# INVESTMENT TRACKER

## A MINI PROJECT REPORT


**Submitted by**

**Fardeen Jamal Jafer** **230701086**

**Dhanush M** **230701070**


In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE


RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105


2023 - 24

# BONAFIDE CERTIFICATE

Certified that this project report "**INVESTMENT TRACKER**" is the bonafide work of **"Fardeen Jamal Jafer (230701086), Dhanush M (230701070)"** who carried out the project work under my supervision

**Submitted for the Practical Examination held on _____**

**SIGNATURE**

**Mrs.Divya.M**
**Assistant Professor,**
**Computer Science and Engineering,**
**Rajalakshmi Engineering College**
**Thandalam, Chennai - 602 105**

**SIGNATURE**

**Mr.Raghu**
**Assistant Professor ,**
**Computer Science and Engineering,**
**Rajalakshmi Engineering College,**
**Thandalam, Chennai - 602 105**

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

**ABSTRACT**:

The Investment Tracker System is a comprehensive and user-friendly application designed to help individuals manage their stock investments. This system enables users to track their portfolio performance, monitor real-time stock prices, and efficiently manage their investment records. By providing a clear and organized interface, the system allows users to easily view the status of their investments, calculate total portfolio value, and make informed decisions about buying or selling stocks.

With the integration of real-time data from the Alpha Vantage API, users can access up-to-date stock prices and analyze market trends. The system supports features like adding, updating, and selling investments, ensuring that users can manage their portfolios effectively. The portfolio is displayed through an interactive chart, providing visual insights into the investment's growth and value over time.

The system also provides a secure login feature, allowing users to create accounts, access their personalized investment data, and securely store their information in a MySQL database. This ensures that users can easily manage their investments, while keeping sensitive data safe.

Additionally, the Investment Tracker System fosters a streamlined user experience by offering a user-friendly interface, complete with intuitive navigation and personalized features. Whether it's tracking individual investments, analyzing portfolio growth, or keeping up with stock market fluctuations, the system serves as a valuable tool for investors looking to make informed financial decisions.

**TABLE OF CONTENTS**

**Chapter 1**

# INTRODUCTION

## 1.1 INTRODUCTION

Investing in stocks is an essential aspect of personal financial management, allowing individuals to build wealth, plan for retirement, and achieve long-term financial goals. However, managing a diverse portfolio can be complex and time-consuming, especially without the right tools. The Investment Tracker System is designed to simplify the process of tracking stock investments, offering a seamless way for users to monitor their portfolio performance, access real-time stock data, and make informed decisions. This system bridges the gap between users and the dynamic stock market, ensuring that users can manage their investments efficiently and stay up-to-date with the latest market trends.

At the heart of the Investment Tracker System is a comprehensive database that stores user investment details, including company ticker symbols, quantities, and total values. This organized repository allows users to view and manage their investments easily. The system also integrates real-time stock prices through the Alpha Vantage API, providing users with accurate and current data to evaluate the performance of their investments. Additionally, the system features a portfolio chart that visually represents the growth and value of investments over time, enabling users to quickly assess their financial progress.

Developed using Java, MySQL, and the Alpha Vantage API, the Investment Tracker System leverages these technologies to create a robust and user-friendly platform. Java serves as the backbone for the application, handling user interactions and displaying investment data in an intuitive manner. MySQL is used for efficient storage and retrieval of user and investment information, ensuring data integrity and security. The Alpha Vantage API provides the real-time stock data, while the graphical user interface (GUI) offers an engaging, interactive experience for users to manage their investments.

This report will detail the development process, system architecture, and the technologies employed in creating the Investment Tracker System. The objective is to demonstrate how the integration of these technologies results in a comprehensive solution for stock investment management, ultimately helping users make informed financial decisions and achieve their investment goals.

## 1.2 Objectives

- To create a centralized database for managing user investment details, including stock tickers, quantities, and total investment values.
- To enable real-time access to accurate stock prices and investment performance using the Alpha Vantage API.
- To provide a visual representation of portfolio growth through interactive charts, helping users track their financial progress over time.
- To allow users to easily add, update, and track their stock investments, making the management of their portfolio more efficient.
- To ensure secure handling of user data, including investment information, by utilizing a robust MySQL database.
- To offer a user-friendly interface that simplifies the process of managing investments, making it accessible to both novice and experienced investors.
- To provide personalized investment insights and enable users to make informed financial decisions based on up-to-date market information.

## 1.3 Modules

### 1. User Registration and Login Module

The User Registration and Login Module is responsible for managing user authentication and account creation. This module captures user details such as name, email, and password, ensuring that only authorized users can access the system. It also includes functionality for secure password storage and verification. The login process ensures users can easily access their personalized investment details, with role-based access control to maintain privacy and security.

### 2. Investment Entry and Management Module

The Investment Entry and Management Module allows users to input and manage their stock investments. Users can add new investments by specifying the company’s ticker symbol and the number of shares owned. The module then fetches real-time stock prices from the Alpha Vantage API to calculate the total value of the investment. Users can also update or delete existing investments, making it easy to maintain an up-to-date portfolio.

### 3. Portfolio Tracking and Visualization Module

The Portfolio Tracking and Visualization Module provides users with a comprehensive view of their investment portfolio. It retrieves data from the database and displays it in an easy-to-understand format, showing current values, quantity of shares, and the total value of each investment. Additionally, this module includes graphical charts to help users track their portfolio's growth over time and analyze trends in their investments.

### 4. Stock Price Integration Module

The Stock Price Integration Module connects to the Alpha Vantage API to fetch real-time stock prices. It allows the system to display up-to-date stock values on the investment entry page and in the portfolio overview. The module ensures the accurate calculation of the total investment value by multiplying the stock's current price with the quantity of shares. This feature helps users make informed decisions based on live market data.

### 5. Investment Calculator Module

The Investment Calculator Module allows users to calculate the potential value of their investments under different market conditions. Users can enter various parameters (e.g., predicted stock prices, future quantities) to simulate investment growth or calculate profits and losses. This module helps users make informed investment decisions and understand the financial implications of their actions.

### 6. Admin Dashboard Module

The Admin Dashboard Module provides administrators with an overview of all user investments. It includes features for monitoring overall system performance, reviewing users' investment portfolios, and generating reports. This module allows admins to view investment trends and take necessary actions to optimize system operations. It also supports role-based access to ensure that administrative functions are restricted to authorized personnel.

### 7. Database Management Module

The Database Management Module is responsible for storing and organizing all investment and user data. This module uses MySQL to manage user accounts, investment records, and other system-related data. It ensures data integrity, security, and efficient retrieval by using structured queries and indexing, making the system reliable and responsive.

**Chapter 2**

# Survey Of Technologies

## 2.1 Software Description

The Investment Tracker system uses Java for the frontend and MySQL for the backend to provide a seamless user experience in tracking investments. It allows users to manage their stock portfolio, view real-time stock data through the Alpha Vantage API, and calculate the value of their investments. The system features a user-friendly interface built with Java Swing, ensuring easy navigation and efficient management of investment records. The MySQL database ensures secure storage and retrieval of user and investment data, while the Alpha Vantage API delivers live market updates to help users make informed decisions.

## 2.2 Languages

The system is primarily developed using Java for the frontend, providing an interactive user interface with Java Swing. MySQL is used for the backend, ensuring secure and efficient data storage and retrieval.

### 2.2.1 Java

Role: Java is the primary programming language used for the frontend of the Investment Tracker system. It is employed to create an interactive and responsive user interface using Java Swing.

Usage: Java handles user interactions on the frontend, including processing input from forms, displaying stock information, and managing user login credentials. It interacts with the MySQL backend for data storage and retrieval.

Advantages:

- Cross-Platform Compatibility: Java is platform-independent, ensuring that the application runs on various operating systems without modification.
- Robust GUI Capabilities: Java Swing allows for the creation of visually appealing and highly customizable graphical user interfaces.
- Object-Oriented: Java’s object-oriented structure promotes code reusability and maintainability, which is ideal for scalable applications.

**MYSQL**

Role: MySQL is the database management system used for storing and managing user data and investment information in the Investment Tracker system.

Usage: MySQL handles the creation and management of databases, tables, and queries for the application. It stores user information, investment details, and historical data on stocks and portfolio performance.

Advantages:

- Efficient Data Retrieval: MySQL uses optimized queries and indexing to ensure fast retrieval of large amounts of data.
- Security: It offers strong data security features, such as user access control and data encryption, which ensures sensitive information is protected.
- Scalability: MySQL is capable of handling large volumes of data, making it suitable for systems that expect growth.

