Cairo University

Faculty of Computers and Artificial Intelligence

CS251

Introduction to Software Engineering

Invest Wise

Software Design Specifications

Version 1.1

Winter2025

CS251: Phase-1
Project: Invest Wise

Software Design Specification

Contents

Team 3

Document Purpose and Audience 3

System Models 4

I. Architecture Diagram 4

II. Class Diagram(s) 5

III. Class Descriptions 5

IV. Sequence diagrams 7

Class - Sequence Usage Table 14

V. State Diagram 16

VI. SOLID Principles 17

VII. Design Patterns 18

Tools 20

Ownership Report 20

CS251: Phase-1
Project: Invest Wise

Software Design Specification

Team

ID	Name	Email	Mobile
20230542	Ahmed sheref sayed	20230542@stud.fci-cu.edu.eg	01091575793
20230544	Hassan Walid Hassan	20230544@stud.fci-cu.edu.eg	01025068020
20231142	Mohammed Sheref Abd-Alazim	20231142@stud.fci-cu.edu.eg	01150600775

Document Purpose and Audience

Purpose:

The objective of this document is to outline the software requirements agreed upon by stakeholders, detailing the essential functionalities of the software to optimize development time and costs by ensuring the requirements are clearly specified.

Audience:

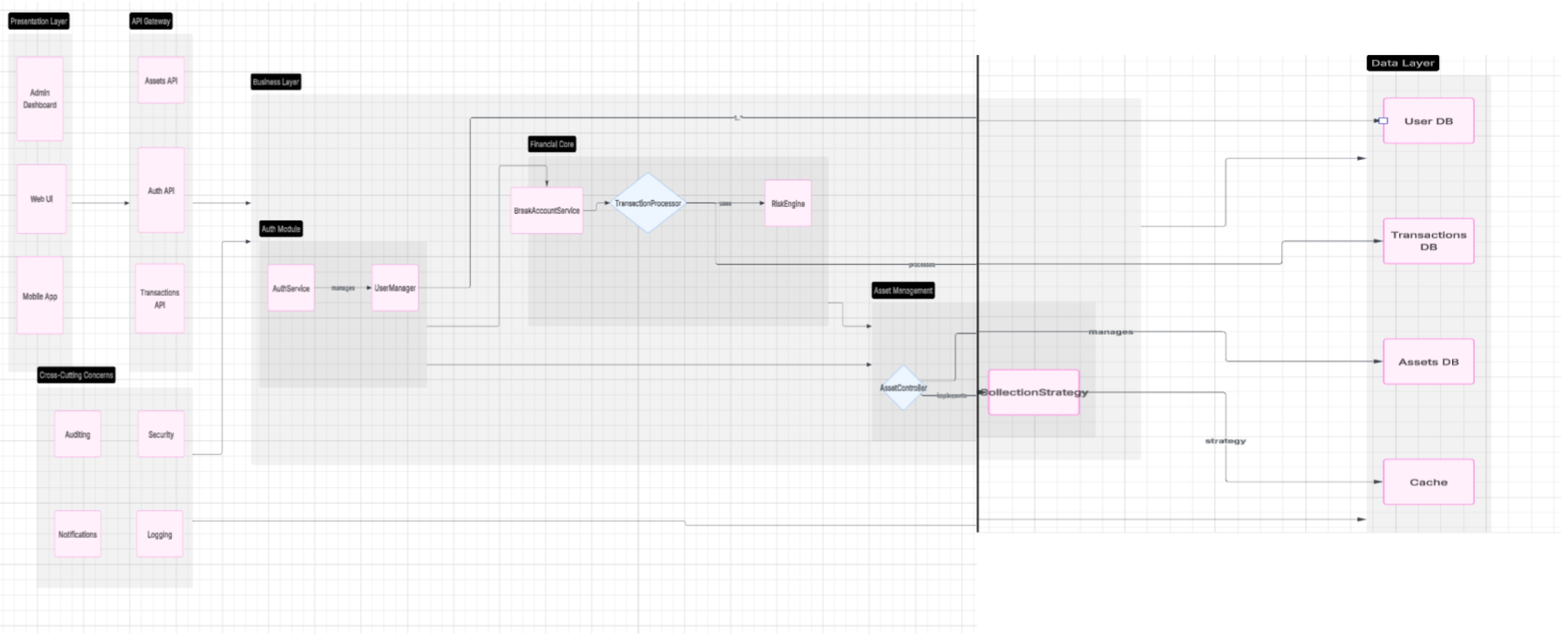
- Coders
- Designers
- Stakeholders

CS251: Phase-1
Project: Invest Wise

Software Design Specification

