RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CB23332
SOFTWARE ENGINEERING LAB

Laboratory Record Note Book

Name : . . . . . .

Year / Branch / Section : . . . . . .

Register No. : . . . . . .

Semester : . . . . . .

Academic Year : . . . . . .

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

RAJALAKSHMI NAGAR, THANDALAM – 602-105

BONAFIDE CERTIFICATE

NAME:_____**REGISTER NO.:** _____

ACADEMIC YEAR: 2024-25 **SEMESTER:** III **BRANCH:**_____B.E/B.Tech

This Certification is the bonafide record of work done by the above student in the

CB23332-SOFTWARE ENGINEERING - Laboratory during the year 2024 – 2025.

Signature of Faculty -in – Charge

Submitted for the Practical Examination held on _____

Internal Examiner

External Examiner

INDEX

S. No.	Name of the Experiment	Expt. Date	Faculty Sign
1.	Preparing Problem Statement		
2.	Software Requirement Specification (SRS)		
3.	Entity-Relational Diagram		
4.	Data Flow Diagram		
5.	Use Case Diagram		
6.	Activity Diagram		
7.	State Chart Diagram		
8.	Sequence Diagram		
9.	Collaboration Diagramt		
10.	Class Diagram		

EX NO:1	WRITE THE COMPLETE PROBLEM STATEMENT
DATE:	

AIM:

To prepare PROBLEM STATEMENT for any project.

ALGORITHM:

1. Initialze the System and Set Up the Database:
    - Create a database to store user details, wallet information, transactions, and market data.
2. Admin User Logs in and Sets Up System Details:
    - Integrates API keys for exchanges and blockchain explorers.
    - Configures default risk settings and system alerts.
3. Investor User Creates an Account or Logs in:
    - Adds wallet addresses or integrates exchange accounts.
    - Views real-time portfolio value and market trends.
    - Sets up alerts for specific market conditions or price thresholds.
4. System Monitors and Analyzes Portfolio:
    - Fetches real-time price data from external APIs.
    - Updates portfolio balance and calculates profit/loss.
    - Provides diversification insights and risk assessments.
5. Investor User Manages Investments:
    - Tracks gains and losses for each cryptocurrency.
    - Downloads reports for tax filing and compliance.
    - Sets and monitors stop-loss or target profit thresholds.
6. Admin Generates Periodic Reports:
    - Generates system-wide analytics, such as user activity, overall portfolio trends, and market insights.
    - Analyzes fees incurred from transactions and API usage.
7. End Program Once All Transactions and Data Entries Are Updated:
    - Backup database with transaction history, portfolio updates, and system logs.

INPUT:

1. **User Information**: Name, email, preferred currency, risk tolerance.
2. **Wallet Information**: Wallet addresses, linked exchanges, portfolio allocation.
3. **Transaction Details**: Transaction IDs, timestamps, amounts, fees, and prices.
4. **Market Data**: Real-time prices, trading volume, and volatility metrics.
5. **Alert Settings**: Price thresholds, market cap changes, or specific news alerts.

Problem:

The current approach to learning cryptocurrency trading and investment is often fraught with challenges for beginners and enthusiasts alike. New investors face risks associated with market volatility, lack of experience, and the absence of safe environments for practicing trading strategies. Complaints frequently stem from financial losses due to untested strategies, difficulty in understanding market dynamics, and the intimidating complexity of cryptocurrency platforms. To address these issues, a robust and interactive Crypto Investment Simulator is needed to provide a safe, user-friendly platform for learning and practicing crypto trading without financial risk.

Background:

Cryptocurrency investment is rapidly gaining popularity, but many aspiring traders and investors lack the resources and experience to navigate this volatile market effectively. Existing tools often focus on live trading, which exposes users to significant financial risks. Additionally, the steep learning curve associated with understanding market indicators, trading techniques, and portfolio management deters many from participating.

A **Crypto Investment Simulator** provides a controlled, risk-free environment where users can:

- Practice trading with virtual funds.
- Experiment with different strategies and understand market trends.
- Learn the functionalities of wallets, exchanges, and blockchain technology.

Such a simulator addresses these challenges by offering real-time market data, a realistic trading experience, and analytics for performance evaluation, enabling users to gain confidence and expertise before engaging in live trading

Objectives:

Analyzing Current Learning Gaps:
Conducting a detailed assessment of the challenges faced by novice traders, such as lack of market knowledge, strategy testing, and risk management skills.

Simulating Realistic Trading Environments:
Creating a virtual trading platform with features that replicate live cryptocurrency exchanges, including order books, market charts, and trading fees.

Providing Virtual Funds:
Allowing users to trade and invest using virtual currency, ensuring a safe environment for experimentation without financial risk.

Real-Time Market Data:
Integrating APIs to fetch real-time cryptocurrency price data and market trends to ensure accurate and dynamic simulations.

Educational Content:
Developing tutorials, guides, and in-app tips to educate users on market indicators, technical analysis, and trading strategies.

Performance Analytics:
Implementing tools to analyze users' trading performance, offering insights on profits, losses, and portfolio diversification.

Risk Management Simulation:
Including features to simulate and teach risk management practices, such as stop-loss orders, portfolio rebalancing, and position sizing.

User Notifications:
Sending alerts and notifications about market changes, investment milestones, and potential risks within the simulation.

Gamification:
Introducing gamified elements like leaderboards, achievements, and challenges to engage users and encourage consistent learning.

Unauthorized Activity Simulation:
Including scenarios for simulated hacks or phishing attempts to teach users about wallet security and safeguarding assets.

Scalability and Future Integration:
Designing the simulator to allow integration with live exchanges and advanced trading features as users progress.

Feedback Mechanisms:
Providing a system to collect user feedback for continuous improvement of the simulator and educational content.

Result:

EX NO:2	WRITE THE SOFTWARE REQUIREMENT SPECIFICATION DOCUMENT
DATE:	

AIM:

To do requirement analysis and develop Software Requirement Specification Sheet(SRS) for any Project.

ALGORITHM:

1. **User Authentication and Authorization**
   - Validate user credentials.
   - Assign access roles (admin, user).
2. **Portfolio Management**
   - Add, update, and remove assets in the simulated portfolio.
   - Implement search and filter functionality for cryptocurrency assets.
3. **Market Data Integration**
   - Fetch real-time or historical market data for selected cryptocurrencies.
   - Update prices and simulate gains or losses based on market data.
4. **Investment Strategy Simulation**
   - Allow users to apply different investment strategies (e.g., dollar-cost averaging, lump-sum investment).
   - Support portfolio rebalancing based on risk tolerance and investment goals.
5. **Risk Analysis and Performance Evaluation**
   - Calculate risk metrics such as volatility, Sharpe ratio, and drawdown.
   - Generate performance reports based on portfolio returns over time.
6. **Report Generation**
   - Generate detailed reports on portfolio performance, risk metrics, and strategy effectiveness.
7. **Notifications and Alerts**
   - Notify users about significant market changes, such as price volatility or portfolio threshold breaches.
   - Send reminders for portfolio rebalancing or strategy adjustments.
8. **Logout**
   - Securely log out of the system.

