System Requirements Specification (SRS)

Smart Finance Tracker

Table of Contents

1. Introduction

1.1. Purpose

This document specifies the functional and non-functional requirements for the Smart Finance Tracker, a web-based personal finance management application. The system helps users track expenses, manage budgets, set savings goals, and analyze financial habits with AI-powered insights.

1.2. Scope

Smart Finance Tracker is a client-side web application that enables users to:

- Track income and expenses with detailed categorization.
- Create and manage personalized budgets
- Set and monitor financial goals with progress tracking
- Receive AI-powered financial insights and recommendations
- Export financial data in multiple formats (CSV, PDF)
- Manage account settings and preferences
- Access the application across devices with responsive design

The system is designed for personal finance management and will run on any modern web browser.

1.3. Abbreviations

SRS: System Requirements Specification

UI: User Interface

UX: User Experience

AI: Artificial Intelligence

API: Application Programming Interface

MERN: MongoDB, Express.js, React.js, Node.js

JWT: JSON Web Token

CRUD: Create, Read, Update, Delete

CSV: Comma-Separated Values

CDN: Content Delivery Network

1.4. Overview

The remainder of this document provides a detailed description of the system's functionality, including user requirements, system features, and technical specifications.

2. Overall Description

This section presents a high-level overview of the Smart Finance Tracker system, describing its context, main features, target users, and operating environment.

2.1. Product Perspective

Smart Finance Tracker is a web-based application accessible through standard web browsers. Users interact with the system through a responsive web interface to manage their personal finances.

- **Frontend**: React-based single-page application with responsive design
- **Backend**: RESTful API built with Node.js and Express.js
- **Database**: MongoDB for flexible document storage
- **Authentication**: JWT-based secure authentication system
- **External Services**: AI API integration for intelligent features

2.2. Product Features

The main features include:

- User authentication with secure password management
- Transaction recording and management (CRUD operations)
- Budget planning and monitoring with visual indicators
- Financial goal tracking with progress visualization
- AI-Powered spending analysis and recommendations
- Category management with default and custom categories
- Settings management including profile and preferences
- Visual reports and analytics with charts
- Data export capabilities (CSV, PDF)
- Responsive dashboard with overview metrics

- Dark mode support for user preference
- Real-time data synchronization

2.3. User Characteristics

Target Users:

- Individuals seeking to manage personal finances
- Users with basic computer and internet literacy
- No specialized financial or technical knowledge required

2.4. Operating Environment

Client-Side:

- Modern web browsers (Chrome 90+, Firefox 88+, Safari 14+, Edge 90+)
- Responsive design supporting devices from 375px to 1920px width
- JavaScript enabled
- LocalStorage support for theme preferences

Server-Side:

- Node.js runtime environment (v16 or higher)
- MongoDB database (v5 or higher)
- Express.js web framework
- Cloud or dedicated hosting environment

Network:

- Internet connection required
- HTTPS support for secure communication
- RESTful API communication

2.5. Assumptions and Dependencies

- Users have access to internet connection
- Users provide accurate financial information
- Users have a valid email address for registration

3. Functional Requirements

This section describes the main functionalities of the Smart Finance Tracker, detailing what the system must do to meet user needs and business objectives.

3.1. User Authentication

Users can register, login, and manage their accounts.

Requirements:

- The system shall allow new users to register with email and password
- The system shall enforce password requirements (minimum 8 characters)
- The system shall authenticate users using JWT tokens
- The system shall maintain secure user sessions with token expiration
- The system shall allow users to logout from any device
- The system shall provide password reset functionality via email
- The system shall display a landing page for unauthenticated users
- The system shall hash passwords using bcrypt before storage
- The system shall validate email format during registration
- The system shall prevent duplicate email registrations
- The system shall provide session timeout after 30 minutes of inactivity
- The system shall redirect unauthorized users to login page

3.2. Dashboard

Central overview page displaying key financial financial metrics and recent activity.

Requirements:

- The system shall display total income for selected period (month/year/all)
- The system shall display total expenses for selected period
- The system shall calculate and show current balance (income - expenses)
- The system shall calculate and show current savings
- The system shall show recent transactions (last 5)
- The system shall display transaction count by type
- The system shall provide period selection filters (month, year, all time)
- The system shall show visual summary cards with color coding
- The system shall display spending by category in chart form
- The system shall show budget status summary

- The system shall refresh data automatically on user actions
- The system shall show empty state
- The system shall display AI-generated financial tips

3.3. Transaction Management

Complete transaction lifecycle management including creation, viewing, editing, and deletion.