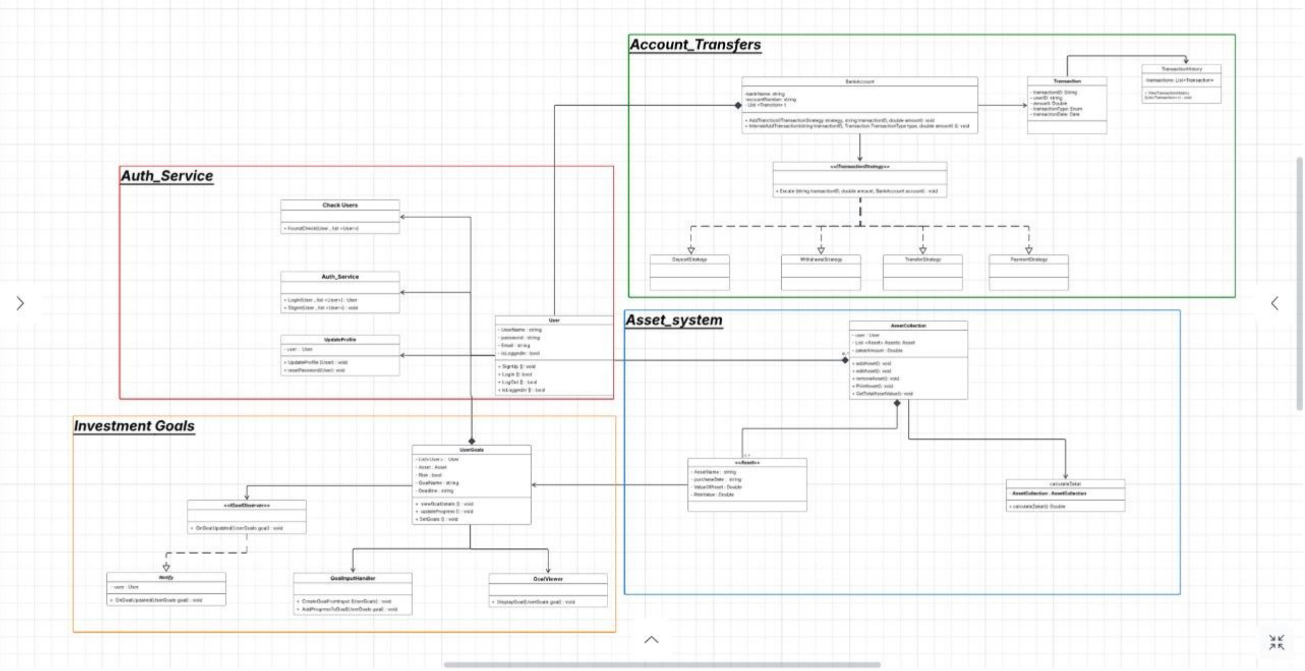
System Models

I. Architecture Diagram



Software Design Specification

II. Class Diagram(s)



III. Class Descriptions

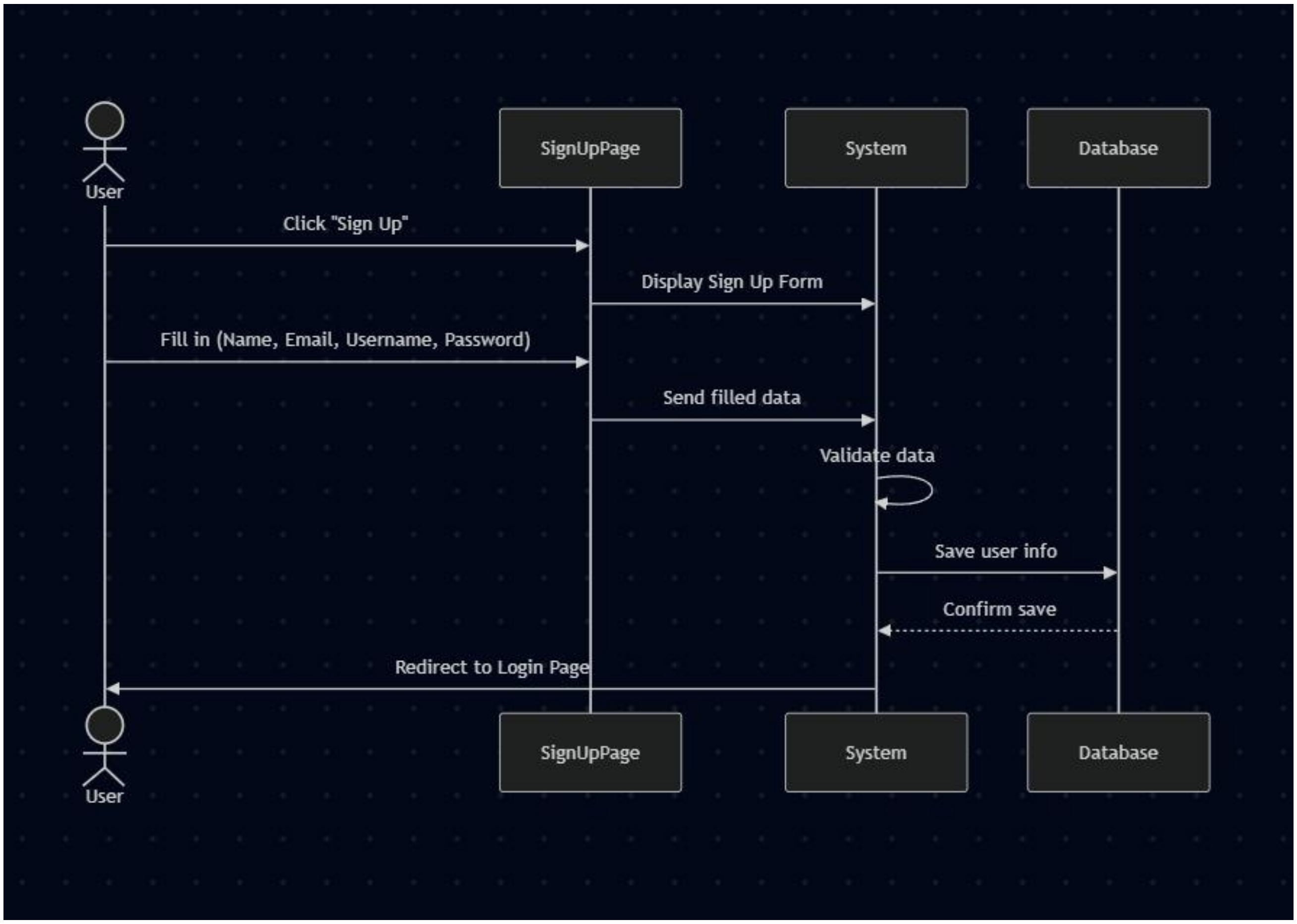
Class ID	Class Name	Description & Responsibility
1.	User	Represents the system's user, storing username, password, email, and login status. Provides functions for signing up, logging in, logging out, and checking if the user is active.
2.	UpdateProfile	Manages editing user information like username, email, and password. Works closely with the User class to keep account data updated and secure.
3.	UserGoals	Links users with their investment goals and associated assets. Sends alerts to users if an asset's risk level changes.
4.	BankAccount	Contains bank account information for users. Provides secure methods to connect a bank and verify the ownership of the account.
5.	ChaseBankProvider	Handles secure connection with Chase Bank. Manages authentication credentials and session control for banking operations.
6.	Transaction	Models financial transactions, tracking amount, type, and date. Allows creating new transactions and viewing transaction records.
7.	TransactionHistory	Stores a user's full history of transactions. Offers features to add new transactions, search by date, and count total transactions.

