Faculty of Computers and Artificial Intelligence

Cairo University

CS251: Intro to Software Engineering

Assignment 2

Dr. Mohammed El Ramly

(Draft Version)

Invest

Name	ID	Section number
Mariam Badr Yahya Abd El-Naby	20230391	S17-S18
John Ayman Demian	20230109	S17-S18
Emad George Mattar Hanna	20230244	S23-S24

April 2025

CS251: <Stock Holders>
Project: <Invest>

Software Design Specification

Contents

Document Purpose and Audience 4
System Models 5
I. Architecture Diagram 5
II. Class Diagram(s) 6
III. Class Descriptions 7
IV. Sequence diagrams 9
V. Stat Diagram: - 17
.... 17
VI. Solid Principle: - 18
1. Single Responsibility Principle (SRP) 18
2. Open/Closed Principle (OCP) 18
3. Dependency Inversion Principle (DIP) 18
VII. Design Patterns: - 18
1. Factory Pattern (Used for creating Assets) 18
2. Strategy Pattern (Used in Risk Assessment and Zakat Calculation) 18
3. Observer Pattern (Used in Notifications) 19
Tools 19
Draw. oi 19
Ownership Report 19
Owner 19
Items 19
Mariam Badr 19
Architecture Diagram, 3 Sequence Diagram & them Table and some classes on class diagram & them Table 19
John Ayman 19
3 Sequence Diagram & them Table, some classes on class diagram & them Table 19
Emad George 19

CS251: <Stock Holders>
Project: <Invest>

Software Design Specification

Stat Diagram, Document Purpose and Audience, SOLID, Design Patterns and Class Responsibility Table: 19

CS251: <Stock Holders>
Project: <Invest>

Software Design Specification

Document Purpose and Audience

Document Purpose:

This document outlines the detailed software design for **Invest**, a Sharia-compliant investment management system tailored to Egyptian users. It describes the overall architecture, system components, interactions between classes, and adherence to well-established design principles. The goal is to ensure consistency, maintainability, and clarity for the developers and stakeholders involved in building and validating the system.

Target Audience:

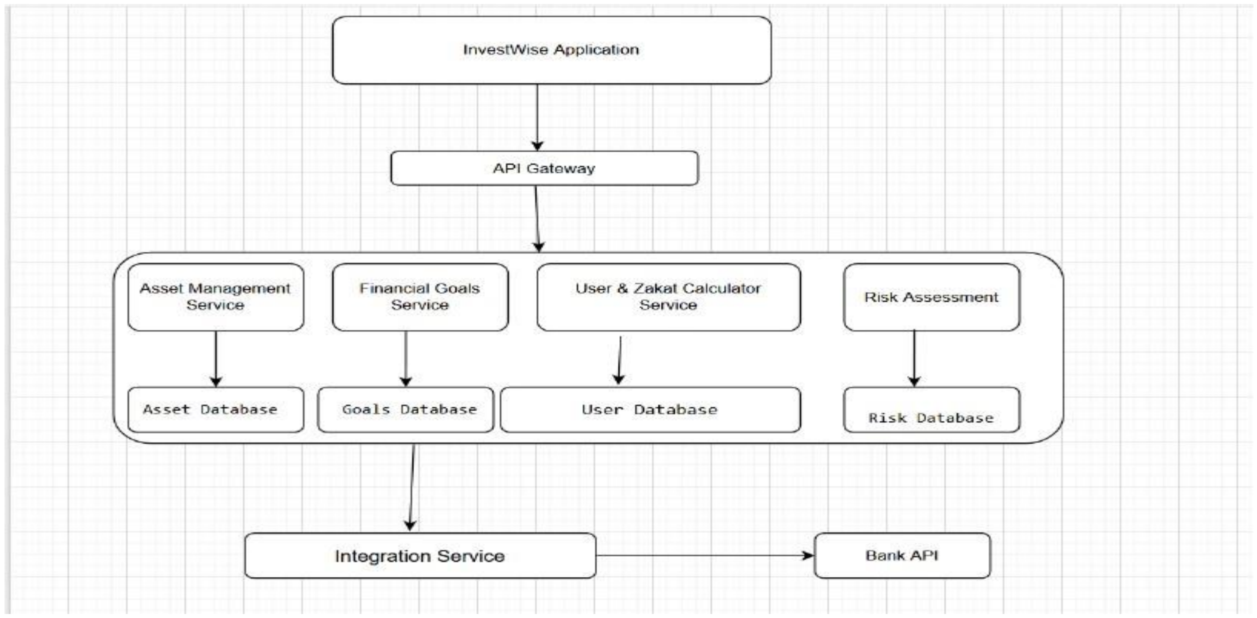
- **Developers:** responsible for frontend, backend, and integration.
- **Quality Assurance teams:** verifying implementation accuracy.
- **Project Managers:** tracking progress and system coherence.
- **Stakeholders and domain experts:** (e.g., Islamic finance advisors) reviewing investment and compliance-related features.