Requirements:

- The system shall provide a form to add new transactions
- The system shall require: amount, category, date, and type (income/expense)
- The system shall allow optional description field
- The system shall validate all inputs before submission
- The system shall save transactions to MongoDB database
- The system shall provide real-time success/error feedback via toasts
- The system shall allow inline category creation without leaving transaction form
- The system shall enable editing of existing transactions via modal
- The system shall allow deletion of transactions with confirmation prompt
- The system shall support bulk filtering by type, category, and date range
- The system shall support sorting by date or amount (ascending/descending)
- The system shall limit transaction listing with pagination or load more
- The system shall display transaction history with visual type indicators
- The system shall show transaction details including category, amount, and date
- The system shall calculate and display totals for filtered transactions

3.4. Budget Planner

Budget creation and monitoring system with alerts and visualizations.

Requirements:

- The system shall allow users to set monthly budget limits per category
- The system shall display budget vs actual spending comparison
- The system shall calculate remaining budget for each category
- The system shall show visual progress bars for budget utilization
- The system shall alert users when approaching 80% of budget limit
- The system shall highlight categories exceeding budget in red

- The system shall allow editing and deletion of budget entries
- The system shall calculate percentage of budget used
- The system shall show month-to-date spending against budget
- The system shall provide AI-powered budget recommendations
- The system shall track budget performance over time
- The system shall allow budget templates for common categories

3.5. Category Management

Dynamic category system with default and user-created categories.

Requirements:

- The system shall provide default categories for income and expense
- The system shall allow users to create custom categories
- The system shall categorize by type (income or expense)
- The system shall prevent duplicate category names per user
- The system shall allow category creation during transaction entry
- The system shall display category list in dropdown format
- The system shall show spending totals per category in reports
- The system shall prevent deletion of categories with existing transactions
- The system shall support both default and user-specific categories
- The system shall validate category names (max 50 characters)

3.6. Reports & Analytics

Comprehensive financial reporting with visualizations and insights.

Requirements:

- The system shall generate monthly spending reports
- The system shall display income vs expense comparison charts
- The system shall show spending trends over time with line graphs
- The system shall display category-wise breakdown in pie/donut charts
- The system shall calculate savings rate percentage
- The system shall show month-over-month comparisons
- The system shall provide year-over-year analysis
- The system shall display top spending categories

- The system shall show spending patterns by day of week
- The system shall provide AI-generated insights and recommendations
- The system shall allow custom date range selection for reports
- The system shall export reports to PDF format

3.7. Financial Goals

Goal setting and tracking system with progress visualization.

Requirements:

- The system shall allow users to create savings goals with target amount
- The system shall require goal name, target amount, and deadline
- The system shall display progress towards each goal with visual indicators
- The system shall calculate percentage completion automatically
- The system shall allow multiple active goals simultaneously
- The system shall show estimated completion date based on savings rate
- The system shall allow editing and deletion of goals
- The system shall track goal history and achievements
- The system shall send notifications when goals are reached
- The system shall provide goal templates (emergency fund, vacation, etc.)
- The system shall show time remaining until deadline
- The system shall calculate required monthly savings to meet goal

3.8. AI-Powered Features

Intelligent features using AI for personalized financial guidance.

Requirements:

- The system shall provide AI-generated financial tips on dashboard
- The system shall offer AI budget suggestions in Budget Planner
- The system shall provide AI insights in Reports & Analytics
- The system shall analyze user spending patterns automatically
- The system shall identify unusual spending behavior
- The system shall suggest budget optimizations based on history
- The system shall predict potential budget overruns
- The system shall offer personalized savings recommendations

- The system shall detect recurring expenses and suggest automation
- The system shall provide spending forecasts for upcoming months
- The system shall compare user spending to category averages
- The system shall suggest categories for uncategorized transactions

3.9. Settings & Profile Management

User preferences and account management interface.

Requirements:

- The system shall allow users to update profile information (name)
- The system shall allow currency preference selection
- The system shall allow users to change password securely
- The system shall provide account deletion option with confirmation
- The system shall save user preferences persistently
- The system shall allow data export before account deletion
- The system shall show account statistics (join date, transaction count)
- The system shall require current password for sensitive changes

3.10. Data Export

Export financial data in various formats for external use.