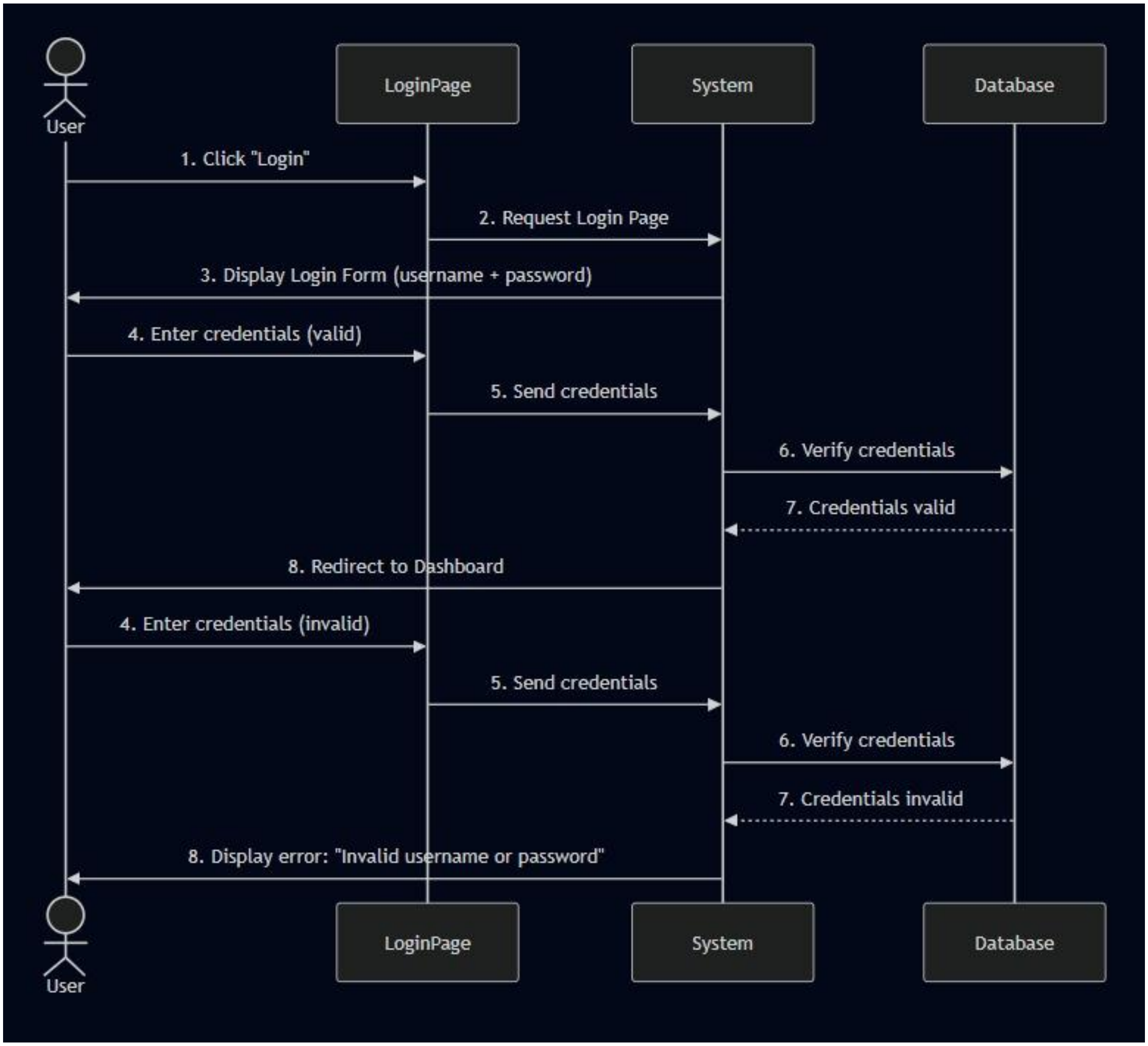
Software Design Specification

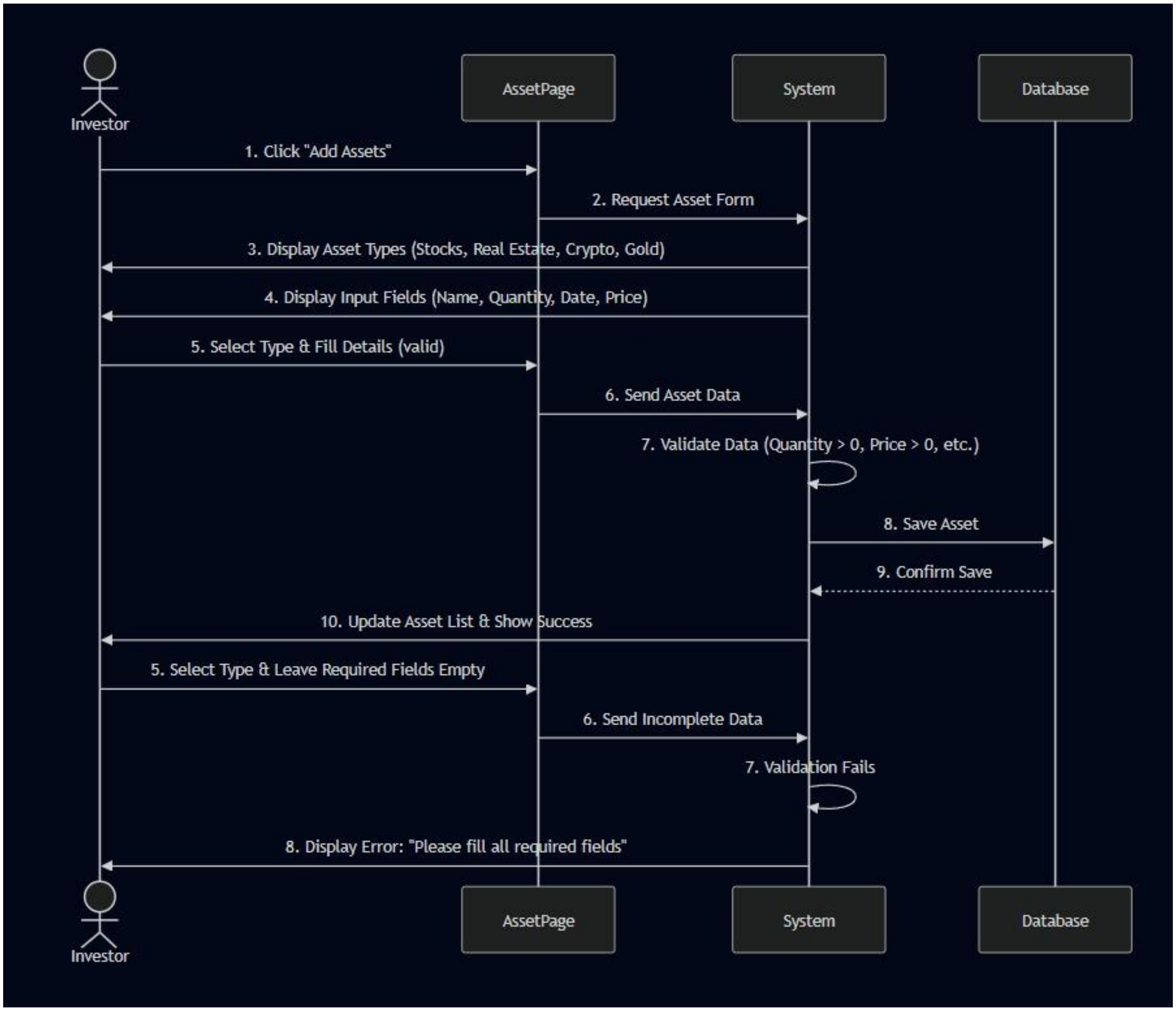
Class ID	Class Name	Description & Responsibility
8.	Asset	Represents an individual asset like stock, real estate, or cryptocurrency. Tracks the asset's purchase date, current value, and risk factor.
9.	AssetCollection	Manages all assets owned by a user. Supports adding and removing assets and calculates the total zakat amount based on asset value.
10.	calculateZakat	Calculates zakat from a user's total assets according to Islamic financial principles, based on the current asset values.

