**Cairo University  
Faculty of Computers and Artificial Intelligence** 

**CS251**

**Introduction to Software Engineering**

Tharwa

Software Design Specifications

Version 1.5

|  |  |  |
| --- | --- | --- |
| Name | ID | Email |
| Fatema El-Zhraa Ahmed Mohamed El-Fiky | 20230280 | fatmaelfeky922@gmail.com |
| Nagham Wael Mohamed | 20231189 | naghamw63@gmail.com |
| Aly El-Deen Yasser Ali | 20231109 | ali.el.badry.747@gmail.com |

April of 2025

**Contents**

[Team 3](#_Toc196646571)

[Document Purpose and Audience 3](#_Toc196646572)

[Purpose. 3](#_Toc196646573)

[Audience. 3](#_Toc196646574)

[System Models 5](#_Toc196646575)

[I. Architecture Diagram 5](#_Toc196646576)

[II. Class Diagram(s) 6](#_Toc196646577)

[III. Class Descriptions 7](#_Toc196646578)

[IV. Sequence diagrams 12](#_Toc196646579)

[Class - Sequence Usage Table 13](#_Toc196646580)

[V. State Diagram 14](#_Toc196646581)

[VI. SOLID Principles 14](#_Toc196646582)

[VII. Design Patterns 14](#_Toc196646583)

[Tools 14](#_Toc196646584)

[Ownership Report 14](#_Toc196646585)

# Team

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Email** | **Mobile** |
| 20230280 | Fatema El-Zhraa Ahmed Mohamed El-Fiky | fatmaelfeky922@gmail.com | 01221990828 |
| 20231189 | Nagham Wael Mohamed | [naghamw63@gmail.com](mailto:naghamw63@gmail.com) | 01007600773 |
| 20231109 | Aly El-Deen Yasser Ali | ali.el.badry.747@gmail.com | 01286964627 |

# Document Purpose and Audience

## Purpose.

This SDS defines the design and structure of the Personal Investment Management Software. The software aims to help users track, analyze, and optimize their investment portfolios, ensuring informed financial decisions. It serves as a reference for consistent development, testing, and future improvements.

## Audience.

1. **Development Team**
   * Software Engineers/Developers: Backend and frontend developers who will implement the system architecture and codebase based on the design specifications.
   * Technical Leads: Senior developers who will oversee implementation and ensure alignment with design decisions.
   * QA Engineers: Testers who will use the design documentation to create test cases and verify system behavior.
2. **System Architects**
   * Professionals responsible for reviewing and approving the high-level system design and ensuring it meets all technical requirements.
3. **Project Stakeholders**
   * *Product Owners*: Non-technical stakeholders who need to understand how design decisions fulfill business requirements.
   * *Islamic Finance Experts*: Domain specialists who will verify Sharia-compliance aspects of the design.
   * *Banking Integration Partners*: Technical representatives from partner institutions (e.g., CIB) who need to understand integration points.
4. **Maintenance Team**
   * Future developers who will maintain, update, or extend the system and need comprehensive design documentation.

# System Models

## I. Architecture Diagram

* **Decide on suitable software architecture for this system. Describe the architecture you chose and why it is suitable for the project.**
* **Divide your system into componenets or packages.**
* **Provide an architecture diagram showing the different components of the system and their relation to each other. Use suitable notation like C4 or arrow and box.**

## II. Class Diagram(s)

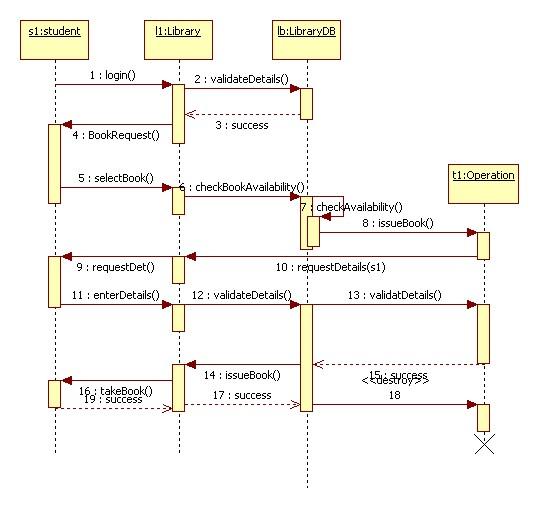
* **You should provide your class diagram. Diagram should show the components and packages in your system as well as all classes, their attributes and operations and their relations.**
* **In case one diagram is so complex, divide it to several ones of reasonable size or draw separate ones, each for one of the components on the system decomposition diagram.**
* **Class diagram is a static diagram and should not represent any dynamic flow of events.**
* **Put stereotypes of the classes to give more information. UML predefines some stereotypes like: <<interface>>, <<type>>, <<implementationClass>>, <<enumeration>>, etc. and you create your own also.**
* **Put Relationships between classes and the types of the relationships.**
* **Put multiplicity.**
* **Put relationship name (e.g. faculty "offer" course).**
* **Put attributes in the classes and their types and visibility.**
* **Put functions, parameters and return types.**
* **Include all domain (entity), boundary and control classes needed to implement the system.**
* **The following is a Shopping Cart Component class diagram example.**

