FINMENTOR AI - MULTIAGENT FINANCIAL ADVISORY PLATFORM

SOFTWARE REQUIREMENTS SPECIFICATIONS

Submitted by

2448046: Roshan Varghese

2448059: Varun Alfred Dsouza

Project Guide: Dr. Saleema J S

For the course

Course Code: MDS581A

Course Name: PROJECT-II

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

This Software Requirements Specification (SRS) defines the functional and non-functional requirements for the Minimum Viable Product (MVP) of FinMentor AI, a simplified financial advisory platform powered by multi-agent AI architecture. This MVP focuses on delivering core educational and advisory features within a 2-month development timeline.

Target Audience: Solo developer/small development team, project stakeholders, and early testers.

1.2 Scope

The system includes:

- Text-based conversational AI financial advisory
- Basic multi-agent system with 4 core agents (Educational, Market Data, Portfolio Builder, Query Router)
- Simple portfolio allocation using Equal Weight strategy (5-8 stocks)
- Real-time market data integration via Yahoo Finance API
- User authentication (email/password only)
- Conversation history and basic context retention
- Web interface
- Static educational content library (manually curated)
- Google Gemini LLM integration

1.3 Definitions and Acronyms

Term	Definition
DSPy	Declarative Self-improving Python - framework for LLM-based reasoning
RAG	Retrieval-Augmented Generation
LLM	Large Language Model (Google Gemini for MVP)
NSE/BSE	National Stock Exchange / Bombay Stock Exchange (India)
Yahoo Finance	Free API for real-time stock market data
Equal Weight	Simple portfolio strategy with equal allocation to each stock

2. OVERALL DESCRIPTION

2.1 Product Perspective

FinMentor AI MVP is a standalone web application designed to provide basic financial education and advisory services to Indian retail investors. The system uses a simplified multi-agent architecture with Google Gemini as the core LLM, connected to Yahoo Finance for market data.

2.2 Product Functions - Core Features

- User Registration and Authentication: Email/password-based login with JWT tokens
- Conversational AI Interface: Text-based chat for financial queries
- Educational Content: Pre-loaded financial literacy content (stocks, mutual funds, basics)
- Market Data Retrieval: Real-time stock prices, basic fundamentals via Yahoo Finance
- Portfolio Recommendations: Simple equal-weight portfolio generation (5-8 Indian stocks)
- Conversation History: Basic storage and retrieval of past conversations

2.3 User Classes

Primary User: Indian investors (age 20-45) with basic financial knowledge seeking educational content and simple portfolio guidance.

The MVP focuses on serving beginners to intermediate investors only.

2.4 Operating Environment

- Client: modern web browsers (Chrome, Safari, Firefox)
- Backend: Ubuntu/Linux server with Python 3.9+
- Database: PostgreSQL 14+
- API Services: Yahoo Finance API, Google Gemini API
- Deployment: Cloud platform (Heroku, Railway, or Hugging Face Spaces)

2.5 Design and Implementation Constraints

- Development Timeline: Maximum 8 weeks for MVP completion
- Budget: Zero-cost or minimal-cost services only (free tiers)
- APIs: Yahoo Finance (free tier), Google Gemini (free tier with rate limits)
- Platform: Focus on web-first approach

- Data Storage: Simplified schema without vector embeddings
- Agent Count: Maximum 4 agents to reduce complexity
- No Real Trading: Educational and informational purposes only

3. SYSTEM FEATURES AND FUNCTIONAL REQUIREMENTS

3.1 User Authentication System

Basic email/password authentication with JWT token-based session management.

- REQ-AUTH-001: System shall allow users to register with email and password
- REQ-AUTH-002: System shall validate email format and password strength (min 8 characters)
- REQ-AUTH-003: System shall hash passwords using bcrypt before storage
- REQ-AUTH-004: System shall generate JWT tokens upon successful login
- REQ-AUTH-005: System shall support logout functionality
- REQ-AUTH-006: JWT tokens shall expire after 24 hours

3.2 Conversational AI Interface

Text-based chat interface allowing users to ask financial questions and receive AI-powered responses.

- REQ-CHAT-001: System shall provide a text input field for user queries
- REQ-CHAT-002: System shall display AI responses in conversation bubbles
- REQ-CHAT-003: System shall maintain conversation context for current session
- REQ-CHAT-004: System shall route queries to appropriate agent via Query Router
- REQ-CHAT-005: System shall display "typing" indicator during AI processing
- REQ-CHAT-006: Response time shall be under 5 seconds for simple queries
- REQ-CHAT-007: System shall handle errors gracefully with user-friendly messages

3.3 Multi-Agent System (4 Core Agents)

Multi-agent architecture with 4 specialized agents coordinated through basic routing logic.