Software Design Specification

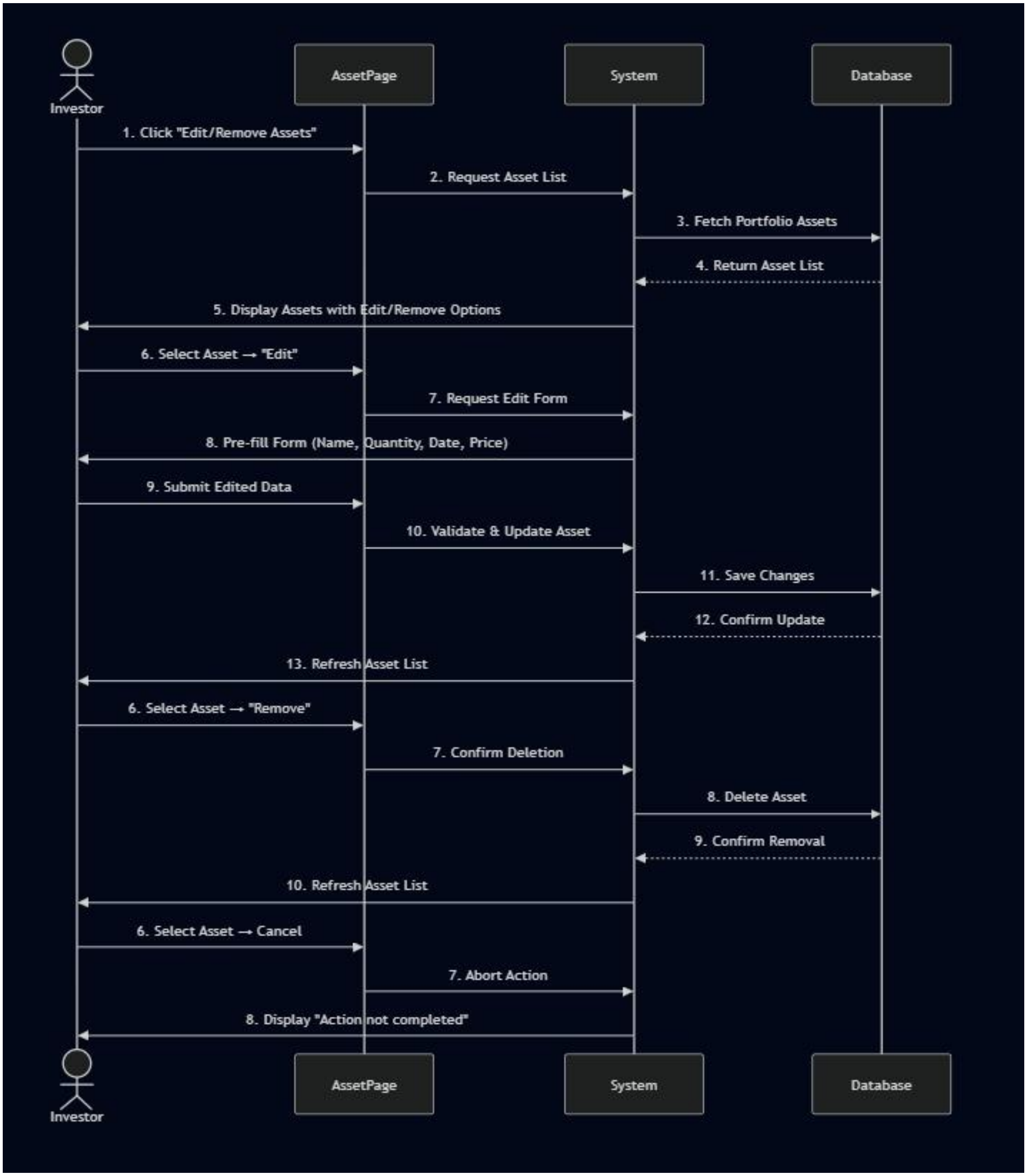
IV. Sequence diagrams



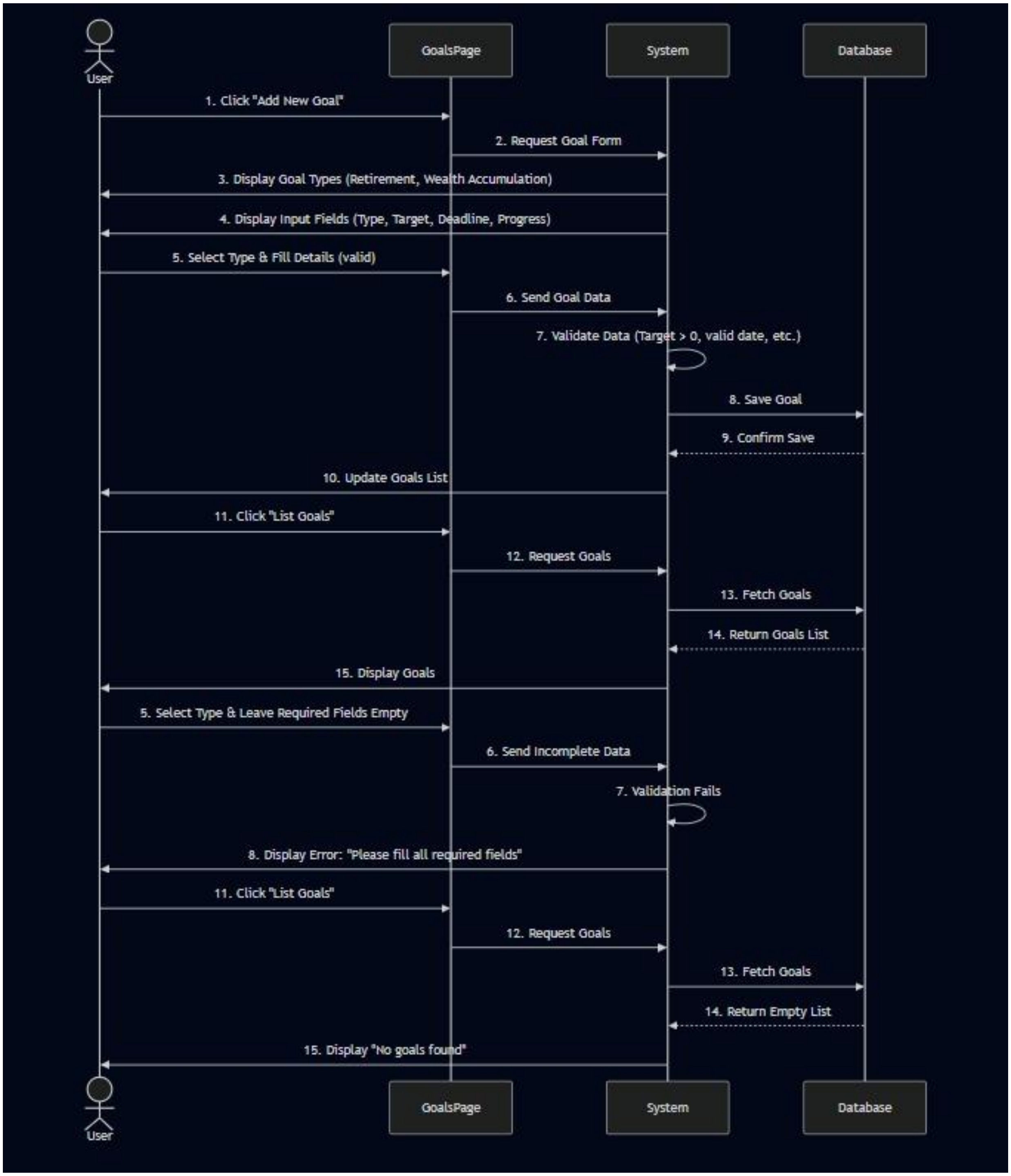

User
LoginPage
System
Database
1. Click "Login"
2. Request Login Page
3. Display Login Form (username + password)
4. Enter credentials (valid)
5. Send credentials
6. Verify credentials
7. Credentials valid
8. Redirect to Dashboard
4. Enter credentials (invalid)
5. Send credentials
6. Verify credentials
7. Credentials invalid
8. Display error: "Invalid username or password"
User
LoginPage
System
Database

