/
6. **Vite Documentation** - https://vitejs.dev/
7. **MDN Web Docs (localStorage)** - https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage
8. **Web Accessibility Guidelines (WCAG)** - https://www.w3.org/WAI/standards-guidelines/wcag/

Appendix A: Project File Structure

```
finance-tracker/
├── index.html
├── package.json
├── vite.config.js
├── tailwind.config.js
├── postcss.config.js
├── .gitignore
├── README.md
│
├── docs/
│   └── Requirement_Specification_Document.md
│
└── src/
    ├── main.jsx
    ├── App.jsx
    ├── index.css
    │
    ├── components/
    │   ├── Charts/
    │   │   ├── ExpensePieChart.jsx
    │   │   ├── IncomeExpenseChart.jsx
    │   │   ├── TrendLineChart.jsx
    │   │   └── PortfolioChart.jsx
    │   │
    │   ├── Common/
    │   │   ├── Button.jsx
    │   │   ├── Card.jsx
    │   │   ├── EmptyState.jsx
    │   │   ├── Input.jsx
    │   │   ├── LoadingSpinner.jsx
    │   │   ├── Modal.jsx
    │   │   ├── ProgressBar.jsx
    │   │   ├── Select.jsx
    │   │   └── Toast.jsx
    │   │
    │   ├── Dashboard/
    │   │   ├── BudgetOverview.jsx
    │   │   ├── QuickActions.jsx
    │   │   ├── RecentTransactions.jsx
    │   │   └── SummaryCard.jsx
    │   │
    │   └── Layout/
    │       ├── Layout.jsx
    │       ├── Navbar.jsx
    │       ├── ProtectedRoute.jsx
    │       └── Sidebar.jsx
    │
    ├── context/
    │   ├── AuthContext.jsx
```

```
    │   ├── FinanceContext.jsx
    │   ├── NotificationContext.jsx
    │   └── ThemeContext.jsx
    │
    ├── data/
    │   └── sampleData.js
    │
    ├── pages/
    │   ├── Bills.jsx
    │   ├── Budgets.jsx
    │   ├── Calculators.jsx
    │   ├── Categories.jsx
    │   ├── Dashboard.jsx
    │   ├── Expenses.jsx
    │   ├── Goals.jsx
    │   ├── Help.jsx
    │   ├── Income.jsx
    │   ├── Investments.jsx
    │   ├── Login.jsx
    │   ├── Profile.jsx
    │   ├── Register.jsx
    │   ├── Reports.jsx
    │   ├── Settings.jsx
    │   └── Transactions.jsx
    │
    └── utils/
        ├── calculators.js
        ├── helpers.js
        ├── localStorage.js
        └── validators.js
```

Appendix B: Sample Screenshots

The application includes the following screens (screenshots to be attached):

1. Login Page
2. Dashboard
3. Income Management
4. Expense Management
5. Budget Tracking
6. Savings Goals
7. Investment Portfolio
8. Bill Reminders
9. Reports & Analytics
10. Financial Calculators
11. Settings Page

Document End

Prepared for B.Sc. (Computer Science) Internship 2025 Submission Date: January 8, 2026