Query Understanding Agent (Router)

- Purpose: Classify user intent and route to appropriate specialized agent

- Capabilities: Classify queries into categories (educational, market_data, portfolio, general)
- Technology: Prompt-based classification with Google Gemini

Educational Content Agent

- Purpose: Provide financial literacy education to users
- Capabilities: Answer questions about stocks, mutual funds, SIP, trading basics, financial planning
- Content Source: Static knowledge base with 50-100 curated Q&A pairs
- Technology: Simple keyword matching + Gemini generation

Market Data Agent

- Purpose: Retrieve and explain real-time market data
- Capabilities: Fetch stock prices, basic fundamentals (P/E ratio, market cap), 52-week high/low
- Data Source: Yahoo Finance Python library (yfinance)
- Supported Markets: NSE and BSE listed stocks

Portfolio Builder Agent

- Purpose: Generate simple portfolio recommendations
- Strategy: Equal-weight allocation across 5-8 stocks
- Selection Criteria: User risk tolerance (low/medium/high) + sector diversification
- Output: List of stock tickers with equal percentage allocation

Functional Requirements

- REQ-AGENT-001: Query Router shall classify intent with 80%+ accuracy
- REQ-AGENT-002: Educational Agent shall respond to 20+ common financial topics
- REQ-AGENT-003: Market Data Agent shall fetch data from Yahoo Finance API
- REQ-AGENT-004: Market Data Agent shall handle API errors gracefully
- REQ-AGENT-005: Portfolio Builder shall generate 5-8 stock recommendations
- REQ-AGENT-006: Portfolio Builder shall ensure sector diversification (max 2 stocks per sector)
- REQ-AGENT-007: All agents shall respond within 5 seconds
- REQ-AGENT-008: Agent responses shall be in simple English suitable for beginners

3.4 Market Data Integration

Integration with Yahoo Finance API to retrieve real-time stock market data for Indian markets.

- REQ-MARKET-001: System shall fetch real-time stock prices via yfinance library
- REQ-MARKET-002: System shall support NSE and BSE stock symbols
- REQ-MARKET-003: System shall retrieve: current price, P/E ratio, market cap, 52-week high/low
- REQ-MARKET-004: System shall cache market data for 5 minutes to reduce API calls
- REQ-MARKET-005: System shall handle stock symbol not found errors
- REQ-MARKET-006: System shall format numbers in Indian numbering system (lakhs/crores)

3.5 Portfolio Generation System

Simple equal-weight portfolio recommendation engine based on user risk profile.

- REQ-PORT-001: System shall ask user for risk tolerance (Low/Medium/High)
- REQ-PORT-002: System shall generate 5 stocks for Low risk, 6-7 for Medium, 8 for High
- REQ-PORT-003: System shall select from predefined universe of 30-50 quality stocks
- REQ-PORT-004: System shall ensure equal weight allocation (e.g., 20% each for 5 stocks)
- REQ-PORT-005: System shall include maximum 2 stocks from same sector
- REQ-PORT-006: System shall provide basic rationale for each stock selection
- REQ-PORT-007: Portfolio recommendations are for educational purposes only (include disclaimer)

3.6 Conversation History

Basic storage and retrieval of user conversation history.

- REQ-HIST-001: System shall store all user messages and AI responses in database
- REQ-HIST-002: System shall link conversations to user accounts
- REQ-HIST-003: System shall display conversation history on app open
- REQ-HIST-004: System shall maintain last 10 messages as context for current session

- REQ-HIST-005: Users shall be able to view past conversations (last 30 days)
- REQ-HIST-006: Users shall be able to start new conversation thread

4. EXTERNAL INTERFACE REQUIREMENTS

4.1 User Interface Requirements

- REQ-UI-001: Chat interface with message input field and conversation display
- REQ-UI-002: Login/Registration screens with email and password fields
- REQ-UI-003: Simple navigation: Home (Chat), History, Profile
- REQ-UI-004: Clean, minimalist design suitable for financial content
- REQ-UI-005: Responsive design for mobile screens (Android)
- REQ-UI-006: Loading indicators during API calls
- REQ-UI-007: Error messages displayed in red banner at top

4.2 API Interface Requirements

The FastAPI backend shall expose the following endpoints sample system:

Endpoint	Method	Purpose	Auth Required
/api/auth/register	POST	User registration	No
/api/auth/login	POST	User login	No
/api/auth/logout	POST	User logout	Yes
/api/chat/message	POST	Send message to AI	Yes
/api/chat/history	GET	Get conversation history	Yes
/api/portfolio/generate	POST	Generate portfolio	Yes
/api/market/stock/{symbol}	GET	Get stock data	Yes
/api/user/profile	GET	Get user profile	Yes