# Chapter 3 REQUIREMENTS AND ANALYSIS

## 3.1 REQUIREMENT SPECIFICATION

### 3.1.1 Functional Requirements

**User Authentication and Authorization**

- User Registration and Login: Allow users to register, create accounts, and log in securely to access their investment portfolios.
- Role-Based Access Control: Define user roles (e.g., investor, admin) and assign specific permissions to ensure data privacy and security.

**Investment Management**

- Investment Tracking: Allow users to add, update, and view investments with company name, ticker symbol, quantity, and current stock price.
- Portfolio Overview: Provide a dashboard where users can view their portfolio performance, including the total investment value and individual stock details.
- Sell Investment: Enable users to sell investments by entering the stock symbol, quantity, and selling price.

**Stock Price Integration**

- Real-Time Stock Prices: Integrate with an external stock API (such as Alpha Vantage) to fetch real-time stock prices and update portfolio values accordingly.
- Price Updates: Automatically update the stock prices and calculate new portfolio values at regular intervals or upon user request.

**Investment Calculator**

- Value Calculation: Allow users to calculate the total value of their investments based on current stock prices, investment quantity, and other relevant data.
- Profit/Loss Tracking: Provide a feature to track profit or loss for each investment based on the current stock price compared to the purchase price.

**User Profile Management**

- Profile Creation and Update: Allow users to create and update their profiles, including their personal details and investment preferences.
- Password Management: Enable users to change their passwords securely.

**Admin Dashboard**

- Comprehensive Overview: Provide an admin dashboard where administrators can monitor all users, view investments, and track application performance.
- Management Tools: Include tools for generating investment reports, tracking stock prices, and ensuring database integrity.

## 3.1.2 Non-Functional Requirements

**Security**

- Data Encryption: Ensure that sensitive data such as user credentials and investment details are encrypted both in transit (using HTTPS) and at rest (using AES encryption).
- Compliance: Adhere to data protection regulations (e.g., GDPR) to ensure that user data is handled securely and responsibly.

**Performance**

- Scalability: Design the system to accommodate a growing user base and increasing transaction volumes, ensuring it performs efficiently even with a large number of investments.
- Response Time: Ensure fast response times, with actions such as login, data retrieval, and updates taking less than 2 seconds under normal load conditions.

**Reliability**

- Availability: Guarantee high system availability, aiming for 99.9% uptime, with minimal downtime for maintenance or updates.
- Data Backup: Implement regular backups of the database to prevent data loss and ensure that recovery is possible in case of a failure.

**Usability**

- User-Friendly Interface: Design a simple, intuitive interface that is easy to navigate for both novice and experienced users, ensuring that users can easily manage their investments and track their portfolio.
- Accessibility: Ensure the system is accessible to all users, including those with disabilities, by adhering to web accessibility standards (WCAG 2.1).

**Maintainability**

- Modular Design: Use a modular design to ensure that the application can be easily maintained, extended, or modified with minimal impact on the overall system.
- Comprehensive Documentation: Provide thorough documentation for both developers and users, including installation guides, API documentation, and user manuals.

**Interoperability**

- System Integration: Ensure the system can integrate with external APIs (e.g., stock market APIs) for real-time data retrieval and other financial tools to enhance the functionality of the investment tracker.

## 3.2 Hardware and Software Requirements

### Hardware Requirements:

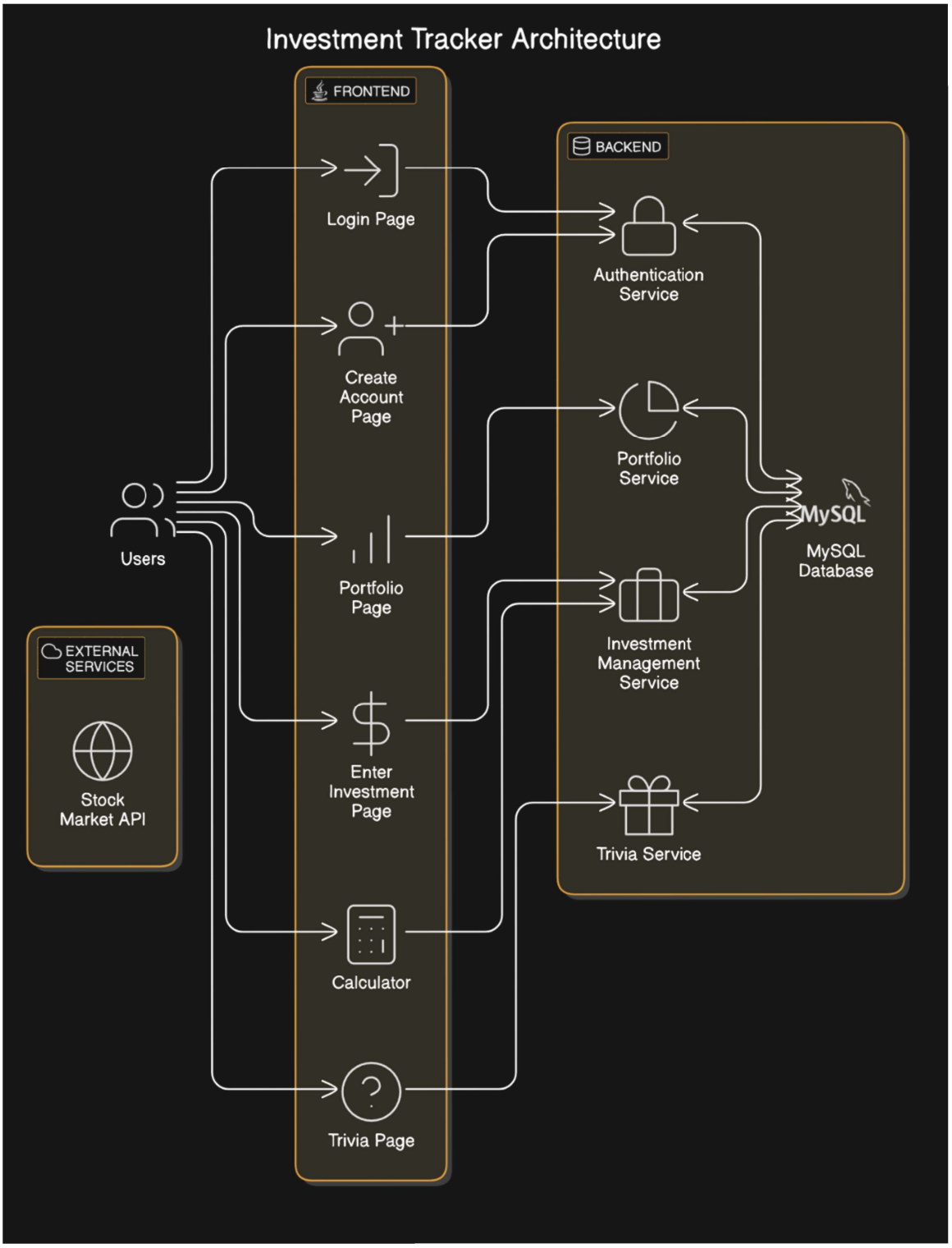
- Desktop PC or Laptop: A reliable desktop PC or laptop to run the Investment Tracker System.
- Processor: Intel® Core™ i3-6006U CPU @ 2.00GHz or equivalent for efficient processing.
- RAM: 4.00 GB or more to handle multiple concurrent users and smooth performance of the application.
- System Architecture: 64-bit operating system with x64-based processor for optimal performance.
- Monitor Resolution: 1024 x 768 or higher for clear and efficient display of the system interface.
- Input Devices: Keyboard and Mouse for user interaction.
- Server: A server with sufficient processing power, storage capacity, and network speed to support the backend operations, database, and API calls.
- Network: Reliable and stable network infrastructure for real-time stock data retrieval and user interaction.

### Software Requirements:

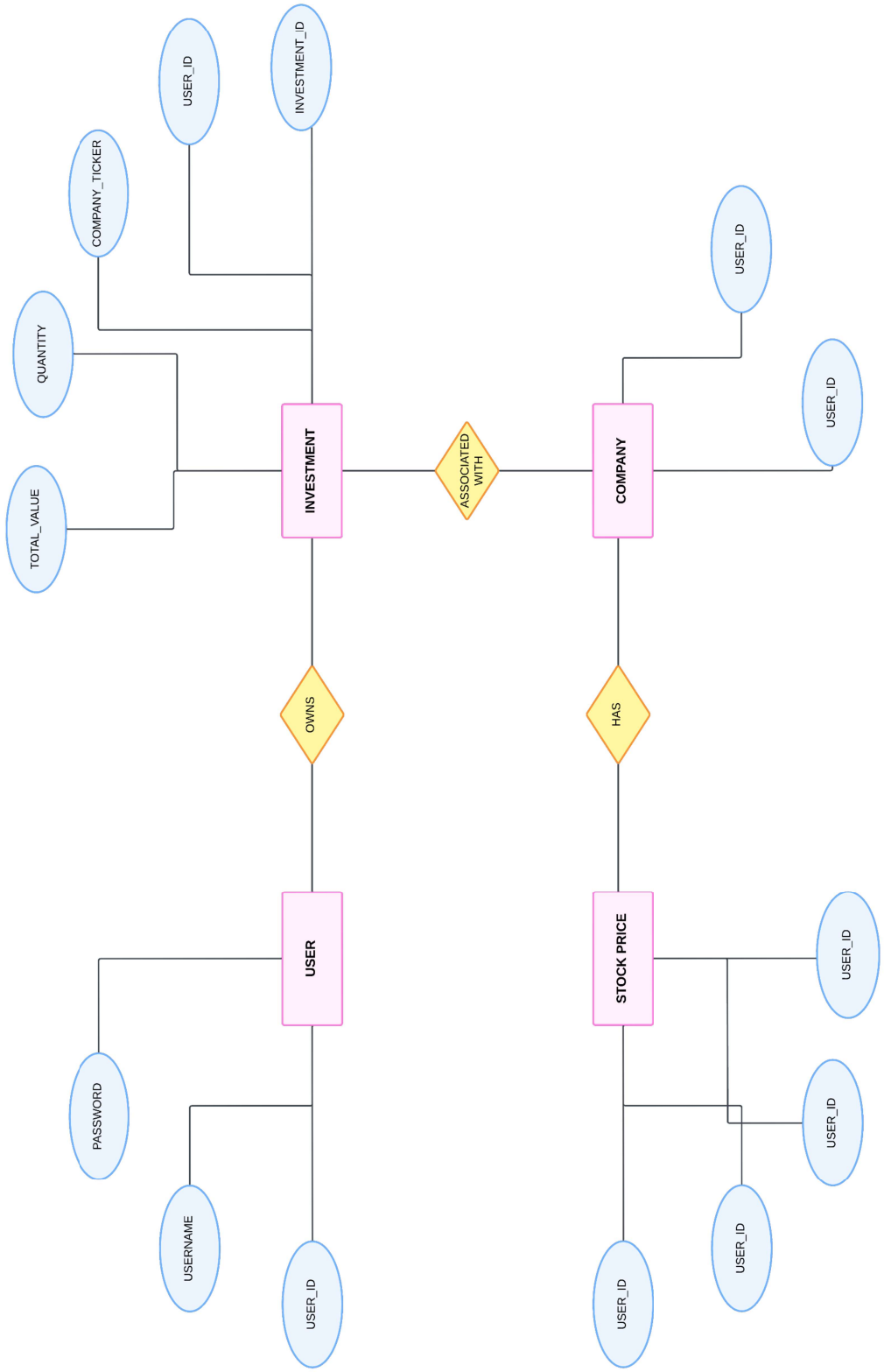
- Operating System: Windows 10 or any compatible operating system for development and deployment.
- Code Editor: NetBeans or Visual Studio Code for writing and editing the application code.
- Front End: Java (for GUI), Java Swing (for UI design), and relevant libraries for user interface components.
- Back End: MySQL for database management and integration.
- Middleware: JDBC (Java Database Connectivity) for connecting the backend database to the frontend application.
- Version Control: Git for source code management and collaboration**.**

# 3.3 Architecture Diagram

A visual diagram that provides an overall view of the Investment Tracker, identifying the external entities that interact with the system and the major data flows between these entities and the system.



## 3.4 ER DIAGRAM



## 3.5 NORMALISATION

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing a database into two or more tables and defining relationships between the tables. The steps to normalize a database table for the INVESTMENT TRACKER are as follows.

**Raw Database**

| Attribute | Datatype | Example Value |
|---|---|---|
| user_id | INT | 2 |
| Username | VARCHAR(255) | Jay |
| Password | VARCHAR(255) | Jay123 |
| Investment_id | INT | 52 |
| Company_Ticker | VARCHAR(10) | AAPL |
| Quantity | DOUBLE | 12 |
| Total_Value | DOUBLE | 5423 |
| Stock_id | INT | 43 |
| Current_Price | DOUBLE | 132.34 |

### First Normal Form (1NF)

To achieve 1NF, we need to ensure that each cell contains only a single value and each record is unique.

| User_ID | Username | Password | Investment_ID | Company_Ticker | Quantity | Total_Value | Stock_ID | Current_Price |
|---|---|---|---|---|---|---|---|---|
| 2 | Jay | Jay123 | 52 | AAPL | 12 | 5423 | 43 | 132.3 |
| 2 | Jay | Jay123 | 53 | MSFT | 10 | 2500 | 44 | 250.0 |

### Second Normal Form (2NF) with Phone Numbers

To achieve 2NF, we need to ensure that all non-key attributes are fully functionally dependent on the primary key. We'll decompose the table into multiple tables to remove partial dependencies.

**Users Table**

| user_id | Username | Password |
|---|---|---|
| 2 | Jay | Jay123 |

**Investments Table**

| Investment_id | user_id | Company_Ticker | Quantity | Total_Value |
|---|---|---|---|---|
| 52 | 2 | AAPL | 12 | 5423 |
| 53 | 2 | MSFT | 10 | 2500 |

**Stocks Table**

| Stock_id | Company_Ticker | Current_Price |
|---|---|---|
| 43 | AAPL | 132.34 |
| 44 | MSFT | 250.00 |

## CHAPTER 4

# PROGRAM CODE

```
import javax.swing.*;
import javax.swing.border.Border;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import javax.swing.table.DefaultTableModel;
import java.net.URL;
import org.json.JSONObject;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.data.category.DefaultCategoryDataset;
import java.util.Random;

public class Investments extends JFrame {
    private static final String DB_URL = "jdbc:mysql://localhost:3306/investment_tracker";
    private static final String DB_USERNAME = "root";
    private static final String DB_PASSWORD = "Investment";
    private static final String API_URL = "https://www.alphavantage.co/query?
function=GLOBAL_QUOTE&symbol=";
    private static final String API_KEY = "X3K268VVLVQPGQG9";

    private static Connection conn;
    private int userId;
    private JTabbedPane tabbedPane; // Make tabbedPane a member variable

    // Trivia questions and answers
    private String[][] triviaQuestions = {
        {"The stock market is a place where investors buy and sell ownership in publicly traded
companies.", "True"},
        {"A stock dividend is a payment made to shareholders in the form of additional shares of
stock.", "False"},
        {"Bonds are typically considered a more risky investment than stocks.", "False"},
        {"The Dow Jones Industrial Average (DJIA) is made up of 50 companies listed on the New
York Stock Exchange.", "False"},
        {"A \"bull market\" refers to a period when stock prices are generally falling.", "False"},
        {"The primary market is where stocks are first issued to the public in an Initial Public Offering
(IPO).", "True"},
        {"In a \"bear market,\" investors tend to be optimistic about the future and purchase stocks
aggressively.", "False"},
        {"A stock split increases the price of a company's shares by dividing the shares into smaller
units.", "False"},
            {"The NASDAQ Composite Index includes all stocks listed on the NASDAQ stock
exchange.", "False"},
        {"Short selling allows an investor to profit from the price increase of a stock.", "False"},
```

```
{"Blue-chip stocks are typically considered high-risk investments.", "False"},
        {"The price-to-earnings (P/E) ratio measures the profitability of a company relative to its stock price.", "True"},
        {"A stock's market capitalization is calculated by multiplying the stock's current price by the total number of shares outstanding.", "True"},
        {"A stockbroker is a person who invests their own money in the stock market on behalf of clients.", "False"},
        {"A limit order is an instruction to buy or sell a stock at a specific price or better.", "True"}
    };

    private String currentQuestion;
    private String currentAnswer;

    public Investments() {
        setTitle("Stock Investment Tracker");
        setSize(1000, 650);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setBackground(new Color(45, 45, 45));

        // Uniform color scheme
        UIManager.put("Panel.background", new Color(30, 30, 30));
        UIManager.put("Label.foreground", Color.WHITE);
        UIManager.put("Button.background", new Color(50, 150, 250));
        UIManager.put("Button.foreground", Color.WHITE);
        UIManager.put("Button.font", new Font("Arial", Font.PLAIN, 14));
        UIManager.put("TextField.background", Color.WHITE);
        UIManager.put("TextField.foreground", Color.BLACK);
        UIManager.put("Table.background", Color.WHITE);
        UIManager.put("Table.foreground", Color.BLACK);
        UIManager.put("OptionPane.background", Color.WHITE);
        UIManager.put("OptionPane.foreground", Color.BLACK);
        tabbedPane = new JTabbedPane(); // Initialize the tabbedPane
        JPanel loginPanel = createLoginPanel();
        JPanel createAccountPanel = createAccountPanel();
        JPanel companyPanel = createCompanyPanel();
        JPanel viewInvestmentsPanel = createViewInvestmentsPanel();
        JPanel triviaPanel = createTriviaPanel(); // Create trivia panel
        tabbedPane.addTab("Login", loginPanel);
        tabbedPane.addTab("Create Account", createAccountPanel);
        tabbedPane.addTab("Enter Investment", companyPanel);
         tabbedPane.addTab("Portfolio", viewInvestmentsPanel);
        tabbedPane.addTab("Trivia", triviaPanel); // Add trivia tab
          // Disable Enter Investment and View Investments tabs initially
        tabbedPane.setEnabledAt(2, false);
        tabbedPane.setEnabledAt(3, false);
        tabbedPane.setEnabledAt(4, false); // Disable Trivia tab

        add(tabbedPane);}
```

```
private JPanel createCompanyPanel() {
        JPanel companyPanel = new JPanel(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);
        gbc.fill = GridBagConstraints.BOTH;

        // Main panel for splitting the layout
        JPanel mainPanel = new JPanel(new GridLayout(1, 3)); // Two equal halves

        // Left side for investment entry
        JPanel investmentPanel = new JPanel();
        investmentPanel.setLayout(new GridBagLayout());
        investmentPanel.setBackground(new Color(30, 30, 30));

        JLabel headingLabel = new JLabel("Enter Investment", JLabel.CENTER);
        headingLabel.setFont(new Font("Serif", Font.BOLD, 22));

        JLabel companyLabel = new JLabel("Enter Company Ticker:");
        JTextField companyField = new JTextField(15);
        JLabel quantityLabel = new JLabel("Enter Quantity:");
        JTextField quantityField = new JTextField(15);
        JButton companySubmitButton = new CustomButton("Submit Investment");
        JButton sellButton = new CustomButton("Sell Investment");
        JButton logoutButton = new JButton("Logout");

        logoutButton.setBackground(Color.RED);
        logoutButton.addActionListener(e -> logout());

        gbc.gridx = 0;
        gbc.gridy = 0;
        investmentPanel.add(headingLabel, gbc);
        gbc.gridy = 1;
        investmentPanel.add(companyLabel, gbc);
        gbc.gridy = 2;
        investmentPanel.add(companyField, gbc);
        gbc.gridy = 3;
        investmentPanel.add(quantityLabel, gbc);
        gbc.gridy = 4;
        investmentPanel.add(quantityField, gbc);
        gbc.gridy = 5;
        investmentPanel.add(companySubmitButton, gbc);
        gbc.gridy = 6;
        investmentPanel.add(sellButton, gbc);
        gbc.gridy = 7;
        investmentPanel.add(logoutButton, gbc);
```

```
companySubmitButton.addActionListener(e -> {
        String ticker = companyField.getText().toUpperCase();
        String quantityStr = quantityField.getText();

        try {
            double quantity = Double.parseDouble(quantityStr);
            double price = fetchStockPrice(ticker);

            if (price != -1) {
                double totalValue = price * quantity;
                showMessageDialog(String.format("Total Investment Value: $%.2f",
totalValue));

                saveOrUpdateInvestment(userId, ticker, quantity, totalValue);
                showMessageDialog("Investment saved successfully!");
            } else {
                showMessageDialog("Failed to fetch stock price. Please try again.");
            }
        } catch (NumberFormatException ex) {
            showMessageDialog("Invalid quantity. Please enter a numeric value.");
        }
    });

    sellButton.addActionListener(e -> {
        String ticker = companyField.getText().toUpperCase();
        String quantityStr = quantityField.getText();

        try {
            double quantity = Double.parseDouble(quantityStr);
            double currentQuantity = getCurrentInvestmentQuantity(userId, ticker);

            if (quantity > currentQuantity) {
                showMessageDialog("Invalid quantity");
            } else {
                sellInvestment(userId, ticker, quantity);
                showMessageDialog("Investment sold successfully!");
            }
        } catch (NumberFormatException ex) {
            showMessageDialog("Invalid quantity. Please enter a numeric value.");
        }
    });

    JPanel separatorPanel = new JPanel();
    separatorPanel.setPreferredSize(new Dimension(0, 0));
    separatorPanel.setBackground(new Color(30, 30, 30));
      // Right side for calculator
    JPanel calcPanel = new JPanel();
    calcPanel.setLayout(new GridBagLayout());
    calcPanel.setBackground(new Color(30, 30, 30)); // Same background as other panels
```

```
JLabel calcHeadingLabel = new JLabel("Investment Calculator");
        calcHeadingLabel.setForeground(Color.WHITE);
        calcHeadingLabel.setFont(new Font("Serif", Font.BOLD, 22));
        gbc.gridx = 0;
        gbc.gridy = 0;
        calcPanel.add(calcHeadingLabel, gbc);

        JLabel investAmountLabel = new JLabel("Investment Amount:");
        JTextField investAmountField = new JTextField(15); // Uniform size
        JLabel expectedReturnLabel = new JLabel("Expected Return (%):");
        JTextField expectedReturnField = new JTextField(15); // Uniform size
        JButton calculateButton = new CustomButton("Calculate");
        JLabel resultLabel = new JLabel(""); // Label to display the result
        resultLabel.setForeground(Color.WHITE);

        // Add spacing for the result label
        gbc.gridy = 1;
        calcPanel.add(investAmountLabel, gbc);
        gbc.gridy = 2;
        calcPanel.add(investAmountField, gbc);
        gbc.gridy = 3;
        calcPanel.add(expectedReturnLabel, gbc);
        gbc.gridy = 4;
        calcPanel.add(expectedReturnField, gbc);
        gbc.gridy = 5;
        calcPanel.add(calculateButton, gbc);
        gbc.gridy = 6;
        calcPanel.add(Box.createVerticalStrut(15), gbc); // Space before result label
        gbc.gridy = 7;
        calcPanel.add(resultLabel, gbc); // Add result label to the panel

        // Add action listener for calculate button
        calculateButton.addActionListener(e -> {
            try {
                double investmentAmount = Double.parseDouble(investAmountField.getText());
                double expectedReturn = Double.parseDouble(expectedReturnField.getText()) /100;
                double estimatedReturn = investmentAmount * expectedReturn;
                resultLabel.setText(String.format("Estimated Return: $%.2f", estimatedReturn));
            } catch (NumberFormatException ex) {
                showMessageDialog("Please enter valid numbers for investment and return.");
            }
        });
          // Add both panels to the main panel
        mainPanel.add(investmentPanel);
        mainPanel.add(separatorPanel);
        mainPanel.add(calcPanel); // Add calculator panel without a separator

        // Add the main panel to the company panel
        companyPanel.add(mainPanel);
22    return companyPanel;}
```

```
private JPanel createTriviaPanel() {
        JPanel triviaPanel = new JPanel();
        triviaPanel.setLayout(new BoxLayout(triviaPanel, BoxLayout.Y_AXIS));
        triviaPanel.setBackground(new Color(30, 30, 30));

        // Randomly select a trivia question
        int randomIndex = new Random().nextInt(triviaQuestions.length);
        currentQuestion = triviaQuestions[randomIndex][0];
        currentAnswer = triviaQuestions[randomIndex][1];

        JLabel questionLabel = new JLabel(currentQuestion);
        questionLabel.setForeground(Color.WHITE);
        questionLabel.setFont(new Font("Arial", Font.BOLD, 24)); // Increased font size
        questionLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
        triviaPanel.add(Box.createVerticalGlue()); // Add vertical glue for centering
        triviaPanel.add(questionLabel);
        triviaPanel.add(Box.createVerticalGlue()); // Add vertical glue for centering

        // Create a panel for the buttons
        JPanel buttonPanel = new JPanel();
        buttonPanel.setBackground(new Color(30, 30, 30));
        buttonPanel.setLayout(new FlowLayout(FlowLayout.CENTER)); // Center buttons
        JButton trueButton = new CustomButton("True");
        JButton falseButton = new CustomButton("False");

        buttonPanel.add(trueButton);
        buttonPanel.add(falseButton);
        triviaPanel.add(buttonPanel); // Add button panel to trivia panel

        // Add action listeners
        trueButton.addActionListener(e -> processAnswer("True", triviaPanel));
        falseButton.addActionListener(e -> processAnswer("False", triviaPanel));

        return triviaPanel;
    }

    private void processAnswer(String userAnswer, JPanel triviaPanel) {
        String message;
        if (userAnswer.equals(currentAnswer)) {
            // Assuming Apple stock as an example
            String ticker = "AAPL";
            double price = fetchStockPrice(ticker);
            saveOrUpdateInvestment(userId, ticker, 1, price); // Add one stock
          // Change trivia panel background color to green
            triviaPanel.setBackground(Color.GREEN);
            message = String.format("Congratulations, you just won 1 stock of %s worth $
%.2f !!", ticker, price);
            showCelebrationEffect(triviaPanel);  // Add celebration effect
        }
```

```
else {
            // Change trivia panel background color to red for wrong answer
            triviaPanel.setBackground(Color.RED);
            message = "Better luck next time!";

            // Reset background to original color after a few seconds
            Timer timer = new Timer(3000, e -> {
                triviaPanel.setBackground(new Color(30, 30, 30));
                triviaPanel.revalidate();
                triviaPanel.repaint();
            });
            timer.setRepeats(false);
            timer.start();
        }

        showMessageDialog(message);
    }

    private void showCelebrationEffect(JPanel triviaPanel) {
        // Create a label for celebration
        JLabel celebrationLabel = new JLabel("✨ ✨ ✨ ✨");
        celebrationLabel.setFont(new Font("Serif", Font.BOLD, 100));
        celebrationLabel.setForeground(Color.WHITE);
        celebrationLabel.setAlignmentX(Component.CENTER_ALIGNMENT);

        triviaPanel.add(celebrationLabel);
        triviaPanel.revalidate(); // Refresh the panel to show the new label
        triviaPanel.repaint();

        // Optional: Add a timer to remove the celebration after a few seconds
        Timer timer = new Timer(3000, e -> {
            triviaPanel.remove(celebrationLabel);
            triviaPanel.setBackground(new Color(30, 30, 30)); // Reset to original color
            triviaPanel.revalidate();
            triviaPanel.repaint();
        });
        timer.setRepeats(false);
        timer.start();
    }

    private JPanel createLoginPanel() {
        JPanel loginPanel = new JPanel(new GridBagLayout());
        JPanel contentPanel = new JPanel(new GridBagLayout());
        contentPanel.setBackground(Color.WHITE); // White background for the content box
        contentPanel.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5)); // Padding
around content

        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);
24      gbc.fill = GridBagConstraints.HORIZONTAL;
```

```
JLabel titleLabel = new JLabel("INVESTMENT TRACKER", JLabel.CENTER);
        titleLabel.setFont(new Font("Serif", Font.BOLD, 22));
        titleLabel.setForeground(Color.BLACK); // Set heading color to black

        JLabel usernameLabel = new JLabel("Username:");
        usernameLabel.setForeground(Color.BLACK); // Set label color to black
        JTextField usernameField = new JTextField(15);
        JLabel passwordLabel = new JLabel("Password:");
        passwordLabel.setForeground(Color.BLACK); // Set label color to black
        JPasswordField passwordField = new JPasswordField(15);
        JButton submitButton = new CustomButton("Login");
        JButton createAccountButton = new CustomButton("Create Account");

        gbc.gridx = 0;
        gbc.gridy = 0;
        contentPanel.add(titleLabel, gbc);
        gbc.gridy = 1;
        contentPanel.add(usernameLabel, gbc);
        gbc.gridy = 2;
        contentPanel.add(usernameField, gbc);
        gbc.gridy = 3;
        contentPanel.add(passwordLabel, gbc);
        gbc.gridy = 4;
        contentPanel.add(passwordField, gbc);
        gbc.gridy = 5;
        contentPanel.add(submitButton, gbc);
        gbc.gridy = 6;
        contentPanel.add(createAccountButton, gbc);

        submitButton.addActionListener(e -> {
            String username = usernameField.getText();
            String password = new String(passwordField.getPassword());

            if (login(username, password)) {
                showWelcomeScreen(username); // Show welcome screen instead of dialog
                tabbedPane.setEnabledAt(2, true); // Enable Enter Investment tab
                tabbedPane.setEnabledAt(3, true); // Enable View Investments tab
                tabbedPane.setEnabledAt(4, true); // Enable Trivia tab
            } else {
                showMessageDialog("Invalid username or password.");
            }
        });
 createAccountButton.addActionListener(e -> tabbedPane.setSelectedIndex(1)); // Switch to
Create Account tab

        loginPanel.add(contentPanel);
        return loginPanel;
    }
```

```
private void showWelcomeScreen(String username) {
        JFrame welcomeFrame = new JFrame("Welcome");
        welcomeFrame.setSize(500, 300);
        welcomeFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        welcomeFrame.setLocationRelativeTo(null);
        welcomeFrame.getContentPane().setBackground(Color.WHITE);

        JLabel welcomeLabel = new JLabel("<html><div style='text-align: center;'><span
style='font-size: 30px; font-weight: bold;'>Welcome " + username +
"!</span><br><br><br><br><span style='font-size: 14px;'>Please note that all stock
market investments carry risks, including the potential loss of capital. Past performance of
stocks or any security is not indicative of future results. Always conduct thorough research
or consult with a financial advisor before making investment decisions. This application
provides information for tracking purposes only and should not be considered as financial
advice.</span></div></html>", JLabel.CENTER);
welcomeLabel.setForeground(Color.BLACK);
welcomeLabel.setFont(new Font("Arial", Font.PLAIN, 14));
welcomeFrame.add(welcomeLabel);


        welcomeFrame.setVisible(true);

        // Set a timer to close the welcome screen after 7 seconds
        Timer timer = new Timer(7000, e -> {
            welcomeFrame.dispose(); // Close the welcome frame
            tabbedPane.setSelectedIndex(2); // Switch to Enter Investment tab
        });
        timer.setRepeats(false);
        timer.start();

        // Additionally, focus the welcome frame
        welcomeFrame.addWindowFocusListener(new WindowFocusListener() {
            @Override
            public void windowGainedFocus(WindowEvent e) {
                // Bring the welcome frame to the front
                welcomeFrame.toFront();
                welcomeFrame.requestFocus();
            }

            @Override
            public void windowLostFocus(WindowEvent e) {
                // Do nothing
            }
        }); }
        private JPanel createAccountPanel() {
        JPanel createAccountPanel = new JPanel(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);
        gbc.fill = GridBagConstraints.HORIZONTAL;
```

```
JLabel headingLabel = new JLabel("Create Account", JLabel.CENTER);
        headingLabel.setFont(new Font("Serif", Font.BOLD, 22));
        JLabel newUsernameLabel = new JLabel("New Username:");
        JTextField newUsernameField = new JTextField(15);
        JLabel newPasswordLabel = new JLabel("New Password:");
        JPasswordField newPasswordField = new JPasswordField(15);
        JButton createButton = new CustomButton("Create Account");
        JButton backButton = new CustomButton("Back to Login");

        gbc.gridx = 0;
        gbc.gridy = 0;
        createAccountPanel.add(headingLabel, gbc);
        gbc.gridy = 1;
        createAccountPanel.add(newUsernameLabel, gbc);
        gbc.gridy = 2;
        createAccountPanel.add(newUsernameField, gbc);
        gbc.gridy = 3;
        createAccountPanel.add(newPasswordLabel, gbc);
        gbc.gridy = 4;
        createAccountPanel.add(newPasswordField, gbc);
        gbc.gridy = 5;
        createAccountPanel.add(createButton, gbc);
        gbc.gridy = 6;
        createAccountPanel.add(backButton, gbc);

        createButton.addActionListener(e -> {
            String newUsername = newUsernameField.getText();
            String newPassword = new String(newPasswordField.getPassword());

            if (createAccount(newUsername, newPassword)) {
                showMessageDialog("Account created successfully!");
                tabbedPane.setSelectedIndex(0); // Switch to Login tab
            } else {
                showMessageDialog("Error creating account. Please try again.");
            }
        });
          backButton.addActionListener(e -> tabbedPane.setSelectedIndex(0)); // Switch to
Login tab

        return createAccountPanel;
    }

    private JPanel createViewInvestmentsPanel() {
        JPanel viewInvestmentsPanel = new JPanel(new BorderLayout());
        JTable investmentsTable = new JTable();
        JScrollPane scrollPane = new JScrollPane(investmentsTable);
        viewInvestmentsPanel.add(scrollPane, BorderLayout.CENTER);
```

```
JLabel totalInvestmentLabel = new JLabel("Total Investment Value: $0.00",
JLabel.CENTER);
        totalInvestmentLabel.setFont(new Font("Serif", Font.BOLD, 16));
        viewInvestmentsPanel.add(totalInvestmentLabel, BorderLayout.SOUTH);

        JPanel chartPanel = createChartPanel();
        viewInvestmentsPanel.add(chartPanel, BorderLayout.NORTH);
        viewInvestmentsPanel.setBackground(new Color(30, 30, 30));

        viewInvestmentsPanel.addComponentListener(new ComponentAdapter() {
            public void componentShown(ComponentEvent e) {
                displayInvestments(investmentsTable, totalInvestmentLabel);
                updateChart(); // Update the chart whenever the panel is shown
            }
        });

        return viewInvestmentsPanel;
    }

    private JPanel createChartPanel() {
        DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        // Generate random sample data for 12 months
        double[] sp500Values = {3850.4, 3990.7, 4075.8, 4120.6, 4205.2, 4300.1, 4415.3,
            4510.8, 4630.9, 4750.6, 4825.1};

          for (int i = 0; i < sp500Values.length; i++) {
    dataset.addValue(sp500Values[i], "Stock Price", "Month " + (i + 1));
          }

        JFreeChart lineChart = ChartFactory.createLineChart(
                "S&P 500",
                "Month",
                "Value",
                dataset
        );
       ChartPanel chartPanel = new ChartPanel(lineChart);
        chartPanel.setPreferredSize(new Dimension(800, 300));
        return chartPanel   }
private void logout() {
        showMessageDialog("You have logged out.");
        userId = -1; // Reset userId
        setVisible(false);
        new Investments().setVisible(true);
    }
private static void connectToDatabase() {
        try {
            conn = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD);
        } catch (SQLException e) {
            e.printStackTrace();}   }
```

```
 private boolean login(String username, String password) {
        try {
            PreparedStatement ps = conn.prepareStatement("SELECT * FROM users WHERE
 username = ? AND password = ?");
            ps.setString(1, username);
            ps.setString(2, password);
            ResultSet rs = ps.executeQuery();

            if (rs.next()) {
                userId = rs.getInt("user_id");
                return true;
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return false;
    }

    private boolean createAccount(String username, String password) {
        try {
            PreparedStatement ps = conn.prepareStatement("INSERT INTO users (username,
 password) VALUES (?, ?)");
            ps.setString(1, username);
            ps.setString(2, password);
            ps.executeUpdate();
            return true;
        } catch (SQLException e) {
            e.printStackTrace(); }
        return false; }

    private double fetchStockPrice(String ticker) {
       try {
            URL url = new URL(API_URL + ticker + "&apikey=" + API_KEY);
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            conn.setRequestMethod("GET");
            BufferedReader in = new BufferedReader(new
 InputStreamReader(conn.getInputStream()));
            String inputLine;
            StringBuilder content = new StringBuilder();

            while ((inputLine = in.readLine()) != null) {
                content.append(inputLine); }
            in.close();

            JSONObject json = new JSONObject(content.toString());
            return json.getJSONObject("Global Quote").getDouble("05. price");
        } catch (Exception e) {
            e.printStackTrace(); }
        return -1; }
```

```
private void saveOrUpdateInvestment(int userId, String ticker, double quantity, double
totalValue) {
    try {
        PreparedStatement checkStmt = conn.prepareStatement(
                "SELECT investment_id, total_value FROM investments WHERE user_id = ?
AND company_ticker = ?");
        checkStmt.setInt(1, userId);
        checkStmt.setString(2, ticker);
        ResultSet rs = checkStmt.executeQuery();

        if (rs.next()) {
            // Retrieve the existing total value
            double existingTotalValue = rs.getDouble("total_value");
            double newTotalValue = existingTotalValue + totalValue; // Update the total value

            PreparedStatement updateStmt = conn.prepareStatement(
                    "UPDATE investments SET quantity = quantity + ?, total_value = ? WHERE
user_id = ? AND company_ticker = ?");
            updateStmt.setDouble(1, quantity);
            updateStmt.setDouble(2, newTotalValue);
            updateStmt.setInt(3, userId);
            updateStmt.setString(4, ticker);
            updateStmt.executeUpdate();
        } else {
            PreparedStatement insertStmt = conn.prepareStatement(
                    "INSERT INTO investments (user_id, company_ticker, quantity, total_value)
VALUES (?, ?, ?, ?)");
            insertStmt.setInt(1, userId);
            insertStmt.setString(2, ticker);
            insertStmt.setDouble(3, quantity);
            insertStmt.setDouble(4, totalValue);
            insertStmt.executeUpdate();
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

    private void sellInvestment(int userId, String ticker, double quantity) {
        try {
            PreparedStatement ps = conn.prepareStatement("SELECT quantity FROM
investments WHERE user_id = ? AND company_ticker = ?");
            ps.setInt(1, userId);
            ps.setString(2, ticker);
            ResultSet rs = ps.executeQuery();

            if (rs.next()) {
                double currentQuantity = rs.getDouble("quantity");
                double newQuantity = currentQuantity - quantity;
```

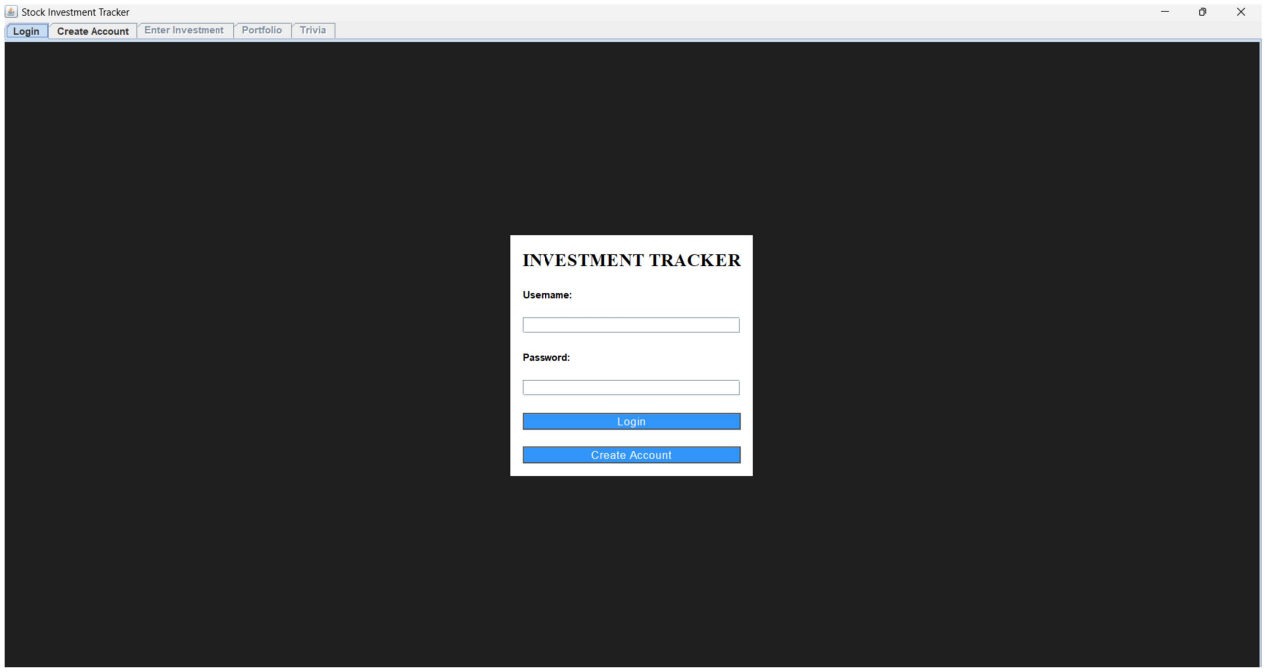
```
if (newQuantity <= 0) {
                PreparedStatement deleteStmt = conn.prepareStatement("DELETE FROM
investments WHERE user_id = ? AND company_ticker = ?");
                deleteStmt.setInt(1, userId);
                deleteStmt.setString(2, ticker);
                deleteStmt.executeUpdate();
            }
        else {
                PreparedStatement updateStmt = conn.prepareStatement("UPDATE investments
SET quantity = ? WHERE user_id = ? AND company_ticker = ?");
                updateStmt.setDouble(1, newQuantity);
                updateStmt.setInt(2, userId);
                updateStmt.setString(3, ticker);
                updateStmt.executeUpdate();
            }
          }
      } catch (SQLException e) {
          e.printStackTrace();
      }
   }

   private double getCurrentInvestmentQuantity(int userId, String ticker) {
      double currentQuantity = 0;
      try {
          PreparedStatement ps = conn.prepareStatement("SELECT quantity FROM
investments WHERE user_id = ? AND company_ticker = ?");
          ps.setInt(1, userId);
          ps.setString(2, ticker);
          ResultSet rs = ps.executeQuery();
          if (rs.next()) {
              currentQuantity = rs.getDouble("quantity");
          }
      } catch (SQLException e) {
          e.printStackTrace();
      }
      return currentQuantity;
   }

   private void displayInvestments(JTable table, JLabel totalInvestmentLabel) {
      try {
          DefaultTableModel model = new DefaultTableModel(new String[]{"Ticker",
"Quantity", "Total Value"}, 0);
          PreparedStatement ps = conn.prepareStatement("SELECT company_ticker, quantity,
total_value FROM investments WHERE user_id = ?");
          ps.setInt(1, userId);
          ResultSet rs = ps.executeQuery();
          double totalInvestment = 0;
        while (rs.next()) {
              String ticker = rs.getString("company_ticker");
              double quantity = rs.getDouble("quantity");
```

```
double totalValue = rs.getDouble("total_value");
            totalInvestment += totalValue;

            model.addRow(new Object[]{ticker, quantity, String.format("$%.2f",
totalValue)}); // Format totalValue
        }

        table.setModel(model);
        totalInvestmentLabel.setText(String.format("Total Investment Value: $%.2f",
totalInvestment)); // Format totalInvestment
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private void showMessageDialog(String message) {
    JDialog dialog = new JDialog();
    dialog.setModal(true);
    dialog.setSize(400, 100); // Increased width of dialog box
    dialog.setLocationRelativeTo(null);

    JPanel panel = new JPanel();
    panel.setBackground(Color.BLACK); // Set dialog background to black
    JLabel label = new JLabel(message);
    label.setForeground(Color.WHITE); // Set text color to white
    panel.add(label);

    JButton okButton = new CustomButton("OK"); // Use CustomButton for consistent
styling
    okButton.addActionListener(e -> dialog.dispose());
    panel.add(okButton);

    dialog.add(panel);
    dialog.setVisible(true);
}

private class CustomButton extends JButton {
    private Color normalColor = new Color(50, 150, 250);
    private Color hoverColor = new Color(30, 130, 230);
    private Color pressedColor = new Color(70, 180, 255);
    private Border normalBorder = BorderFactory.createEtchedBorder(Color.GRAY,
Color.DARK_GRAY);
    private Border hoverBorder = BorderFactory.createLineBorder(Color.WHITE);
public CustomButton(String text) {
        super(text);
        setBackground(normalColor);
        setForeground(Color.WHITE);
        setFont(new Font("Arial", Font.PLAIN, 14));
```

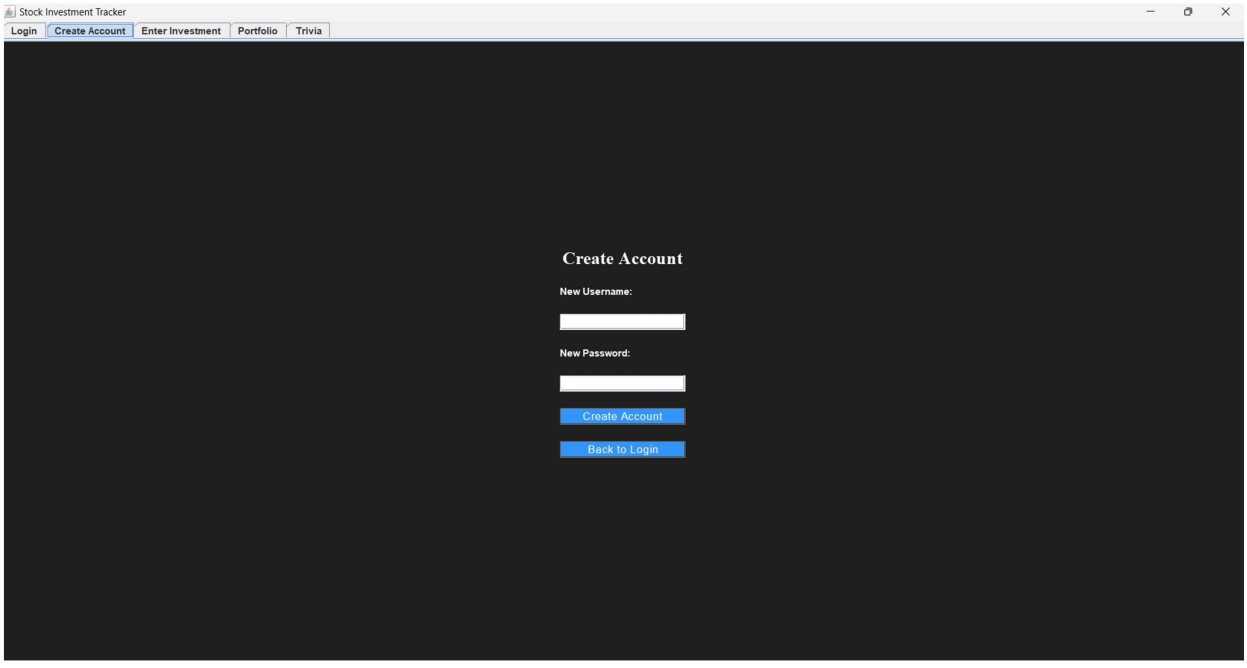
```
            setBorder(normalBorder);
            setFocusPainted(false);
            setContentAreaFilled(false);
            setOpaque(true);
            setRolloverEnabled(true);
            addMouseListener(new MouseAdapter() {
                @Override
                public void mouseEntered(MouseEvent e) {
                    setBackground(hoverColor);
                    setBorder(hoverBorder);
                }

                @Override
                public void mouseExited(MouseEvent e) {
                    setBackground(normalColor);
                    setBorder(normalBorder);
                }

                @Override
                public void mousePressed(MouseEvent e) {
                    setBackground(pressedColor);
                }

                @Override
                public void mouseReleased(MouseEvent e) {
                    setBackground(hoverColor);
                }
            });
        }
    }

    public static void main(String[] args) {
        connectToDatabase();
        SwingUtilities.invokeLater(() -> new Investments().setVisible(true));
    }
}
```

## Chapter 5 RESULTS AND DISCUSSION

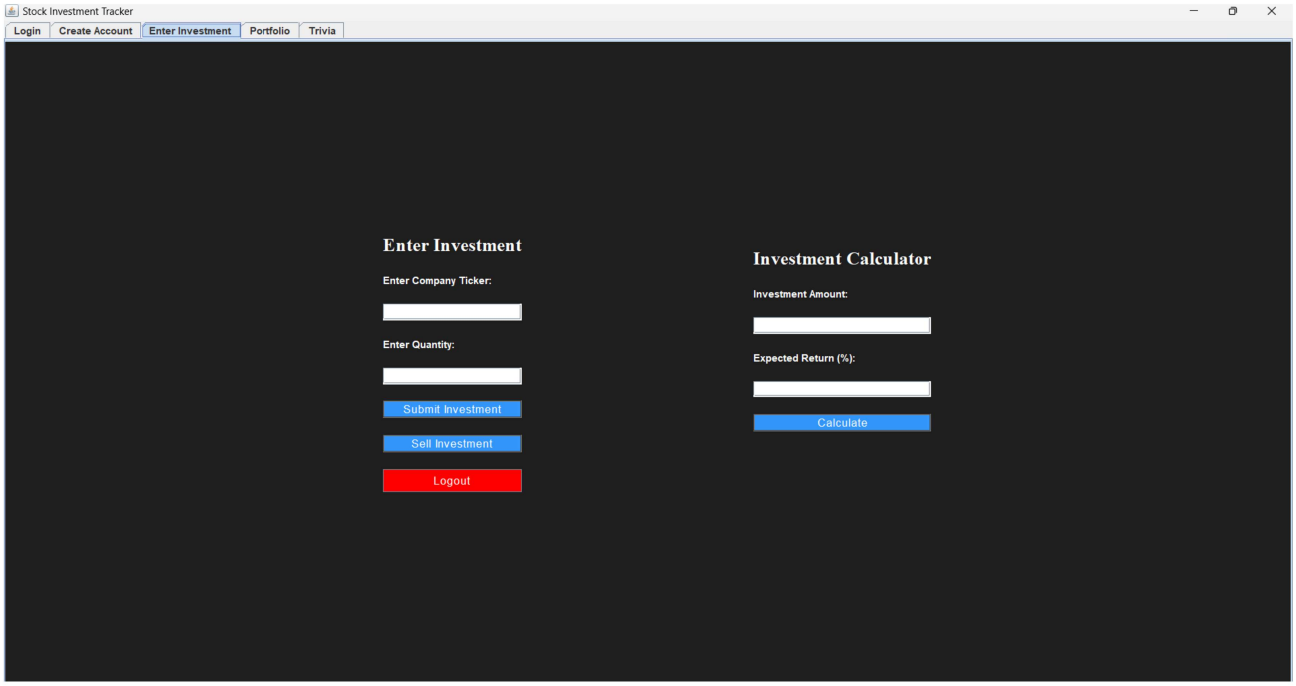
# Login Page:



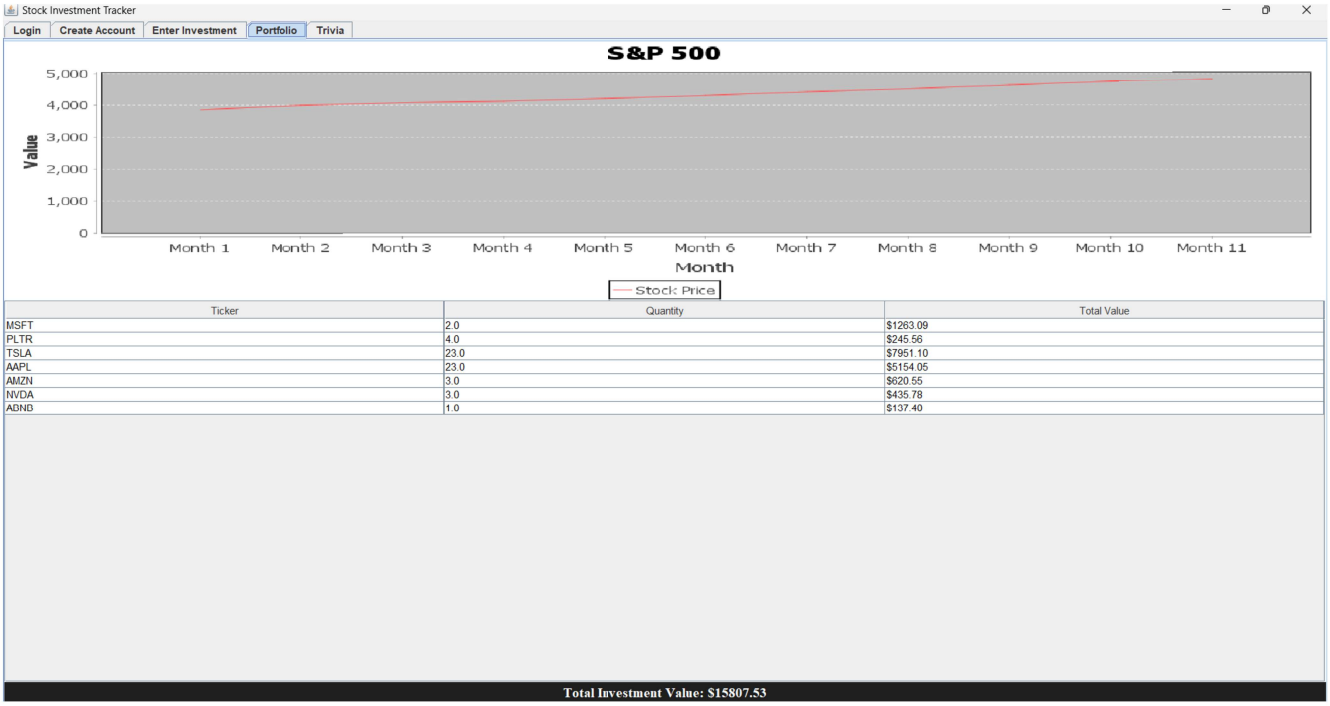

# Account Creation Page:



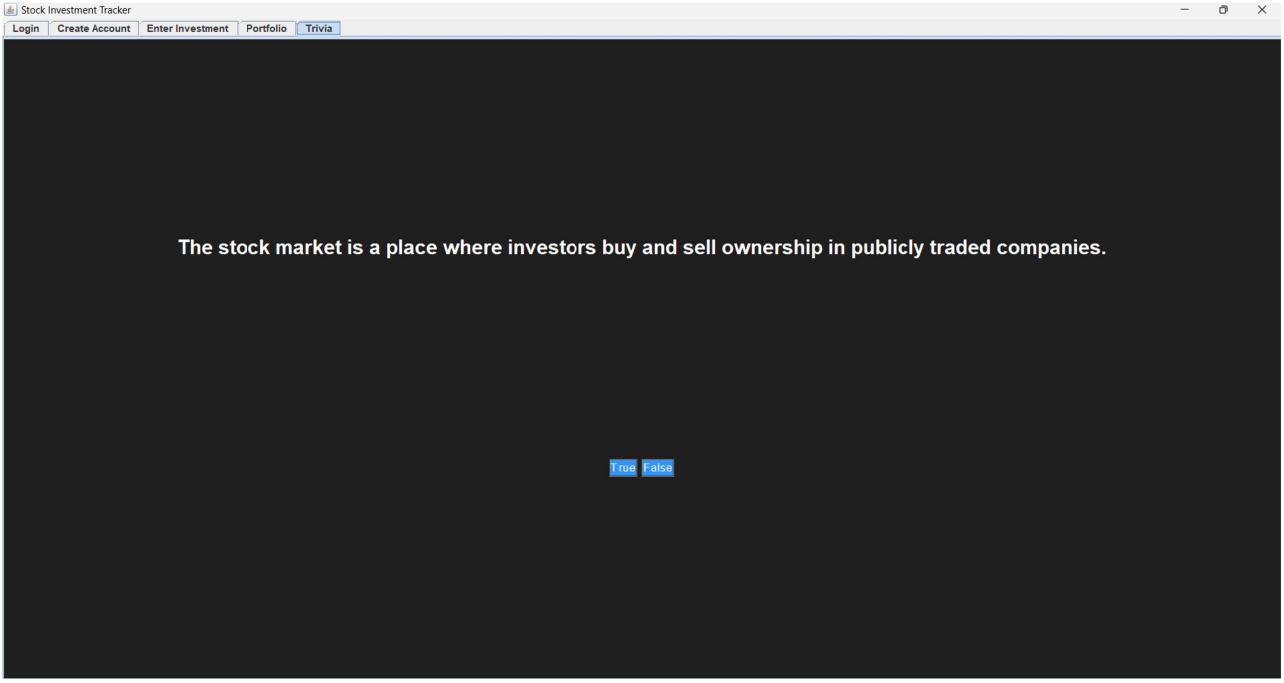
# Enter Investment Page:




# Portfolio Page:



# Trivia Page:


Stock Investment Tracker
Login
Create Account
Enter Investment
Portfolio
Trivia
The stock market is a place where investors buy and sell ownership in publicly traded companies.
True
False

# Chapter 6

## CONCLUSION

The development of the Investment Tracker application represents a significant achievement in streamlining investment management and enhancing user engagement with personal finance. Leveraging the Alpha Vantage API and MySQL for reliable data handling, the application delivers real-time investment tracking, empowering users to make informed decisions on their portfolios.

Using Java for the frontend provides a robust structure for secure data processing, while the integration of a visually appealing GUI ensures an intuitive user experience. The project meets all essential functional requirements, including user authentication, investment addition, stock valuation, and portfolio management. Non-functional requirements further ensure that the system is secure, responsive, and scalable, facilitating future upgrades and the accommodation of increased user demands.

In conclusion, the Investment Tracker application effectively addresses the complexities of personal finance management through a well-integrated set of technologies, positioning it as a powerful tool for users seeking detailed insights into their investments. This scalable, adaptable solution is well-prepared to meet evolving needs while delivering reliable and valuable performance for the end-user.

# Chapter 7 REFERENCES

## 7.1 REFERENCES

Alpha Vantage. (n.d.). Alpha Vantage API Documentation. Retrieved from https://www.alphavantage.co/documentation/.

Bartleby. (n.d.). Trivia Questions and Answers. Retrieved from https://www.bartleby.com.

Stack Overflow. (n.d.). Developer Q&A and Problem Solving Forum. Retrieved from https://stackoverflow.com.