CS251: <Stock Holders>
Project: <Invest>

Software Design Specification

System Models

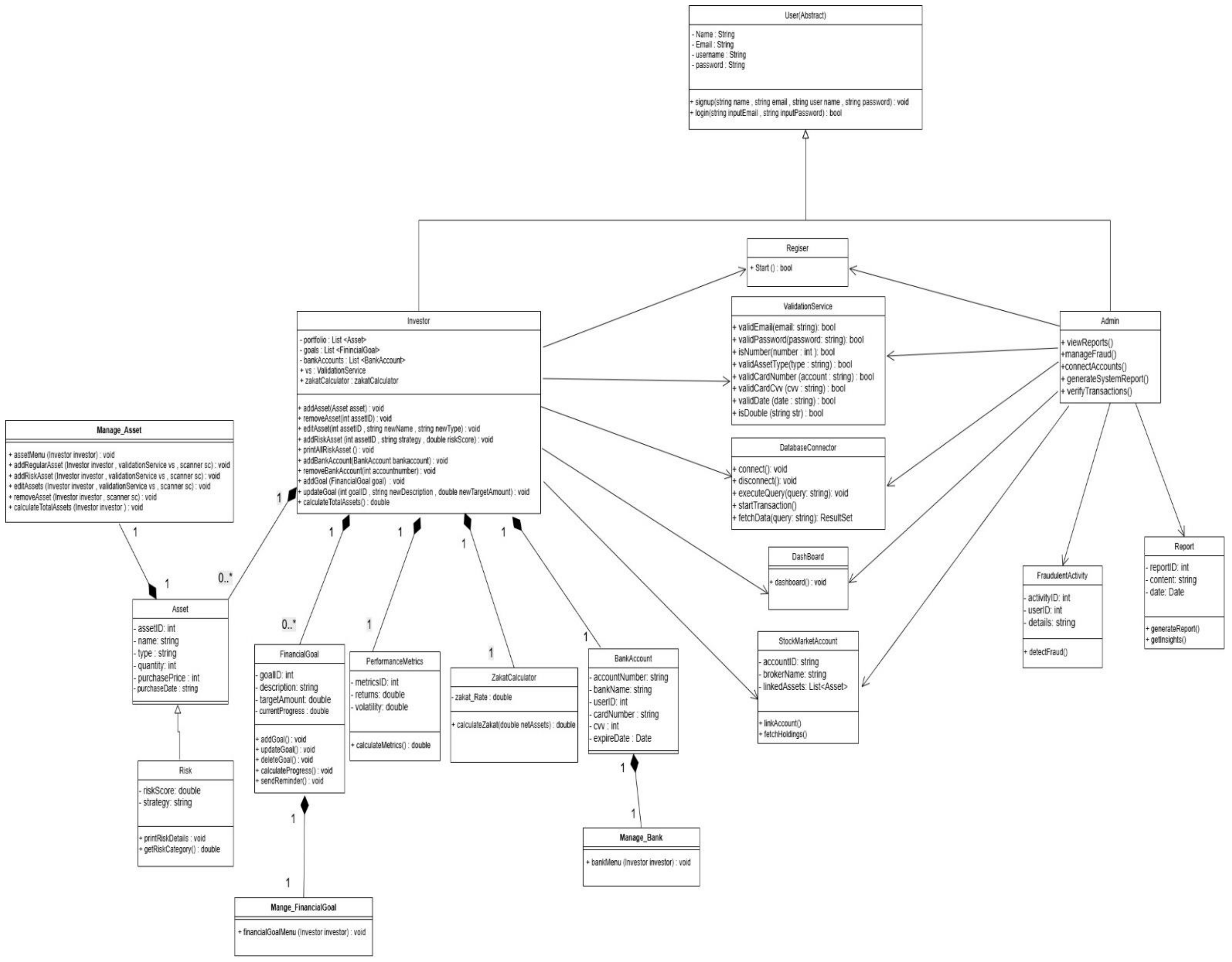
I. Architecture Diagram



CS251: <Stock Holders>
Project: <Invest>

Software Design Specification

II. Class Diagram(s)


User(Abstract)
- Name : String
- Email : String
- username : String
- password : String
+ signup(string name , string email , string user name , string password) : void
+ login(string inputEmail , string inputPassword) : bool
Regiser
+ Start () : bool
ValidationService
+ validEmail(email: string): bool
+ validPassword(password: string): bool
+ isNumber(number : int): bool
+ validAssetType(type : string) : bool
+ validCardNumber (account : string) : bool
+ validCardCvv (cvv : string) : bool
+ validDate (date : string) : bool
+ isDouble (string str) : bool
Admin
+ viewReports()
+manageFraud()
+connectAccounts()
+ generateSystemReport()
+ verifyTransactions()
Investor
- portfolio : List <Asset>
- goals : List <FinincialGoal>
- bankAccounts : List <BankAccount>
+ vs : ValidationService
+ zakatCalculator : zakatCalculator
+ addAsset(Asset asset) : void
+ removeAsset(int assetID) : void
+ editAsset(int assetID , string newName , string newType) : void
+ addRiskAsset (int assetID , string strategy , double riskScore) : void
+ printAllRiskAsset () : void
+ addBankAccount(BankAccount bankaccount) : void
+ removeBankAccount(int accountnumber) : void
+ addGoal (FinancialGoal goal) : void
+ updateGoal (int goalID , string newDescription , double newTargetAmount) : void
+ calculateTotalAssets() : double
DatabaseConnector
+ connect(): void
+ disconnect(): void
+ executeQuery(query: string): void
+ startTransaction()
+ fetchData(query: string): ResultSet
Manage_Asset
+ assetMenu (Investor investor) : void
+ addRegularAsset (Investor investor , validationService vs , scanner sc) : void
+ addRiskAsset (Investor investor , validationService vs , scanner sc) : void
+ editAssets (Investor investor , validationService vs , scanner sc) : void
+ removeAsset (Investor investor , scanner sc) : void
+ calculateTotalAssets (Investor investor) : void
1
1
1
0..*
1
1
1
1
1
DashBoard
+ dashboard() : void
Report
- reportID: int
- content: string
- date: Date
+ generateReport()
+ getInsights()
FraudulentActivity
- activityID: int
- userID: int
- details: string
+ detectFraud()
Asset
- assetID: int
- name: string
- type : string
- quantity: int
- purchasePrice : int
- purchaseDate : string
0..*
1
1
1
StockMarketAccount
- accountID: string
- brokerName: string
- linkedAssets: List<Asset>
+ linkAccount()
+ fetchHoldings()
FinancialGoal
- goalID: int
- description: string
- targetAmount: double
- currentProgress : double
+ addGoal() : void
+ updateGoal() : void
+ deleteGoal() : void
+ calculateProgress() : void
+ sendReminder() : void
PerformanceMetrics
- metricsID: int
- returns: double
- volatility: double
+ calculateMetrics() : double
ZakatCalculator
- zakat_Rate : double
+ calculateZakat(double netAssets) : double
BankAccount
- accountNumber: string
- bankName: string
- userID: int
- cardNumber : string
- cvv : int
- expireDate : Date
Risk
- riskScore: double
- strategy: string
+ printRiskDetails : void
+ getRiskCategory() : double
1
1
1
1
Manage_Bank
+ bankMenu (Investor investor) : void
Mange_FinancialGoal
+ financialGoalMenu (Investor investor) : void