4.3 External API Integration

4.3.1 Yahoo Finance API

- Library: yfinance Python package (unofficial Yahoo Finance API wrapper)
- Rate Limits: No strict limits but implement 5-minute caching
- Data Retrieved: Stock price, P/E ratio, market cap, 52-week range
- Error Handling: Graceful fallback if API unavailable

4.3.2 Google Gemini API

- Model: gemini-1.5-flash (free tier)
- Rate Limits: 60 requests per minute (free tier)
- Use Cases: Query classification, educational responses, portfolio rationale generation
- Error Handling: Retry logic with exponential backoff

5. NON-FUNCTIONAL REQUIREMENTS

5.1 Performance Requirements

- REQ-PERF-001: API response time shall be < 3 seconds for simple queries
- REQ-PERF-002: System shall handle 10 concurrent users
- REQ-PERF-003: Database queries shall execute in < 500ms
- REQ-PERF-004: App startup time shall be < 3 seconds
- REQ-PERF-005: Market data caching shall reduce repeated API calls by 70%

5.2 Security Requirements

- REQ-SEC-001: Passwords shall be hashed using bcrypt with salt
- REQ-SEC-002: JWT tokens shall be used for authentication
- REQ-SEC-003: HTTPS shall be enforced for all API communication
- REQ-SEC-004: API keys shall be stored in environment variables (not hardcoded)
- REQ-SEC-005: User data shall not be shared with third parties
- REQ-SEC-006: SQL injection prevention via parameterized queries

5.3 Reliability Requirements

- REQ-REL-001: System uptime target: 95%
- REQ-REL-002: Graceful degradation if Yahoo Finance API is unavailable
- REQ-REL-003: Database backups daily
- REQ-REL-004: Error logging to track issues

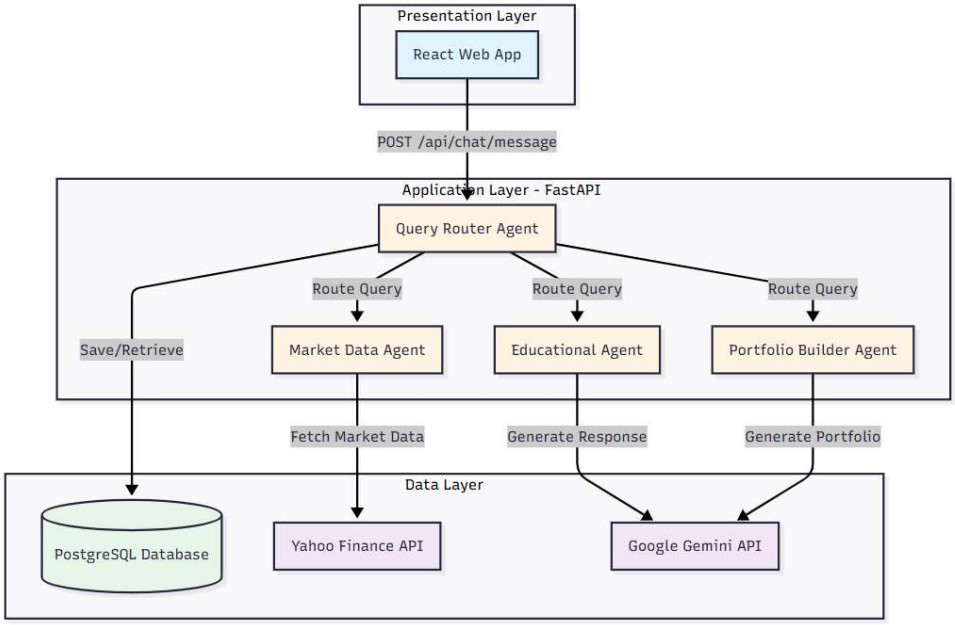
5.4 Maintainability Requirements

- REQ-MAIN-001: Code shall follow PEP 8 style guidelines (Python)
- REQ-MAIN-002: Functions shall have docstrings explaining purpose
- REQ-MAIN-003: API endpoints shall be documented with OpenAPI/Swagger
- REQ-MAIN-004: Configuration shall be externalized via environment variables
- REQ-MAIN-005: Git commits shall have descriptive messages

6. SYSTEM ARCHITECTURE

6.1 High-Level Architecture

Simplified 3-tier architecture:



- Presentation Layer: React web app
- Application Layer: FastAPI backend with 4 AI agents
- Data Layer: PostgreSQL database + External APIs (Yahoo Finance, Gemini)