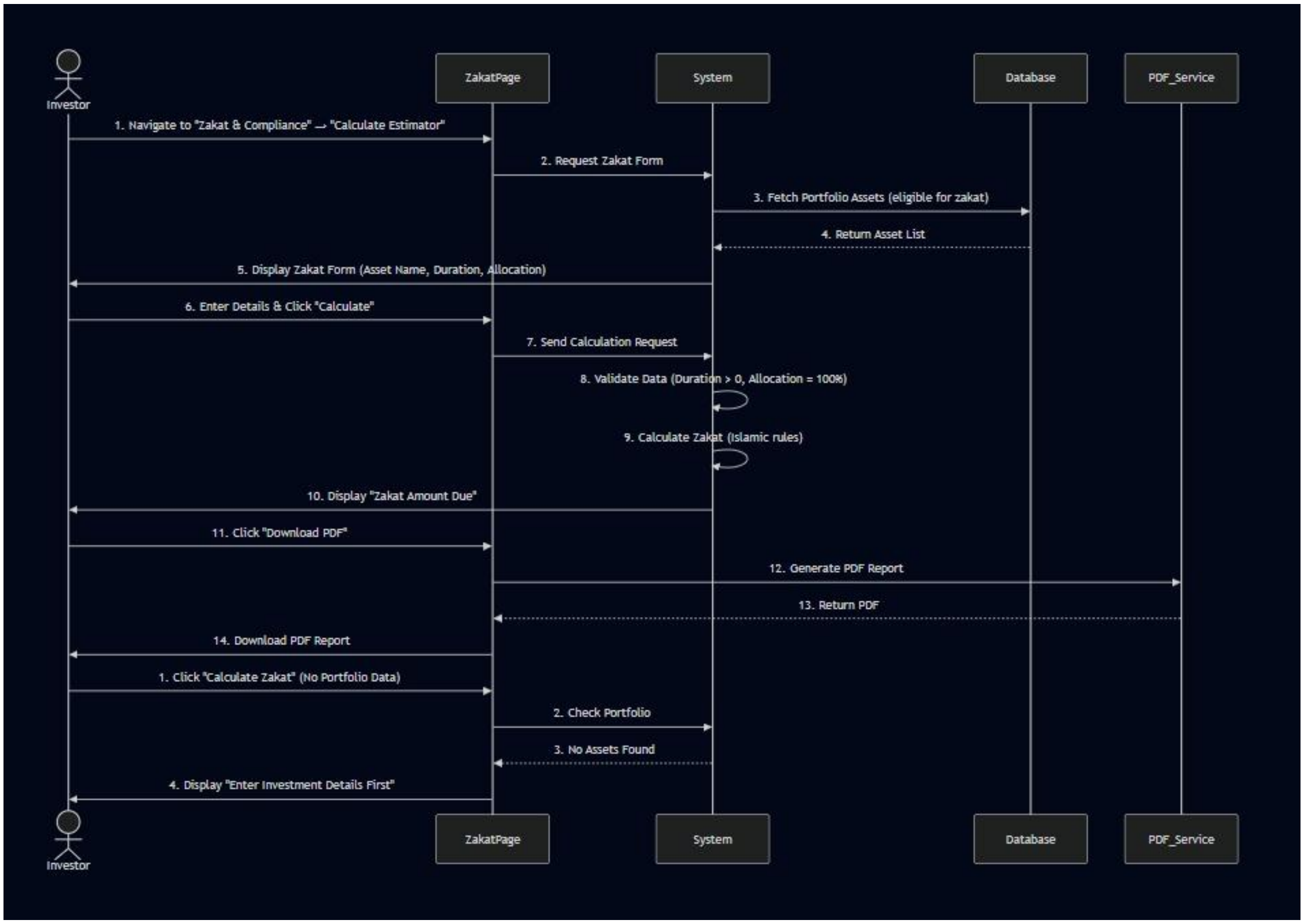

Investor
AssetPage
System
Database
1. Click "Add Assets"
2. Request Asset Form
3. Display Asset Types (Stocks, Real Estate, Crypto, Gold)
4. Display Input Fields (Name, Quantity, Date, Price)
5. Select Type & Fill Details (valid)
6. Send Asset Data
7. Validate Data (Quantity > 0, Price > 0, etc.)
8. Save Asset
9. Confirm Save
10. Update Asset List & Show Success
5. Select Type & Leave Required Fields Empty
6. Send Incomplete Data
7. Validation Fails
8. Display Error: "Please fill all required fields"
Investor
AssetPage
System
Database