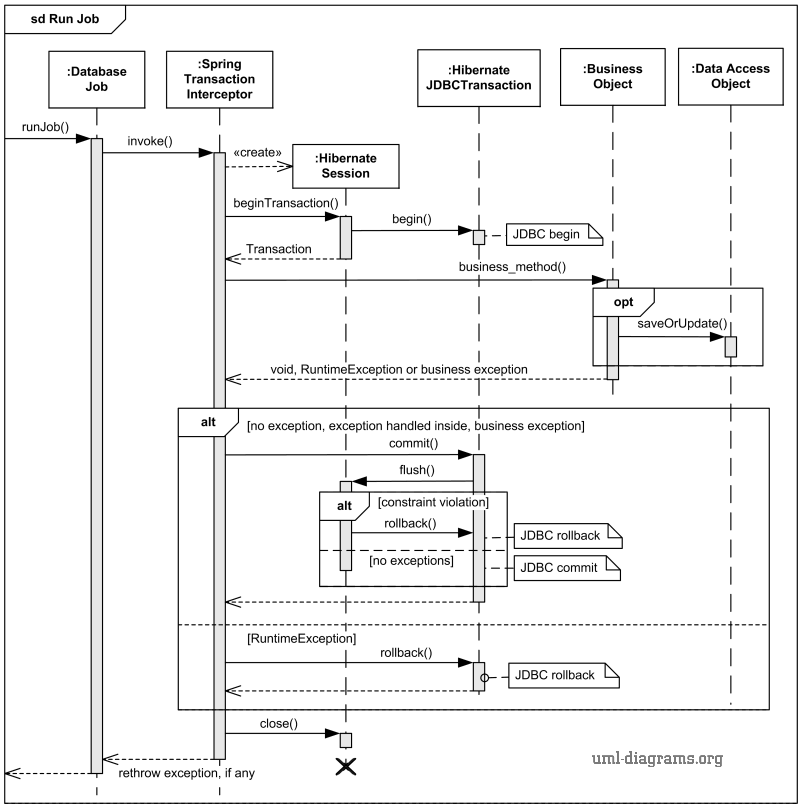
## III. Class Descriptions

| **ID** | **Class Name** | **Description & Responsibility** |
| --- | --- | --- |
| 1. | **Asset** | **Description**: Represents a financial asset (stocks, real estate, crypto, gold) in a user’s portfolio.  **Responsibilities**:   * Store asset details (Name, Type, Quantity, purchaseTime). * Link to financial goals (Goals: Vector<Goal>). |
| 2. | **Person** | **Description**: Base class storing common attributes for all human actors.  **Responsibilities**:   * Manage core attributes (Username, Name, Password, Email). |
| 3. | **User** (Extends Person) | **Description**: Represents an investor with linked assets and bank accounts.  **Responsibilities**:   * Maintain collections of Assets and BankAccounts (Accounts: Vector<BankAccount>). * Provide methods to fetch assets/accounts. |
| 4. | **BankAccount** | **Description**: Stores linked bank/card details for transactions.  **Responsibilities**:   * Secure sensitive data (OTP, CVV, cardNumber). |
| 5. | **Goal** | **Description**: Tracks financial objectives (e.g., retirement savings).  **Responsibilities**:   * Store goal metrics (targetAmount, deadline, currentProgress). * Update progress via setters (setProgress()). |
| 6. | **AuthController** | **Description**: Handles authentication and user sessions.  **Responsibilities**:   * Verify credentials (Verification(user: Person): boolean). * Manage signup/login flows (SignUp(), Login()). * Interact with UserRepo for persistence. |
| 7. | **ZakatPanel** (Interface) | **Description**: Defines contracts for Sharia-compliant zakat operations.  **Responsibilities**:   * Declare methods for zakat calculation (ZakatCalculation()). * Enforce halal screening (HalalInvestScreen()). |
| 8. | **RiskAndAdvicePanel** (Interface) | **Description**: Specifies risk assessment and investment advice features.  **Responsibilities**:   * Define risk analysis methods (AssetRisk()). * Outline strategy optimization (OptimizeStrategy()). |
| 9. | **FinGoalPanel** (Interface) | **Description**: Template for financial goal tools  **Responsibilities**:   * Declare goal tracking methods (ViewGoals(), TrackProgress()). * Support adding new goals (AddNewGoal()). |
| 10. | **Admin(Extends Person)** | **Description**  Represents an administrator in the system.  **Responsibilities:**  Potentially manage users, verify fraud activities, and oversee the system’s operations. |
| 11. | **AdminRepo<<Implementation>>** | **Description**  Represent the interface that the admin will interact with  Responsibilities:   * Check the Authorization of Admin License * Verify the Frauds of user (VerfiyFrauts ( user : User )) |
| 12. | **DashboardController** | **Description**  Controls the interaction between the user and the system's dashboard, coordinating with user data through a repository.  **Responsibilities**   * Manage the current User session. * Handle user operations by interacting with the UserRepo. * Display the dashboard menu to the user using ViewMenu() |
| 13. | **UserRepo<Implementation>** | **Description**  Repository class that stores all system users and their associated bank APIs.  **Responsibilities:**   * Maintain a Vector of User objects and a reference to a Bank API. * Provide access to user data through the method GetRepo(User user). |