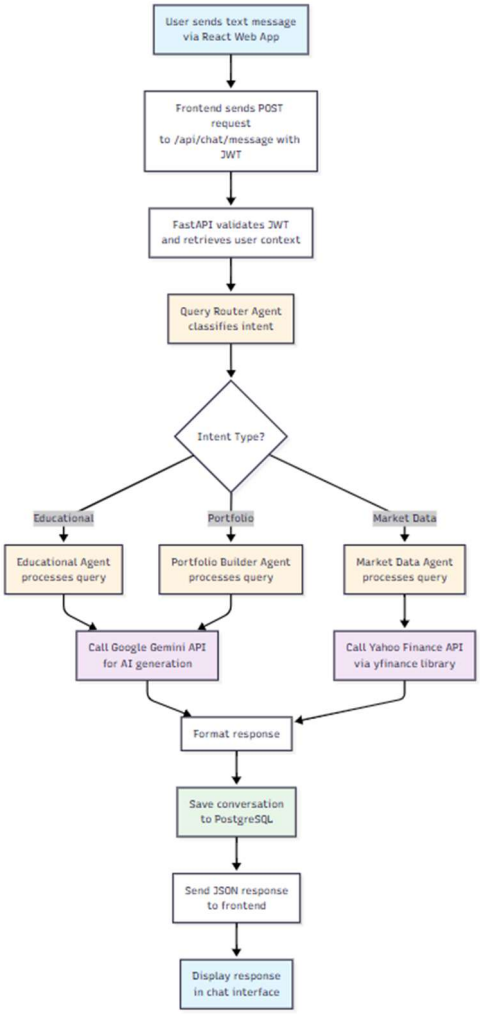
6.2 Technology Stack

Component	**Technology**
Frontend (Web Alternative)	React + Vite + Tailwind CSS
Backend Framework	FastAPI (Python 3.9+)
Database	PostgreSQL 14+
ORM	SQLAlchemy
Authentication	JWT (PyJWT library)
LLM Framework	Google Gemini API (direct REST calls)
Agent Framework	LangChain (simplified - optional)
Market Data	yfinance Python library
Deployment	Heroku / Railway / Render (free tier)
Version Control	Git + GitHub

6.3 Data Flow - Simplified

User Query Processing Flow:

- User sends text message via Web app
- Frontend sends POST request to /api/chat/message with JWT token
- FastAPI backend validates JWT and retrieves user context
- Query Router Agent classifies intent (educational/market_data/portfolio)
- Appropriate specialized agent processes query
- If market data needed: Call Yahoo Finance API via yfinance
- If AI generation needed: Call Google Gemini API
- Response formatted and sent back to frontend
- Frontend displays response in chat interface
- Conversation saved to PostgreSQL database



7. DATABASE SCHEMA

7.1 Table Definitions

Simplified schema with 4 core tables:

Table: users

Column	Type	Constraints	Description
user_id	UUID	PRIMARY KEY	Unique user identifier
email	VARCHAR(255)	UNIQUE, NOT NULL	User email address
password_hash	VARCHAR(255)	NOT NULL	Bcrypt hashed password
created_at	TIMESTAMP	DEFAULT NOW()	Account creation time
last_login	TIMESTAMP	NULL	Last login timestamp

Table: conversations

Column	Type	Constraints	Description
conversation_id	UUID	PRIMARY KEY	Unique conversation ID
user_id	UUID	FOREIGN KEY	Reference to users table
started_at	TIMESTAMP	DEFAULT NOW()	Conversation start time
last_message_at	TIMESTAMP	NULL	Last message timestamp

Table: messages

Column	Type	Constraints	Description
message_id	UUID	PRIMARY KEY	Unique message ID
conversation_id	UUID	FOREIGN KEY	Reference to conversations
sender	VARCHAR(10)	NOT NULL	"user" or "ai"
content	TEXT	NOT NULL	Message text content
agent_used	VARCHAR(50)	NULL	Which agent processed query
created_at	TIMESTAMP	DEFAULT NOW()	Message timestamp

Table: portfolios

Column	Type	Constraints	Description
portfolio_id	UUID	PRIMARY KEY	Unique portfolio ID
user_id	UUID	FOREIGN KEY	Reference to users
risk_level	VARCHAR(20)	NOT NULL	Low/Medium/High
stocks	JSONB	NOT NULL	Array of stock recommendations
created_at	TIMESTAMP	DEFAULT NOW()	Generation timestamp