Software Design Specification

III. Class Descriptions: -

Class ID	Class Name	Description & Responsibility	Refactoring Recommendations
1	Controller	Central control unit for managing interactions between other classes.	Divide if handling unrelated tasks; otherwise, clarify responsibilities.
2	Database Connector	Handles database operations (connect, execute queries, fetch data, manage transactions).	Well-defined. No changes needed.
3	Visitations Service	Validates user inputs (email, password, fields) and transaction data.	Rename to Validation Service for clarity. Keep as is.
4	User (Abstract)	Base class for shared user attributes (name, email, etc.) and actions (login, signUp).	Well-structured. No changes needed.
5	Investor	Extends User to manage portfolios, financial goals, and bank accounts. Tracks progress/zakat.	Well-defined. No changes needed.
6	Stock Market Account	Manages stock market account details (broker, linked assets) and fetches holdings.	Small but specific. Merge with Investor if overlapping responsibilities.
7	Admin	Extends User for administrative tasks (fraud management, reports, transaction verification).	Fix typo in romance Accounts (likely manage Accounts). Clarify or remove ambiguous methods.
8	Asset	Represents financial assets (ID, name, value) and supports CRUD operations.	Well-defined. No changes needed.
9	Financial Goal	Tracks financial goals (target amount, description) and calculates	Well-structured. No changes needed.

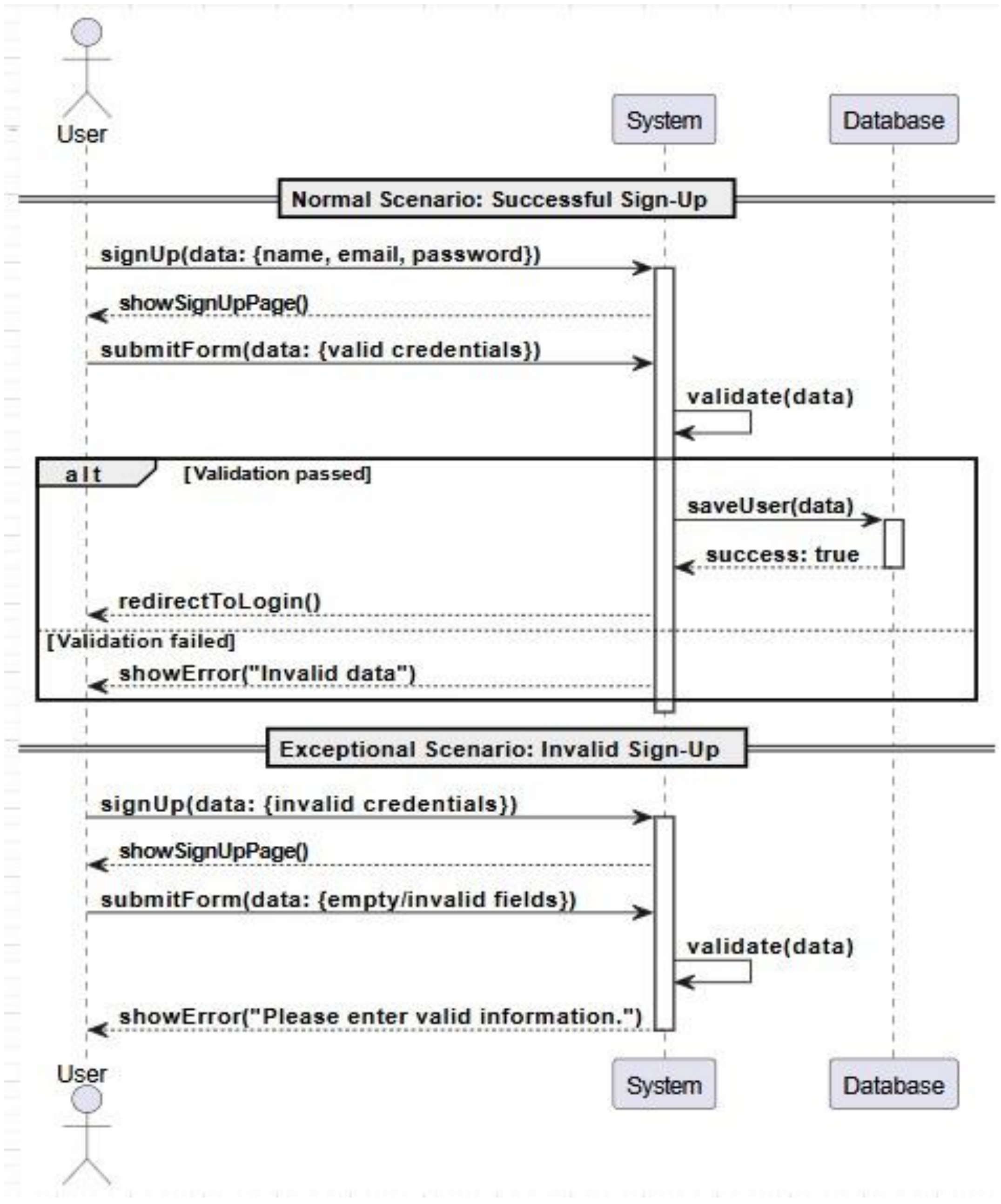
		progress/reminders.	
10	Bank Account	Stores bank account details (number, bank name, card info).	Fix typo income Bank (likely link Bank). Small but necessary.
11	Risk Assessment	Calculates risk scores and strategies for investments.	Very small. Merge with Performance Metrics or Investor.
12	Performance Metrics	Tracks financial metrics (returns, volatility) and calculates performance.	Very small. Merge with Risk Assessment into a Financial Analysis Service.
13	Zakat Calculator	Calculates zakat dues based on assets. Incomplete (methods missing).	Merge with Investor or expand into a full service.

