Optivest

Portfolio Optimization and Investment Recommendation System

Submitted By:

Bhoomi Gupta 234013

Prabhleen Kaur 234021

Teacher: Dr. Supreet Kaur Sahi

Acknowledgement

Apart from the efforts of the team, the success of any project depends largely on the encouragement and guidance of many others. We take this opportunity to express our gratitude to those who have been instrumental in the successful completion of this project. The completion of any interdisciplinary project relies on cooperation, coordination, and the combined efforts of various sources of knowledge. We are eternally grateful to our teacher, **Dr. Supreet Kaur Sahi**, for her unwavering willingness to provide valuable advice and direction, under which we executed this project. Her constant guidance and eagerness to share her vast knowledge deepened our understanding of the project and its implications, helping us to complete the assigned tasks effectively.

Bhoomi Gupta 234013

Prabhleen Kaur 234021

Certificate

This is to certify that the Software Engineering project report entitled "**Optivest: Portfolio Optimization and Investment recommendation system**" is the work carried out by **Bhoomi Gupta & Prabhleen Kaur**, students of B.Sc.(H) Computer Science (V) Semester, Sri Guru Gobind Singh College of Commerce, University of Delhi, under the supervision of **Dr. Supreet Kaur Sahi**, Professor, Department of Computer Science, Sri Guru Gobind Singh College.

This report has not been submitted to any other organization/institution for the award of any other degree or diploma.

Dr. Supreet Kaur Sahi

(Project Guide)

Index

S. No.	Task	Page No.
1	Problem Statement	5
2	Process Model	6
3	SRS	10
4	Use Case	12
5	Use Case Description	12
6	Product Functions	14
7	Context Level Diagram	15
8	DFD Level - 1	16
9	DFD Level - 2	16
10	Sequence Diagrams	20
11	Data Dictionary	23
12	ER Diagram	24
13	Database Design	25
14	Project Scheduling	27
15	Computing FP	29
16	Risk Analysis	30
17	Architectural Design	32
18	Pseudocode of a Module	35
19	Python Code of a Module	41
21	Cyclomatic Complexity	43
22	Test Cases	45
23	Conclusion	46
24	Future Scope	47
25	Appendix	48

Problem Statement

In today’s financial landscape, investors face critical challenges in managing their portfolios effectively while balancing risk and return. The complexity of market fluctuations, lack of accessible analytical tools, and limited understanding of financial risks make it difficult for individuals to make informed investment decisions. Traditional portfolio management methods, such as manual record-keeping, basic spreadsheets, or ad-hoc decisions, are inefficient, error-prone, and provide little support for predictive analysis.

This project addresses the problem of enabling secure, data-driven, and user-friendly portfolio management by integrating risk prediction and visualization tools. It aims to overcome issues such as lack of risk awareness, ineffective portfolio diversification, and poor decision-making by providing an automated, analytics-driven solution that empowers investors.

Proposed System

The proposed system will focus on:

1. **User Authentication & Access Control:** Ensuring only authorized users can access and manage their investment portfolios.
2. **Portfolio Management:** Allowing users to add, update, and track their investments in different asset classes.
3. **Risk Prediction:** Leveraging analytical models to assess portfolio risk, volatility, and exposure to market fluctuations.
4. **Data Security:** Safeguarding sensitive financial data through encryption and secure storage mechanisms.
5. **Visualization:** Presenting portfolio performance and risk insights through interactive dashboards, charts, and reports for better decision-making.

By integrating these features, the system aims to provide investors with actionable insights, minimize financial risks, and enhance decision-making capabilities. Ultimately, it promotes financial literacy, builds trust, and supports secure, data-driven investment management in an increasingly dynamic market.

Process Model

Introduction

The development of the *Investment Recommendation and Portfolio Optimization System* follows the **Agile Software Development Model**, which emphasizes iterative progress, continuous user involvement, and flexible adaptation to change. Since the project involves financial data analysis, real-time testing, and optimization modules, an adaptive and incremental approach is best suited to ensure continuous improvement and accuracy.

Justification for Choosing Agile Model

The Agile model was selected because it allows for **incremental development**, **frequent evaluation**, and **continuous enhancement** of each system component. The project requires iterative testing of data analytics algorithms, risk prediction methods, and visualization tools. Using Agile ensures that user feedback can be quickly incorporated to improve the functionality and reliability of the system.

Key reasons for choosing Agile include:

- Frequent interaction with end-users or investors to validate system features.
- Flexibility to accommodate changing data sources or analytical models.
- Early demonstration of working prototypes for feedback and refinement.
- Reduced overall project risk through incremental testing and review.

Phases of the Agile Process Model

The development process is divided into several **iterative and incremental phases** as described below:

1. Requirement Analysis

- Gather system requirements from potential users, financial analysts, and domain experts.
- Define both functional and non-functional requirements such as data processing, visualization, risk analysis, and portfolio optimization.

- Deliverable: *Software Requirement Specification (SRS) Document.*

2. System Design

- Design the overall architecture of the system, including data flow, database schema, and process interaction.
- Identify major modules like Data Input, Risk Analysis, Optimization, and Visualization.
- Deliverable: *System Architecture Diagram, ER Diagram, and DFDs.*

3. Iteration 1 – Data Collection and Preprocessing

- Develop a module to collect and clean historical financial data.
- Integrate APIs (like Yahoo Finance) for stock data extraction.
- Deliverable: *Cleaned and structured dataset.*

4. Iteration 2 – Portfolio Optimization Module

- Implement **Markowitz's Modern Portfolio Theory (MPT)** for optimizing risk and return.
- Allow user input for risk tolerance levels (Conservative, Balanced, Aggressive).
- Deliverable: *Portfolio Optimization Engine.*

5. Iteration 3 – Visualization and Reporting

- Create visual dashboards for portfolio performance and the efficient frontier.
- Generate summary reports displaying expected returns and associated risks.
- Deliverable: *Interactive Visualizations and Reports.*

6. Testing and Validation

- Perform **unit testing**, **integration testing**, and **performance evaluation**.
- Ensure correctness of results and verify system stability under different datasets.
- Deliverable: *Test Report.*

7. Deployment and Feedback

- Deploy the system prototype and collect user feedback from investors and analysts.
- Implement suggested improvements in the next Agile sprint.
- Deliverable: *Refined Prototype.*

8. Maintenance

- Continuously update the system to handle new market data and financial indicators.
- Maintain documentation and ensure model adaptability.
- Deliverable: *Enhanced and Updated System Version.*

Advantages of the Agile Model

- **Flexibility:** Adapts to evolving requirements and data changes.
- **User-Centric Development:** Continuous feedback ensures practical usability.
- **Early Delivery:** A functional version is available early in the lifecycle.
- **Improved Quality:** Frequent testing enhances reliability and performance.
- **Incremental Progress:** Modules are developed and refined step-by-step.

Requirement Analysis

1) Software Requirement Specification
2) Use Case
3) Product Functions
4) Data Flow
5) Sequence Diagrams
6) Data Dictionary
7) ER Diagram
8) Database Design

Software Requirements

The *Investment Recommendation and Portfolio Optimization System* is designed to assist investors in making data-driven decisions by providing optimized portfolios and risk-aware investment recommendations. From a product perspective, the system focuses on delivering a seamless, interactive, and intelligent investment planning experience.

The product allows users to log in securely and input their investment preferences, such as risk tolerance and expected return. It automatically collects and processes historical stock market data using APIs like Yahoo Finance. Through its backend analytics engine, it calculates essential financial indicators including average returns, volatility, covariance, and correlation between assets. Using Markowitz's Modern Portfolio Theory, the system identifies the optimal combination of assets that minimizes risk while maximizing expected return.

Users can view the results through interactive charts and visualizations such as the efficient frontier, portfolio allocation charts, and risk-return plots. The product also integrates crash risk metrics like Value-at-Risk (VaR) and Maximum Drawdown to alert investors about potential downside risks. Based on this analysis, it recommends suitable investment portfolios categorized as conservative, balanced, or aggressive, depending on the user's chosen profile.

The system is designed to be fast, scalable, and secure, ensuring accurate results and smooth performance even when processing large datasets. It emphasizes usability through a clean and intuitive interface, making it accessible to both novice and experienced investors. The platform can be deployed on any standard operating system and integrates with popular Python-based data science libraries such as Pandas, NumPy, and Matplotlib for computation and visualization.

Overall, the product aims to deliver an intelligent, evidence-based decision-support tool that enhances investor confidence, reduces emotional bias, and provides continuous insights into portfolio performance and market risk.

System Interfaces

The *Investment Recommendation and Portfolio Optimization System* consists of multiple system interfaces that ensure smooth interaction between the user, the system modules, and external components. Each interface is designed to provide clarity, security, and efficiency in data flow and functionality.

The **user interface** serves as the main point of interaction for investors. It allows users to register, log in, and manage their profiles. Once authenticated, users can input their investment preferences such as risk tolerance and expected returns. The interface is designed to be user-friendly, with an intuitive dashboard that displays visual analytics, charts, and portfolio summaries. Graphs like the efficient frontier, risk-return plots, and allocation pie charts help users easily interpret investment recommendations.

The **data interface** handles communication with external financial data sources. It connects to APIs such as Yahoo Finance or Alpha Vantage to fetch live or historical stock data. The data is imported in formats like CSV or JSON, cleaned, and stored in the system's internal database. This interface ensures that the system has continuous access to updated market data required for accurate analysis and recommendations.

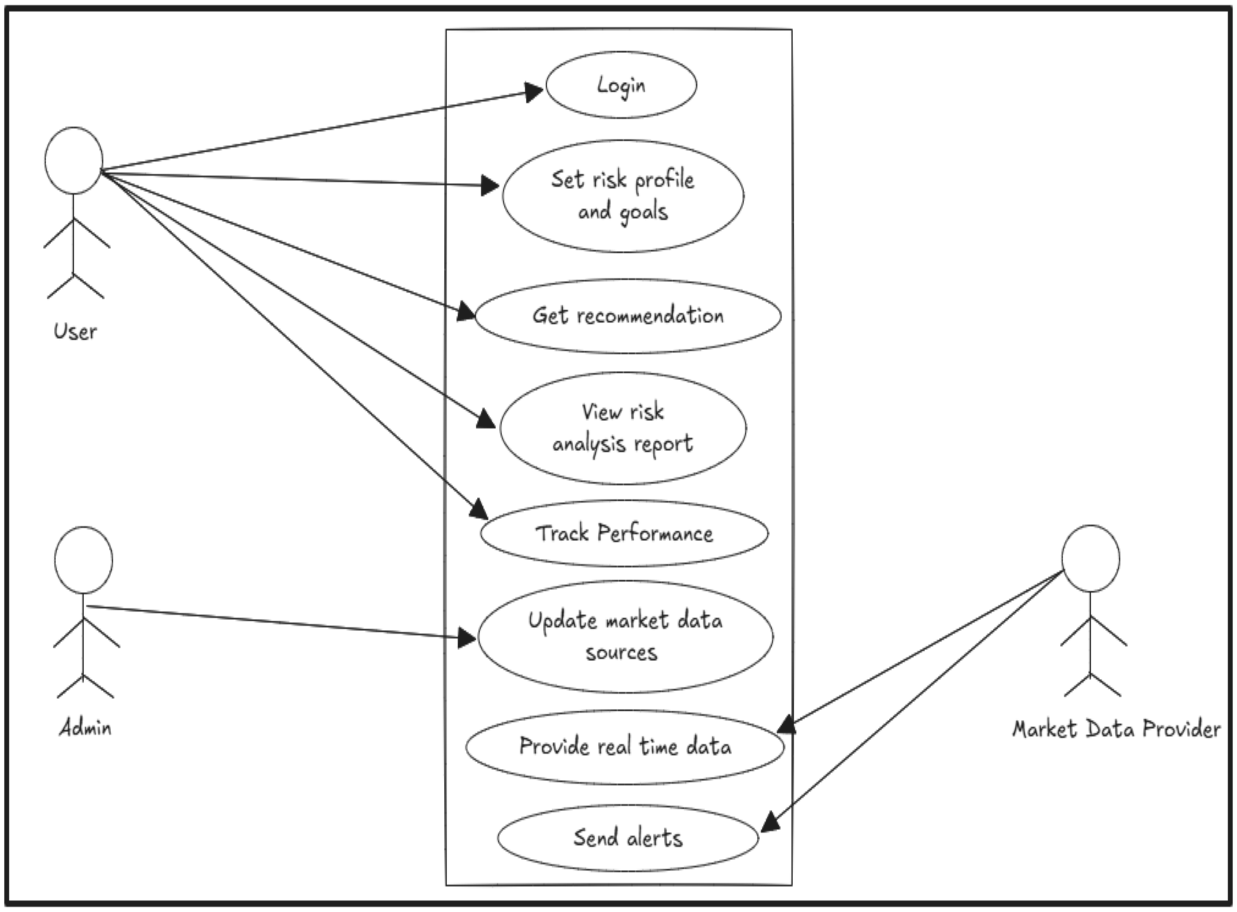
The **database interface** manages interaction between the application and its underlying database, which stores user information, stock data, risk metrics, and portfolio records. SQL queries or ORM (Object Relational Mapping) functions handle data retrieval, storage, and updates. The database interface ensures data integrity, security, and efficient access during processing and report generation.