8. IMPLEMENTATION PLAN

8.1 Week 1-2: Foundation & Backend Setup

- Set up development environment (Python, PostgreSQL, Git)
- Initialize FastAPI project structure
- Implement database models with SQLAlchemy
- Create user registration and authentication endpoints
- Implement JWT token generation and validation
- Set up Google Gemini API integration
- Basic error handling and logging
- Deploy backend to Heroku/Railway for testing

8.2 Week 3-4: Agent Development

- Implement Query Router Agent with intent classification
- Build Educational Content Agent with static knowledge base
- Create Market Data Agent with yfinance integration
- Develop Portfolio Builder Agent with equal-weight logic
- Test agent coordination and routing
- Implement conversation context management
- Add market data caching mechanism

8.3 Week 5-6: Frontend Development

- Choose platform: React (Web)
- Build authentication screens (login/register)
- Create chat interface with message input
- Implement API integration with backend
- Build conversation history view

- Add loading states and error handling
- Implement JWT storage and session management
- Basic UI/UX polish

8.4 Week 7: Integration & Testing

- End-to-end testing of all user flows
- Test all 4 agents with real queries
- Verify market data accuracy
- Test portfolio generation with different risk levels
- Security testing (authentication, authorization)
- Performance testing and optimization
- Fix critical bugs

8.5 Week 8: Deployment & Documentation

- Final deployment to production environment
- Create user documentation/help guide
- Add legal disclaimers (investment advice warning)
- Set up basic analytics/monitoring
- Create demo video
- Conduct user acceptance testing with 3-5 beta users
- Prepare handover documentation

9. TESTING STRATEGY

9.1 Unit Testing

- Authentication functions (password hashing, JWT generation)
- Query Router classification logic
- Portfolio generation algorithm
- Database CRUD operations
- Target: 60% code coverage minimum

9.2 Integration Testing

- API endpoint testing (all endpoints)
- Yahoo Finance API integration
- Google Gemini API integration
- Database connection and queries
- Frontend-backend communication

9.3 User Acceptance Testing

Test Scenarios:

- New user registration and first login
- Ask educational question and receive response
- Request stock price information
- Generate portfolio for different risk levels
- View conversation history
- Logout and login again (session persistence)

9.4 Testing Tools

- Backend: pytest for Python unit tests
- API Testing: Postman or Thunder Client
- Frontend: React Testing Library
- Manual Testing: Physical Android device or browser

10. RISKS AND MITIGATION STRATEGIES

Risk	Probability	Impact	Mitigation Strategy
Timeline overrun	High	High	Cut features aggressively; focus on 3 agents instead of 4 if needed
Google Gemini API rate limits	Medium	High	Implement request queuing and caching; upgrade to paid tier if needed
Yahoo Finance API reliability	Medium	Medium	Add retry logic; prepare fallback message for users
Agent response quality	Medium	Medium	Extensive prompt engineering and testing; use pre-defined responses
Database performance	Low	Medium	Add indexes on frequently queried columns; limit conversation history
Security vulnerabilities	Low	High	Follow OWASP guidelines; use established libraries (bcrypt, PyJWT)

11. EXPLORATORY DATA ANALYSIS (EDA)

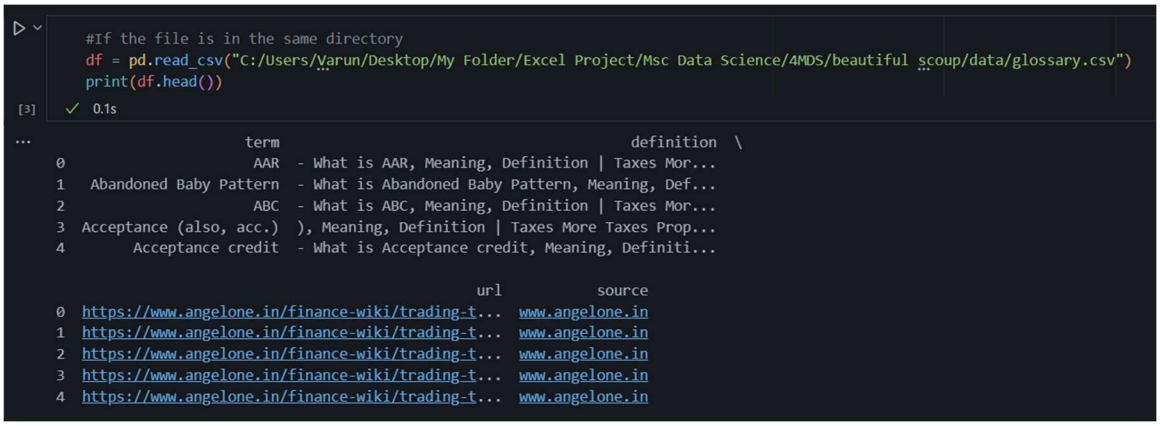
11.1 Data Structure and Integrity

Basic Information

- Data:The data consists of scraped data from angelone website
- Total Entries: 538
- Total Columns: 4 which include term, definition, url, source
- Term : The column consists of all the technical and fundamental terms related to capital market.
- Definition: Refers to meaning of each of the term with respect to the term.
- URL: This consists of Uniform Resource Locator for each of the term in the angelone website.
- Source: The common webpage of angelone site.