CS251: <Stock Holders>
Project: <Invest>

Software Design Specification

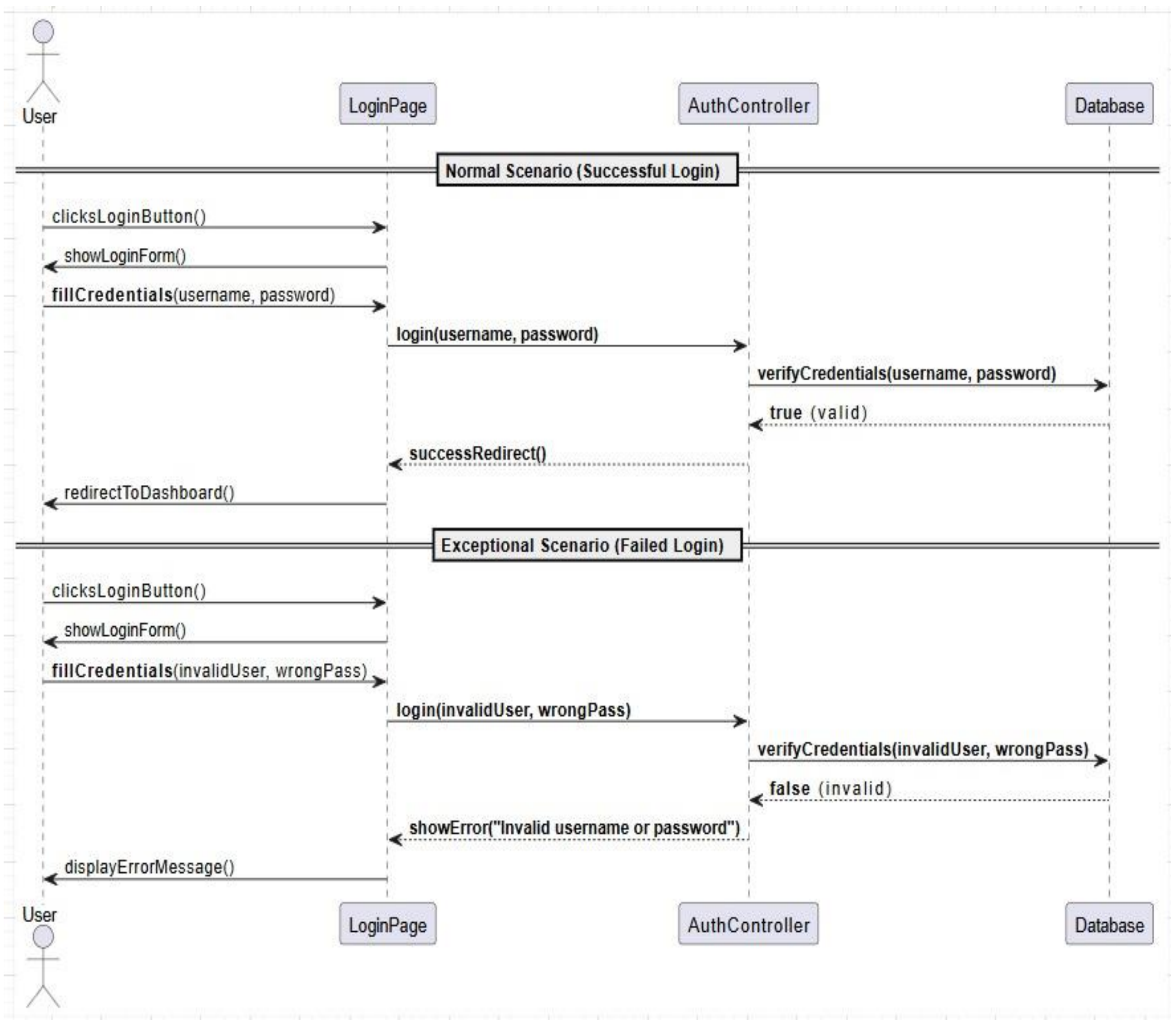
IV. Sequence diagrams

User Story #1


User
System
Database
Normal Scenario: Successful Sign-Up
signUp(data: {name, email, password})
showSignUpPage()
submitForm(data: {valid credentials})
validate(data)
alt
[Validation passed]
saveUser(data)
success: true
redirectToLogin()
[Validation failed]
showError("Invalid data")
Exceptional Scenario: Invalid Sign-Up
signUp(data: {invalid credentials})
showSignUpPage()
submitForm(data: {empty/invalid fields})
validate(data)
showError("Please enter valid information.")
User
System
Database