The **analytics and computation interface** connects the front-end modules with backend algorithms. It takes input data from the user and processed financial data from the database to perform calculations such as expected return, portfolio variance, and Value-at-Risk. This interface acts as a bridge between user commands and computational models implemented using Python libraries like NumPy and Pandas.

The **reporting and visualization interface** allows the system to display results in a clear, graphical format. It generates real-time visual reports that summarize risk, return, and portfolio composition. Libraries like Matplotlib or Plotly are integrated to create interactive dashboards and exportable PDF reports for investors.

The **security interface** ensures safe user authentication, data encryption, and controlled access. It prevents unauthorized data manipulation and ensures that user credentials and financial information remain confidential.

Use Case Diagram



Description

1. **Login**
   The user starts by logging into the system using secure authentication credentials. The system validates the entered username and password against the stored user records in the database. On successful authentication, the user gains access to personalized features such as portfolio tracking and investment recommendations. This ensures that all activities are user-specific and securely managed.

2. **Set Risk Profile and Goals**
   Once logged in, the user defines their investment objectives and risk tolerance level—such as conservative, balanced, or aggressive. The system collects additional details like investment horizon, target returns, and preferred sectors. These inputs are stored and used to tailor portfolio optimization strategies according to the user's preferences.
3. **Get Recommendation**
   Based on the user's defined risk profile and market data, the system applies Markowitz's Modern Portfolio Theory to generate an optimized investment portfolio. The algorithm computes the best asset allocation that balances return and risk, displaying the expected performance and asset weights for the recommended portfolio.
4. **View Risk Analysis Report**
   The system provides a detailed risk analysis report highlighting key financial indicators, including Value-at-Risk (VaR), Sharpe Ratio, and Maximum Drawdown. This report helps investors understand potential losses under adverse market conditions and assess the stability of their investment portfolio.
5. **Track Performance**
   The user can view real-time updates on the performance of their portfolio through charts and analytics dashboards. The system fetches live market data to calculate updated returns and volatility, allowing investors to monitor deviations from expected performance and make timely adjustments.
6. **Update Market Data Sources**
   This function is performed by the admin, who ensures the system has access to the latest market data feeds. The admin can modify or verify data API sources, ensuring that the information used for analysis is accurate and current.
7. **Provide Real-Time Data**
   The Market Data Provider continuously supplies live stock and index data to the system. This enables accurate portfolio calculations, efficient frontier plotting, and risk modeling. The data flow between the provider and the system ensures high reliability and minimal latency in updates.
8. **Send Alerts**
   The system automatically sends alerts and notifications to users regarding market fluctuations, portfolio performance, or threshold breaches (e.g., if risk exceeds the user's tolerance). These alerts can be delivered through email or dashboard notifications, enhancing user engagement and decision-making.

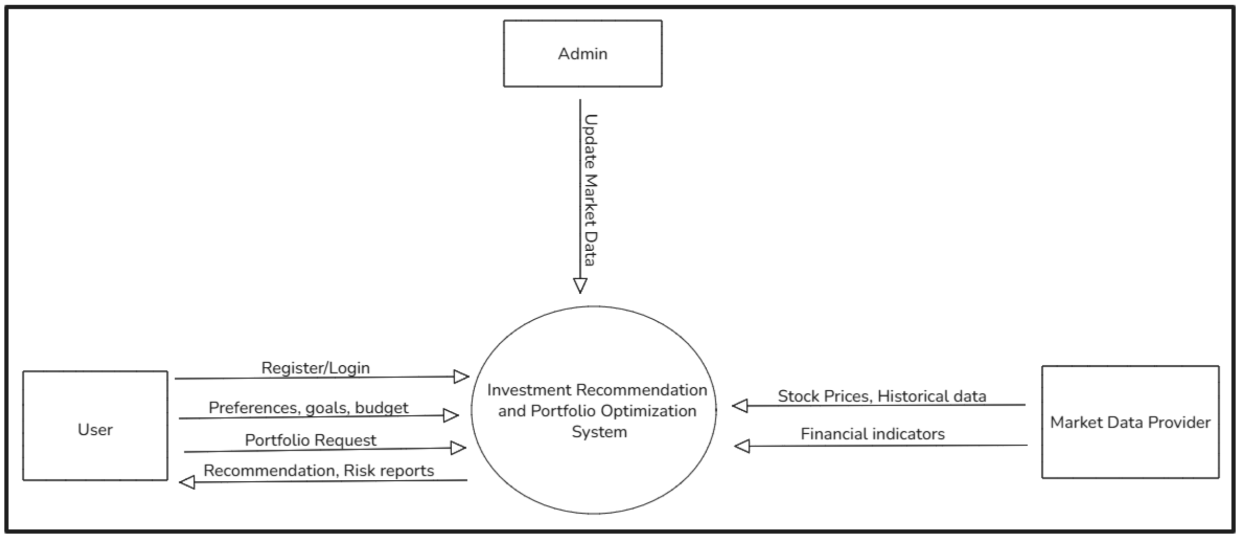
Product Functions

1. **User Registration and Login**
   The system allows new users to register by providing basic details such as name, email, and password. Returning users can securely log in using encrypted authentication. This ensures data privacy and user-specific access to portfolios and preferences.
2. **Risk Profile and Goal Setup**
   Users can define their investment preferences by setting financial goals (e.g., long-term growth, income generation) and risk tolerance levels (conservative, moderate, aggressive). This information forms the basis for generating personalized investment recommendations.
3. **Portfolio Optimization**
   The system applies *Markowitz's Modern Portfolio Theory* to calculate optimal asset allocations. It analyzes historical returns, volatility, and correlations among assets to recommend portfolios that maximize returns for a given risk level.
4. **Investment Recommendation Generation**
   Based on the user's risk profile and market data, the system generates an investment recommendation report. The report includes optimal asset weights, expected return, risk score, and investment categories suitable for the user's objectives.
5. **Risk Analysis and Reporting**
   The system performs advanced financial risk assessments using indicators like *Value-at-Risk (VaR)*, *Sharpe Ratio*, and *Maximum Drawdown*. A detailed risk report is displayed, helping investors understand possible losses and portfolio sensitivity to market changes.
6. **Market Data Integration**
   The system connects to external *Market Data Providers* through APIs to retrieve live or historical market prices of stocks, bonds, and mutual funds. This ensures that analysis and recommendations are based on accurate and up-to-date information.
7. **Portfolio Performance Tracking**
   Users can monitor the ongoing performance of their portfolio through dynamic dashboards. The system updates returns, volatility, and key performance metrics using real-time market data to help users make informed decisions.
8. **Visualization and Insights**
   The system generates visual analytics such as *efficient frontier plots*, *risk-return scatter charts*, and *performance comparison graphs*. These visualizations make it

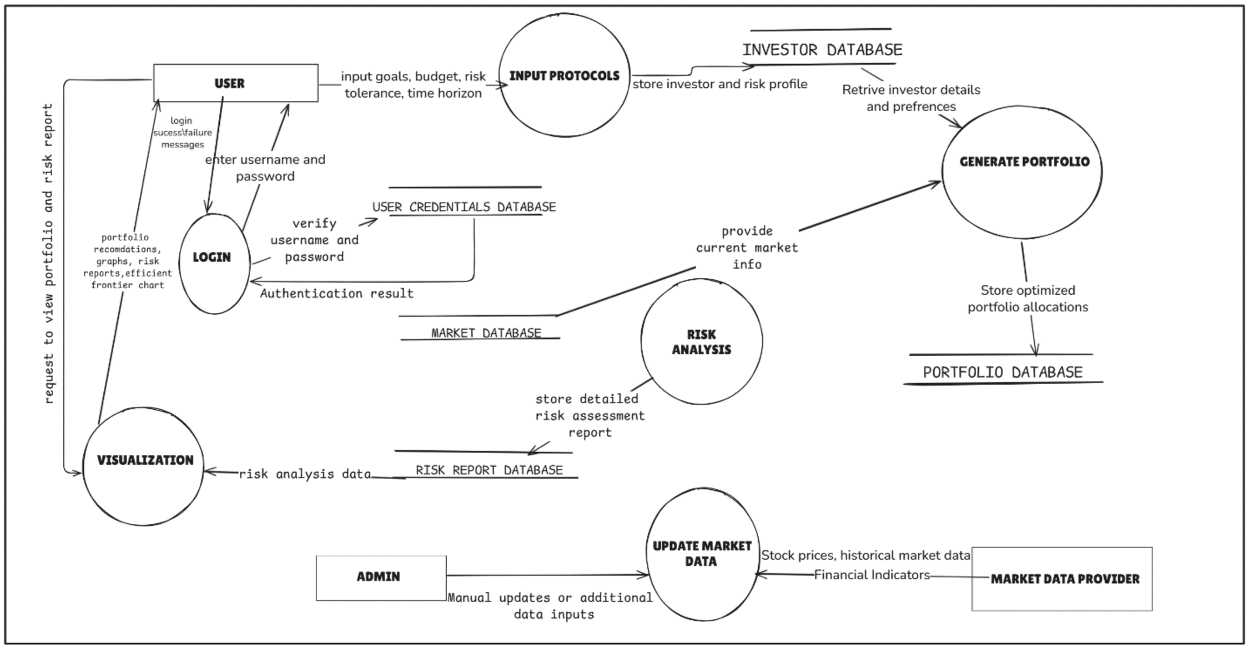
easier for users to interpret their investment strategy and understand diversification benefits.

9. **Alerts and Notifications**
   Users receive automated alerts when market conditions change significantly or when their portfolio deviates from optimal allocation. Notifications are sent for major losses, portfolio rebalancing suggestions, and news affecting their assets.
10. **Admin Data Management**
    The admin oversees the management of user accounts, database maintenance, and market data sources. The admin ensures continuous data availability, updates model parameters if needed, and validates system functionality.

Context Diagram

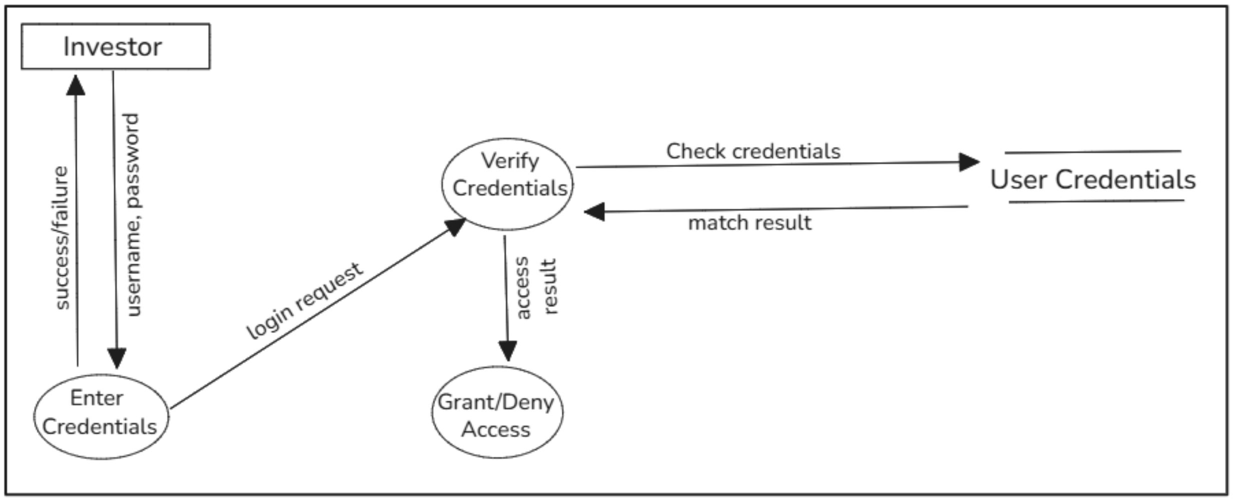

Admin
Update Market Data
User
Register/Login
Preferences, goals, budget
Portfolio Request
Recommendation, Risk reports
Investment Recommendation and Portfolio Optimization System
Stock Prices, Historical data
Financial indicators
Market Data Provider

DFD Level 1

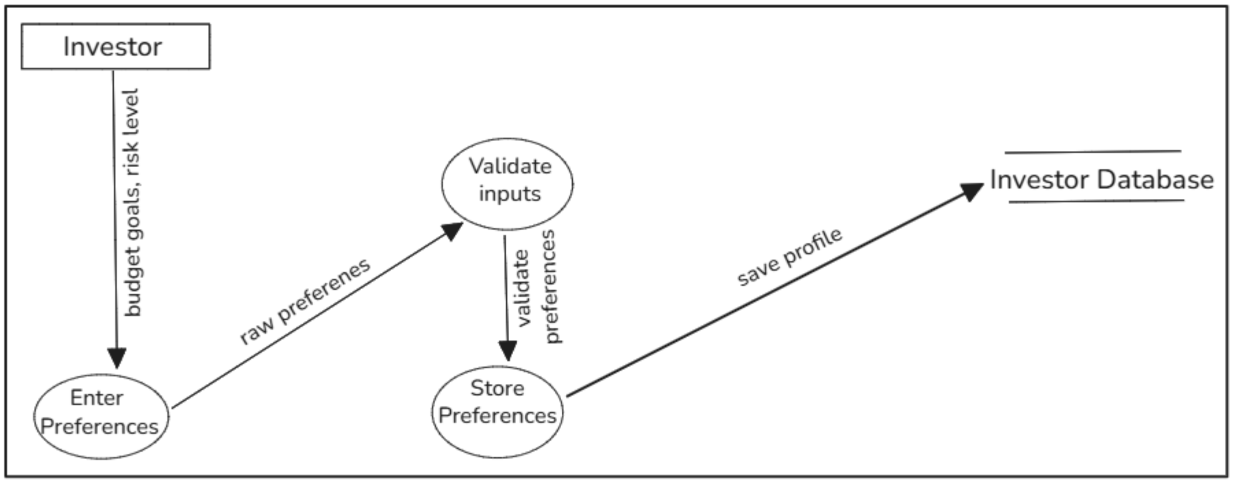

USER
input goals, budget, risk tolerance, time horizon
INPUT PROTOCOLS
store investor and risk profile
INVESTOR DATABASE
Retrive investor details and prefrences
GENERATE PORTFOLIO
login sucess\failure messages
enter username and password
verify username and password
USER CREDENTIALS DATABASE
LOGIN
Authentication result
request to view portfolio and risk report
portfolio recomdations, graphs, risk reports,efficient frontier chart
provide current market info
MARKET DATABASE
Store optimized portfolio allocations
PORTFOLIO DATABASE
RISK ANALYSIS
store detailed risk assessment report
RISK REPORT DATABASE
risk analysis data
VISUALIZATION
UPDATE MARKET DATA
ADMIN
Manual updates or additional data inputs
Stock prices, historical market data
Financial Indicators
MARKET DATA PROVIDER

DFD Level 2

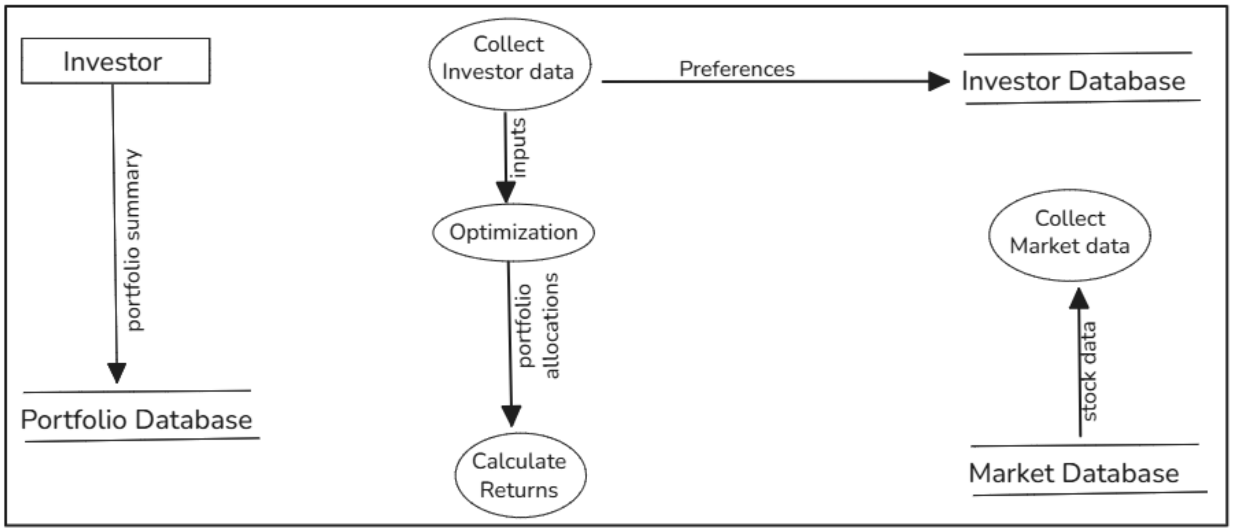
Process 1: Login/Authentication



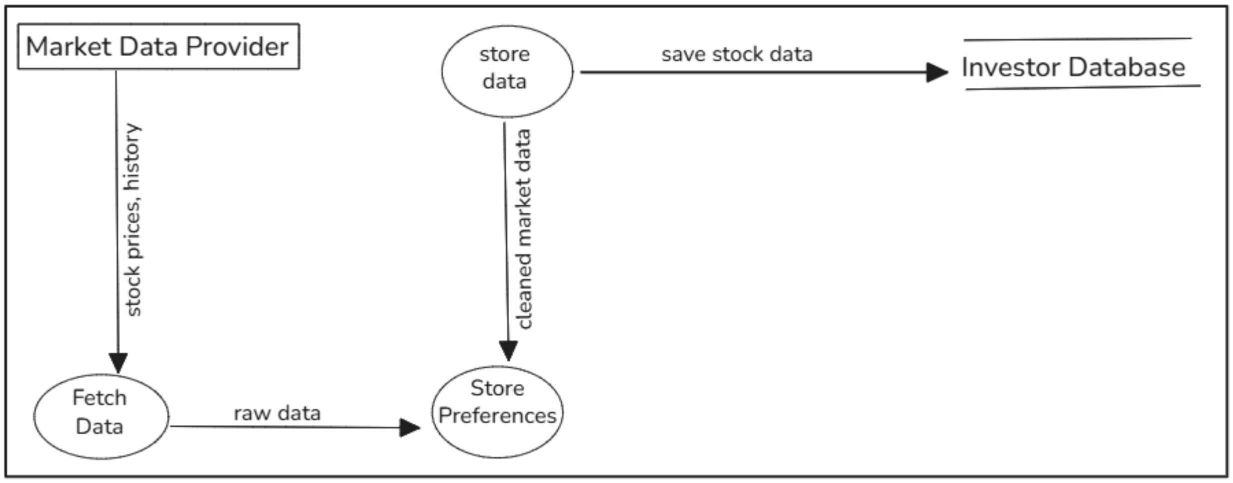
Process 2: Input Preferences



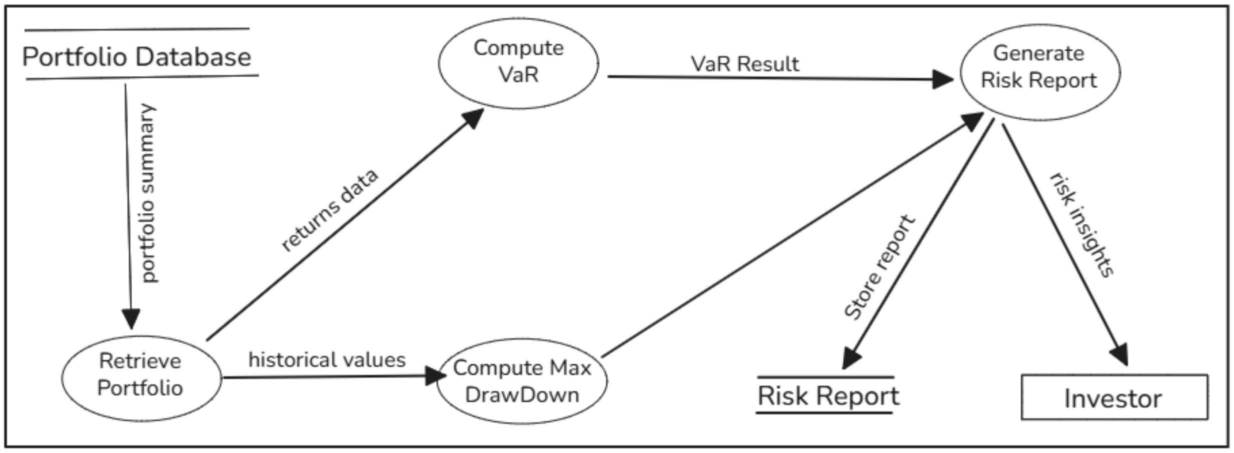
Process 3: Generate Portfolio



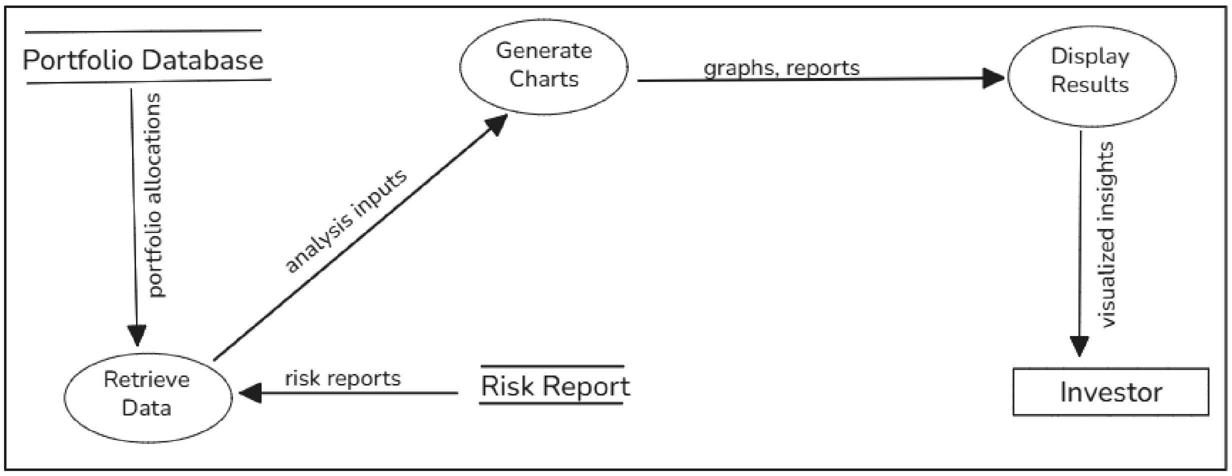
Process 4: Update market data



Process 5: Perform risk analysis

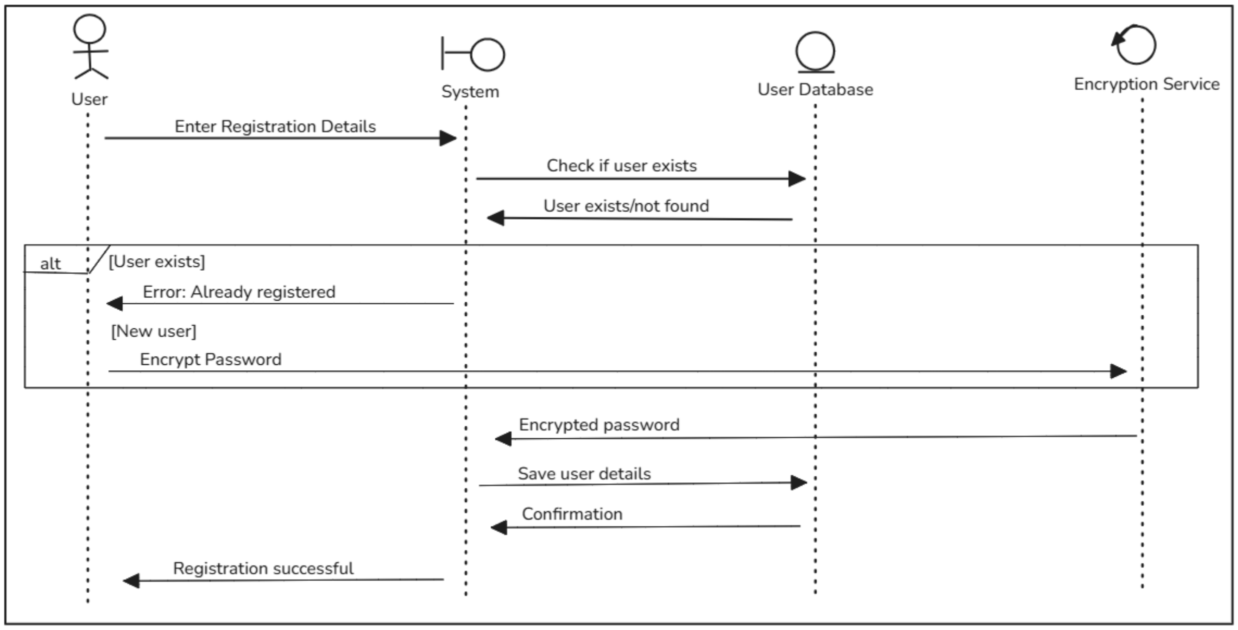


Process 6: Visualize results

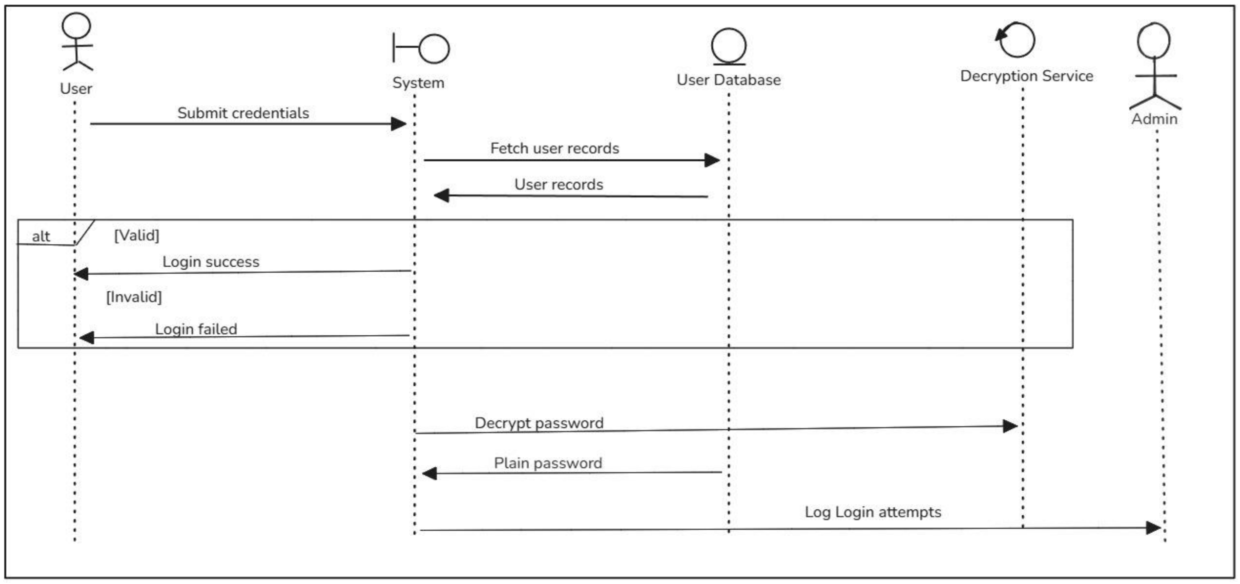


SEQUENCE DIAGRAMS

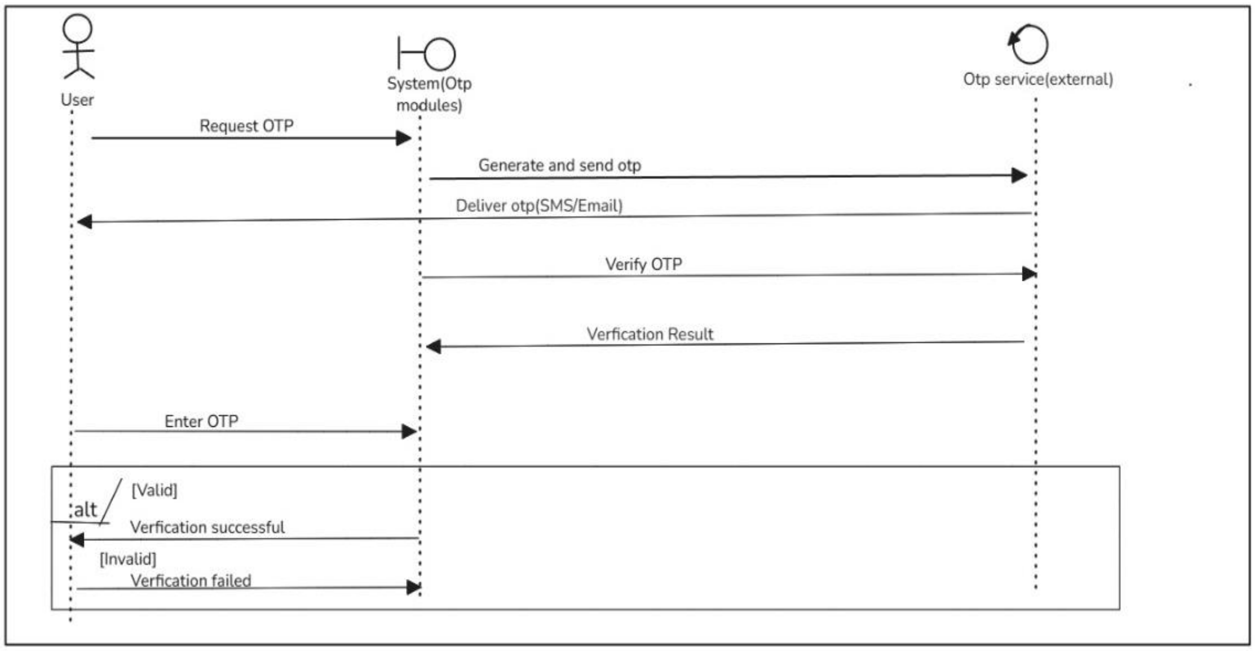
1. *Registration*



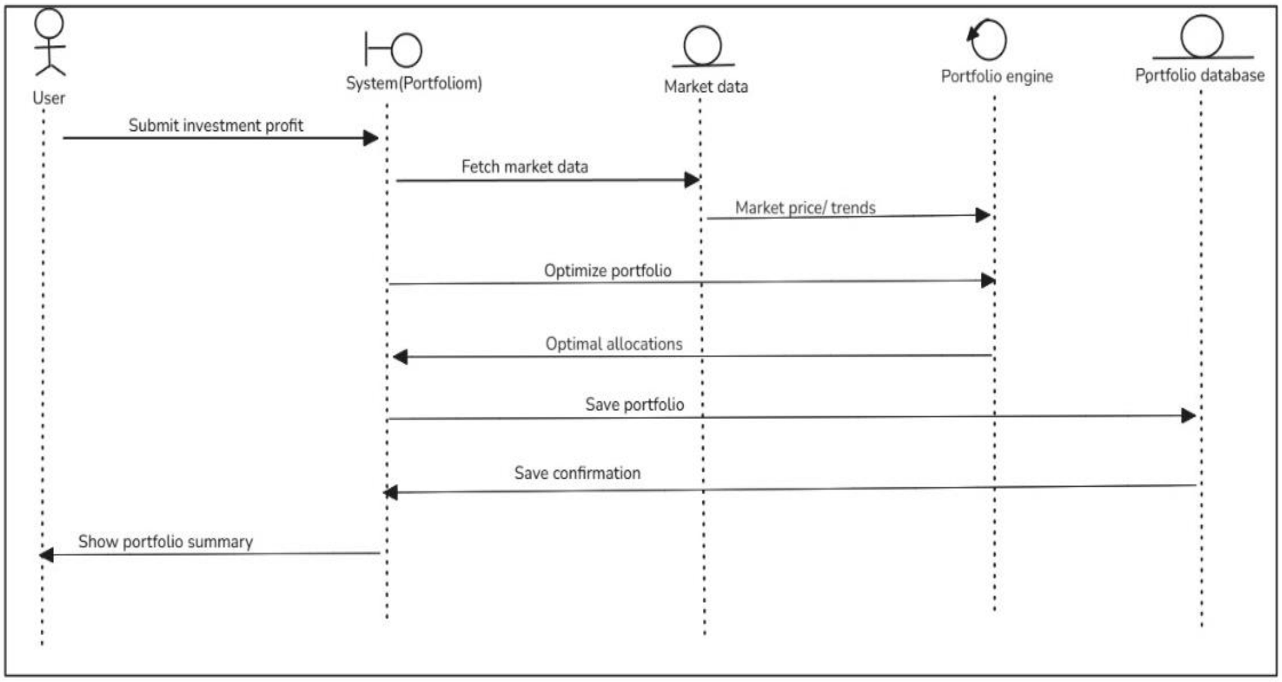
2. *Login*



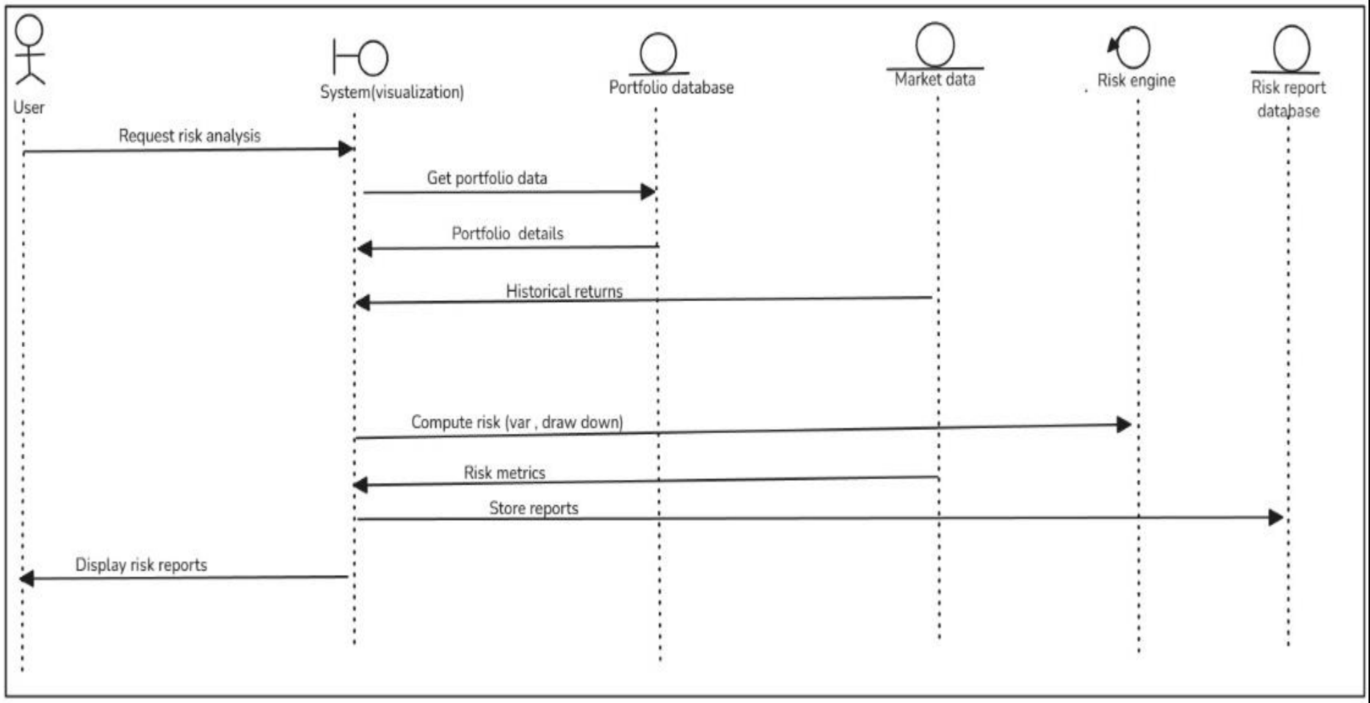
3. *OTP Verification*



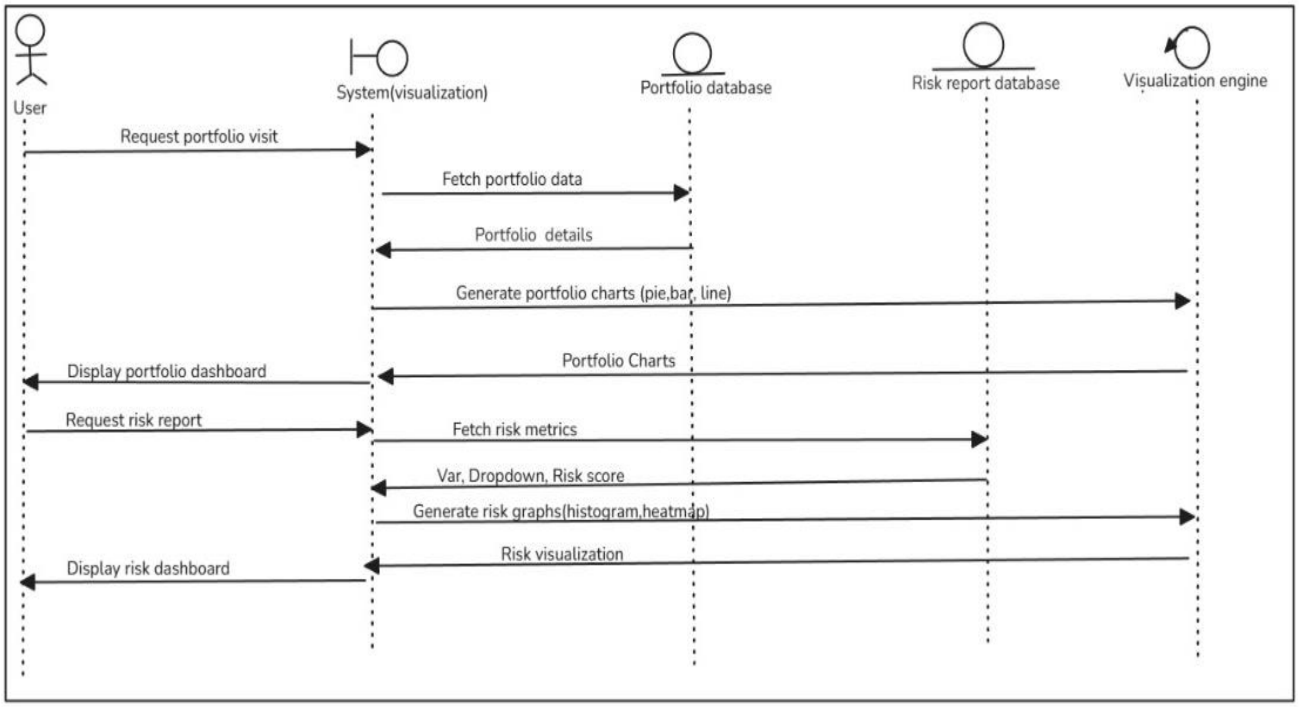
4. *Portfolio Management*



5. *Risk Prediction*



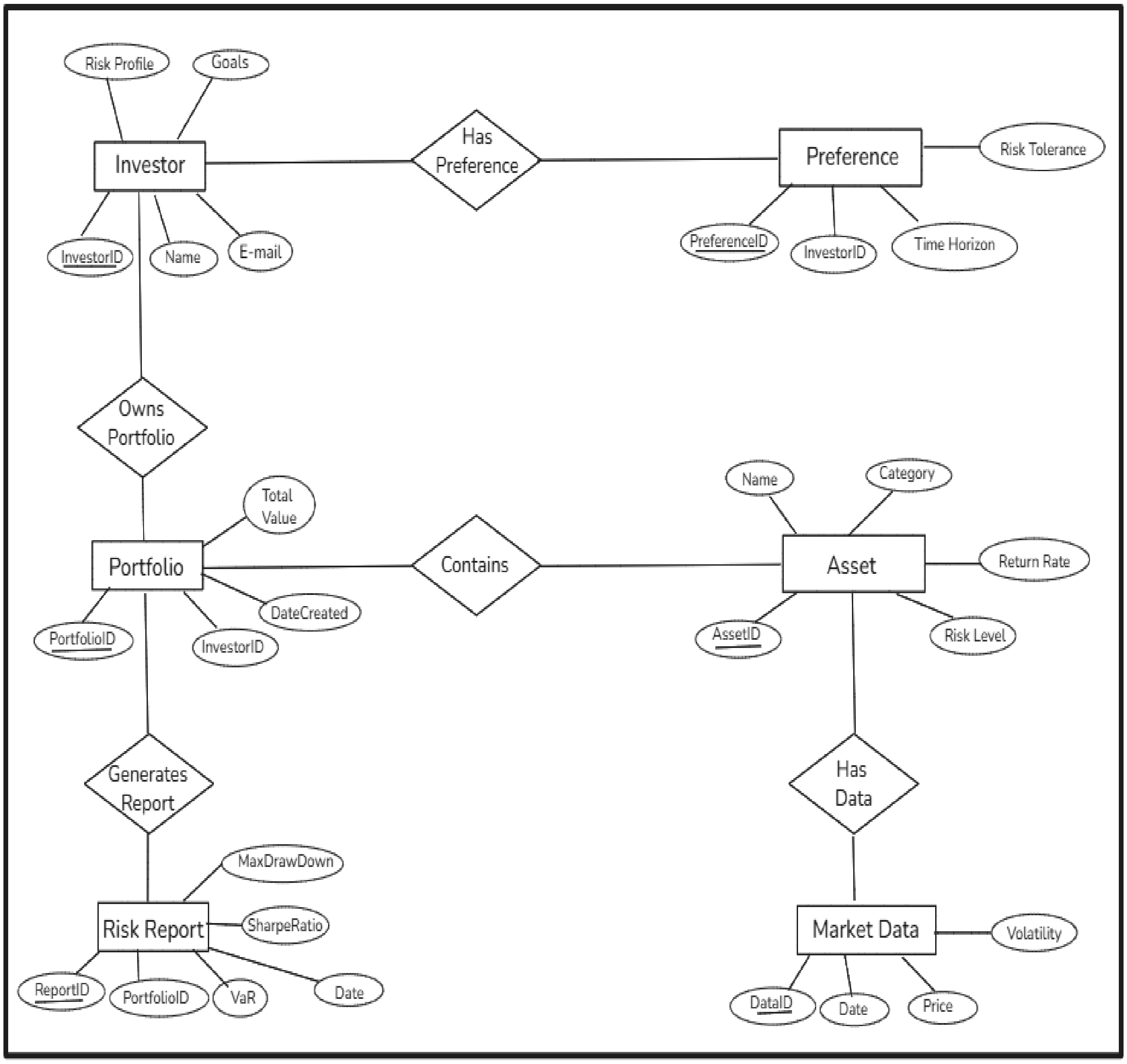
6. *Visualization*



<u>Data Dictionary</u>

Entity	**Attribute**	**Data Type**	**Description**
Investor	InvestorID	Integer	Unique identifier for each investor
Investor	Name	Varchar(100)	Name of the investor
Investor	Email	Varchar(100)	Email address of the investor
Preference	PreferenceID	Integer	Unique identifier for investor's preferences
Preference	RiskTolerance	Varchar(50)	Investor's risk appetite (Low/Medium/High)
Preference	TimeHorizon	Integer	Investment time period in years
Portfolio	PortfolioID	Integer	Unique portfolio identifier
Portfolio	TotalValue	Decimal(15,2)	Total value of the portfolio
Portfolio	DateCreated	Date	Date the portfolio was created
Asset	AssetID	Integer	Unique identifier for each asset
Asset	Name	Varchar(100)	Name of the financial asset
Asset	Category	Varchar(50)	Category of the asset (Stock, Bond, etc.)
Asset	ReturnRate	Decimal(5,2)	Average return percentage of the asset
Risk Report	ReportID	Integer	Unique identifier for each risk report
Risk Report	MaxDrawDown	Decimal(5,2)	Maximum portfolio loss percentage
Risk Report	SharpeRatio	Decimal(5,2)	Risk-adjusted return measure
Market Data	DataID	Integer	Unique identifier for market data entry
Market Data	Price	Decimal(10,2)	Current market price of the asset

ER Diagram


Risk Profile
Goals
Investor
Has Preference
Preference
Risk Tolerance
InvestorID
Name
E-mail
PreferenceID
InvestorID
Time Horizon
Owns Portfolio
Name
Category
Total Value
Portfolio
Contains
Asset
Return Rate
DateCreated
PortfolioID
InvestorID
AssetID
Risk Level
Generates Report
Has Data
MaxDrawDown
Risk Report
SharpeRatio
Market Data
Volatility
ReportID
PortfolioID
VaR
Date
DataID
Date
Price

Database Design

Investor

Attribute Name	Data Type	Description
InvestorID (PK)	INT	Unique identifier for each investor
Name	VARCHAR(100)	Name of the investor
Email	VARCHAR(100)	Email address of the investor
RiskProfile	VARCHAR(50)	Investor's risk-taking behavior
Goals	VARCHAR(200)	Investment goals

Preference

Attribute Name	Data Type	Description
PreferenceID (PK)	INT	Unique ID for each preference
InvestorID (FK)	INT	References InvestorID from Investor table
RiskTolerance	VARCHAR(50)	Investor's tolerance for risk
TimeHorizon	VARCHAR(50)	Investment duration preference

Portfolio

Attribute Name	Data Type	Description
PortfolioID (PK)	INT	Unique identifier
InvestorID (FK)	INT	References InvestorID from
TotalValue	FLOAT	Total current value of portfolio
DateCreated	DATE	Date when the portfolio was created

Asset

Attribute Name	Data Type	Description
AssetID (PK)	INT	Unique ID for each asset
Name	VARCHAR(100)	Name of the asset
Category	VARCHAR(50)	Type/category of asset (e.g., stock, bond)
ReturnRate	FLOAT	Average return rate of the asset
RiskLevel	VARCHAR(50)	Level of risk associated with the asset

MarketData

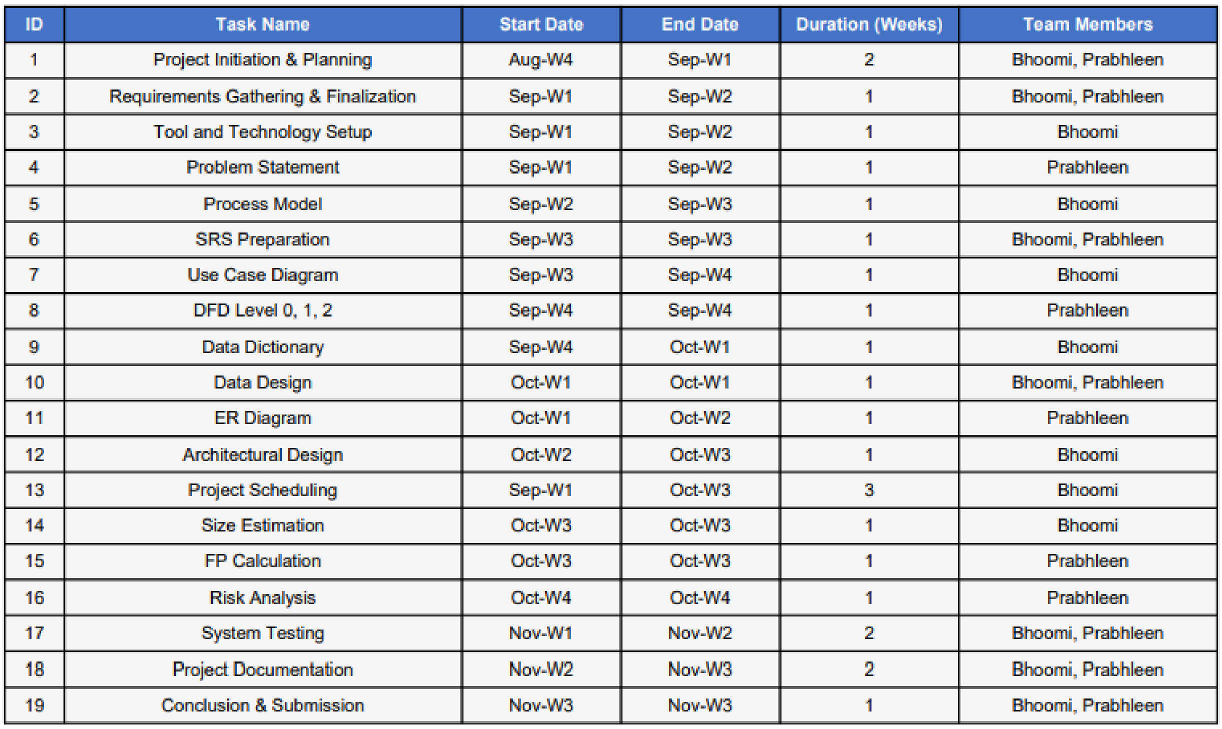
Attribute Name	Data Type	Description
DataID (PK)	INT	Unique identifier for each market data entry
Date	DATE	Date of market data record
Price	FLOAT	Current price of the asset
Volatility	FLOAT	Market volatility measure

RiskReport

Attribute Name	Data Type	Description
ReportID (PK)	INT	Unique ID for each risk report
PortfolioID (FK)	INT	References PortfolioID from Portfolio table
VaR	FLOAT	Value at Risk
SharpeRatio	FLOAT	Risk-adjusted return measure
MaxDrawDown	FLOAT	Maximum loss from a peak
Date	DATE	Date of report generation

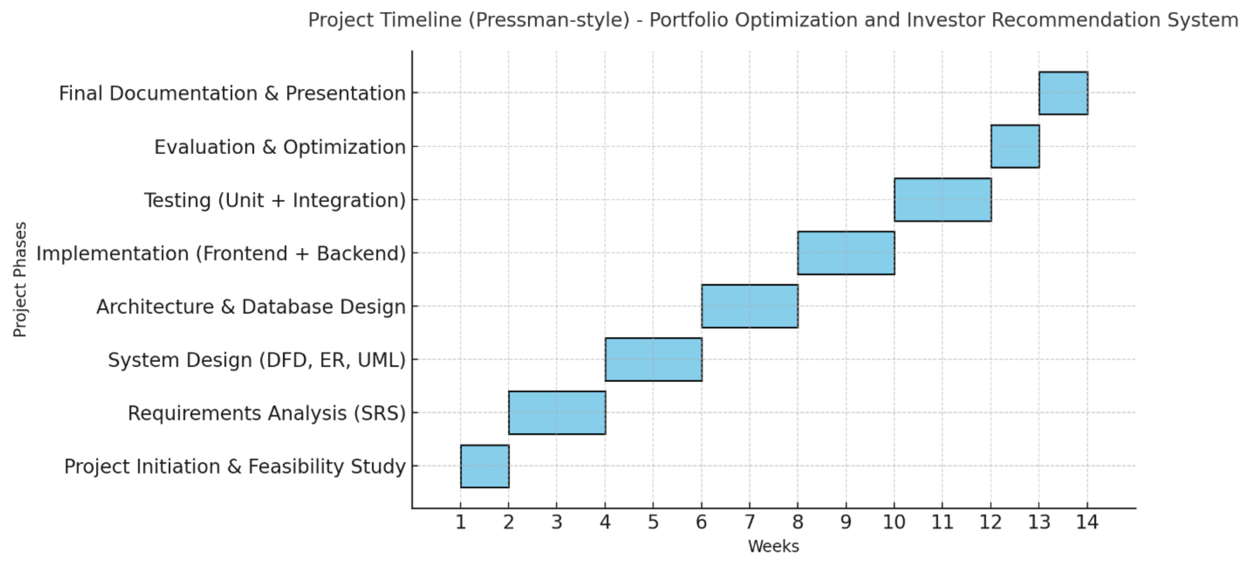
Project Scheduling/Timeline Chart (Gantt Chart)

Phase/Task	Timeline	Team Members
Problem Statement	AUG-W4, AUG-W4, SEP-W1, SEP-W1	Bhoomi, Prabhleen
Process Model	SEP-W1, SEP-W1, SEP-W1	Bhoomi, Prabhleen
SRS	SEP-W2, SEP-W2, SEP-W2, SEP-W2	Bhoomi, Prabhleen
Use Case	SEP-W2, SEP-W2, SEP-W2, SEP-W2	Bhoomi, Prabhleen
Use Case Description	SEP-W3	Bhoomi
DFD Level 0	SEP-W3, SEP-W3, SEP-W3	Prabhleen
DFD Level 1	SEP-W3, SEP-W3, SEP-W3	Bhoomi, Prabhleen
DFD Level 2	SEP-W4, SEP-W4, SEP-W4, SEP-W4	Bhoomi, Prabhleen
Sequence Diagrams	SEP-W4, SEP-W4, SEP-W4, SEP-W4	Bhoomi
Data Dictionary	SEP-W4, SEP-W4, SEP-W4, SEP-W4	Prabhleen
Data Design	OCT-W1, OCT-W1, OCT-W1, OCT-W1	Bhoomi, Prabhleen
ER Diagram	OCT-W1, OCT-W1, OCT-W1	Bhoomi, Prabhleen
Architectural Design	OCT-W2, OCT-W2, OCT-W2	Bhoomi
Project Scheduling	SEP-W1, SEP-W1, NOV-W4, NOV-W4	Prabhleen
Size Estimation	OCT-W3, OCT-W3, OCT-W3	Bhoomi
FP Calculation	OCT-W3, OCT-W3, OCT-W3	Bhoomi
Risk Analysis	OCT-W4, OCT-W4, OCT-W4	Prabhleen
Testing	NOV-W2, NOV-W2, NOV-W2, NOV-W2	Bhoomi, Prabhleen
Conclusion	NOV-W3, NOV-W3, NOV-W3	Bhoomi, Prabhleen



ID	Task Name	Start Date	End Date	Duration (Weeks)	Team Members
1	Project Initiation & Planning	Aug-W4	Sep-W1	2	Bhoomi, Prabhleen
2	Requirements Gathering & Finalization	Sep-W1	Sep-W2	1	Bhoomi, Prabhleen
3	Tool and Technology Setup	Sep-W1	Sep-W2	1	Bhoomi
4	Problem Statement	Sep-W1	Sep-W2	1	Prabhleen
5	Process Model	Sep-W2	Sep-W3	1	Bhoomi
6	SRS Preparation	Sep-W3	Sep-W3	1	Bhoomi, Prabhleen
7	Use Case Diagram	Sep-W3	Sep-W4	1	Bhoomi
8	DFD Level 0, 1, 2	Sep-W4	Sep-W4	1	Prabhleen
9	Data Dictionary	Sep-W4	Oct-W1	1	Bhoomi
10	Data Design	Oct-W1	Oct-W1	1	Bhoomi, Prabhleen
11	ER Diagram	Oct-W1	Oct-W2	1	Prabhleen
12	Architectural Design	Oct-W2	Oct-W3	1	Bhoomi
13	Project Scheduling	Sep-W1	Oct-W3	3	Bhoomi
14	Size Estimation	Oct-W3	Oct-W3	1	Bhoomi
15	FP Calculation	Oct-W3	Oct-W3	1	Prabhleen
16	Risk Analysis	Oct-W4	Oct-W4	1	Prabhleen
17	System Testing	Nov-W1	Nov-W2	2	Bhoomi, Prabhleen
18	Project Documentation	Nov-W2	Nov-W3	2	Bhoomi, Prabhleen
19	Conclusion & Submission	Nov-W3	Nov-W3	1	Bhoomi, Prabhleen

Timeline Chart



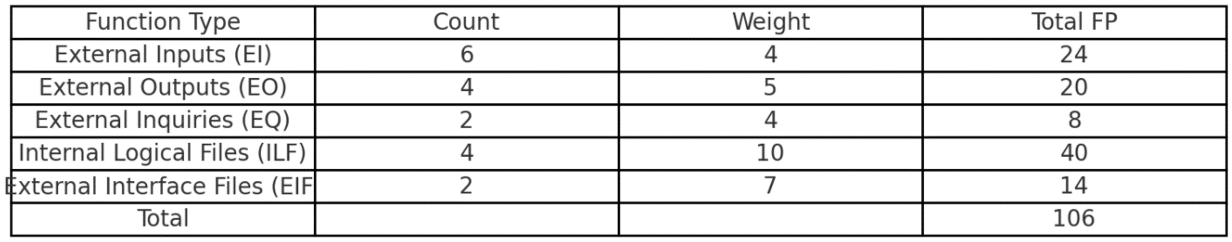
Computing FP

Size Estimation (FUNCTION BASED METRICS)

Information domain values are defined in the following manner:

- **Number of external inputs (EIs)** - Each external input originates from a user or is transmitted from another application and provides distinct application-oriented data or control information. Inputs are often used to update internal logical files (ILFs). Inputs should be distinguished from inquiries which are counted separately.
- **Number of external outputs (EOs)** - Each external output is derived data within the application that provides information to the user. In this context external output refers to reports, screens, error messages, etc. Individual data items within a report are not counted separately.
- **Number of external inquiries (EQs)** - An external inquiry is defined as an online input that results in the generation of some immediate software response in the form of an online output (often retrieved from an ILF).
- **Number of internal logical files (ILFs)** - Each internal logical file is a logical grouping of data that resides within the application's boundary and is maintained via external inputs.
- N**umber of external interface files (EIFs)** - Each external interface file is a logical grouping of data that resides external to the application but provides information that may be of use to the application.

Calculation Table



Function Type	Count	Weight	Total FP
External Inputs (EI)	6	4	24
External Outputs (EO)	4	5	20
External Inquiries (EQ)	2	4	8
Internal Logical Files (ILF)	4	10	40
External Interface Files (EIF	2	7	14
Total			106

Final Function Point (FP) Estimate:

106 FP

Risk Analysis Table

S.No.	Risk	Category	Probability (P)	Impact (I)	Mitigation Strategy
1	Key stakeholder becomes unavailable	Stakeholder Risk	15%	Medium	Identify backup stakeholders, maintain thorough documentation, and ensure regular updates to all stakeholders.
2	Misalignment between team expectations	Team Risk	20%	High	Conduct kickoff meetings, set clear roles and responsibilities, and establish an internal communication plan.
3	Data breach due to poor access management	Security Risk	25%	Critical	Enforce strict access controls, implement two-factor authentication, and conduct regular audits of access permissions.
4	Delay in obtaining critical resources	Operational Risk	20%	Medium	Build a buffer in timelines, maintain a list of alternative vendors/suppliers, and regularly track procurement progress.
5	High staff turnover during the project	Team Risk	10%	High	Create a knowledge repository, conduct thorough handovers, and

					provide training for new team members.
6	Inaccurate cost estimation leads to budget overrun	Financial Risk	30%	High	Conduct detailed cost analysis, regularly review budgets, and implement a contingency fund for unexpected expenses.
7	Regulatory or compliance issues	Compliance Risk	15%	High	Engage legal/compliance experts, stay updated on regulations, and conduct periodic compliance reviews.
8	Failure of key third-party service providers	Vendor Risk	20%	High	Evaluate service providers carefully, have contracts with SLAs in place, and establish backup vendors where possible.

Architectural Design

The proposed *Profile Optimization and Investment Recommendation System* is based on the **MVC (Model-View-Controller)** architecture. This architectural pattern divides the application into three main layers — Model, View, and Controller — to ensure **modularity**, **scalability**, and **security**.
By separating data handling, user interface, and application logic, the system enhances maintainability and supports future integration of machine learning models for investment predictions and portfolio optimization.

Model Details

The **Model** layer manages the system's core functionality, including data processing, business logic, and database operations. It ensures secure handling of user credentials, investment records, and analytical computations.

Key components include:

- **User Authentication & Access Control:**
  The Model validates user credentials during login and registration using secure hashing algorithms such as **bcrypt** or **Argon2**. It manages session tokens and ensures only authorized users can access portfolio data.
- **Portfolio Management Module:**
  This component allows storage and retrieval of user investments across asset classes (stocks, bonds, mutual funds, etc.). It supports operations like adding, updating, or deleting investment entries while maintaining consistency in the database.
- **Risk Prediction Engine:**
  Analytical models are implemented to calculate **portfolio risk metrics**, such as volatility, Sharpe ratio, and asset correlation. Machine learning algorithms may be integrated for predictive analysis of future risk trends.
- **Data Security Mechanisms:**
  The Model incorporates encryption for sensitive financial data both in transit (using **SSL/TLS**) and at rest (using **AES-256**). Database queries are parameterized to prevent SQL injection attacks.

View Details

The **View** layer handles all user interface (UI) and data visualization aspects of the system. It presents a clear and interactive interface for users to manage portfolios and view insights.

Key elements include:

- **User Dashboards:**
  Personalized dashboards display portfolio summaries, asset distribution, and total returns.
- **Visualization Components:**
  Interactive **charts and graphs** (e.g., pie charts for asset allocation, line charts for performance trends) help users analyze portfolio performance and risk.
- **Forms and Notifications:**
  The interface includes forms for registration, login, and investment entry, along with alerts for success, warnings, or errors (e.g., *"Invalid login credentials"*, *"Portfolio updated successfully"*).
- **Responsive Design:**
  Built with responsive web technologies (HTML5, CSS3, and JavaScript frameworks), ensuring compatibility across desktops, tablets, and mobile devices.

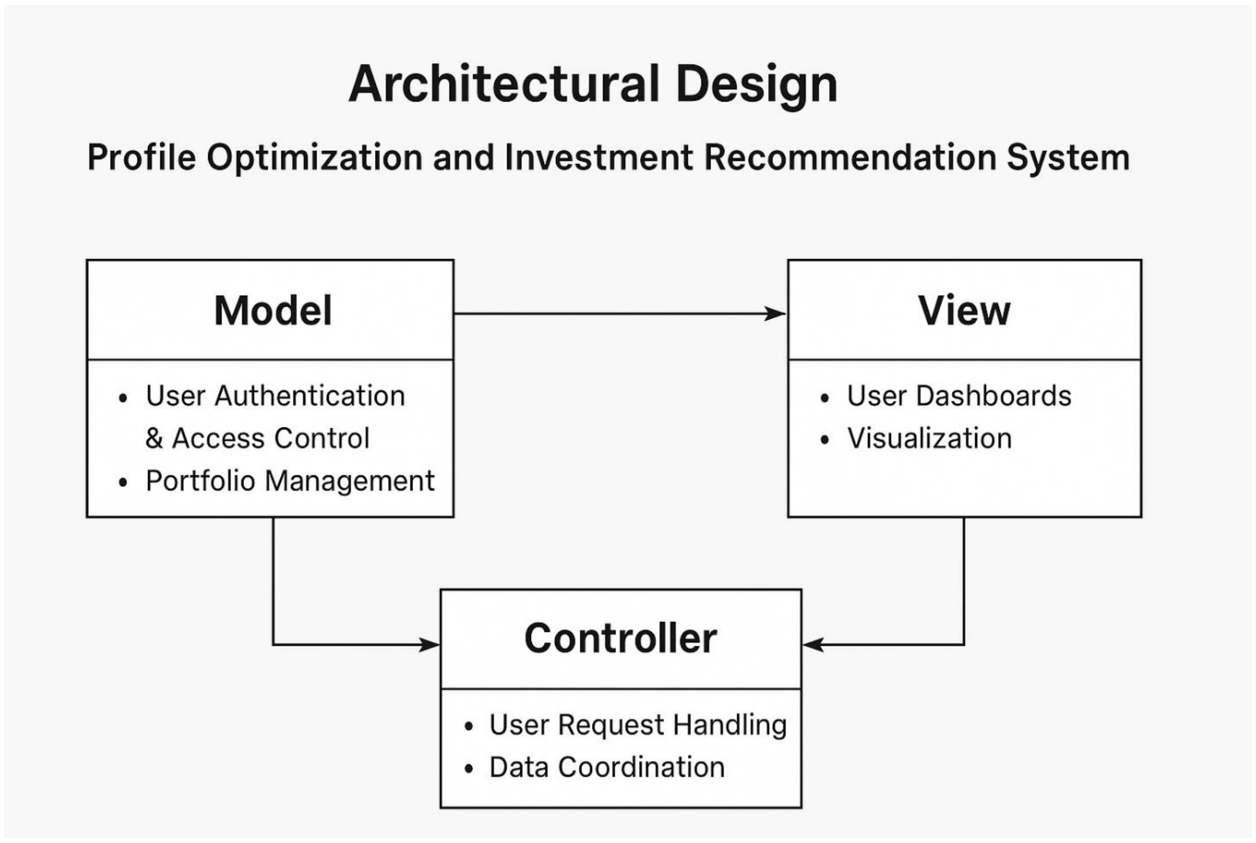
Controller Role

The **Controller** serves as the intermediary between the Model and View, managing user requests, coordinating business logic, and updating the interface accordingly.

Core responsibilities include:

- **User Request Handling:**
  The Controller processes inputs from users (e.g., login, adding an investment) and invokes corresponding Model functions to execute the required logic.
- **Data Coordination:**
  It retrieves computed data (e.g., predicted risk levels, investment summaries) from the Model and passes it to the View for presentation.
- **Access Control Enforcement:**
  Before executing any operation, the Controller checks user authentication and authorization levels, ensuring secure data access.

- **Dynamic Updates:**
  Based on the results from the Model, the Controller dynamically updates the user interface with real-time charts, performance metrics, or notification messages.



Pseudocode

1. User Authentication & Access Control

```
START
FUNCTION UserAuthentication()
    INPUT username, password
    hashed_password = HASH(password)

    user_record = DATABASE.GET_USER(username)

    IF user_record == NULL THEN
        DISPLAY "User not found"
    ELSE
        IF hashed_password == user_record.hashed_password THEN
            token = GENERATE_SESSION_TOKEN()
            STORE token IN SESSION_TABLE
            DISPLAY "Login Successful"
        ELSE
            DISPLAY "Invalid Password"
        END IF
    END IF
END FUNCTION

FUNCTION UserRegistration()
    INPUT username, email, password
    IF DATABASE.USER_EXISTS(username, email) THEN
        DISPLAY "User already exists"
    ELSE
        hashed_password = HASH(password)
        DATABASE.INSERT_USER(username, email, hashed_password)
        DISPLAY "Registration Successful"
    END IF
END FUNCTION
END
```

2. Portfolio Management

```
START
FUNCTION AddInvestment(user_id, asset_name, amount, category)
   IF VALIDATE_INPUT(asset_name, amount) THEN
      DATABASE.INSERT("Portfolio", user_id, asset_name, amount, category)
      DISPLAY "Investment Added Successfully"
   ELSE
      DISPLAY "Invalid Input"
   END IF
END FUNCTION

FUNCTION UpdateInvestment(portfolio_id, new_amount)
   record = DATABASE.GET_RECORD("Portfolio", portfolio_id)
   IF record != NULL THEN
      record.amount = new_amount
      DATABASE.UPDATE(record)
      DISPLAY "Investment Updated"
   ELSE
      DISPLAY "Portfolio Not Found"
   END IF
END FUNCTION

FUNCTION ViewPortfolio(user_id)
   portfolio_list = DATABASE.GET_PORTFOLIO(user_id)
   DISPLAY portfolio_list
END FUNCTION
END
```

3. Risk Prediction

```
START
FUNCTION CalculateRiskScore(portfolio_id)
   portfolio = DATABASE.GET_ASSETS(portfolio_id)
   total_value = SUM(portfolio.asset_values)
   risk_score = 0

   FOR EACH asset IN portfolio DO
```

```
        market_data = DATABASE.GET_MARKET_DATA(asset.asset_id)
        volatility = market_data.volatility
        risk_score += (asset.value / total_value) * volatility
    END FOR

    RETURN risk_score
END FUNCTION

FUNCTION ClassifyRiskLevel(risk_score)
    IF risk_score < 0.3 THEN
        RETURN "Low Risk"
    ELSE IF risk_score < 0.6 THEN
        RETURN "Medium Risk"
    ELSE
        RETURN "High Risk"
    END IF
END FUNCTION
END
```

4. Data Security

```
START
FUNCTION EncryptData(data)
    key = GET_ENCRYPTION_KEY()
    encrypted_data = AES_ENCRYPT(data, key)
    RETURN encrypted_data
END FUNCTION

FUNCTION DecryptData(encrypted_data)
    key = GET_ENCRYPTION_KEY()
    decrypted_data = AES_DECRYPT(encrypted_data, key)
    RETURN decrypted_data
END FUNCTION

FUNCTION SecureStorage(user_data)
    encrypted_data = EncryptData(user_data)
    DATABASE.STORE(encrypted_data)
    DISPLAY "Data stored securely"
END FUNCTION
```

```
END
```

4. Visualization and Reporting

```
START
FUNCTION GenerateReport(user_id)
   portfolio = DATABASE.GET_PORTFOLIO(user_id)
   risk_score = CalculateRiskScore(portfolio.id)
   risk_level = ClassifyRiskLevel(risk_score)

   chart_data = ANALYZE_PERFORMANCE(portfolio)
   DISPLAY_CHART(chart_data)
   DISPLAY "Risk Level: ", risk_level
   DISPLAY "Total Portfolio Value: ", SUM(portfolio.asset_values)
END FUNCTION

FUNCTION ShowDashboard(user_id)
   portfolio_data = DATABASE.GET_PORTFOLIO(user_id)
   recent_trends = DATABASE.GET_MARKET_TRENDS()
   DISPLAY_DASHBOARD(portfolio_data, recent_trends)
END FUNCTION
END
```

Flow of Operations

1. User Authentication and Access Control

1. The system starts by presenting the **Login/Registration screen** to the user.
2. If the user is new, they register by entering their name, email, and password.
   - The password is **hashed and securely stored** in the database.
3. Returning users enter their credentials.
   - The system verifies the username and password.
   - On successful authentication, a **session token** is generated to maintain secure access.
4. Unauthorized users are denied access with an appropriate error message.

Outcome:
Only verified users gain access to their personalized dashboard.

---

2. Portfolio Management

1. Once authenticated, the user navigates to the **Portfolio Management** section.
2. The user can:
   - **Add** a new investment (by entering asset name, amount, category, and date).
   - **Update** or **delete** existing investments.
   - **View** a summary of all assets in the portfolio.
3. The system stores these transactions in the **Portfolio** table linked to the user's ID.
4. The total portfolio value and asset-wise distribution are dynamically updated.

Outcome:
The user has a clear and updated view of their investment portfolio.

---

3. Risk Prediction and Analysis

1. The system retrieves the user's portfolio and related **market data**.
2. For each asset:
   - The system calculates **volatility**, **return rate**, and **risk exposure**.
3. Using the aggregated data, a **risk score** is computed using a weighted formula based on asset values and market volatility.
4. The risk score is then classified into **Low**, **Medium**, or **High** categories.
5. A **risk report** is generated, summarizing the overall risk, performance, and diversification level.

Outcome:
The user receives an accurate analysis of portfolio risk to guide investment decisions.

---

4. Profile Optimization

1. Based on the calculated risk level and user's **preferences** (e.g., risk tolerance, financial goals, and time horizon),
   the system generates **investment recommendations**.
2. The algorithm suggests:
   - Rebalancing portfolios to reduce risk exposure.
   - Potential new assets or mutual funds to improve returns.

    - Optimal asset allocation percentages (e.g., 60% equity, 30% bonds, 10% cash).
3. The user can accept or reject recommendations and make modifications accordingly.

Outcome:
The portfolio becomes better aligned with the user's risk profile and investment goals.

---

5. Data Security

1. All user data (personal and financial) is **encrypted** before being stored in the database.
2. During communication, **SSL/TLS protocols** ensure secure data transmission.
3. Session management prevents unauthorized access or data tampering.
4. Regular encryption key rotation and access control policies are enforced.

Outcome:
Data integrity and privacy are maintained throughout the system.

---

6. Visualization and Reporting

1. After optimization, the system displays the results through an **interactive dashboard**.
2. Components of the dashboard include:
    - Pie charts showing asset allocation.
    - Line graphs showing portfolio performance over time.
    - Risk vs Return comparison plots.
3. The user can generate and download a **detailed report** summarizing:
    - Portfolio composition
    - Risk analysis results
    - Recommended changes and performance projections

Outcome:
Users gain actionable insights through graphical visualization and summary reports.

Risk Prediction Module Python Code

Functionality:

This module:

- Takes portfolio assets with returns and volatility.
- Calculates **portfolio risk (variance & standard deviation)**.
- Classifies risk level as **Low**, **Medium**, or **High**.
- Prints results neatly in the command prompt.

Python Code:

```
# risk_prediction.py

import numpy as np

def calculate_portfolio_risk(returns, weights, covariance_matrix):
    """
    Calculates portfolio risk using variance-covariance method.
    returns: list of mean returns for each asset
    weights: list of portfolio weights
    covariance_matrix: matrix showing covariance between assets
    """
    portfolio_variance = np.dot(weights, np.dot(covariance_matrix, weights.T))
    portfolio_std_dev = np.sqrt(portfolio_variance)
    return portfolio_std_dev

def classify_risk(risk_value):
    """Classifies portfolio risk level"""
    if risk_value < 0.10:
        return "Low Risk"
    elif 0.10 <= risk_value < 0.20:
        return "Moderate Risk"
    else:
        return "High Risk"

# Example Data
```

```
assets = ["Stock A", "Stock B", "Stock C"]
returns = np.array([0.12, 0.08, 0.06])      # Expected returns
weights = np.array([0.5, 0.3, 0.2])         # Portfolio weights
cov_matrix = np.array([
    [0.04, 0.01, 0.00],
    [0.01, 0.03, 0.01],
    [0.00, 0.01, 0.02]
])  # Covariance matrix

# Risk Calculation
portfolio_risk = calculate_portfolio_risk(returns, weights, cov_matrix)
risk_level = classify_risk(portfolio_risk)

# Output Results
print("------- Portfolio Risk Analysis -------")
for i in range(len(assets)):
    print(f"Asset: {assets[i]}, Expected Return: {returns[i]*100:.2f}%, Weight:
{weights[i]*100:.1f}%")
print("---------------------------------------")
print(f"Portfolio Risk (Std Dev): {portfolio_risk*100:.2f}%")
print(f"Risk Category: {risk_level}")
```

What is Testing

Software testing is the process of evaluating a software application to ensure it functions correctly, meets requirements, and is free of defects before it is released. It involves verifying and validating the software's functionality, performance, security, and usability to build trust and mitigate risks. Key techniques include white box testing (checking the internal code) and black box testing (checking functionality based on requirements without knowledge of the code).

Types of Testing

White Box Testing (also known as Clear Box, Glass Box, or Structural Testing):

White box testing involves examining the internal workings of a system, including its code, architecture, and design. The tester has full knowledge of the system's internal structure and logic.

- **Focus**: Internal logic, code paths, data structures, and security vulnerabilities within the code.
- **Performed by**: Typically developers or QA engineers with strong programming skills.
- **Techniques**: Code coverage analysis (e.g., statement coverage, branch coverage, path coverage), unit testing, integration testing.
- **Advantages**: Can uncover hidden defects, optimize code, and improve code quality.
- **Disadvantages:** Requires in-depth technical knowledge, can be time-consuming for complex systems.

Black Box Testing (also known as Behavioral, Functional, or Opaque Box Testing):

Black box testing focuses solely on the external behavior of the system without any knowledge of its internal structure. The tester interacts with the system as an end-user, providing inputs and verifying outputs against the specified requirements.

Focus: Functionality, user requirements, and overall system behavior.

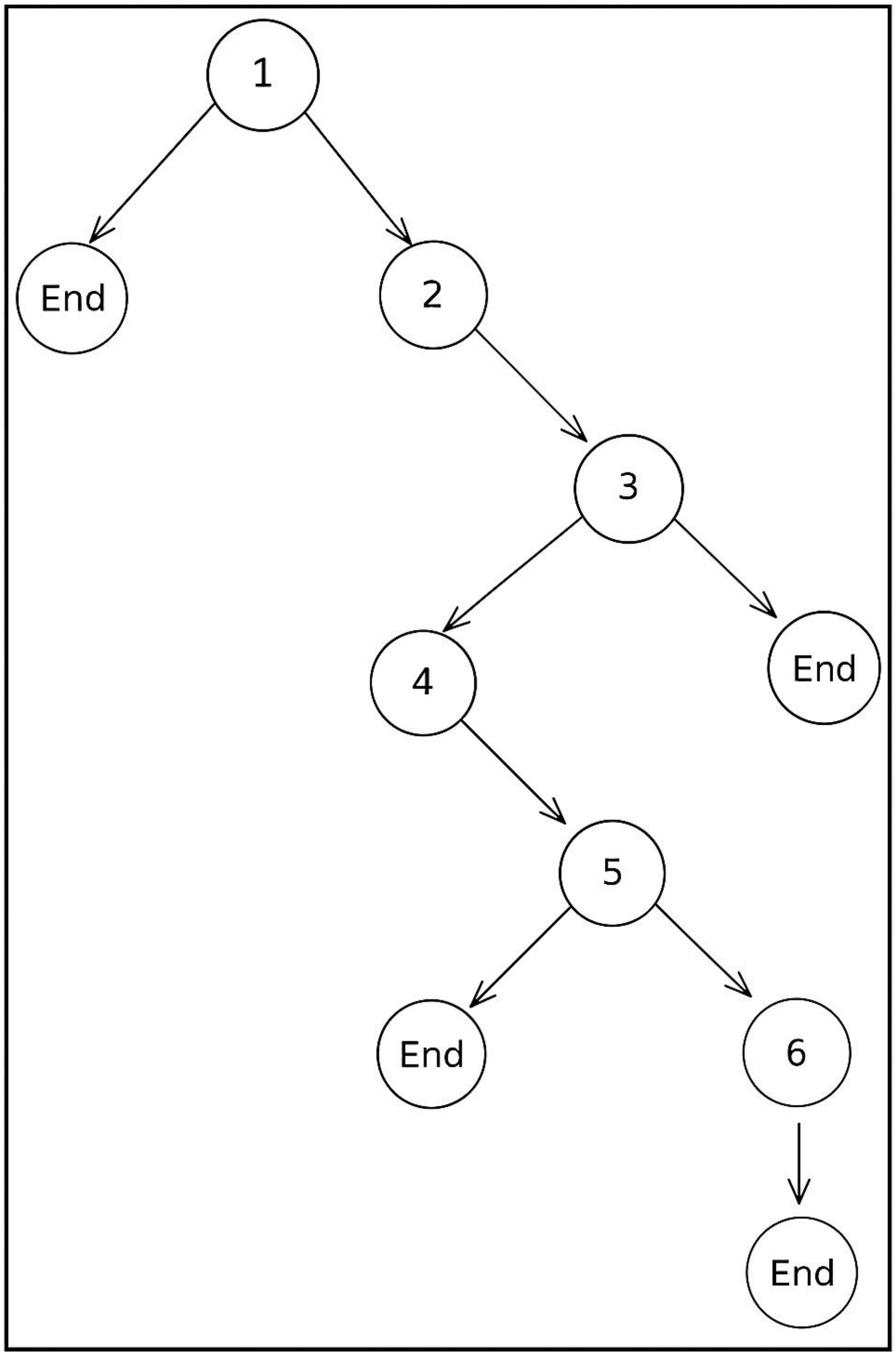
Performed by: Independent QA testers or end-users.

Techniques: Functional testing, system testing, acceptance testing, regression testing.

Advantages: Does not require programming knowledge, can be performed by non-technical personnel, validates user-facing features.

Disadvantages: May miss internal defects or vulnerabilities if not exposed through external behavior.

Flow Diagram



Cyclomatic Complexity V(G)

1) Way 1: Using edges, nodes, and independent components

Formula:
Cyclomatic Complexity = $e - n + 2P$

Where:

- **$e = 12$** (number of edges)
- **$n = 9$** (number of nodes)
- **$P = 1$** (number of independent components)

Cyclomatic Complexity = $12 - 9 + 2 \times 1 = 5$

---

2) Way 2: Using decision nodes

Formula:
Cyclomatic Complexity = (Number of decision nodes) + 1

Where:

- **Decision nodes = 4** (conditional checks in data validation, risk calculation, threshold evaluation, and recommendation logic)

Cyclomatic Complexity = $4 + 1 = 5$

---

3) Way 3: Using regions

Formula:
Cyclomatic Complexity = Number of regions in the flow graph

Where:

- **Number of regions = 5**

Cyclomatic Complexity = 5

Summary:

Hence, the **Cyclomatic Complexity (V(G)) = 5**, indicating a **moderate program complexity** and **manageable code structure**.
This level ensures the program is maintainable and testable with a limited number of independent paths.

Independent Paths in Control Flow

1. Input data validation fails → Return error message
2. Input valid → Proceed to risk calculation → Return “Low Risk”
3. Input valid → Proceed to risk calculation → Return “Moderate Risk”
4. Input valid → Proceed to risk calculation → Return “High Risk”
5. Input valid → Risk calculated → Generate Investment Recommendation → Return “Recommended Portfolio Generated Successfully”

Test Cases

Module: Risk Prediction

Section-I (Before Execution)	**Section-II (After Execution)**
Purpose:	Execution History:
To validate that the system can successfully predict portfolio risk based on asset volatility.	The test case was executed successfully.
Pre-Condition:	Results:
Portfolio data (stocks, bonds, etc.) is already available in the database.	The system accurately predicted the risk level as ‘Moderate’.
Inputs:	Any other observations:
Stock: TCS, Reliance; Investment: ₹5,00,000	None
Expected Outputs:	Post Conditions:
Risk level displayed as ‘Moderate’, risk percentage between 10–20%.	Predicted risk value stored in database for analysis.
If fails, any possible reasons:	Suggestions:
Invalid or missing asset data.	Ensure complete and correct financial data before running the prediction.
Written by: Bhoomi Gupta	Run by: Krrish Gupta
Date: 1/11/2025	Date: 1/11/2025

Conclusion

The *Profile Optimization and Investment Recommendation System* successfully integrates data-driven analytics, intelligent risk assessment, and secure user management to assist investors in making informed financial decisions. By combining advanced technologies such as data visualization, machine learning-based prediction models, and secure authentication protocols, the system offers a comprehensive platform for portfolio monitoring and optimization.

Through effective portfolio management, risk prediction, and personalized investment recommendations, users can efficiently track their investments, analyze performance trends, and mitigate potential losses. The project emphasizes the importance of data security and user privacy, ensuring all sensitive financial information is safeguarded through encryption and controlled access mechanisms.

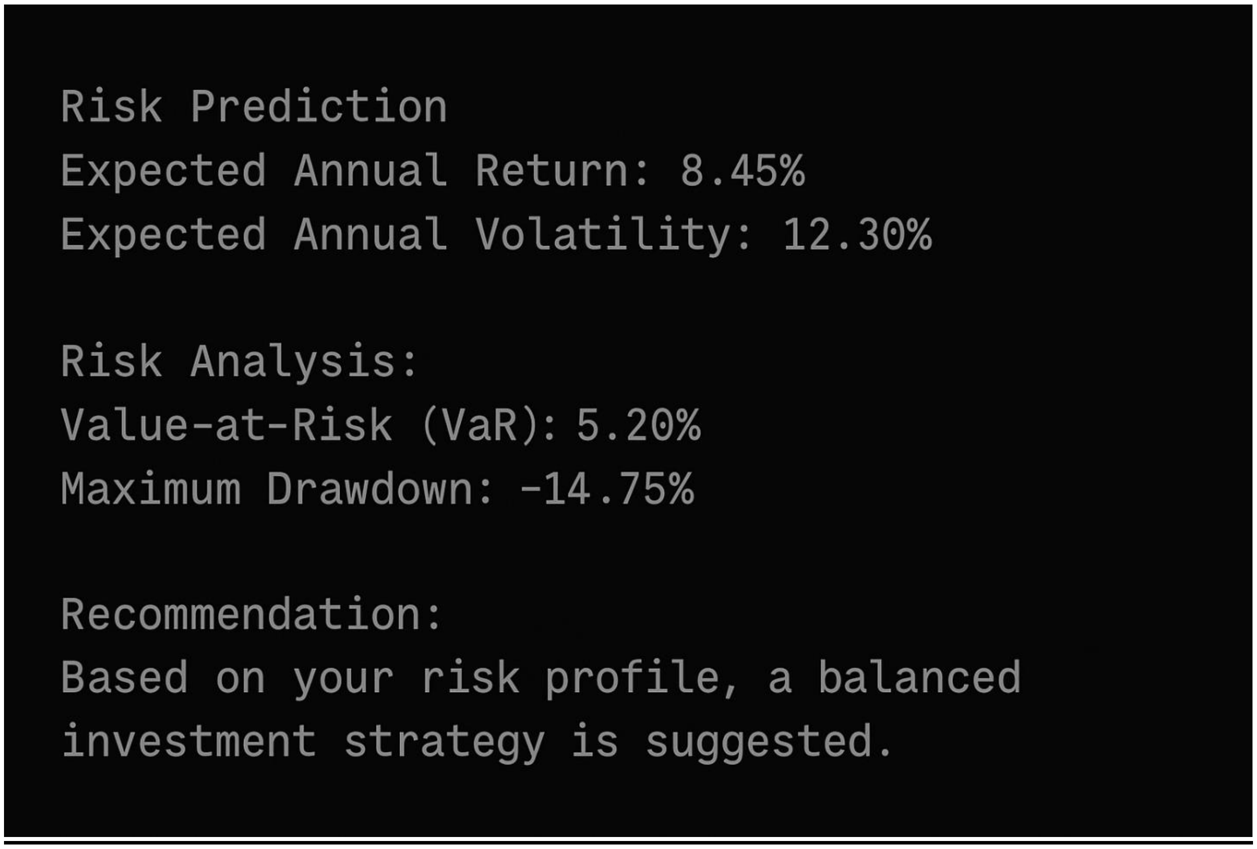
Overall, the system demonstrates how digital solutions can simplify complex investment decisions, improve financial literacy, and promote strategic wealth management. The project can be further enhanced by incorporating real-time market data integration, AI-based advisory insights, and mobile accessibility to make it even more interactive and impactful for end-users.

Future Scope

The *Profile Optimization and Investment Recommendation System* can be further enhanced and expanded in several ways to increase its efficiency, accuracy, and user engagement. Some potential future developments include:

1. **Integration of Real-Time Market Data:**
   Incorporating live stock market feeds, mutual fund NAVs, and cryptocurrency updates to provide real-time insights and dynamic portfolio recommendations.
2. **AI-Powered Investment Advisor:**
   Using artificial intelligence and machine learning algorithms to offer personalized investment advice, automate rebalancing strategies, and predict market trends more accurately.
3. **Mobile Application Development:**
   Developing a mobile-friendly version or dedicated app to allow users to manage their portfolios and receive instant notifications about performance and risk alerts.
4. **Enhanced Security Features:**
   Implementing biometric authentication, multi-factor security layers, and blockchain-based transaction logging to strengthen user data protection and transparency.
5. **Integration with Banking and Payment Systems:**
   Enabling users to link bank accounts and perform transactions directly from the platform for a seamless investment experience.
6. **Social & Community Investment Insights:**
   Adding a feature that allows users to view trending investment patterns, expert opinions, and community-driven investment recommendations.
7. **Comprehensive Financial Planning Tools:**
   Expanding the system to include tax-saving plans, retirement planning, insurance recommendations, and loan optimization options for holistic financial management.

Appendix



```
Risk Prediction
Expected Annual Return: 8.45%
Expected Annual Volatility: 12.30%

Risk Analysis:
Value-at-Risk (VaR): 5.20%
Maximum Drawdown: -14.75%

Recommendation:
Based on your risk profile, a balanced
investment strategy is suggested.
```