Missing and Duplicate Values

- Missing Values: None. All 538 entries are complete across all 4 columns, which indicates a robust initial scrape.
- Duplicate Rows: The initial dataset contained duplicates, which were successfully removed.
- Source Frequency: All 538 entries originate from the same source: www.angelone.in



```
#If the file is in the same directory
df = pd.read_csv("C:/Users/Varun/Desktop/My Folder/Excel Project/Msc Data Science/4MDS/beautiful scoup/data/glossary.csv")
print(df.head())
```

[3] ✓ 0.1s

```
...                    term                                         definition  \
0                       AAR  - What is AAR, Meaning, Definition | Taxes Mor...
1   Abandoned Baby Pattern  - What is Abandoned Baby Pattern, Meaning, Def...
2                       ABC  - What is ABC, Meaning, Definition | Taxes Mor...
3  Acceptance (also, acc.)  ), Meaning, Definition | Taxes More Taxes Prop...
4        Acceptance credit  - What is Acceptance credit, Meaning, Definiti...

                                                 url          source
0  https://www.angelone.in/finance-wiki/trading-t...  www.angelone.in
1  https://www.angelone.in/finance-wiki/trading-t...  www.angelone.in
2  https://www.angelone.in/finance-wiki/trading-t...  www.angelone.in
3  https://www.angelone.in/finance-wiki/trading-t...  www.angelone.in
4  https://www.angelone.in/finance-wiki/trading-t...  www.angelone.in
```

11.2 Content Characteristics Analysis

To understand the linguistic properties of the glossary, the length of the terms and definitions was analyzed.

Summary Statistics for Text Length

The dataset contains 538 entries. The financial terms are quite concise, with an average length of about 15 characters and a median of 13 characters; however, they range widely, from 2 to 54 characters. In contrast, the definitions are consistently detailed and substantial, averaging 784 characters with a median of 801 characters. This tight clustering is demonstrated by a low standard deviation (≈126) and a narrow interquartile range, meaning 50% of all definitions fall between 736 and 859 characters, confirming a highly standardized, long-form approach to the content.

```
=== SUMMARY STATISTICS ===
       term_length  definition_length
count   538.000000         538.000000
mean     14.836431         784.050186
std       7.876326         126.113452
min       2.000000         216.000000
25%      10.000000         736.250000
50%      13.000000         801.000000
75%      19.000000         859.000000
max      54.000000        1076.000000
```

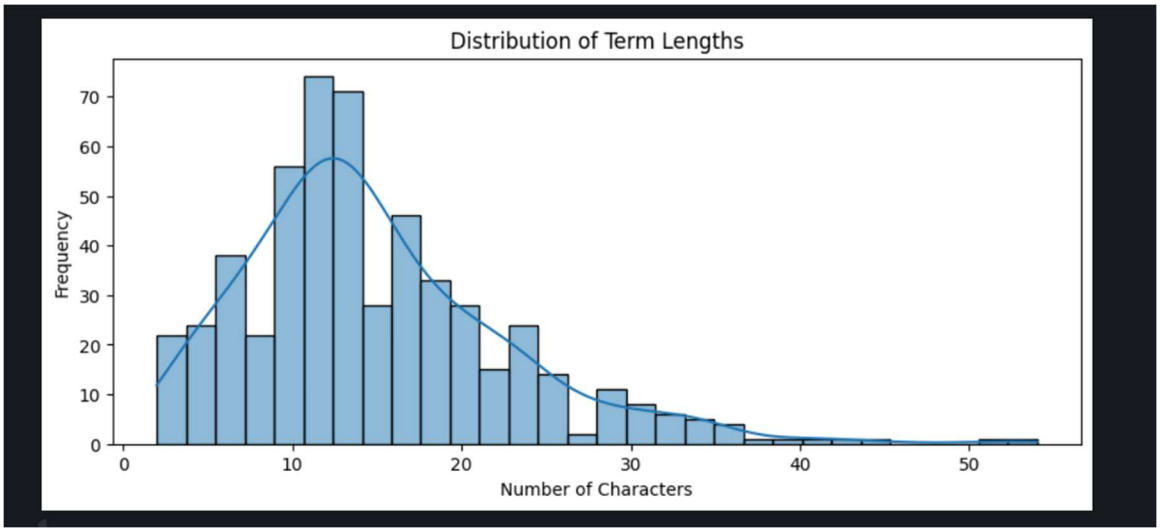
1. Term Length: Terms are relatively concise, averaging about 15 characters, but range significantly from very short acronyms (2 characters) to long phrases (up to 54 characters).