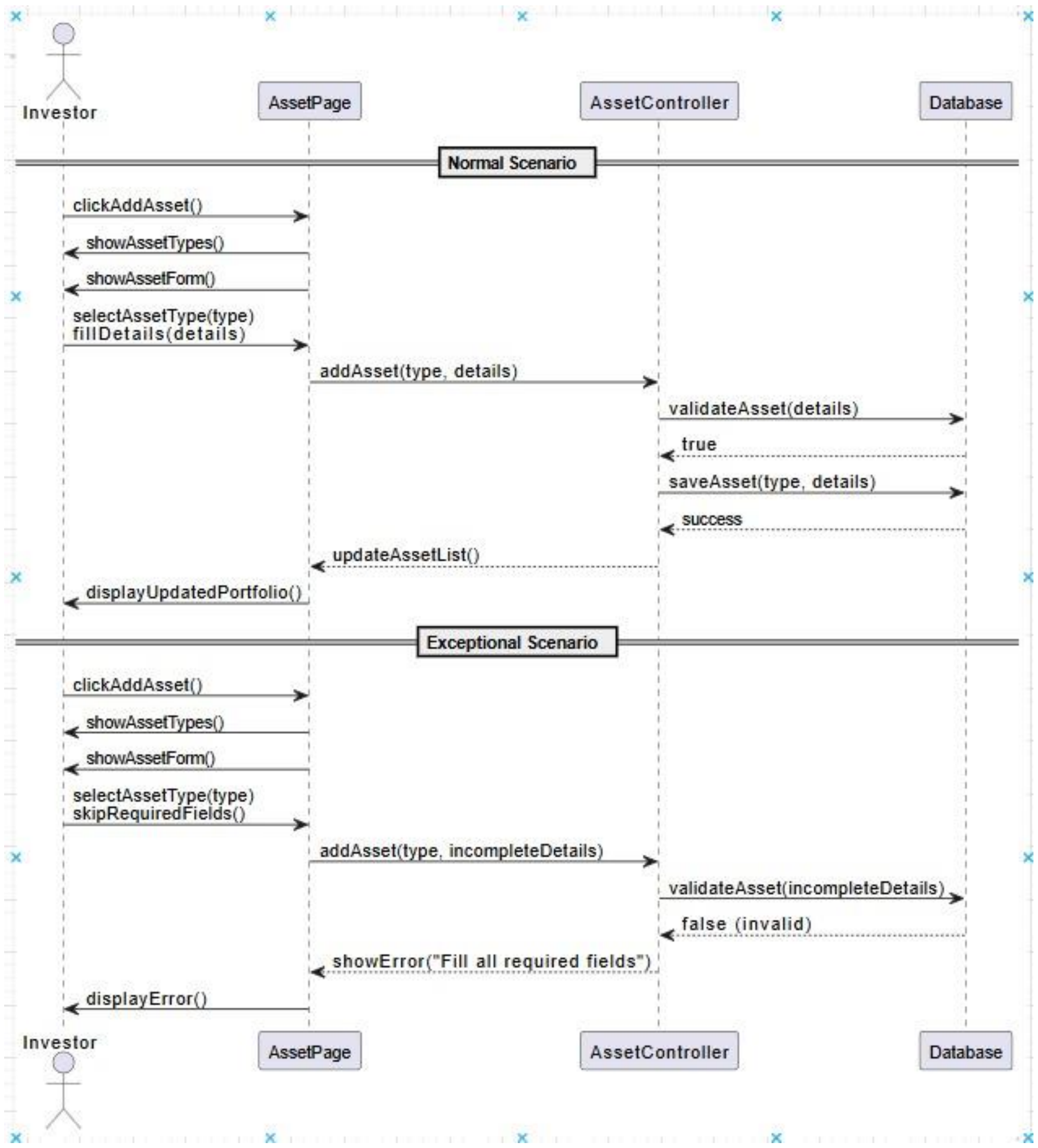
Software Design Specification

User Story #2

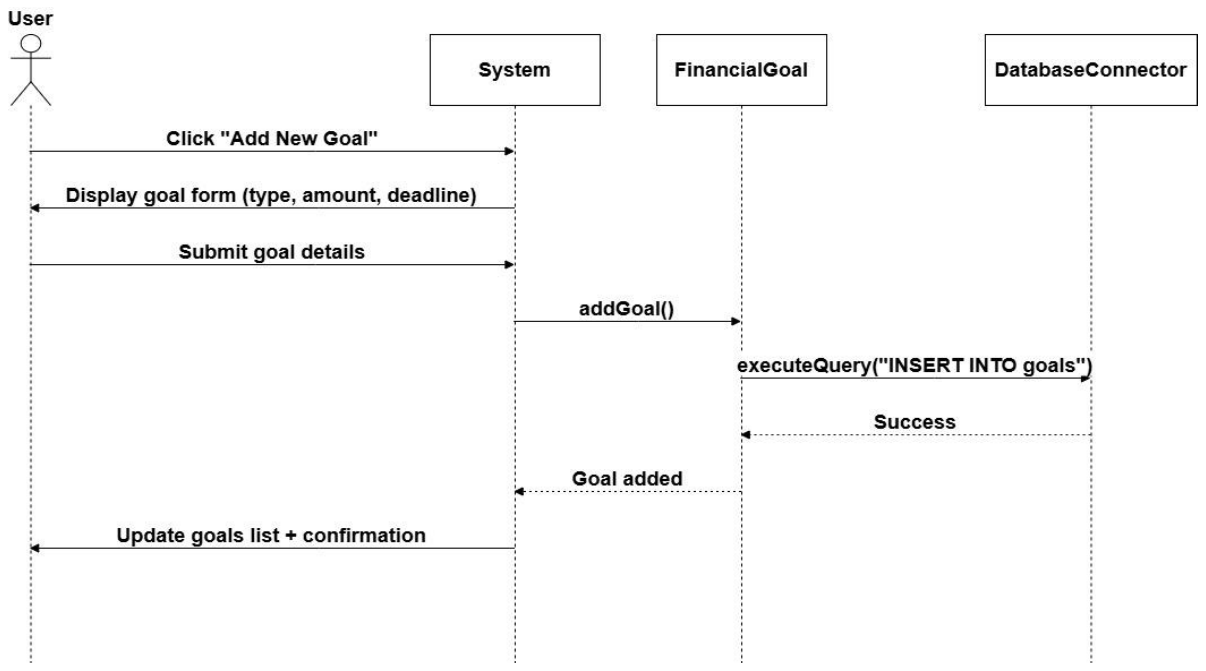

User
LoginPage
AuthController
Database
Normal Scenario (Successful Login)
clicksLoginButton()
showLoginForm()
fillCredentials(username, password)
login(username, password)
verifyCredentials(username, password)
true (valid)
successRedirect()
redirectToDashboard()
Exceptional Scenario (Failed Login)
clicksLoginButton()
showLoginForm()
fillCredentials(invalidUser, wrongPass)
login(invalidUser, wrongPass)
verifyCredentials(invalidUser, wrongPass)
false (invalid)
showError("Invalid username or password")
displayErrorMessage()
User
LoginPage
AuthController
Database