Requirements:

- The system shall export transaction data to CSV format
- The system shall allow date range selection for export
- The system shall allow category filtering for export
- The system shall generate summary reports in PDF format
- The system shall include all transaction details in exports
- The system shall format dates consistently in exports
- The system shall include headers in CSV exports
- The system shall calculate totals in exported reports
- The system shall allow monthly, yearly, or custom range exports
- The system shall include charts and graphs in PDF reports

3.11. Error Handling and Session Management

Comprehensive error handling and session management system.

Requirements:

- The system shall display custom 404 error page for invalid URLs
- The system shall provide session timeout page after 30 minutes inactivity
- The system shall display logout confirmation page
- The system shall redirect users appropriately after session expiry
- The system shall provide clear navigation options from error pages
- The system shall handle network errors gracefully
- The system shall display user-friendly error messages
- The system shall log errors for debugging purposes
- The system shall validate all API responses
- The system shall retry failed requests automatically (with limits)

4. Non-Functional Requirements

This section defines the quality attributes and constraints of the system, specifying how the system should perform its functions.

4.1. Performance

- Page load time shall not exceed 3 seconds on stable network connection
- API response time shall not exceed 500ms for 95% of requests
- The system shall support at least 5000 transactions per user
- The system shall handle 100 concurrent users without degradation
- Database queries shall be optimized with proper indexing
- Frontend shall implement code splitting for faster loads
- Images and assets shall be optimized for web delivery

4.2. Security

- User passwords shall be hashed using bcrypt (10+ salt rounds)
- The system shall validate all user inputs on client and server
- Sessions shall timeout after 30 minutes of inactivity
- JWT tokens shall expire after 30 days
- The system shall use HTTPS for all communications in production
- API endpoints shall implement rate limiting

- The system shall prevent SQL/NoSQL injection attacks
- The system shall implement CORS properly
- Sensitive data shall not be exposed in API responses
- The system shall use environment variables for secrets

4.3. Usability

- The interface shall be intuitive with minimal learning curve
- Navigation shall be consistent across all pages
- Error messages shall be clear and actionable
- The system shall provide confirmation for destructive actions
- Forms shall provide inline validation feedback
- Loading states shall be indicated for all async operations
- The system shall be accessible
- Keyboard navigation shall be supported throughout
- The system shall provide helpful tooltips for complex features
- Empty states shall guide users on next actions

4.4. Reliability

- The ystem shall maintain data integrity during failures
- The system shall handle errors without crashing
- Database backups shall be performed daily
- The system shall be available 99% of the time
- Failed transactions shall be rolled back automatically
- The system shall implement proper error logging
- The system shall recover gracefully from API failures
- Data validation shall occur at multiple layers

4.5. Compatibility

- The system shall work on Chrome, Firefox, Safari, and Edge (latest versions)
- The system shall be responsive for desktop, tablet, and mobile
- The system shall support screen sizes from 375px to 1920px width
- The system shall work on iOS and Android mobile browsers

- The system shall degrade gracefully on older browsers
- The system shall support touch and mouse interactions
- The system shall work with screen readers

4.6. Maintainability

- Code shall be well-documented with comments
- The system shall follow best coding practices
- Database schema shall be properly structured
- The system shall use modular code architecture
- The system shall implement proper Git workflow
- The system shall include automated tests (unit, integration)
- The system shall use linting and code formatting tools
- API documentation shall be maintained (Swagger/Postman)
- The system shall implement proper logging

4.7. Scalability

- The system shall support horizontal scaling
- Database connections shall be pooled efficiently
- The system shall implement caching where appropriate
- Static assets shall be served via CDN
- The system shall handle growing user base without rewrite
- API design shall version endpoints for future changes

5. System Architecture

This section outlines the technical architecture of the system, including technology choices, database structure, and system workflow.

5.1. Technology Stack

Frontend:

- React.js 18+ (UI framework)
- Tailwind CSS (styling)
- React Router (navigation)

- Axios (HTTP client)
- React-Toastify (notifications)
- Recharts or Chart.js (data visualization)TML5 for structure

Backend:

- Node.js 16+ (runtime)
- Express.js (web framework)
- MongoDB (database)
- Mongoose (Object Data Mapper)
- JWT (authentication)
- Bcrypt.js (password hashing)
- Crypto (password reset tokens)

Additional Libraries:

- AI API (OpenAI/Claude/Gemini) for intelligent features

Development Tools:

- Git (version control)
- ESLint (code linting)
- Prettier (code formatting)
- Postman (API testing)

5.2. Database Design

Primary Tables:

1. users

- _id (ObjectId)
- name (String, required)
- email (String, unique, required)
- password (String, hashed, required)
- currency (String, default: 'XAF')
- resetPasswordToken (String)
- resetPasswordExpire (Date)
- isActive (Boolean)

- createdAt (Date)
- updatedAt (Date)

2. transactions

- _id (ObjectId)
- userId (ObjectId, ref: 'User')
- type (String, enum: ['income', 'expense'])
- amount (Number, required)
- category (ObjectId, ref: 'Category')
- description (String)
- transactionDate (Date)
- createdAt (Date)
- updatedAt (Date)

3. categories

- _id (ObjectId)
- userId (ObjectId, ref: 'User', nullable for defaults)
- name (String, required)
- type (String, enum: ['income', 'expense'])
- isDefault (Boolean)
- createdAt (Date)
- updatedAt (Date)

4. budgets

- _id (ObjectId)
- userId (ObjectId, ref: 'User')
- category (ObjectId, ref: 'Category')
- amount (Number, required)
- period (String, default: 'monthly')
- startDate (Date)
- endDate (Date)
- createdAt (Date)
- updatedAt (Date)

5. goals

- _id (ObjectId)
- userId (ObjectId, ref: 'User')
- name (String, required)
- targetAmount (Number, required)
- currentAmount (Number, default: 0)
- deadline (Date)
- status (String, enum: ['active', 'completed', 'cancelled'])
- createdAt (Date)
- updatedAt (Date)

5.3. API Architecture

Base URL: /api

Authentication Endpoints:

- POST /auth/register - User registration
- POST /auth/login - User login
- GET /auth/me - Get current user
- POST /auth/forgot-password - Request password reset
- POST /auth/reset-password/:token - Reset password

Transaction Endpoints:

- GET /transactions - Get all user transactions (with filters)
- POST /transactions - Create transaction
- PUT /transactions/:id - Update transaction
- DELETE /transactions/:id - Delete transaction
- GET /transactions/summary - Get summary statistics

Category Endpoints:

- GET /categories - Get all categories

- POST /categories - Create category

Budget Endpoints:

- GET /budgets - Get all budgets
- POST /budgets - Create budget
- PUT /budgets/:id - Update budget
- DELETE /budgets/:id - Delete budget

Goal Endpoints:

- GET /goals - Get all goals
- POST /goals - Create goal
- PUT /goals/:id - Update goal
- DELETE /goals/:id - Delete goal

Report Endpoints:

- GET /reports/monthly - Get monthly report
- GET /reports/yearly - Get yearly report
- GET /reports/category - Get category breakdown

Export Endpoints:

- GET /export/csv - Export to CSV
- GET /export/pdf - Export to PDF

5.4. System Flow

- User visits landing page
- User registers/logs in (JWT token issued)
- User redirected to dashboard
- User performs actions (CRUD operations)
- Frontend sends API requests with JWT token
- Backend validates token and processes request
- Database operations performed via Mongoose

- Response sent back to frontend
- UI updated based on response
- Toast notifications show success/error

6. User Interface Requirements

This section specifies the design and interaction requirements for the user interface to ensure consistency and usability.

6.1. Navigation

- Top navigation bar with logo and main menu links
- Responsive hamburger menu for mobile devices (< 768px)
- Active page indicator in navigation
- Dark mode toggle in navigation
- User profile/name display in navigation
- Logout button accessible from all authenticated pages
- Landing page as entry point for unauthenticated users
- Breadcrumb navigation for nested pages
- Footer with links and copyright information

6.2. Forms

- Clear labels for all input fields
- Inline validation with real-time feedback
- Required field indicators (*)
- Helpful placeholder text
- Error messages displayed below fields
- Success feedback via toast notifications
- Submit and cancel buttons clearly labeled
- Loading states for async submissions
- Disabled state during submission
- Date pickers for date fields
- Dropdown selects for categories
- Number inputs with min/max validation

6.3. Visual Design

- Clean and modern interface with consistent spacing
- Consistent color scheme matching brand identity
- Appropriate use of icons (Bootstrap Icons)
- Responsive layout adapting to all screen sizes
- Charts and graphs for data visualization
- Card-based layouts for content sections
- Visual hierarchy with typography
- Hover states for interactive elements
- Smooth transitions and animations
- Empty states with helpful guidance
- Loading skeletons for async content
- Modal dialogs for focused interactions

6.4. Accessibility

- Semantic HTML elements
- ARIA labels where needed
- Keyboard navigation support
- Focus indicators on interactive elements
- Sufficient color contrast ratios
- Screen reader compatible
- Alt text for images
- Skip to content link

7. Implementation Constraints

This section identifies technical and business limitations that must be considered during system development.

7.1. Technical Constraints

- Must use MERN stack (MongoDB, Express, React, Node.js)

- Must implement RESTful API architecture
- Must use JWT for authentication
- Must integrate AI API for intelligent features
- Must implement responsive design
- Must support modern browsers
- Must use Git for version control
- Must deploy frontend and backend separately
- Must use environment variables for configuration
- Must implement proper error handling

7.2. Business Rules

- Each user can only access their own data
- Transactions must have valid dates (not future dates)
- Budget amounts must be positive numbers
- Categories cannot be deleted if transactions exist
- Goals must have target amounts greater than zero
- Password reset tokens expire after 10 minutes
- User sessions expire after 30 minutes of inactivity
- Email addresses must be unique across system
- Transaction amounts must be positive
- Users must be 13 years or older (terms of service)

7.3. Development Constraints

- Project must be completed within allocated timeline
- Code must follow team's style guide
- All features must be tested before deployment
- Documentation must be maintained throughout

8. Appendices

Supplementary information including page lists and terminology definitions.

8.1. Page/Module List

Public Pages:

1. Landing Page
2. Sign In
3. Sign Up
4. Forgot Password
5. Confirm Reset Password
6. Session Timeout/Logout
7. 404 Error Page

Protected Pages:

8. Dashboard (Overview)
9. Transactions (List & Management)
10. Add/Edit Transaction (Modal)
11. Budget Planner
12. Reports & Analytics
13. Financial Goals
14. Settings/Profile
15. Export Data

8.2. User Roles

A registered user:

- Can create, read, update, delete own transactions
- Can set budgets and goals
- Can view reports and analytics
- Can export own data
- Can manage own profile

8.3. Glossary

Backend: Server-side application logic and database interactions

Budget: A planned spending limit set for a specific category over a defined time period.

Category: A classification system used to organize transactions into groups such as Transportation, Entertainment, Bills, etc.

Dashboard: The main overview page that displays key financial metrics, recent transactions, and summary information.

Expense: Money spent or costs incurred, recorded as a negative transaction in the system.

Export: The process of converting and downloading financial data from the system into external formats (CSV, PDF).

Frontend: Client-side user interface built with React.

Goal: A target savings amount with a specified deadline.

Income: Money received or earned, recorded as a positive transaction in the system.

JWT: JSON Web Token used for stateless authentication.

Middleware: Functions that execute during the request-response cycle in Express.

MongoDB: NoSQL database used for data persistence.

REST API: Representational State Transfer API for client-server communication.

Savings Rate: The percentage of income that is not spent, calculated as (Income - Expenses) / Income × 100.

Session: The period during which a user is actively logged into the system.

Transaction: A record of financial activity (income or expense) with amount, category, date, and description.

User Authentication: The process of verifying user identity through credentials.

8.4. Assumptions

- Users have modern devices with internet connectivity
- Users provide accurate financial information
- Users understand basic financial concepts
- MongoDB Atlas free tier sufficient for initial deployment
- AI API costs within budget constraints
- Users consent to data processing for AI features
- Email delivery service available for password resets
- Hosting platforms support Node.js and MongoDB

8.5. Future Enhancement

Features planned for future versions:

1. **Receipt scanning and OCR**: Upload photo of receipt, extract amount, date etc using OCR, AI category suggestion, one-click to add transaction.
2. **Recurring transaction automation**: Setup recurring bills and utilities and get reminders before due date.
3. **Smart Budget Auto-Adjust:** AI learns your patterns and suggests realistic budgets
4. **Financial health score**: Calculate score based on savings rate, budget adherence etc and show on dashboard with suggestions on how to improve.
5. **Push notifications, Offline mode, Transaction templates**
6. **Mobile native apps (React Native)**
7. **Social features**: Compare with friends, track expenses with roommates/partners, split bills, track who owes, family/shared accounts.
8. **Gamification (achievements, badges):** Challenges like 30-Day no eating out challenge, cancel one subscription challenge, no spend weekend etc
9. **Onboarding Flow:** Guided setup for new users (welcome screen, set currency & budget categories, add first transaction(tutorial), set first budget, create first goal, celebrate completion)