2. Definition Length: Definitions are substantial, with an average length of approximately 750 characters. The tight standard deviation () suggests that definitions are consistently detailed and uniform in their explanation length.

Distribution of Text Lengths

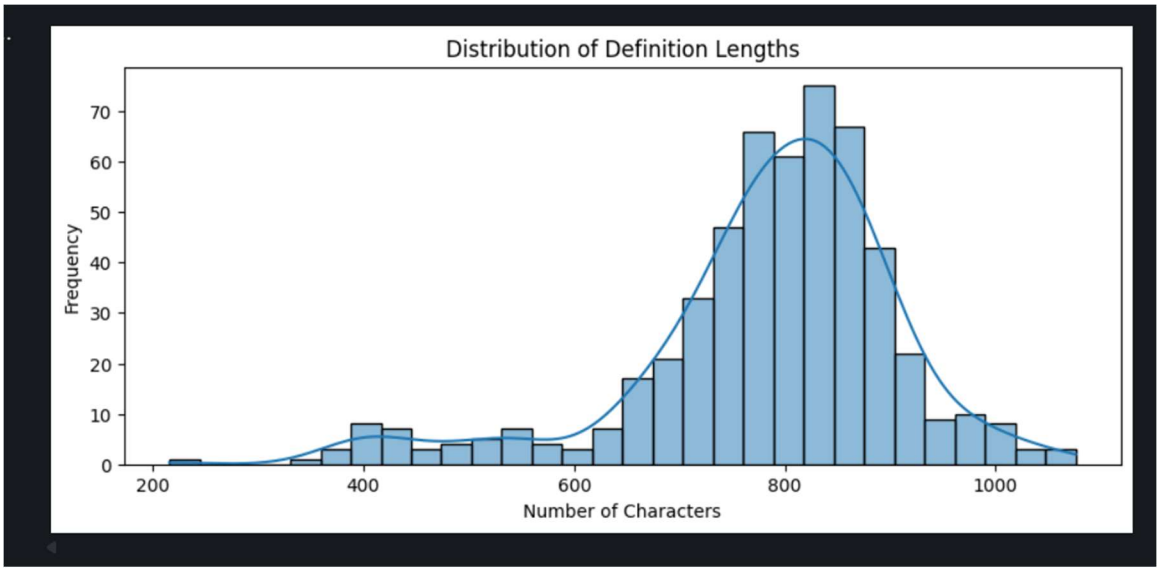
Term Length Distribution

- The distribution of Term Lengths is slightly right-skewed, peaking sharply between 10 and 15 characters. This confirms that most glossary entries are short to medium-length phrases or single words, with fewer entries being very long or very short (acronyms).



Definition Length Distribution

- The distribution of Definition Lengths is approximately normal (bell-shaped) and tightly clustered, peaking around 800-850 characters. This is a strong indicator of content consistency, suggesting a standardized, detailed template or approach was used when writing the definitions on the source website.

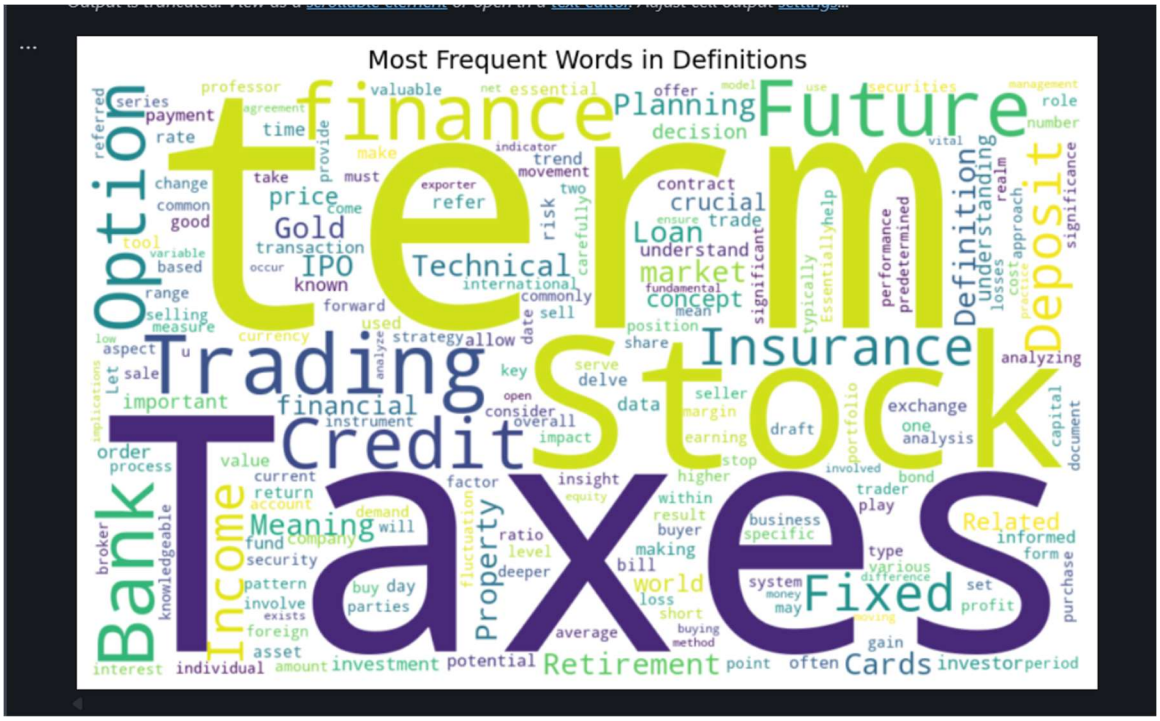


11.3 Lexical Analysis: Most Frequent Words

The word cloud generated from the combined text of all definitions highlights the core themes and topics covered in the glossary.

Most Prominent Words:

- TERM (appears largest, likely due to terms being referenced within definitions)
- FINANCE
- TRADING
- TAXES
- CREDIT
- INSURANCE
- STOCK
- FIXED
- DEPOSIT
- FUTURE
- OPTION
- BANK



The most frequent words confirm that the glossary covers a broad spectrum of the financial world, including core concepts like stock and trading, investment products (option, deposit, future), and fundamental financial instruments (credit, insurance, fixed). The high frequency of Taxes suggests that regulatory and fiscal aspects are heavily emphasized in the definitions.

12. APPENDIX

12.1 Sample Stock Universe (30-50 Quality Stocks)

Portfolio Builder will select from this predefined list:

Sector	Example Stocks (NSE)	Risk Level
Banking	HDFCBANK, ICICIBANK, KOTAKBANK	Low-Medium
IT Services	TCS, INFY, WIPRO	Medium
FMCG	HINDUNILVR, ITC, NESTLEIND	Low
Pharma	SUNPHARMA, DRREDDY	Medium
Auto	MARUTI, TATAMOTORS	Medium-High
Energy	RELIANCE, ONGC	Medium
Telecom	BHARTIARTL	Medium
Metals	TATASTEEL, HINDALCO	High

12.2 Sample Educational Topics

Educational Agent knowledge base will cover:

- What are stocks and how do they work?
- Difference between NSE and BSE
- What is a mutual fund vs. ETF?
- Understanding P/E ratio and market cap
- What is SIP (Systematic Investment Plan)?
- Risk vs. Return basics
- Diversification principles
- Long-term investing vs. trading
- Understanding demat accounts
- Tax implications of equity investments (basic)
- What is a 52-week high/low?
- Bull market vs. Bear market
- What are dividends?
- How to read annual reports (basic)
- Common investing mistakes for beginners

12.3 Development Resources

Useful resources for implementation:

- FastAPI Documentation: https://fastapi.tiangolo.com
- yfinance Library: https://github.com/ranaroussi/yfinance
- Google Gemini API Docs: https://ai.google.dev/docs
- SQLAlchemy ORM: https://docs.sqlalchemy.org
- LangChain: https://python.langchain.com

REFERENCES

[1] Choi, C., Kwon, J., Ha, J., Choi, H., Kim, C., Lee, Y., Sohn, J., & Lopez-Lira, A. (2025). FinDER: Financial dataset for question answering and evaluating retrieval-augmented generation. *arXiv preprint arXiv:2504.15800*. https://doi.org/10.48550/arXiv.2504.15800

[2] Xiao, Y., Sun, E., Luo, D., & Wang, W. (2024). TradingAgents: Multi-agents LLM financial trading framework. *arXiv preprint arXiv:2412.20138*. https://doi.org/10.48550/arXiv.2412.20138

[3] Angel One. (2025). *Online trading & stock broking in India*. https://www.angelone.in

[4] National Stock Exchange of India. (2025). *NSE - National Stock Exchange of India Ltd*. https://www.nseindia.com