Software Design Specification

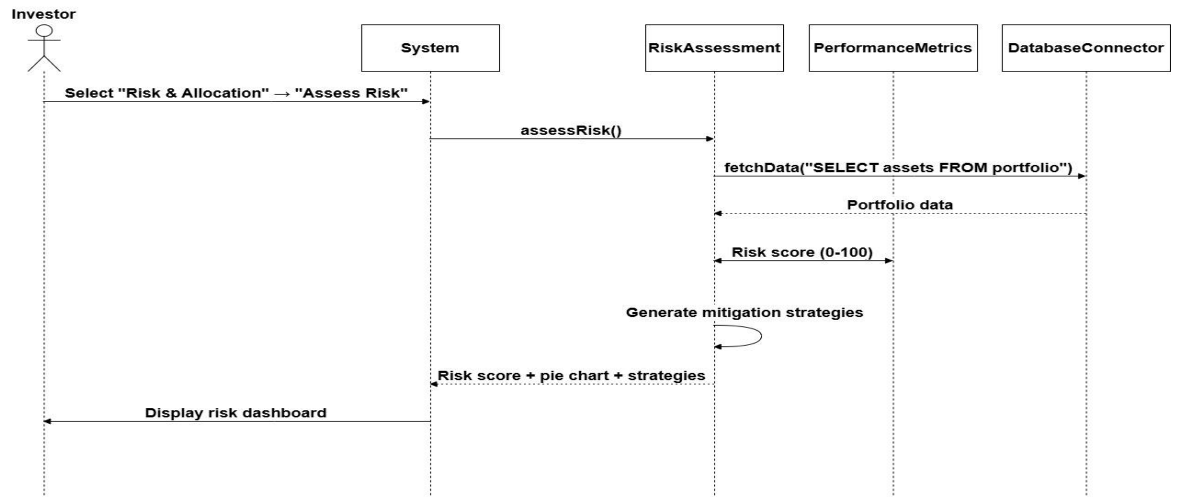
User Story #3



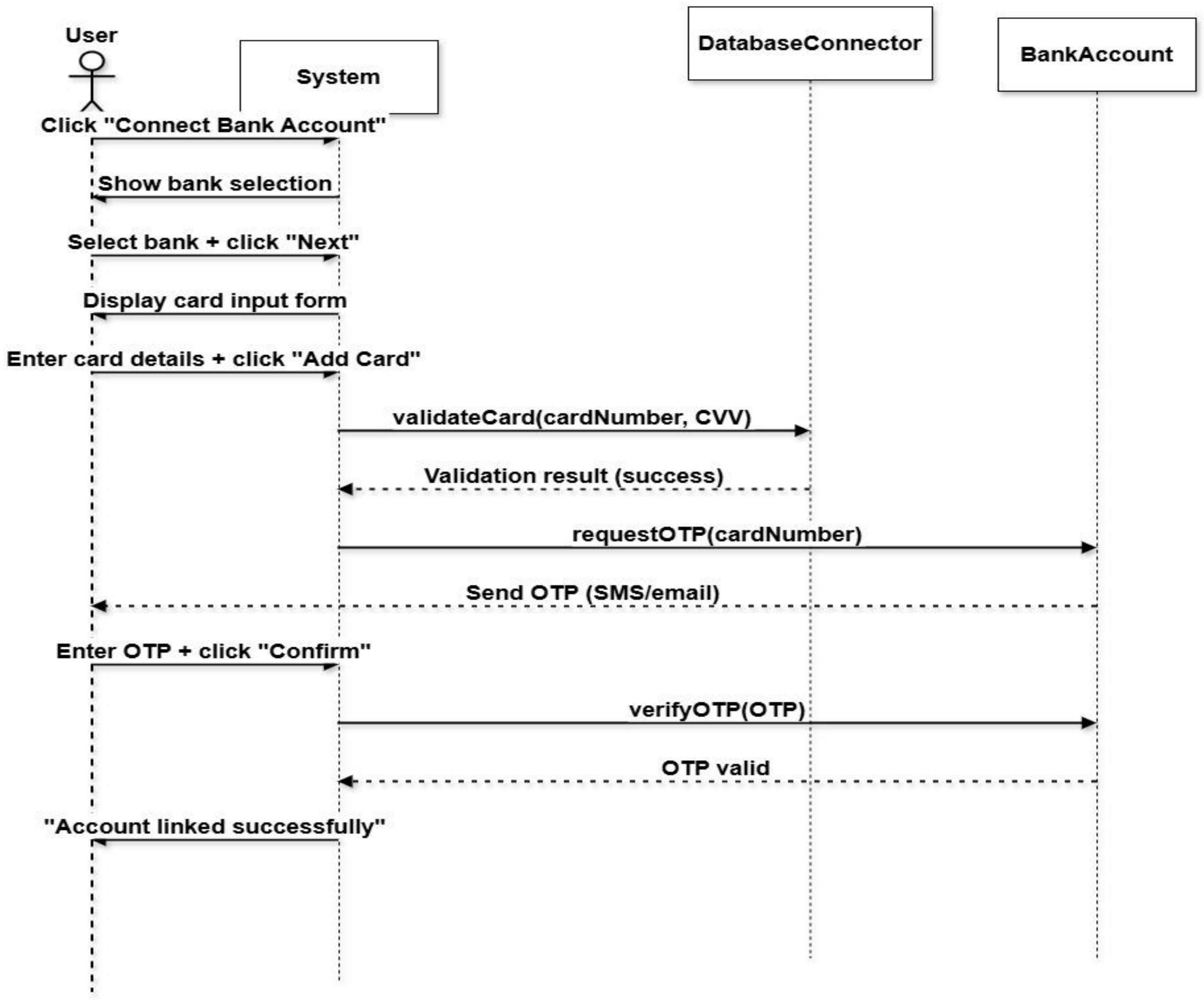
User Story #5


User
System
FinancialGoal
DatabaseConnector
Click "Add New Goal"
Display goal form (type, amount, deadline)
Submit goal details
addGoal()
executeQuery("INSERT INTO goals")
Success
Goal added
Update goals list + confirmation

User Story #6


Investor
System
RiskAssessment
PerformanceMetrics
DatabaseConnector
Select "Risk & Allocation" → "Assess Risk"
assessRisk()
fetchData("SELECT assets FROM portfolio")
Portfolio data
Risk score (0-100)
Generate mitigation strategies
Risk score + pie chart + strategies
Display risk dashboard

User Story #10


User
System
DatabaseConnector
BankAccount
Click "Connect Bank Account"
Show bank selection
Select bank + click "Next"
Display card input form
Enter card details + click "Add Card"
validateCard(cardNumber, CVV)
Validation result (success)
requestOTP(cardNumber)
Send OTP (SMS/email)
Enter OTP + click "Confirm"
verifyOTP(OTP)
OTP valid
"Account linked successfully"

CS251: <Stock Holders>
Project: <Invest>

Software Design Specification

Class - Sequence Usage Table

Sequence Diagram	Classes Used	All Methods Used
User Story #1	User	login(), viewProfile()
	Investor	getRiskProfile()
	Portfolio	calculateTotalValue()
	Risk Analyzer	assessRisk()
User Story #2	User	login()
	Investor	addFinancialGoal()
	Financial Goal	setTargetAmount(), calculateProgress()
	Portfolio	syncWithBankAccount()
	Bank Account	fetchTransactions()
User Story #3	User	login()
	Investor	addAsset()
	Asset (Abstract)	create() (Factory Method)
	Stock	setSymbol(), fetchRealTimePrice()
	Stock API	getPrice()
User Story #5	User	login()
	Investor	requestReport()
	Portfolio	generateReport()
	Report	exportToPDF()
	Notification	sendAlert()

User Story #6	User	login()
	Investor	calculateZakat()
	Portfolio	getTotalValue()
	Zakat Calculator	computeZakat()
	Notification	sendAlert()
User Story #10	User	login()
	Investor	checkCompliance()
	Halal Screener	validateShariaCompliance()
	Stock	getComplianceStatus()
	Notification	sendAlert()