Basic SRS Outline

1. **Introduction**
    - Purpose: Define the simulator's purpose in allowing users to simulate crypto investments.
    - Document Conventions: Include conventions used in the document.
    - Intended Audience and Reading Suggestions: Identify the primary audience.
    - Project Scope: Define the scope and limits of the crypto simulator.
    - References: List related documents, standards, or references.
2. **Overall Description**
    - Product Perspective: The simulator as an independent application for testing crypto strategies.
    - Product Features: Overview of the main features, such as portfolio management and strategy simulation.
    - User Classes and Characteristics: Define user roles like investors and admins.
    - Operating Environment: Define system compatibility with web browsers.
    - Design and Implementation Constraints: Include security and scalability requirements.
    - Assumptions and Dependencies: List key assumptions, like users' familiarity with crypto investing.
3. **System Features**
4. **External Interface Requirements**
    - User Interfaces: Define requirements for the web UI, dashboards, and data entry forms.
    - Software Interfaces: Describe integration with APIs for market data.
    - Communication Interfaces: Specify secure data transmission protocols (e.g., HTTPS).
5. **Nonfunctional Requirements**
    - Performance Requirements: Define load handling and response times.
    - Safety Requirements: Include data recovery and disaster response.
    - Security Requirements: List encryption requirements and data protection compliance.

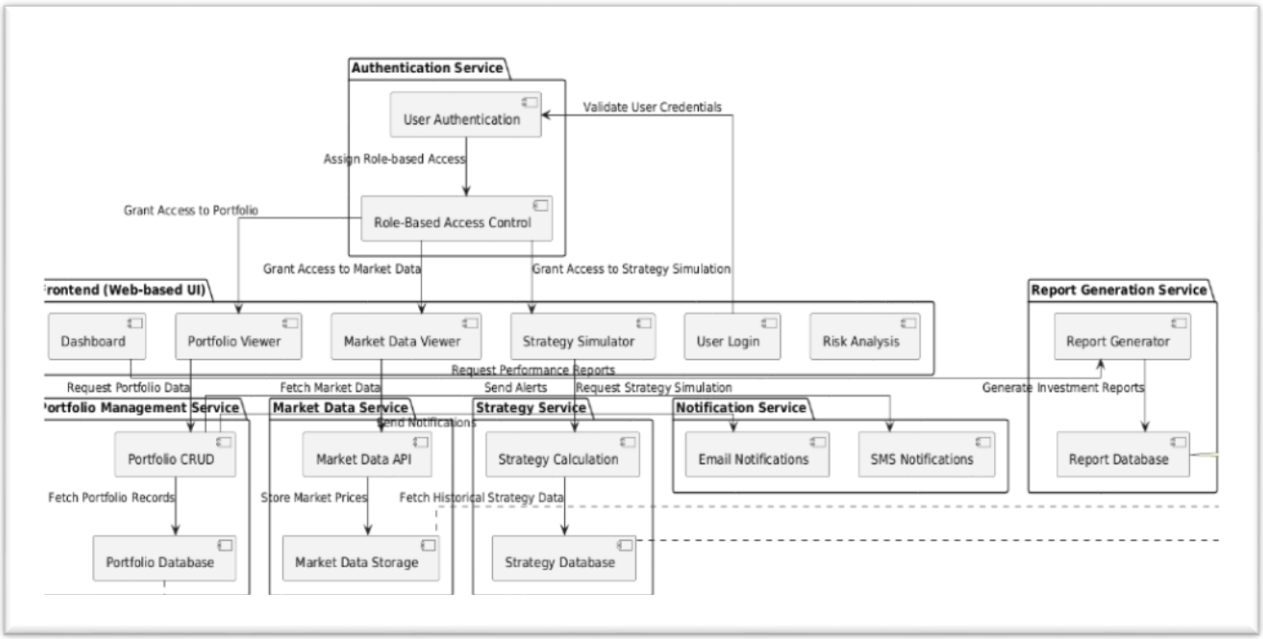
Table of Contents for SRS Document

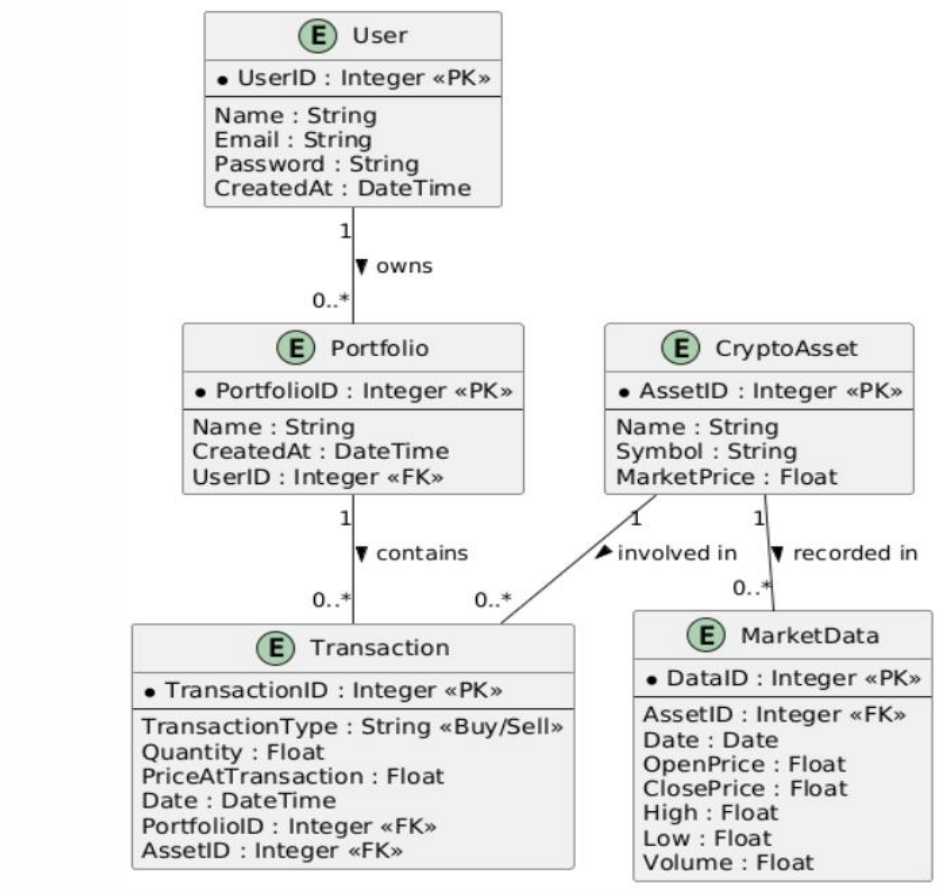
1. **Introduction**
    - Purpose: Outline the purpose of the Crypto Investment Simulator (CIS) for users to experiment with virtual investments.
    - Document Conventions: Explain document terms and identifiers.
    - Intended Audience: Define the primary users.
    - Project Scope: Describe the CIS's functionality in tracking and simulating crypto investments.
    - References: Include relevant guidelines for security and data integrity.

2. **Overall Description**
    - Product Perspective: An independent simulator with options for tracking and strategy testing.
    - Product Features: Portfolio management, market data integration, risk analysis, and notifications.
    - User Classes and Characteristics: Users (individual investors) and Admins.
    - Operating Environment: Cloud-based with browser compatibility.
    - Design Constraints: Compliance with data protection and scalability.
    - Assumptions: Users have a basic understanding of cryptocurrency.
3. **System Features**
    - Portfolio Management: Add, view, and modify simulated crypto assets.
    - Market Data Integration: Access real-time or historical data.
    - Investment Strategy Simulation: Implement and track strategies.
    - Risk and Performance Analysis: Generate performance and risk metrics.
    - Notifications: Real-time alerts on significant events.
    - Logout: Secure user logout process.
4. **External Interface Requirements**
    - User Interfaces: Web-based UI with dashboards and interactive graphs.
    - Software Interfaces: API integration for live market data.
    - Communication Interfaces: Encrypted data transmissions.
5. **Nonfunctional Requirements**
    - Performance: Support for concurrent users with low latency.
    - Safety: Data backup and secure disaster recovery.
    - Security: Data encryption in transit and at rest, compliance with security standards.

Sample Output

1. Purpose: This document specifies requirements for the Crypto Investment Simulator (CIS). The CIS provides a platform for virtual investing, enabling users to simulate cryptocurrency investment strategies, evaluate risks, and track simulated portfolio performance.
2. Document Conventions:
    - [REQ-1.0]: Functional requirements.
    - [NFR-1.0]: Non-functional requirements.
3. Intended Audience and Reading Suggestions: Developers, investors, project managers, and IT staff.
4. Project Scope: A platform for virtual investment tracking and strategy testing in cryptocurrency markets.
5. References: ISO/IEC 27001 for security, market data API documentation.
6. Overall Description: A cloud-based crypto investment simulator, with key features for strategy simulation and performance analysis.
7. System Constraints: Designed as a web-based platform hosted on a cloud server.


Authentication Service
User Authentication
Validate User Credentials
Assign Role-based Access
Role-Based Access Control
Grant Access to Portfolio
Grant Access to Market Data
Grant Access to Strategy Simulation
rontend (Web-based UI)
Dashboard
Portfolio Viewer
Market Data Viewer
Strategy Simulator
User Login
Risk Analysis
Report Generation Service
Report Generator
Request Performance Reports
Request Portfolio Data
Fetch Market Data
Send Alerts
Request Strategy Simulation
Generate Investment Reports
ortfolio Management Service
Market Data Service
Strategy Service
Notification Service
Send Notifications
Portfolio CRUD
Market Data API
Strategy Calculation
Email Notifications
SMS Notifications
Report Database
Fetch Portfolio Records
Store Market Prices
Fetch Historical Strategy Data
Portfolio Database
Market Data Storage
Strategy Database


User
• UserID : Integer «PK»
Name : String
Email : String
Password : String
CreatedAt : DateTime
1
owns
0..*
Portfolio
• PortfolioID : Integer «PK»
Name : String
CreatedAt : DateTime
UserID : Integer «FK»
CryptoAsset
• AssetID : Integer «PK»
Name : String
Symbol : String
MarketPrice : Float
1
contains
0..*
1
involved in
0..*
1
recorded in
0..*
Transaction
• TransactionID : Integer «PK»
TransactionType : String «Buy/Sell»
Quantity : Float
PriceAtTransaction : Float
Date : DateTime
PortfolioID : Integer «FK»
AssetID : Integer «FK»
MarketData
• DataID : Integer «PK»
AssetID : Integer «FK»
Date : Date
OpenPrice : Float
ClosePrice : Float
High : Float
Low : Float
Volume : Float

Result:

EX NO:3	DRAW THE ENTITY RELATIONSHIP DIAGRAM
DATE:	

AIM:

To Draw the Entity Relationship Diagram for any project.

ALGORITHM:

1. Mapping of Regular Entity Types: Identify and map each entity with a primary key.
2. Mapping of Weak Entity Types: Determine any dependent entities and map them accordingly.
3. Mapping of Binary : Map relationships with exactly one-to-one cardinality if they exist.
4. Mapping of Binary 1: Map relationships with one-to-many cardinality.
5. Mapping of Binary M: Map relationships where multiple instances in each entity relate to multiple instances in another.
6. Mapping of Multivalued Attributes: Identify attributes that could have multiple values for a single entity instance.

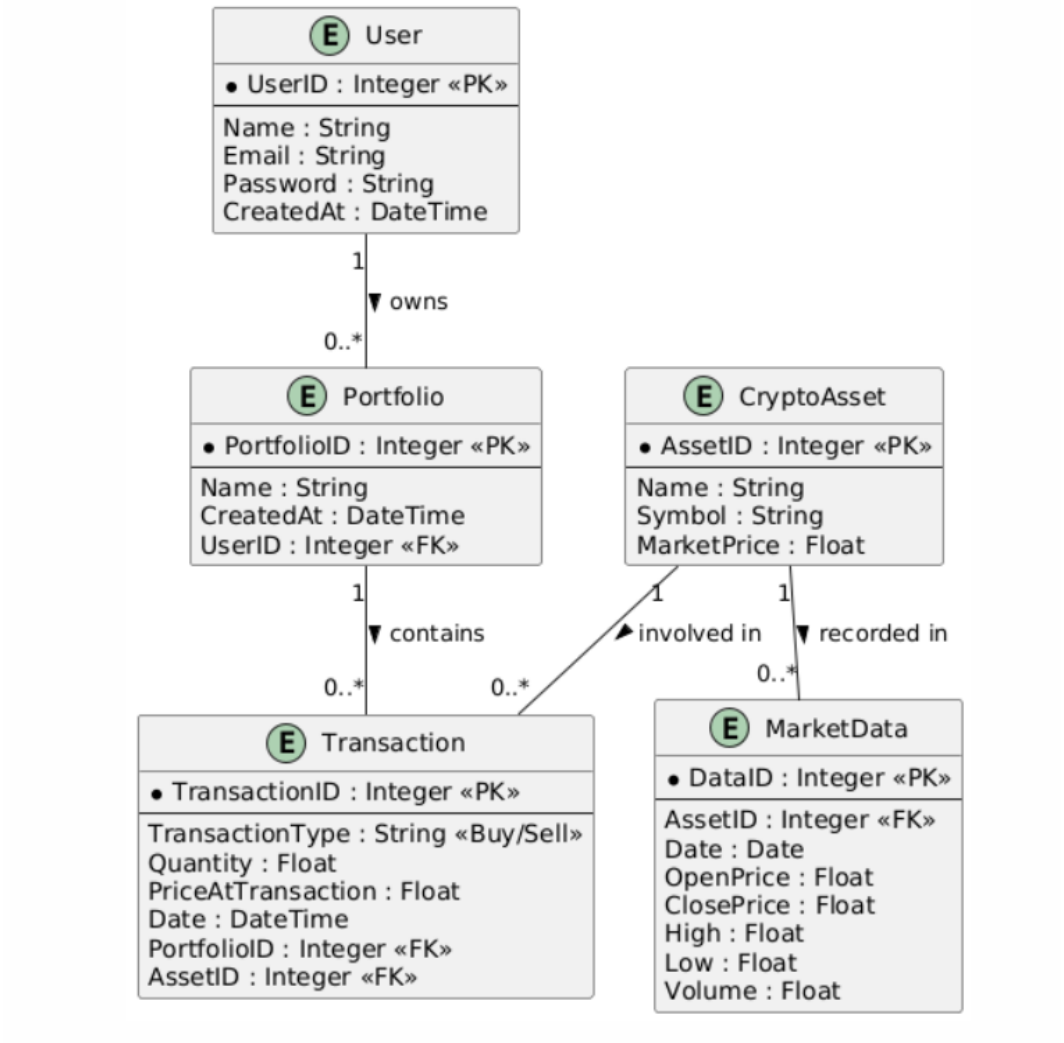
INPUT:

1. Entities and Their Primary Keys

Entity	Description	Primary Key
User	Represents users on the platform	UserID
Portfolio	Represents each user's investment portfolio	PortfolioID
CryptoAsset	Represents different cryptocurrencies	AssetID
Transaction	Represents buy/sell transactions for assets	TransactionID
MarketData	Represents historical market data for each asset	DataID

2. Attributes for Each Entity

Entity	Attributes
User	UserID, Name, Email, Password, CreatedAt
Portfolio	PortfolioID, Name, CreatedAt, UserID
CryptoAsset	AssetID, Name, Symbol, MarketPrice
Transaction	TransactionID, TransactionType (Buy/Sell), Quantity, PriceAtTransaction, Date, PortfolioID, AssetID
MarketData	DataID, AssetID, Date, OpenPrice, ClosePrice, High, Low, Volume


User
• UserID : Integer «PK»
Name : String
Email : String
Password : String
CreatedAt : DateTime
1
owns
0..*
Portfolio
• PortfolioID : Integer «PK»
Name : String
CreatedAt : DateTime
UserID : Integer «FK»
CryptoAsset
• AssetID : Integer «PK»
Name : String
Symbol : String
MarketPrice : Float
1
contains
0..*
1
involved in
0..*
1
recorded in
0..*
Transaction
• TransactionID : Integer «PK»
TransactionType : String «Buy/Sell»
Quantity : Float
PriceAtTransaction : Float
Date : DateTime
PortfolioID : Integer «FK»
AssetID : Integer «FK»
MarketData
• DataID : Integer «PK»
AssetID : Integer «FK»
Date : Date
OpenPrice : Float
ClosePrice : Float
High : Float
Low : Float
Volume : Float

Symbols Used:

- **Primary Key: Denoted by PK**
- **Foreign Key: Denoted by FK**

Result:

EX NO:4	DRAW THE DATA FLOW DIAGRAMS AT LEVEL 0 AND LEVEL 1
DATE:	

AIM:

To Draw the Data Flow Diagram for any project and List the Modules in the Application.

ALGORITHM:

1. Open Visual Paradigm or another DFD tool (e.g., Lucidchart).
2. Select a data flow diagram template.
3. Name the data flow diagram (e.g., "Crypto Investment Simulator DFD").
4. Add external entities that interact with the system.
5. Add main processes for the system's core functionalities.
6. Add data stores for each process's data storage requirements.
7. Continue to add additional items to the DFD as needed.
8. Add data flows between entities, processes, and data stores.
9. Name each data flow clearly.
10. Customize the DFD with colors and fonts if desired.
11. Add a title to the DFD and export or share as needed.

INPUT:

1. Processes:

- User Account Management: Handles user registration, login, and profile management.
- Portfolio Management: Manages user portfolios, including holdings and portfolio details.
- Transaction Processing: Manages buy/sell transactions for cryptocurrencies.
- Market Data Update: Updates and retrieves real-time or historical market data for each cryptocurrency.
- Report Generation: Generates performance reports, profit/loss statements, and transaction summaries.

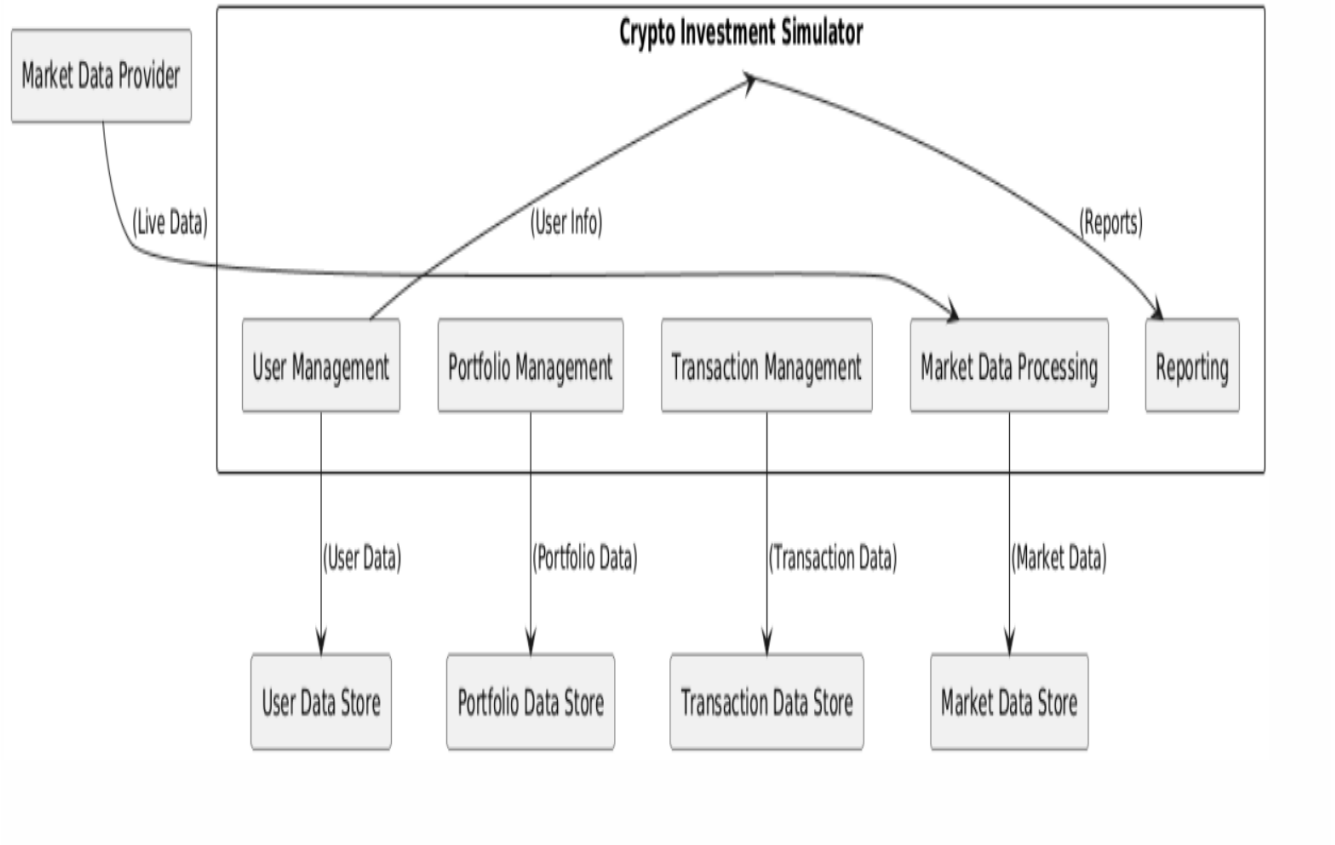
2. Data Stores:

- User Database: Stores user account details, including profiles, credentials, and preferences.
- Portfolio Database: Stores information on user portfolios, asset holdings, and history.
- Transaction Database: Stores transaction records for each buy/sell event.
- Market Data Database: Stores historical and live market data for all cryptocurrencies.
- Report Database: Stores generated reports and related data.

3. External Entities:

- Users: Represents the end-users who interact with the platform.

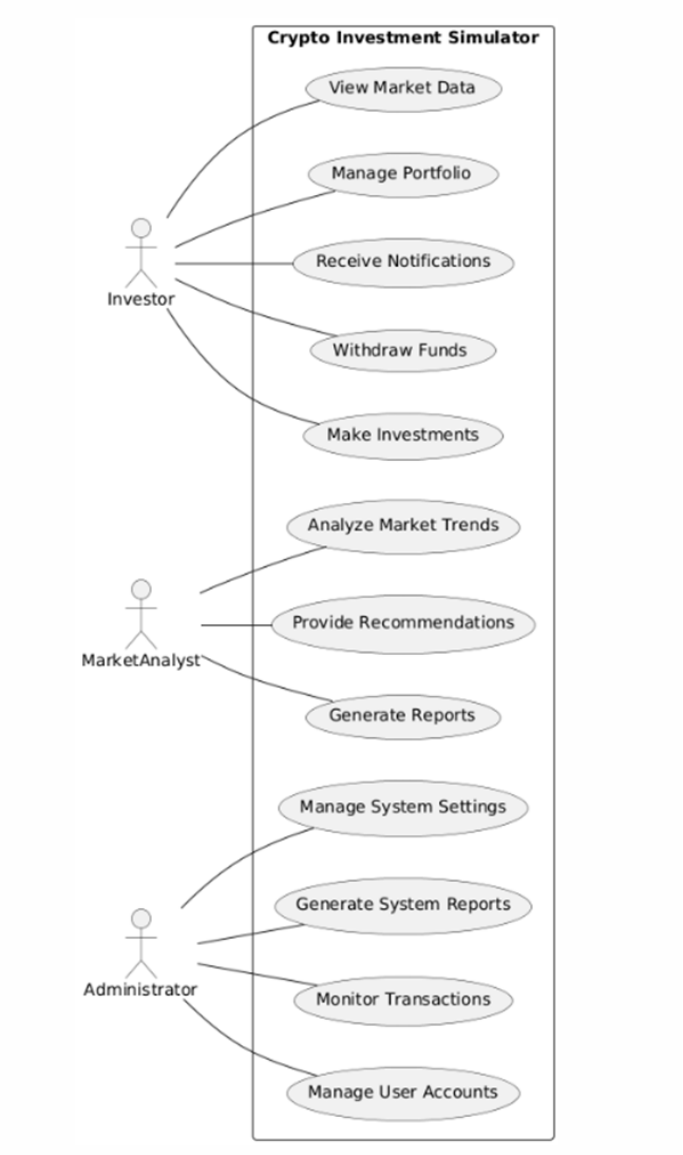
Data flow diagram :



- Crypto Exchange API: Provides live market data and facilitates buy/sell transactions.
- Admin: Manages and monitors the system, updates configurations, and handles report generation.

Result:

Use case Diagram:



EX NO:5	DRAW USE CASE DIAGRAM
DATE:	

AIM:

To Draw the Use Case Diagram for any project

ALGORITHM:

1. Identify Actors
2. Identify Use Cases
3. Connect Actors and Use Cases
4. Add System Boundary
5. Define Relationships
6. Review and Refine
7. Validate

INPUTS:

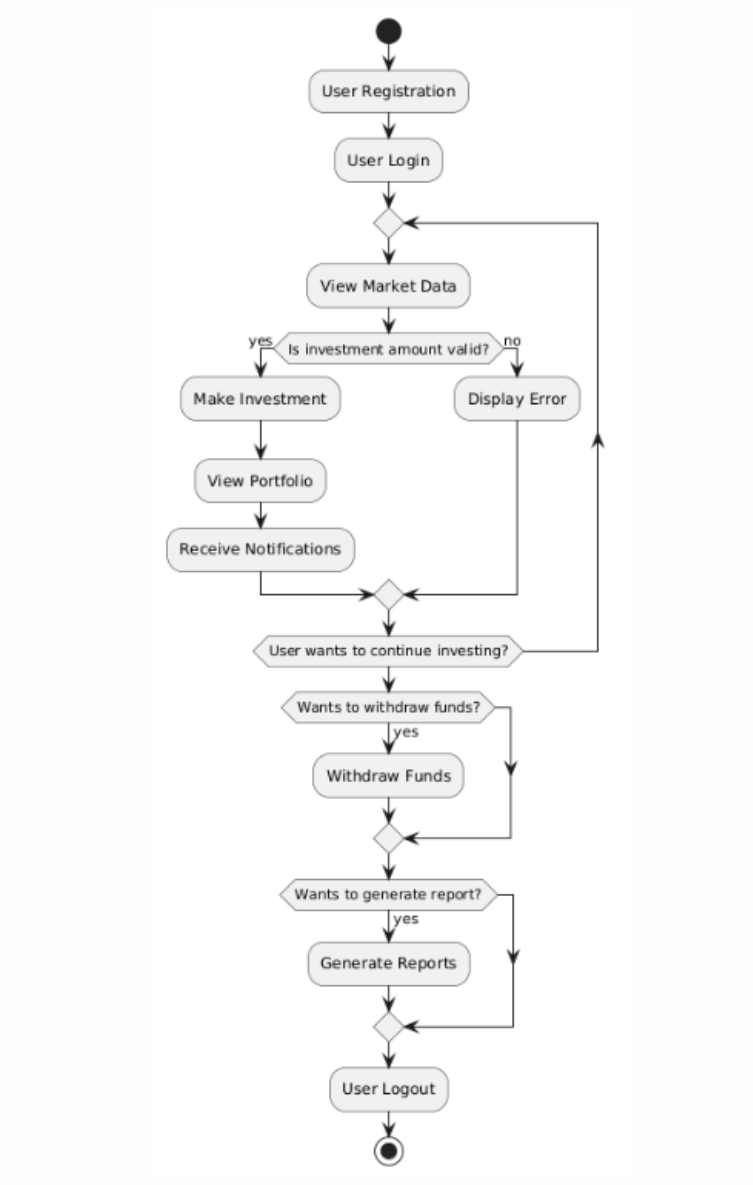
Actors

Use Cases

Relations

Result:

Activity Diagram:



EX NO:6	DRAW ACTIVITY DIAGRAM OF ALL USE CASES.
DATE:	

AIM:

To Draw the activity Diagram for any project

ALGORITHM:

Step 1: Identify the Initial State and Final States

Step 2: Identify the Intermediate Activities Needed

Step 3: Identify the Conditions or Constraints

Step 4: Draw the Diagram with Appropriate Notations

INPUTS:

Activities

Decision Points

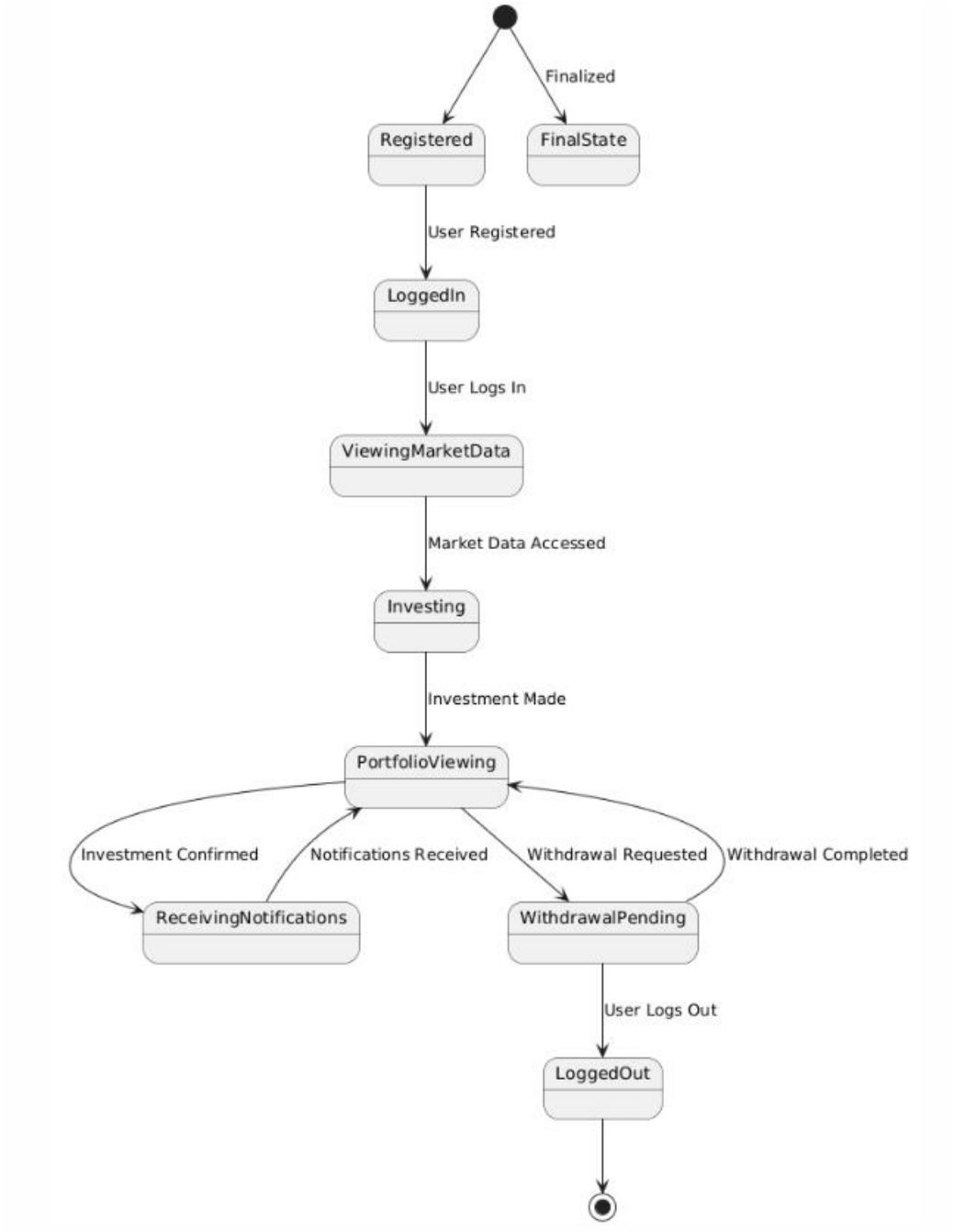
Guards

Parallel Activities

Conditions

Result:

State Chart Diagram:



EX NO:7	DRAW STATE CHART DIAGRAM OF ALL USE CASES.
DATE:	

AIM:

To Draw the State Chart Diagram for any project

ALGORITHM:

STEP-1: Identify the important objects to be analysed.

STEP-2: Identify the states.

STEP-3: Identify the events.

INPUTS:

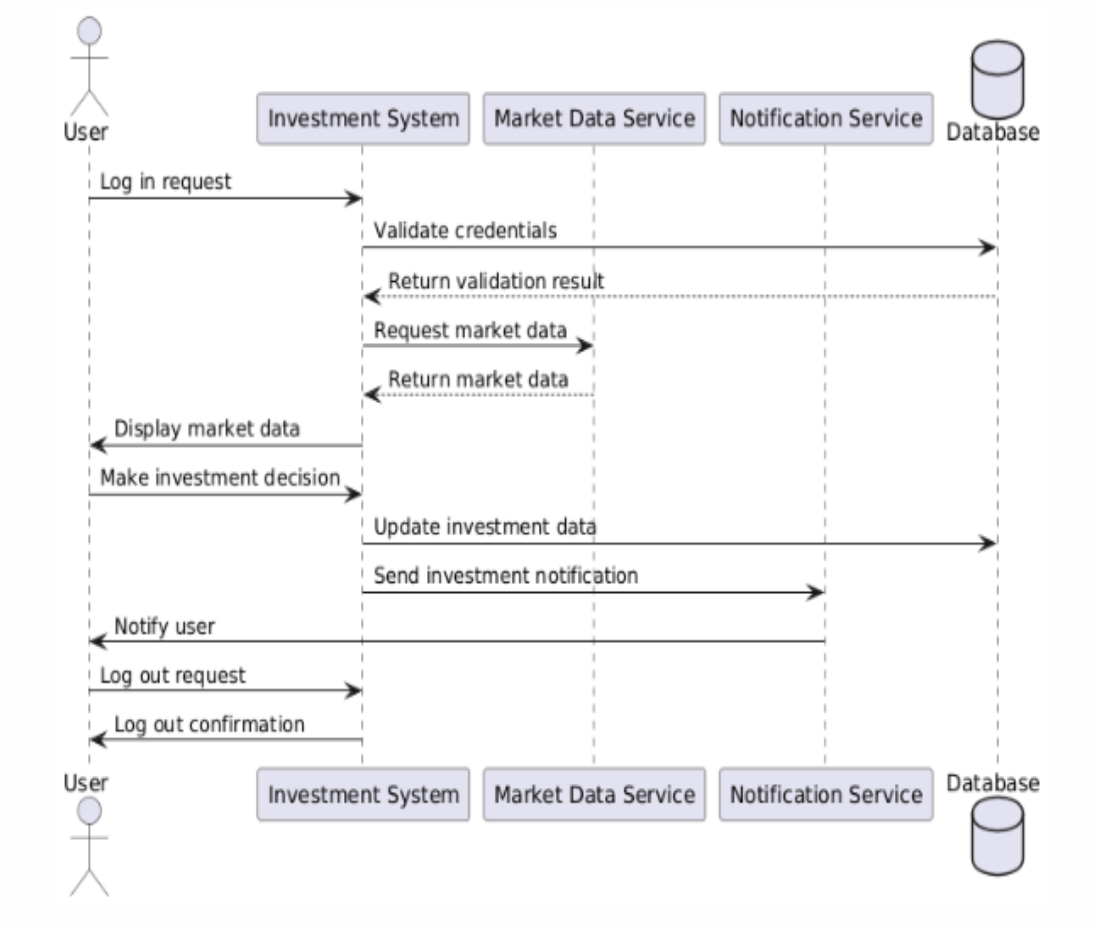
Objects

States

Events

Result:

Sequence Diagram:



EX NO:8	DRAW SEQUENCE DIAGRAM OF ALL USE CASES.
DATE:	

AIM: To Draw the Sequence Diagram for any project

ALGORITHM:

1. Identify the Scenario
2. List the Participants
3. Define Lifelines
4. Arrange Lifelines
5. Add Activation Bars
6. Draw Messages
7. Include Return Messages
8. Indicate Timing and Order
9. Include Conditions and Loops
10. Consider Parallel Execution
11. Review and Refine
12. Add Annotations and Comments
13. Document Assumptions and Constraints
14. Use a Tool to create a neat sequence diagram

INPUTS:

Objects taking part in the interaction.

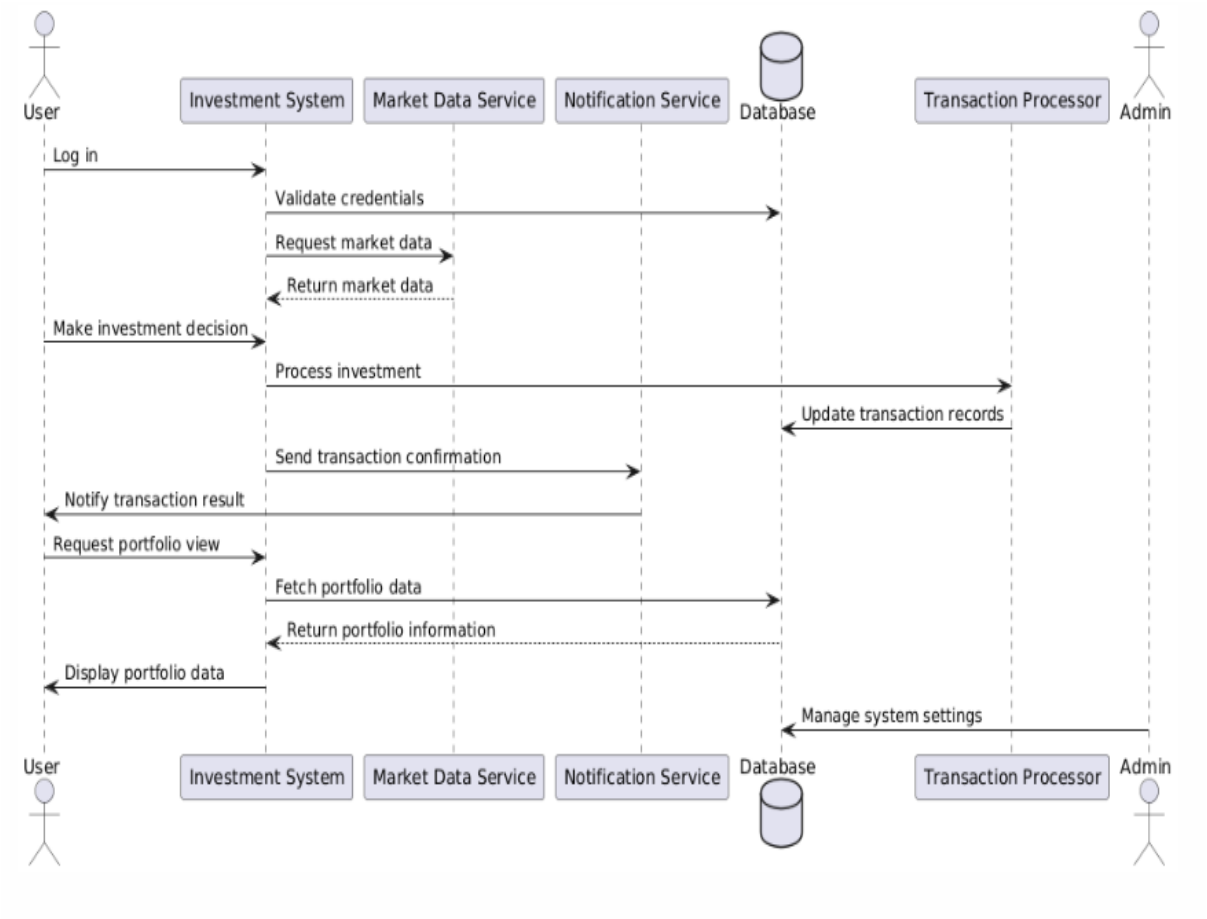
Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

Result:

Collaboration Diagram:



EX NO:9	DRAW COLLABORATION DIAGRAM OF ALL USE CASES
DATE:	

AIM:

To Draw the Collaboration Diagram for any project

ALGORITHM:

Step 1: Identify Objects/Participants

Step 2: Define Interactions

Step 3: Add Messages

Step 4: Consider Relationships

Step 5: Document the collaboration diagram along with any relevant explanations or annotations.

INPUTS:

Objects taking part in the interaction.

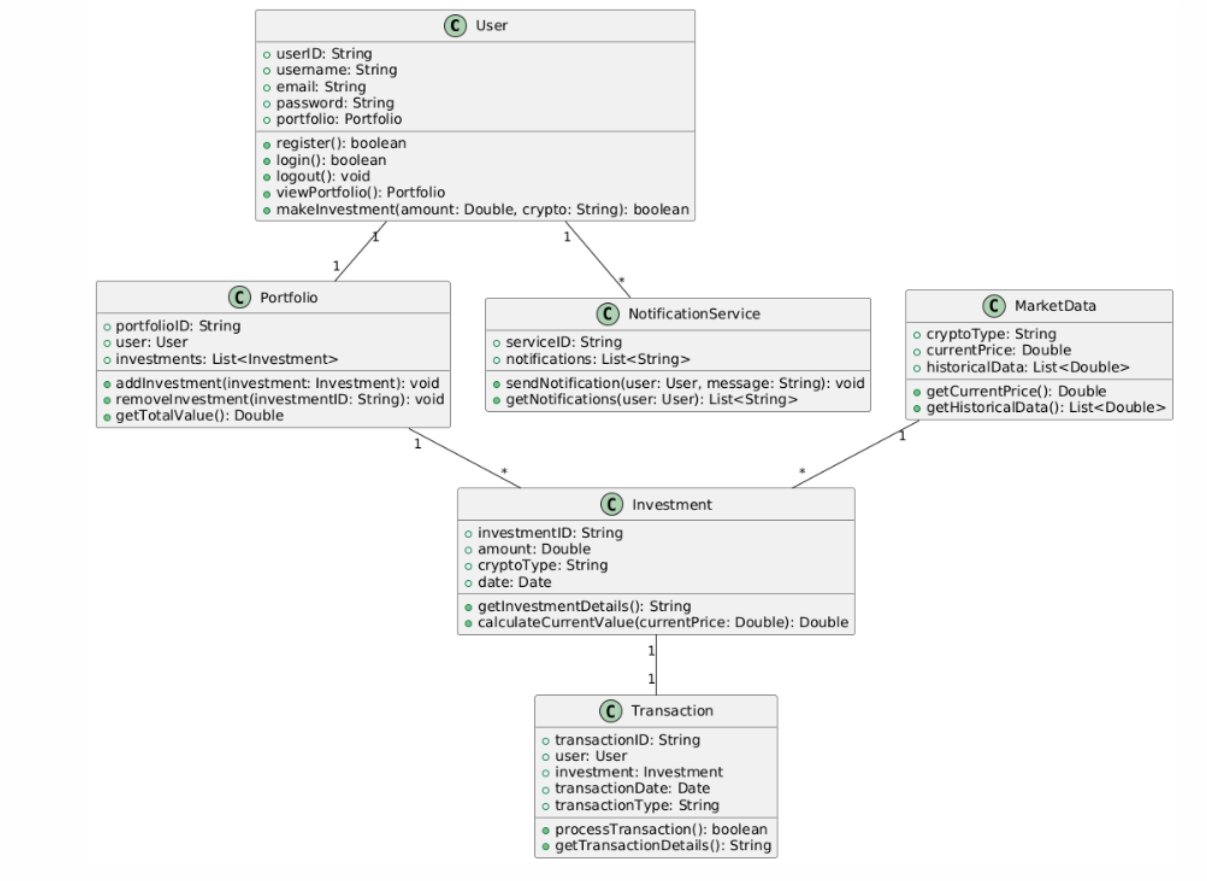
Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

Result:

Class Diagram :



EX NO:10	ASSIGN OBJECTS IN SEQUENCE DIAGRAM TO CLASSES AND MAKE CLASS DIAGRAM.
DATE:	

AIM:

To Draw the Class Diagram for any project

ALGORITHM:

1. Identify Classes
2. List Attributes and Methods
3. Identify Relationships
4. Create Class Boxes
5. Add Attributes and Methods
6. Draw Relationships
7. Label Relationships
8. Review and Refine
9. Use Tools for Digital Drawing

INPUTS:

1. Class Name
2. Attributes
3. Methods
4. Visibility Notation

RESULT:

EX NO:11	MINI PROJECT- PARKING MANAGEMENT SYSTEM
DATE	

AIM:

To develop a **Crypto Investment Simulator** using Streamlit and MySQL, enabling users to simulate cryptocurrency investments, monitor portfolio performance, analyize market trends, and visualize financial growth. The project focuses on creating an interactive platform to educate users about cryptocurrency trading while ensuring simplicity and engagement.

ALGORITHM:

1. Database Connection Initialization

2. Streamlit Interface Setup

3. Operation Selection

4. Database Query Execution

5. Data Visualization

6. Risk and Reward Analysis

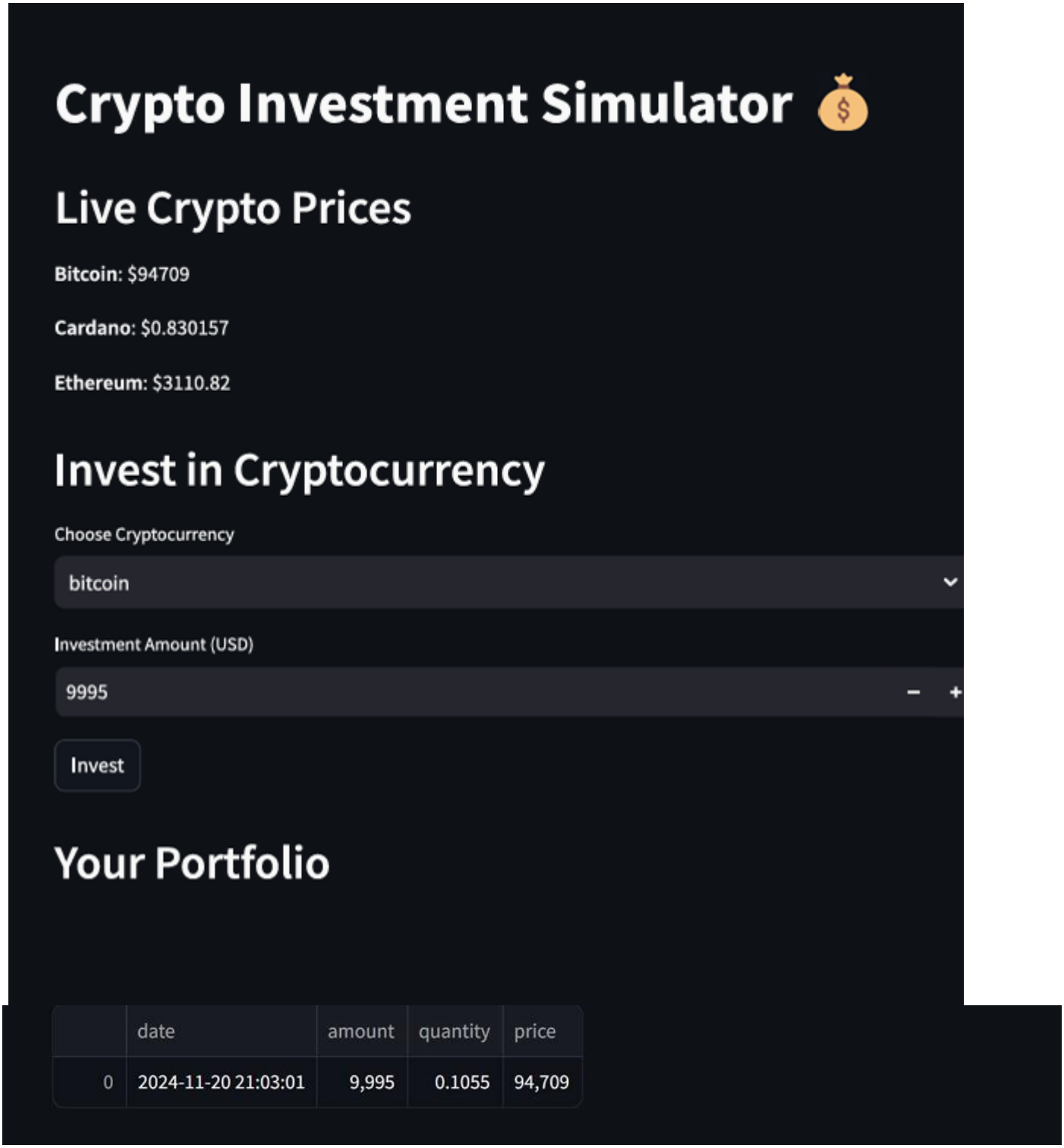
7. Email Notifications (Optional)

8. Feedback Display

9. Application Termination

PROGRAM:

```
import streamlit as st

import pandas as pd

import requests

import json

import os

import matplotlib.pyplot as plt

from datetime import datetime

def load_portfolio():

    if os.path.exists("portfolio.json"):

        with open("portfolio.json", "r") as f:

            return json.load(f)
```


Crypto Investment Simulator
Live Crypto Prices
Bitcoin: $94709
Cardano: $0.830157
Ethereum: $3110.82
Invest in Cryptocurrency
Choose Cryptocurrency
bitcoin
Investment Amount (USD)
9995
Invest
Your Portfolio
date
amount
quantity
price
0
2024-11-20 21:03:01
9,995
0.1055
94,709

```
    return {}
def save_portfolio(portfolio):
    with open("portfolio.json", "w") as f:
        json.dump(portfolio, f, indent=4)
def get_crypto_prices():
    url = "https://api.coingecko.com/api/v3/simple/price?ids=bitcoin,ethereum,cardano&vs_currencies=usd"
    response = requests.get(url)
    if response.status_code == 200:
        return response.json()
    else:
        return {}
st.title("Crypto Investment Simulator 💰")
portfolio = load_portfolio()
crypto_prices = get_crypto_prices()
st.header("Live Crypto Prices")
if crypto_prices:
    for crypto, data in crypto_prices.items():
        st.write(f"**{crypto.capitalize()}**: ${data['usd']}")
else:
    st.error("Failed to fetch live prices. Check your internet connection.")
st.header("Invest in Cryptocurrency")
crypto = st.selectbox("Choose Cryptocurrency", ["bitcoin", "ethereum", "cardano"])
amount = st.number_input("Investment Amount (USD)", min_value=1, step=1)
if st.button("Invest"):
    if crypto in crypto_prices:
        price = crypto_prices[crypto]["usd"]
        quantity = amount / price
        if crypto not in portfolio:
```

```
    portfolio[crypto] = []
        portfolio[crypto].append({
            "date": datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
            "amount": amount,
            "quantity": quantity,
            "price": price,
        })
        save_portfolio(portfolio)
        st.success(f"Invested ${amount} in {crypto.capitalize()} at ${price:.2f}/unit!")
    else:
        st.error("Failed to invest. Try again later.")
st.header("Your Portfolio")
if portfolio:
    for crypto, transactions in portfolio.items():
        st.subheader(crypto.capitalize())
        df = pd.DataFrame(transactions)
        st.write(df)
```

Conclusion:

The **Crypto Investment Simulator** developed using Streamlit and MySQL provides an interactive and user-friendly platform for users to simulate cryptocurrency investments, track portfolio performance, and analyze market trends. This system centralizes key functionalities such as simulating buy/sell transactions, monitoring investment growth, visualizing market data, and analyzing risk and reward. By offering a safe and educational environment, the simulator helps users better understand cryptocurrency trading and portfolio management, empowering them to make informed financial decisions in real-world scenarios.