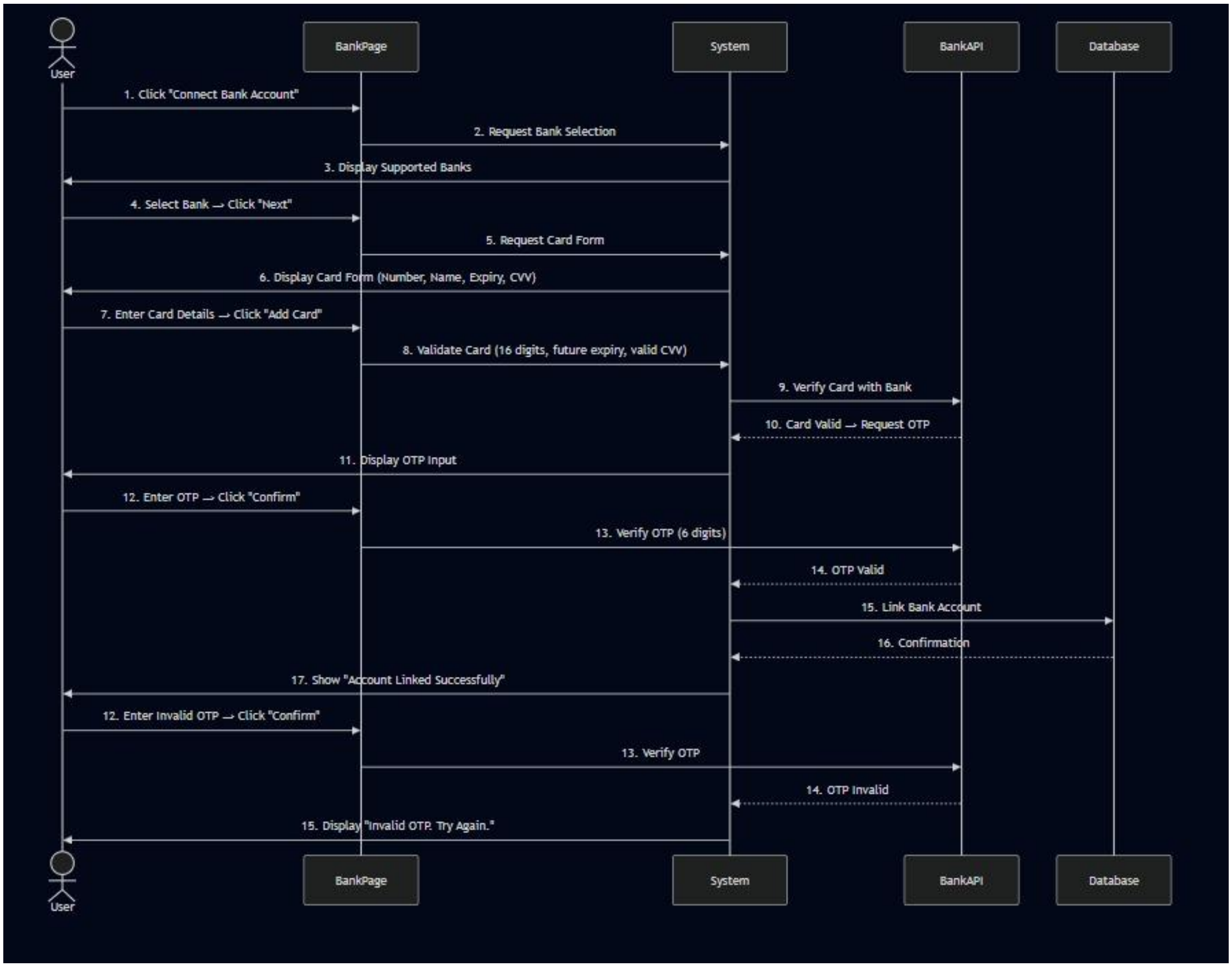
Software Design Specification



Software Design Specification




Investor
ZakatPage
System
Database
PDF_Service
1. Navigate to "Zakat & Compliance" → "Calculate Estimator"
2. Request Zakat Form
3. Fetch Portfolio Assets (eligible for zakat)
4. Return Asset List
5. Display Zakat Form (Asset Name, Duration, Allocation)
6. Enter Details & Click "Calculate"
7. Send Calculation Request
8. Validate Data (Duration > 0, Allocation = 100%)
9. Calculate Zakat (Islamic rules)
10. Display "Zakat Amount Due"
11. Click "Download PDF"
12. Generate PDF Report
13. Return PDF
14. Download PDF Report
1. Click "Calculate Zakat" (No Portfolio Data)
2. Check Portfolio
3. No Assets Found
4. Display "Enter Investment Details First"
Investor
ZakatPage
System
Database
PDF_Service


User
BankPage
System
BankAPI
Database
1. Click "Connect Bank Account"
2. Request Bank Selection
3. Display Supported Banks
4. Select Bank → Click "Next"
5. Request Card Form
6. Display Card Form (Number, Name, Expiry, CVV)
7. Enter Card Details → Click "Add Card"
8. Validate Card (16 digits, future expiry, valid CVV)
9. Verify Card with Bank
10. Card Valid → Request OTP
11. Display OTP Input
12. Enter OTP → Click "Confirm"
13. Verify OTP (6 digits)
14. OTP Valid
15. Link Bank Account
16. Confirmation
17. Show "Account Linked Successfully"
12. Enter Invalid OTP → Click "Confirm"
13. Verify OTP
14. OTP Invalid
15. Display "Invalid OTP. Try Again."
User
BankPage
System
BankAPI
Database

Software Design Specification

Class - Sequence Usage Table

Sequence Diagram	Classes Used	All Methods Used

1. SignUpPage	Class SignUpPage
	Class System
	Class Database
		displaySignUpForm(), sendFilledData()
		validateData(), saveUserInfo()
		confirmSave(), redirectToLoginPage()

2. LoginPage	Class LoginPage
	Class System
	Class Database
	Class User
		displayLoginForm(), sendCredentials()
		verifyCredentials(), handleRedirect()
		checkUserExists(), validatePassword()
		getUsername(), getPassword()

3. Investor Asset Addition	Class Investor
	Class AssetPage
	Class System
	Class Database
		clickAddAssets(), selectAssetType(), fillAssetDetails()
		displayAssetTypes(), displayInputFields(), sendAssetData(), displayError()
		validateAssetData(), updateAssetList()
		saveAsset(), confirmSave()

4. Asset Management	Class Investor
	Class AssetPage
	Class System
	Class Database
		clickEditRemoveAssets(), selectAssetAction()
		requestAssetList(), displayAssetsWithOptions(), requestEditForm() ,prefillForm(), submitEditedData(), confirmDeletion(), displayActionMessage()
		patchPortfolioAssets(), validateAndUpdateAsset(), saveChanges() ,deleteAsset()
		returnAssetList(), confirmUpdate(), confirmRemoval(), refreshAssetList()

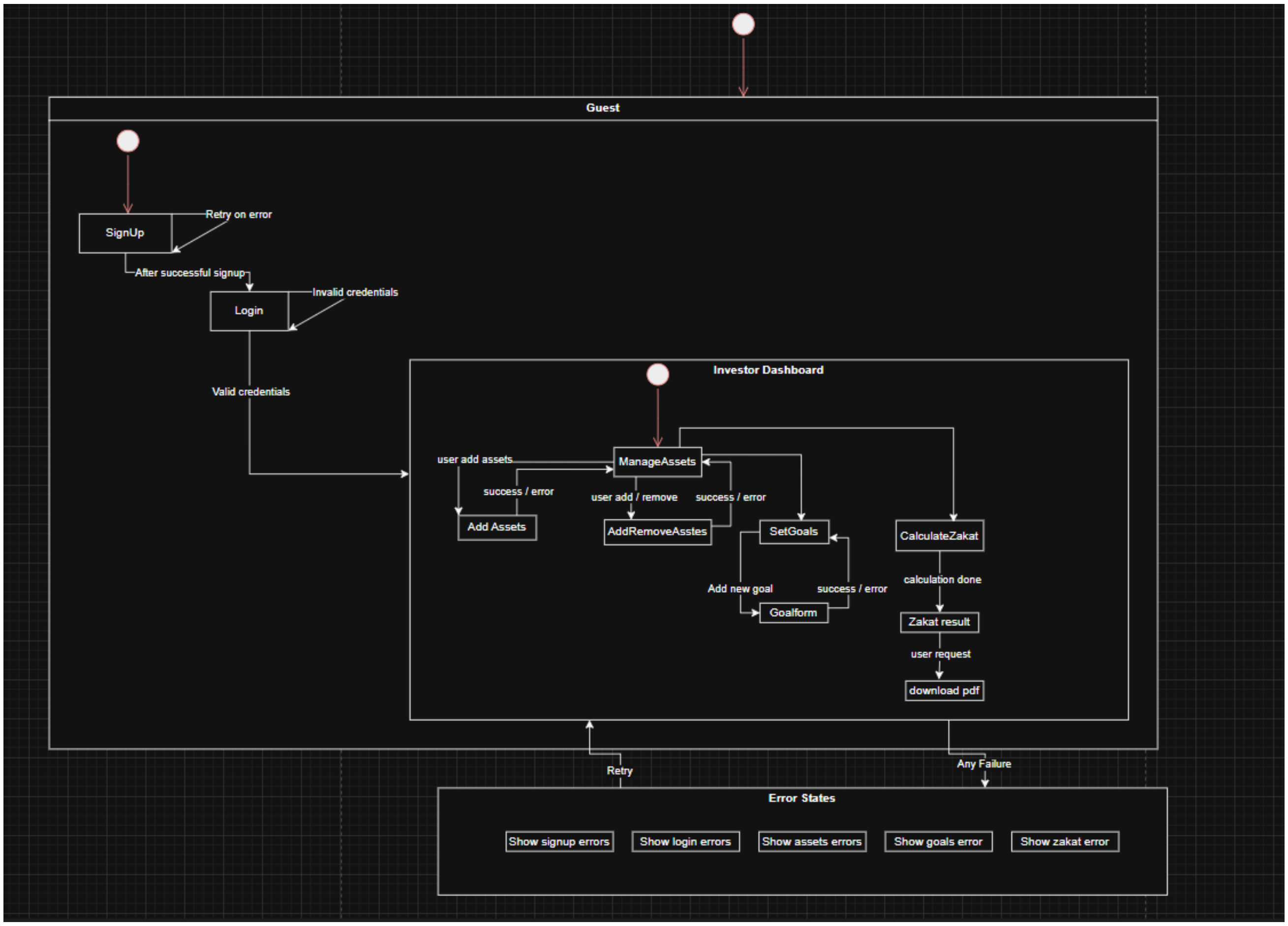
5. Goal Management	Class GoalSteps
	Class System
	Class Database
		clickAddNewGoal(), requestGoalForm(), displayGoalTypes(), displayInputFields(), sendGoalData(), displayError(), clickListGoals(), displayGoals(), displayNoGoalsMessage()
		validateGoalData(), saveGoal(), updateGoalsList(), fetchGoals()
		confirmSave(), returnGoalsList(), returnEmptyList()

6. Zakat Calculation	Class ZakatPage
	Class System
	Class Database
	Class PDF_Service
		navigateToZakatCalculator(), registerZakatForm(), displayZakatForm(), enterCalculationDetails(), displayZakatAmount(), clickDownloadPDF(), downloadPDFReport(), displayNoAssetsMessage()
		fetchPortfolioAssets(), validateZakatData(), calculateZakat(), checkPortfolio()
		returnAssetList(), returnNoAssetsFound()
		generatePDFReport(), returnPDF()

Sequence Diagram	Classes Used	All Methods Used
7. Bank Account Linking	Class RankPage	clickConnectBankAccount(), requestBankSelection(), displaySupportedBanks(), requestCardForm(), displayCardForm(), validateUserInput(), displayOTPInput(), showSuccessMessage(), showOTPError()
	Class System	verifyCardDetails(), requestOTP(), verifyOTP(), linkBankAccount()
	Class BankAPI	validateCardWithBank(), sendOTP(), confirmOTP()
	Class Database	storeBankAccount(), updateAccountStatus()

Software Design Specification

V. State Diagram



VI. SOLID Principles

Class Descriptions

- **User**
  Represents the system's user, storing username, password, email, and login status. Provides functions for signing up, logging in, logging out, and checking if the user is active.

- **UpdateProfile**
  Handles editing user information like username, email, and password. Works closely with the User class to keep account data updated and secure.

- **UserGoals**
  Links users with their investment goals and associated assets. Sends alerts to users if an asset's risk level changes.

- **BankAccount**
  Contains bank account information for users. Provides secure methods to connect a bank and verify the ownership of the account.

- **ChaseBankProvider**
  Handles secure connection with Chase Bank. Manages authentication credentials and session control for banking operations.

- **Transaction**
  Models financial transactions, tracking amount, type, and date. Allows creating new transactions and viewing transaction records.

- **TransactionHistory**
  Stores a user's full history of transactions. Offers features to add new transactions, search by date, and count total transactions.

- **Asset**
  Represents an individual asset like stock, real estate, or cryptocurrency. Tracks the asset's purchase date, current value, and risk factor.

- **AssetCollection**
  Manages all assets owned by a user. Supports adding and removing assets and calculates the total zakat amount based on asset value.

- **calculateZakat**
  Calculates zakat from a user's total assets according to Islamic financial principles, based on the current asset values.

- **Crypto** *(Subclass of Asset)*
  Includes crypto-specific details like wallet address and blockchain network. Calculates value based on market data.

- **RealEstate** *(Subclass of Asset)*
  Represents property. Includes fields like location and rental income, and supports yield calculation.

- **Gold** *(Subclass of Asset)*
  Represents gold holdings. Includes purity and weight, with methods for calculating value by karat.

- **NotifyUsers**
  Observes UserGoals and sends notifications when changes occur, such as high-risk levels or deadlines approaching.

- **FinancialGoal**
  A specific type of investment goal targeting financial milestones like reaching a certain net worth or saving goal.

- **EmergencyFundGoal** *(Subclass of UserGoals)*
  Focused on building a liquid emergency reserve fund. Tracks saved amount versus target.

VII. Design Patterns

1. Design Patterns Used

▪ Observer Pattern

- Used between **UserGoals** and **NotifyUsers**.

- Enables automatic user notification when a goal's risk level or deadline changes.

Software Design Specification

- Promotes loose coupling and event-driven architecture.

▪ **Strategy Pattern**

- Used in the **Asset** system via subclasses: **Crypto**, **RealEstate**, and **Gold**.
- Each subclass implements its own method for calculating value, zakat, or yield.
- Promotes flexibility and open extension of new asset types.

---

2. SOLID Principles Applied

▪ **Single Responsibility Principle (SRP)**

- Each class is responsible for one purpose (e.g., **User** handles login, **UpdateProfile** handles profile updates).

▪ **Open/Closed Principle (OCP)**

- **Asset** class is open for extension via subclasses like **Crypto** and **RealEstate**, but closed for modification.

▪ **Liskov Substitution Principle (LSP)**

- Subclasses of **Asset** (**Crypto**, **RealEstate**, **Gold**) can be used anywhere an **Asset** is expected without

Software Design Specification

Tools

For designing diagrams, we used Lucidchart to create detailed classes diagrams and Mermaid.js to create Sequence diagrams.

For State diagrams: https://app.diagrams.net

Ownership Report

Item	Owners
Ahmed Sheref Sayed	Class Diagram, Apply design pattern
Hassan Walid Hassan	Sequence Diagrams, State Diagram
Mohamed Sheref	Architecture Diagram, Apply Solid Principles