Software Requirements Specification (SRS)

AI Financial Advisor

Date: 11 January 2026

Project by:

- Mukul Bhardwaj

Table of Contents

---

1. Introduction

1.1 Purpose

This SRS document describes the AI-Based Financial Analyzer system - a web application for tracking and analyzing personal finances using AI-powered insights.

1.2 Scope

The system provides:

- **Data Input:** Manual entry and CSV file upload for monthly financial data
- **Visual Analytics:** Interactive dashboard with pie charts and bar graphs
- **Financial Metrics:** Automated calculation of income, expenses, savings, and savings rate
- **AI Analysis:** Health scoring (0-100) and personalized recommendations via Groq AI (LLaMA 3.1-8B)
- **Interactive Chat:** AI chatbot for financial advice
- **Local Deployment:** Flask-based server for privacy

Target Users: Individuals seeking to track and manage personal finances (students, professionals, families, freelancers)

1.3 Definitions

Term	Definition
SRS	Software Requirements Specification
API	Application Programming Interface

Term	Definition
CSV	Comma-Separated Values
EMI	Equated Monthly Installment
Flask	Python web framework
Groq	AI inference API provider

1.4 References

- Flask: https://flask.palletsprojects.com/
- Groq API: https://console.groq.com/docs
- Chart.js: https://www.chartjs.org/
- Pandas: https://pandas.pydata.org/

---

2. Overall Description

2.1 Product Perspective

Standalone web application, integrating Groq API for AI features. Supports local execution and serverless deployment on Vercel.

System Context:

- **Frontend:** HTML5, CSS3, JavaScript with Chart.js
- **Backend:** Python Flask framework
- **AI Engine:** Groq API with LLaMA 3.1-8B-Instant
- **Data Storage:** Session-based (temporary)
- **File Processing:** Pandas library

2.2 Product Functions

1. **Financial Data Entry**
   - Manual form input (9 expense categories)
   - CSV file upload with parsing
   - Input validation
2. **Data Processing**
   - Calculate total expenses, savings, savings rate
   - Expense categorization and percentages
   - Financial health score (0-100)
3. **Visual Dashboard**
   - Interactive pie chart (expense distribution)
   - Bar graph (category breakdown)
   - Summary cards (income, expenses, savings)

    - Colour-coded indicators

4. **AI-Powered Analysis**
    - Financial health assessment with scoring
    - Top 3 personalized recommendations
    - Warning signs and positive highlights
    - Fallback rule-based analysis

5. **Interactive AI Chat**
    - Context-aware chatbot
    - Quick question buttons
    - Real-time responses

2.3 User Characteristics

- **Age Range:** All age groups
- **Technical Skills:** Basic to intermediate
- **Financial Literacy:** Beginner to intermediate
- **Primary Goal:** Track expenses and improve financial habits

2.4 Constraints

- Requires Python 3.8+
- Internet needed for AI features
- Modern browser required
- Max CSV size: 16MB
- Serverless file system is temporary in deployed environments
- Session-based storage (data lost on restart)

2.5 Dependencies

- **Groq API:** Valid API key required for AI
- **Libraries:** Flask, Pandas, Groq SDK, Werkzeug
- **Chart.js CDN:** Internet required
- **JavaScript:** Must be enabled

---

3. Functional Requirements

3.1 User Interface

FR-UI-001: Landing Page

- Display logo, tagline, and hero section
- Show three feature cards
- "Get Started Now" CTA button
- Developer credits in footer

FR-UI-002: Navigation Routes: `/` (home), `/input` (data entry), `/dashboard` (results), `/chat-page` (AI chat)

3.2 Data Input

FR-DI-001: Manual Entry

- Form fields: Income (required), Rent, Food, Transportation, Shopping, Entertainment, EMI, Utilities, Healthcare, Others (all optional)
- Validation: All numeric, non-negative
- Process: Validate → Calculate → Store in session → Redirect to dashboard

FR-DI-002: CSV Upload

- Required format: Category, Amount columns
- Validation: .csv extension, ≤16MB, valid columns
- Process: Upload → Parse with Pandas → Categorize → Calculate → Store → Redirect

FR-DI-003: Input Method Toggle

- Switch between "Manual Entry" and "Upload CSV"
- Smooth animations

3.3 Data Processing

FR-DP-001: Financial Calculations

- Total Expenses = Σ(all categories)
- Savings = Income - Total Expenses
- Savings Rate = (Savings/Income) × 100
- Expense Percentages = (Amount/Income) × 100

FR-DP-002: Financial Health Score Four-factor scoring (0-100):

- **Savings Rate (40 pts):** ≥30%=40, ≥20%=30, ≥10%=20, ≥5%=10
- **Positive Savings (20 pts):** >0=20, ≥-10%=10
- **EMI Burden (20 pts):** 0%=20, ≤30%=15, ≤40%=10, ≤50%=5
- **Balanced Expenses (20 pts):** Max≤30%=20, ≤40%=15, ≤50%=10, ≤60%=5

Status: 80-100=Excellent, 60-79=Good, 40-59=Fair, 0-39=Needs Improvement

3.4 Dashboard

FR-DV-001: Summary Cards Display: Monthly Income, Total Expenses, Monthly Savings, Savings Rate (all colour-coded)

FR-DV-002: Charts

- Pie chart: Expense distribution with tooltips
- Bar graph: Category breakdown with Y-axis formatting

FR-DV-003: Navigation Buttons: "Chat with AI" (primary), "New Analysis" (secondary)

3.5 AI Analysis

FR-AI-001: AI-Powered Analysis

- Model: LLaMA 3.1-8B-Instant via Groq API
- Input: Financial data from session
- Output: Health assessment, top 3 recommendations, warnings, positive highlights
- Parameters: temperature=0.7, max_tokens=1500

FR-AI-002: Fallback Analysis

- Triggers: Missing API key, connection failure, timeout
- Rule-based analysis covering savings rate, EMI, expenses, emergency fund

FR-AI-003: Display

- "Analyze My Finances with AI" button
- Loading spinner
- Health score circle with colour
- Formatted analysis text

3.6 AI Chat

FR-CH-001: Chat Interface

- Welcome message with 4 quick question buttons
- User/AI message bubbles (right/left aligned)
- Typing indicator with animation
- Scrollable history

FR-CH-002: Chat Processing

- Add financial context to queries
- Model: LLaMA 3.1-8B-Instant
- Parameters: temperature=0.8, max_tokens=800
- Indian context with ₹ currency

FR-CH-003: Chat Controls

- Disable input during processing
- Auto-scroll to new messages
- Enter key to send
- Auto-focus on input

3.7 Use Cases

Use Case 1: Analyze Finances

1. User navigates to landing page → Clicks "Get Started Now"
2. Enters data (manual or CSV) → Submits
3. System validates → Calculates → Stores → Redirects to dashboard
4. User views charts and cards → Clicks "Analyze with AI"
5. System queries Groq API → Displays score and recommendations

Use Case 2: Chat with AI

1. User clicks "Chat with AI" from dashboard
2. Types question or clicks quick button → Sends
3. System adds context → Queries Groq API → Displays response
4. User asks follow-up questions

---

4. Non-Functional Requirements

4.1 Performance

- Page load: ≤2s
- Dashboard render: ≤3s
- CSV process: ≤5s (16MB)
- AI analysis: ≤10s
- Chat response: ≤8s
- Memory: ≤500MB
- CPU: ≤50%

4.2 Security

- API keys in .env (never in code/repo)
- Server-side input validation
- secure_filename() for uploads
- .csv only, ≤16MB
- Session encryption
- No persistent data storage
- No logging of financial data

4.3 Reliability

- Graceful AI degradation (fallback to rules)
- User-friendly error messages
- No crashes on invalid input
- Handle edge cases (zero income, negative savings)
- 99% uptime target

4.4 Usability

- Intuitive navigation
- First-time use: <5 minutes
- Responsive design (mobile/desktop)
- Loading indicators
- Clear error messages
- Indian context (₹ currency)

4.5 Maintainability

- Modular code (3 main modules)

- Function docstrings
- Consistent naming
- Environment-based config

4.6 Portability

- Browser-based (no OS-specific UI)
- Python 3.8+ compatible
- Cross-browser support
- Simple deployment: `python app.py`

---

5. System Architecture

5.1 Three-Tier Architecture

Presentation Layer (Frontend)

- HTML5, CSS3, JavaScript, Chart.js
- Files: index.html, input.html, dashboard.html, chat.html
- Responsibilities: UI rendering, input collection, visualization, AJAX

Application Layer (Backend)

- Flask framework (app.py)
- Routes: `/`, `/input`, `/process-manual`, `/process-csv`, `/dashboard`, `/analyze`, `/chat-page`, `/chat`
- Session management, request/response handling

Business Logic Layer

- **data_processor.py:** process_manual_data(), process_csv_data(), calculate_financial_health()
- **ai_analyzer.py:** get_client(), analyze_finances(), chat_with_ai(), get_rule_based_analysis()

5.2 Data Flow

```
User Input → Flask Server → Data Processor → Session Storage
                                  ↓
                          AI Analyzer → Groq API
                                  ↓
                          Dashboard ← Chart.js
```

5.3 Component Diagram

```
┌──────────────────────────────────────┐
│         Presentation Layer              │
│  [index] [input] [dashboard] [chat] │
└──────────────────────────────────────┘
                 │
                 ↓
```

```
┌────────────────────────────────────┐
│    Flask Application (app.py)        │
│  - Routing, Session, File Handling   │
└──────┬─────────────┬───────────────┘
       ↓             ↓
┌──────────────┐ ┌─────────────────┐
│data_processor │ │   ai_analyzer    │
│- CSV parsing  │ │- Groq API calls  │
│- Calculations │ │- Rule-based      │
└──────────────┘ └────────┬────────┘
                          ↓
                 ┌───────────────┐
                 │   Groq AI API  │
                 └───────────────┘
```

5.4 Session Storage

```
session['financial_data'] = {
    'income': float,
    'expenses': {'Rent': float, 'Food': float, ...},
    'total_expenses': float,
    'savings': float,
    'savings_rate': float,
    'expense_percentages': dict,
    'month': string,
    'timestamp': string,
    'source': 'manual' or 'csv'
}
```

---

6. Appendices

6.1 Technology Stack

Component	Technology	Version
Backend	Flask	3.0.0
AI Service	Groq API (LLaMA 3.1-8B)	-
Data Processing	Pandas	2.1.0
File Security	Werkzeug	3.0.0
Frontend	HTML5/CSS3/JS	-
Visualization	Chart.js	Latest
Runtime	Python	3.8+

6.2 API Endpoints

Route	Method	Purpose
`/`	GET	Landing page
`/input`	GET	Input page
`/process-manual`	POST	Process manual data
`/process-csv`	POST	Process CSV file
`/dashboard`	GET	Display dashboard
`/analyze`	POST	AI analysis
`/chat-page`	GET	Chat interface
`/chat`	POST	Chat message

6.3 Environment Setup

Requirements:

- Python 3.8+
- Modern web browser
- Internet (for AI features)

Installation:

```
cd ai-financial-advisor
pip install -r requirements.txt
python app.py
# Access: http://localhost:5000
```

6.4 CSV Template

```
Category,Amount
Income,50000
Rent,15000
Food,8000
Transportation,3000
Shopping,5000
Entertainment,2000
EMI,0
Utilities,2000
Healthcare,1000
Others,500
```

6.5 Error Codes

Code	Message	Resolution

Code	Message	Resolution
400	No file uploaded	Upload valid CSV
400	Wrong file type	Use .csv only
400	No financial data	Enter data first
500	AI API failure	Check API key/retry

6.6 Glossary

Term	Definition
CSV	Comma-Separated Values file
EMI	Equated Monthly Installment
Flask	Python web framework
Groq	AI API provider
LLaMA	Large Language Model by Meta
Savings Rate	% of income saved

6.7 Acknowledgments

Project by:

- Mukul Bhardwaj

Special Thanks:

- Groq AI
- Open-source community

Document Revision History

Version	Date	Changes
1.0	11 Jan 2026	Initial document

Date: 11 January 2026

End of Document