| 14. | **InvestDashboardPanel (Interface)** | **Description**  Interface defining user investment-related operations on the dashboard.  **Responsibilities:**   * + **Add new** assets to a user’s profile.   + Remove assets from a user’s profile.   + View a user's current assets.   + Edit user assets.   + Evaluate **the user's investment performance.** |
| 15. | **Performance (Interface)** | **Description:**  Interface for measuring different aspects of user performance.  **Responsibilities:**   * Define a common implement() method that will be customized for various performance evaluations like ROI (Return on Investment), Asset Distribution, and Valuation. |
| 16. | **ROI** | **Description:**  A concrete class that implements performance evaluation based on Return on Investment.  **Responsibilities:**   * Implement the implement() method to calculate and analyze the user's return on their investments. |
| 17. | **AssestDistrib** | **Description:**  A concrete class that implements performance evaluation based on the distribution of a user’s assets.  **Responsibilities:**   * Implement the implement() method to analyze how the user's assets are spread across different categories. |
| 18. | **Valuation** | **Description:**  A concrete class that implements performance evaluation based on the overall valuation of a user's assets.  **Responsibilities:**   * Implement the implement() method to compute the total value of the user’s assets. |

## IV. Sequence diagrams

* **Usually each use case is represented by a sequence diagram or more.**
* **Draw a sequence diagram for the most important SIX use cases (user stories) that have complex interaction.**
* **Overall, all the diagrams should represent all requirements and possible flows for the use case.**
* **Make sure that each object in the sequence diagram has a corresponding class in the class description table above. If not, it will be REJECTED.**
* **Put actual function calls with proper parameters and return types corresponding to class diagrams.**
* **Following are couple of examples for small / medium examples. We expect such diagrams, however there is a missing thing in them. Most of calls don’t have parameters. Please always specify the parameters in the call, matching the class diagram.**





### Class - Sequence Usage Table

* **In this table, we will list the sequence diagrams you drew. For each one, list all the classes used in this sequence. For each class list all the methods you used in this class. Every method or object on a sequence diagram must belong to an existing class in the class diagram and be shown there. If sequence diagrams do not reflect actual classes and methods, they will be REJECTED.**

| **Sequence Diagram** | **Classes Used** | **All Methods Used** |
| --- | --- | --- |
| 1. Book Field | Class Field  Class Player | Methods …..  Methods …. |

## V. State Diagram

## VI. SOLID Principles

## 

## VII. Design Patterns

# 

# Tools

* Draw.io

# Ownership Report

|  |  |
| --- | --- |
| **Item** | **Owners** |
|  | Fatema El-Zhraa Ahmed Mohamed El-Fiky |
|  | Aly El-Deen Yasser Ali |
|  | Nagham Wael |