CS251: <Stock Holders>
Project: <Invest>

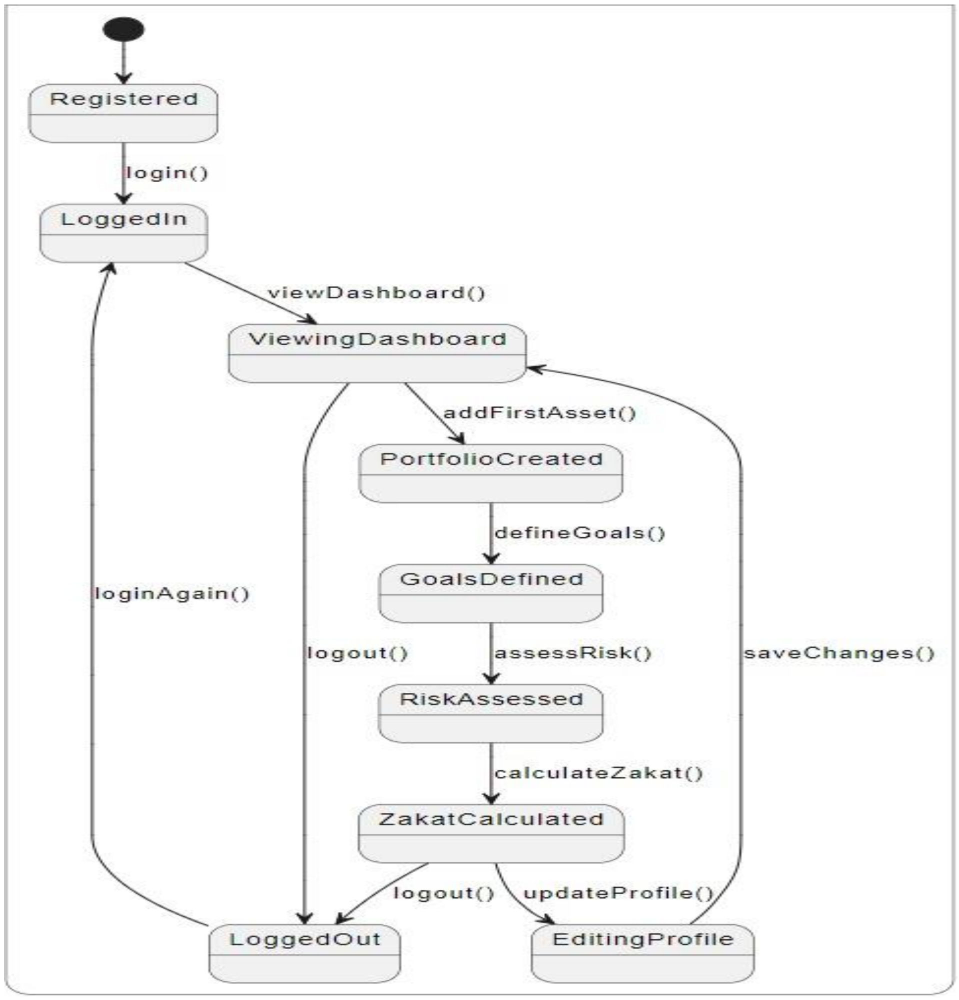
Software Design Specification

V. Class Responsibility Table:

Class ID	Class Name	Description & Responsibility
1	User	Represents an investor in the Investment system. Responsible for managing user authentication by handling sign-up and login processes
2	Portfolio	Represents a user's investment portfolio. Responsible for tracking total value, calculating net worth, and syncing with bank transactions for real-time updates.
3	Goal	Models a user's financial goal (e.g., retirement savings). Responsible for storing goal details and tracking progress toward the target amount by the deadline.
4	Risk Engine	Analyzes portfolio risk. Responsible for calculating a risk score and assessing risk distribution to support investment decisions.
5	Report Generator	Handles report generation for user portfolios. Responsible for creating financial reports in PDF and Excel formats for performance analysis.
6	Asset	Abstract class representing a generic investment asset. Responsible for defining the common structure for specific asset types.
7	Real Estate	Represents a real estate investment. Responsible for storing property-specific details like location and square footage.
8	Crypto	Represents a cryptocurrency investment. Responsible for storing crypto-specific details like ticker and blockchain.
9	Gold	Represents a gold investment. Responsible for storing gold-specific details like weight and karat.
10	Stock	Represents a stock investment. Responsible for storing stock-specific details like ticker and exchange, and fetching real-time prices via Stock API.

Class ID	Class Name	Description & Responsibility
11	Bank API	Interfaces with external banking systems (e.g., CIB). Responsible for syncing transaction data to update portfolio values.
12	Stock API	Interfaces with external stock market platforms (e.g., Thndr). Responsible for fetching real-time stock prices to update asset values.

V. Stat Diagram: -



Software Design Specification

VI. Solid Principle: -

1. Single Responsibility Principle (SRP)

Each class in the system has a well-defined responsibility. For example:

- **Zakat Calculator:** only calculates zakat.
- **Portfolio:** manages aggregation of assets.
- **Report:** handles report generation.
  This makes the system modular and easier to test and maintain.

2. Open/Closed Principle (OCP)

The system is designed to be extendable without modifying existing code.

- The **Asset** class is abstract, and new asset types **(e.g., Gold, Crypto)** can be added without changing the base class.
- Future asset types like Bonds or ETFs can be supported with minimal effort.

3. Dependency Inversion Principle (DIP)

High-level classes like **Portfolio** depend on abstractions **(Asset, Report),** not concrete implementations.

- **Portfolio** interacts with **Zakat Calculator** and **Risk Analyzer** through defined interfaces, enabling substitution with different implementations if needed.

VII. Design Patterns: -

1. Factory Pattern (Used for creating Assets)

The system uses a Factory method to instantiate different types of **Asset** objects **(Stock, Crypto, Gold)** based on user input, keeping creation logic centralized and clean.

2. Strategy Pattern (Used in Risk Assessment and Zakat Calculation)

Different strategies can be applied for risk analysis or zakat calculation.

- **Risk Analyzer** can plug in different algorithms for conservative vs. aggressive investors.
- **Zakat Calculator** could support region-based zakat rules in the future.

3. Observer Pattern (Used in Notifications)

The Notification class observes changes in **Portfolio** or **Zakat Calculator**.
When a report is generated or zakat is due, users receive a notification automatically.

Tools

Draw. oi

Ownership Report

Owner	Items
Mariam Badr	Architecture Diagram, 3 Sequence Diagram & them Table and some classes on class diagram & them Table
John Ayman	3 Sequence Diagram & them Table, some classes on class diagram & them Table
Emad George	Stat Diagram, Document Purpose and Audience, SOLID, Design Patterns and Class Responsibility Table: