# CHAPTER 4: METHODOLOGY

## 4.1 System Requirements

### 4.1.1 Hardware Requirements (Minimum)

- Processor: Dual-core CPU or better
- RAM: 4 GB (8 GB recommended)
- Storage: 1 GB free space
- Internet: Required (market data + news sentiment)

### 4.1.2 Software Requirements

- Operating System: Windows / Linux / macOS
- Python: 3.7+
- Libraries: as listed in requirements.txt

## 4.2 Data Collection and Preprocessing

### 4.2.1 Market Data Acquisition

For a selected ticker symbol, the system:

- checks local {TICKER}.csv cache and reuses it if up-to-date,
- otherwise downloads ~2 years of historical price data using yfinance,
- falls back to Alpha Vantage if the primary source fails.

### 4.2.2 Data Cleaning

- Missing values are removed (dropna).
- Relevant fields such as Close are used as the primary signal.

## 4.3 Linear Regression Forecasting Method

The system implements a Linear Regression model following established methodologies in the literature [1], [2], [3]. The forecasting approach includes:

- A forecast horizon of 7 days is selected for short-term prediction.
- A target column is created using shift operation: Close after n days.
- Feature engineering extracts relevant price patterns from historical data.
- Standard scaling (StandardScaler) is applied to normalize features [2].
- Linear Regression model is trained on 80% of historical data.
- The model forecasts the next 7 days of closing prices.
- Model performance is evaluated using RMSE on a held-out test split (20% of data).
- A 4% adjustment factor is applied to predictions to account for market trends.

This approach balances prediction accuracy with computational efficiency, making it suitable for real-time web applications [1], [4].

## 4.4 Sentiment Analysis Method

- Recent headlines/news are gathered for the given ticker.
- Sentiment scores are computed and aggregated.
- Final outputs include polarity and distribution counts (positive/negative/neutral).

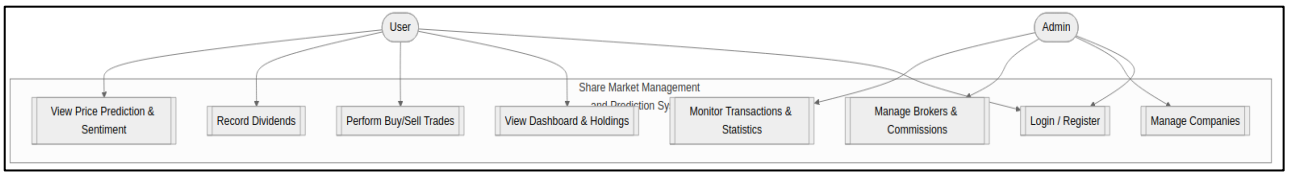
## 4.5 System Architecture and Design

### 4.5.1 Architectural Overview

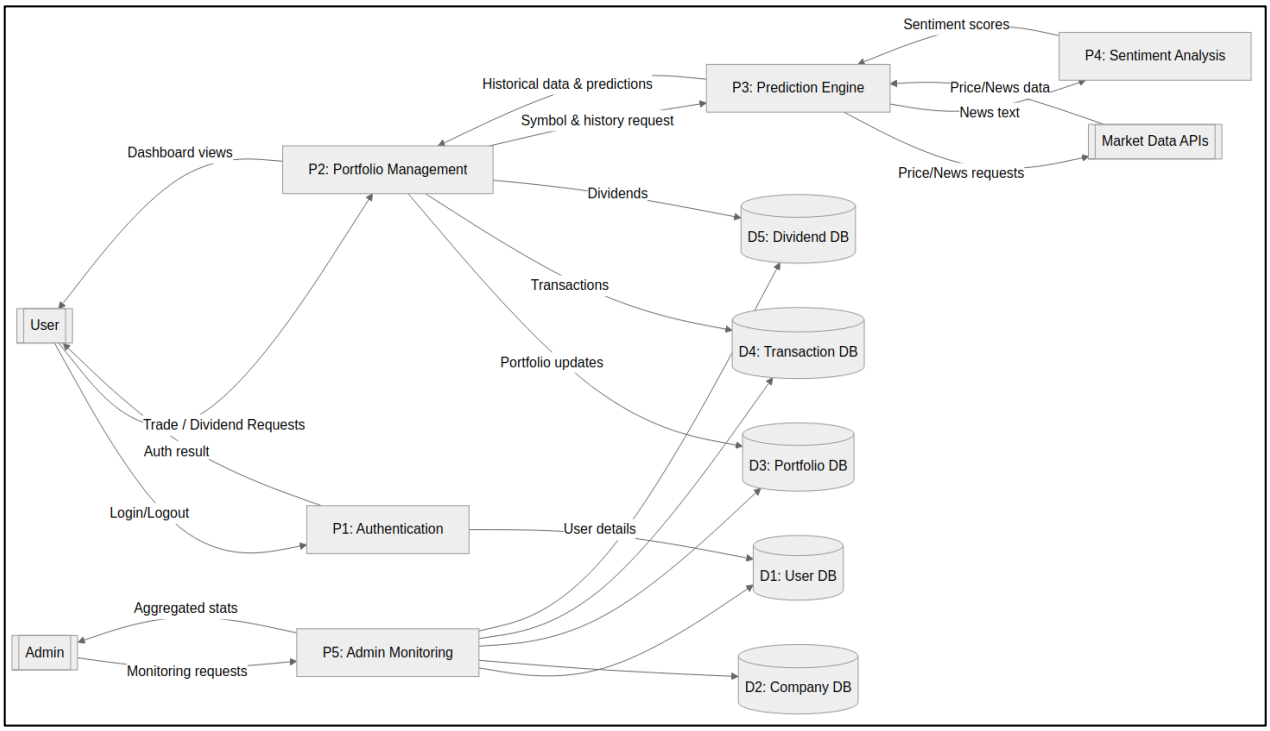
The application follows a three-tier architecture:

- Presentation Layer: HTML templates and dashboards
- Application Layer: Flask routes and business logic
- Data Layer: SQLite database via SQLAlchemy

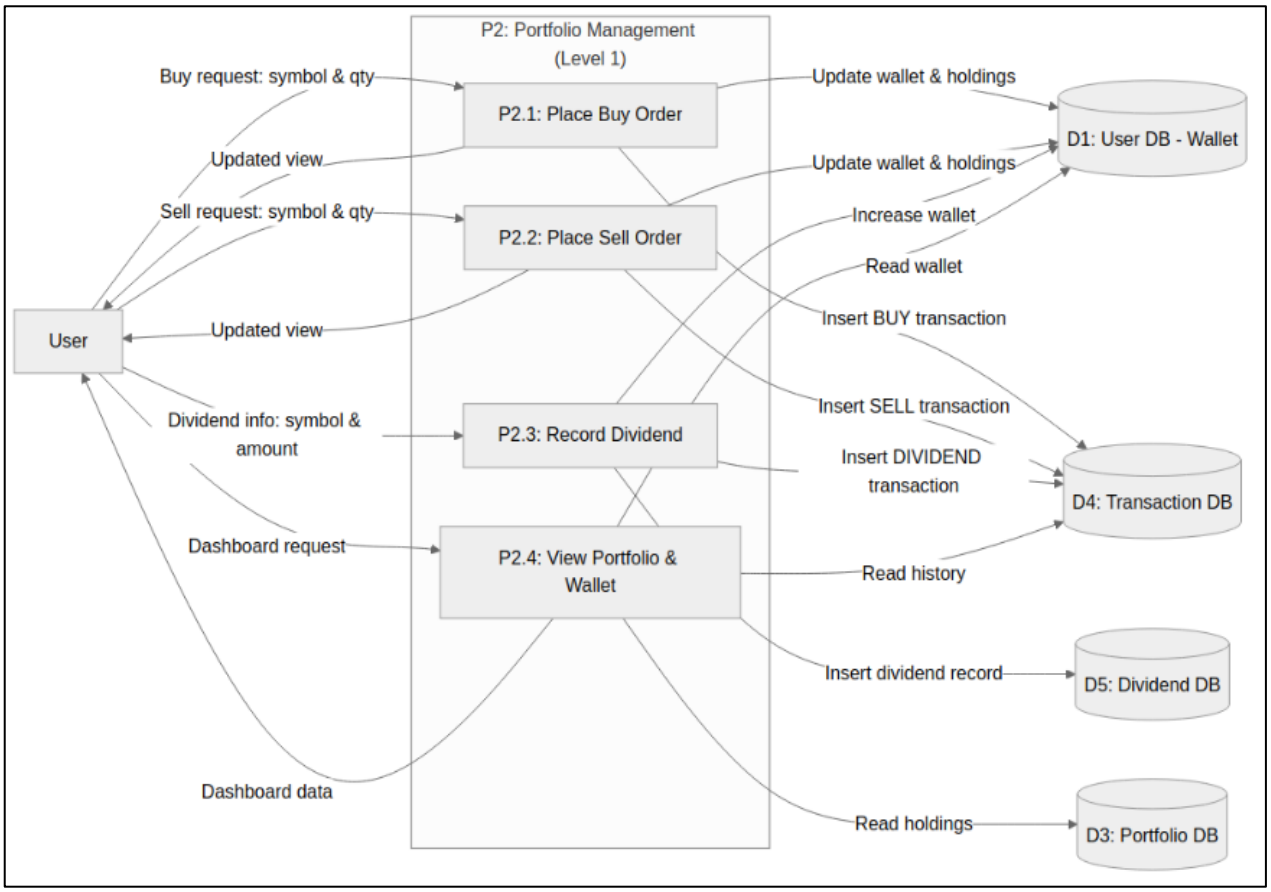
### 4.5.2 Use Case Diagram



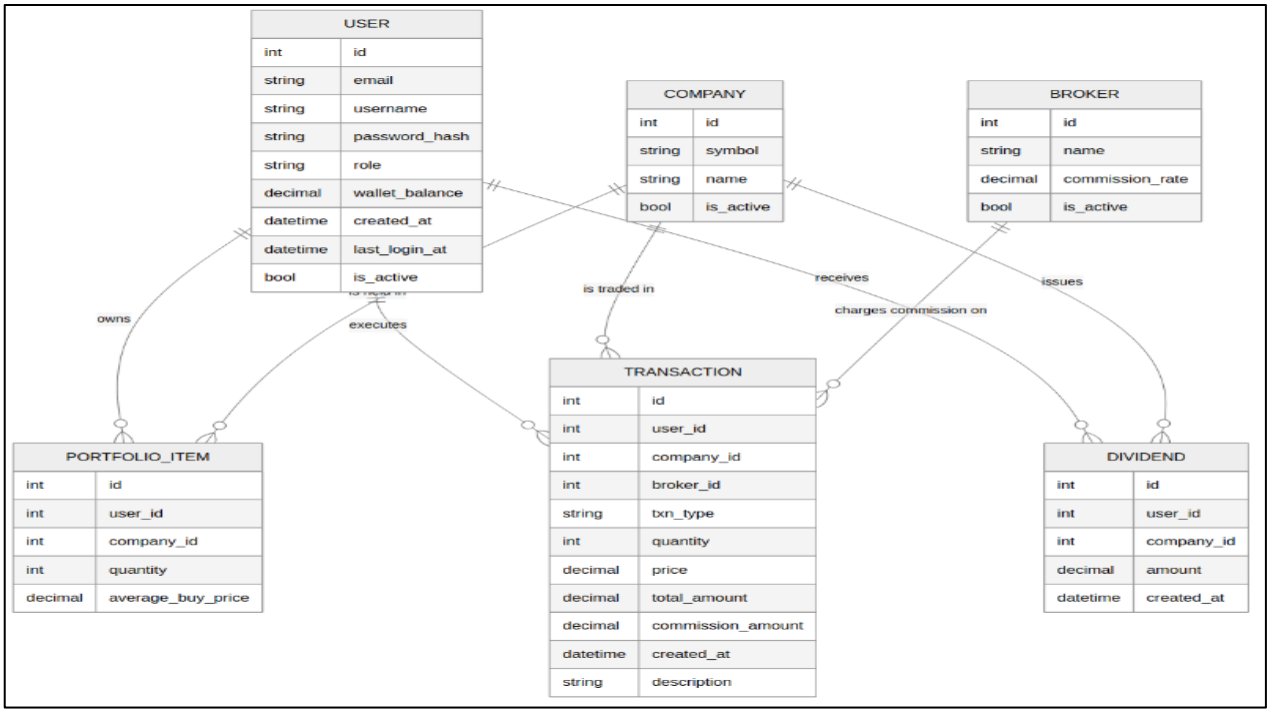

### 4.5.3 Data Flow Diagram (Level 0)



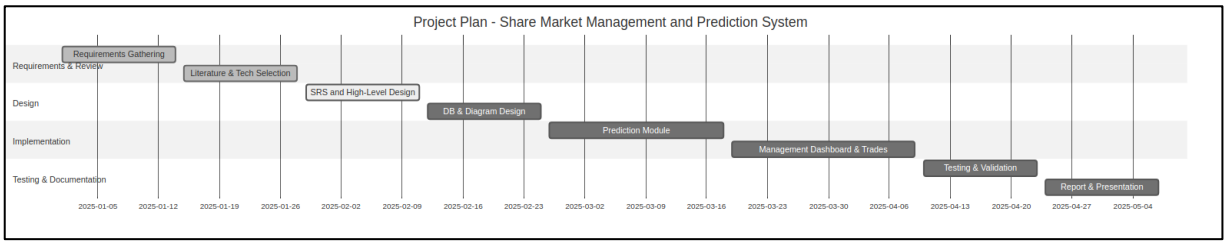
### 4.5.4 Portfolio DFD (Level 1)




### 4.5.5 Database Design (ER Diagram)



### 4.5.6 Project Plan (Gantt Chart)




## 4.6 Implementation Overview

- main.py: Flask app entry point (routes, DB models, prediction orchestration)
- news_sentiment.py: sentiment analysis module
- templates folder: prediction results, user dashboard, admin dashboard
- static folder: styling and client-side scripts (including charts)