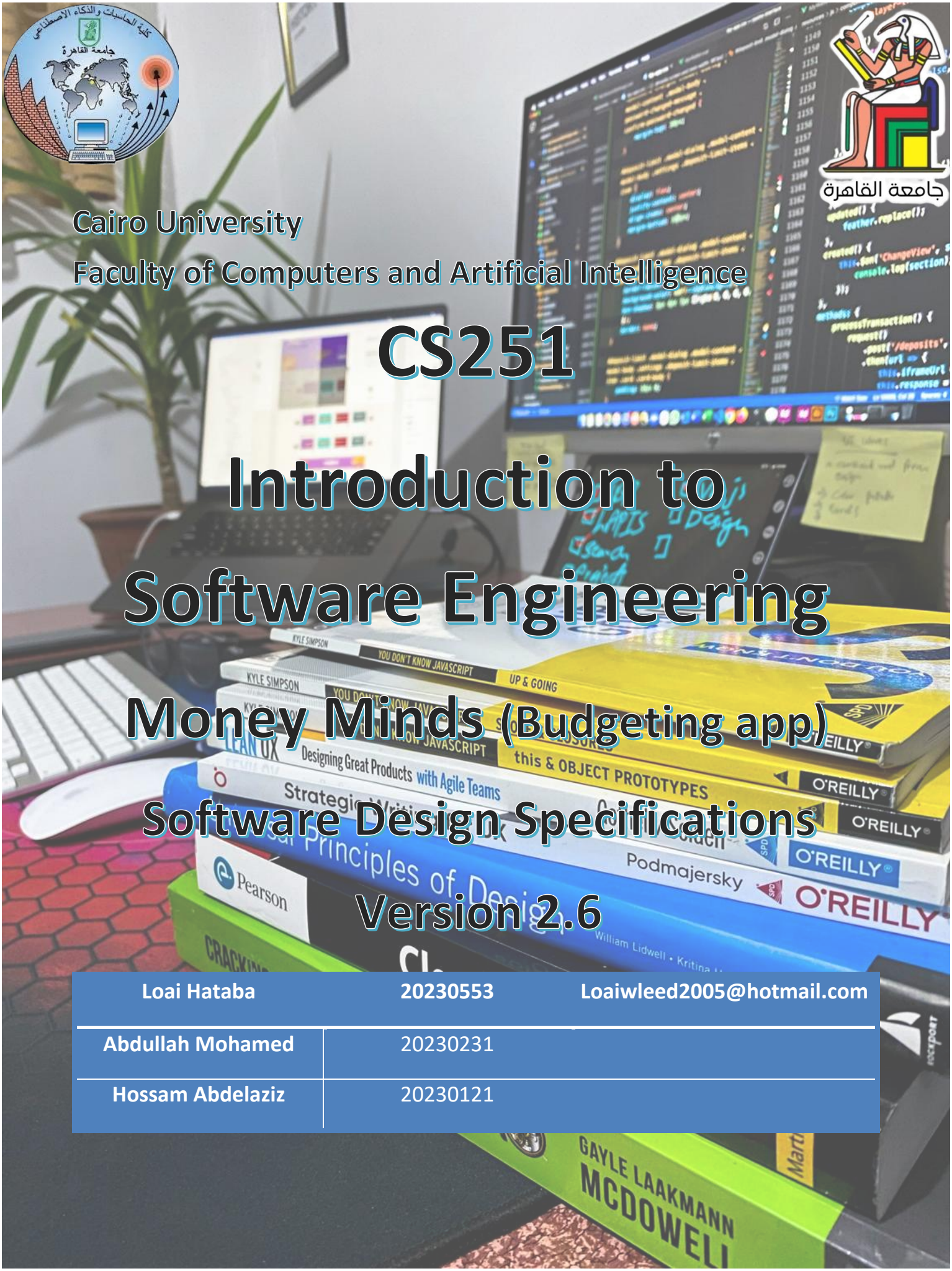


Cairo University

Faculty of Computers and Artificial Intelligence

CS251

Introduction to Software Engineering

Money Minds (Budgeting app)

Software Design Specifications

Version 2.6

Loai Hataba	**20230553**	**Loaiwleed2005@hotmail.com**
Abdullah Mohamed	20230231	
Hossam Abdelaziz	20230121	

CS251: HoodRatz
Project: Money Minds

Software Design Specification

Contents

CS251: HoodRatz
Project: Money Minds

Software Design Specification

Team

ID	Name	Email
20230553	Loai Hataba	20230553@stud.fci-cu.edu.eg
20230231	Abdullah Mohamed	20230231@stud.fci-cu.edu.eg
20230121	Hossam Abdelaziz	20230121@stud.fci-cu.edu.eg

Document Purpose and Audience

Purpose

- This document describes the design, structure, & functionality of the Budget Manager application.
- It explains how users can track their incomes, expenses, and generate financial reports.
- It outlines the main components, their responsibilities, and how they interact with each other.

Audience

- Developers – to understand the system architecture and build the application.
- Project Manager – to oversee the project development and ensure requirements are met.
- Testers/QA Team – to reference expected functionalities during testing.
- Potential Stakeholders (optional) – to review the overall app structure and features.

CS251: HoodRatz
Project: Money Minds

Software Design Specification

System Models

I. Architecture Diagram

Software Architecture Choice

For the Budget Manager application, we selected an **architecture** consisting of the **Frontend**, **Backend**, and **Database** layers, connected through APIs and supported by Authentication and Analytics services.
This architecture is suitable for the project because it provides:

- **Separation of concerns**: each layer has a specific responsibility (UI, business logic, data storage).
- **Scalability**: the application can grow by upgrading each tier independently.
- **Security**: user data can be protected through centralized authentication mechanisms.
- **Maintainability**: the structure simplifies debugging, updates, and future enhancements.

System Components

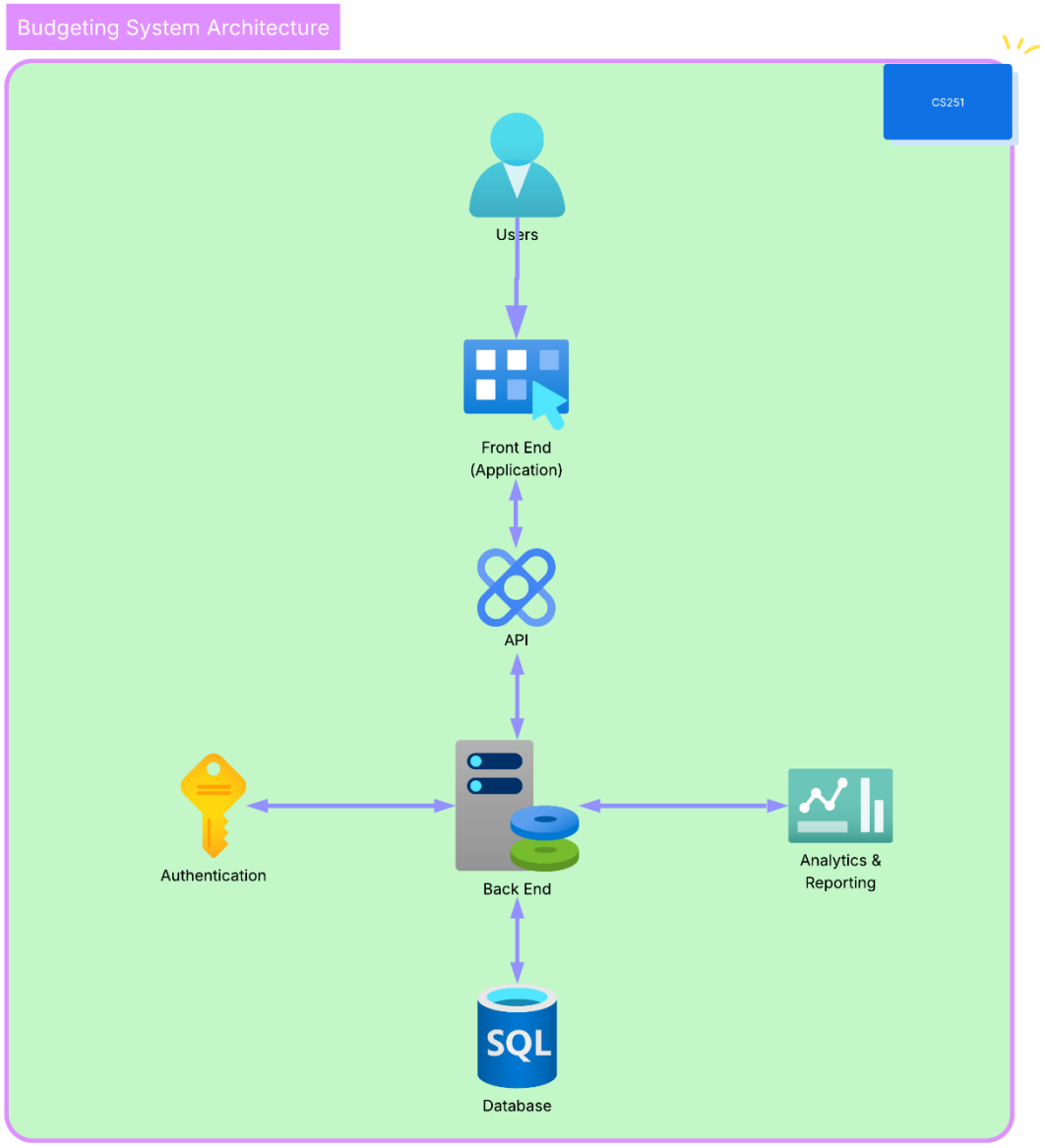
The system is divided into the following main components:

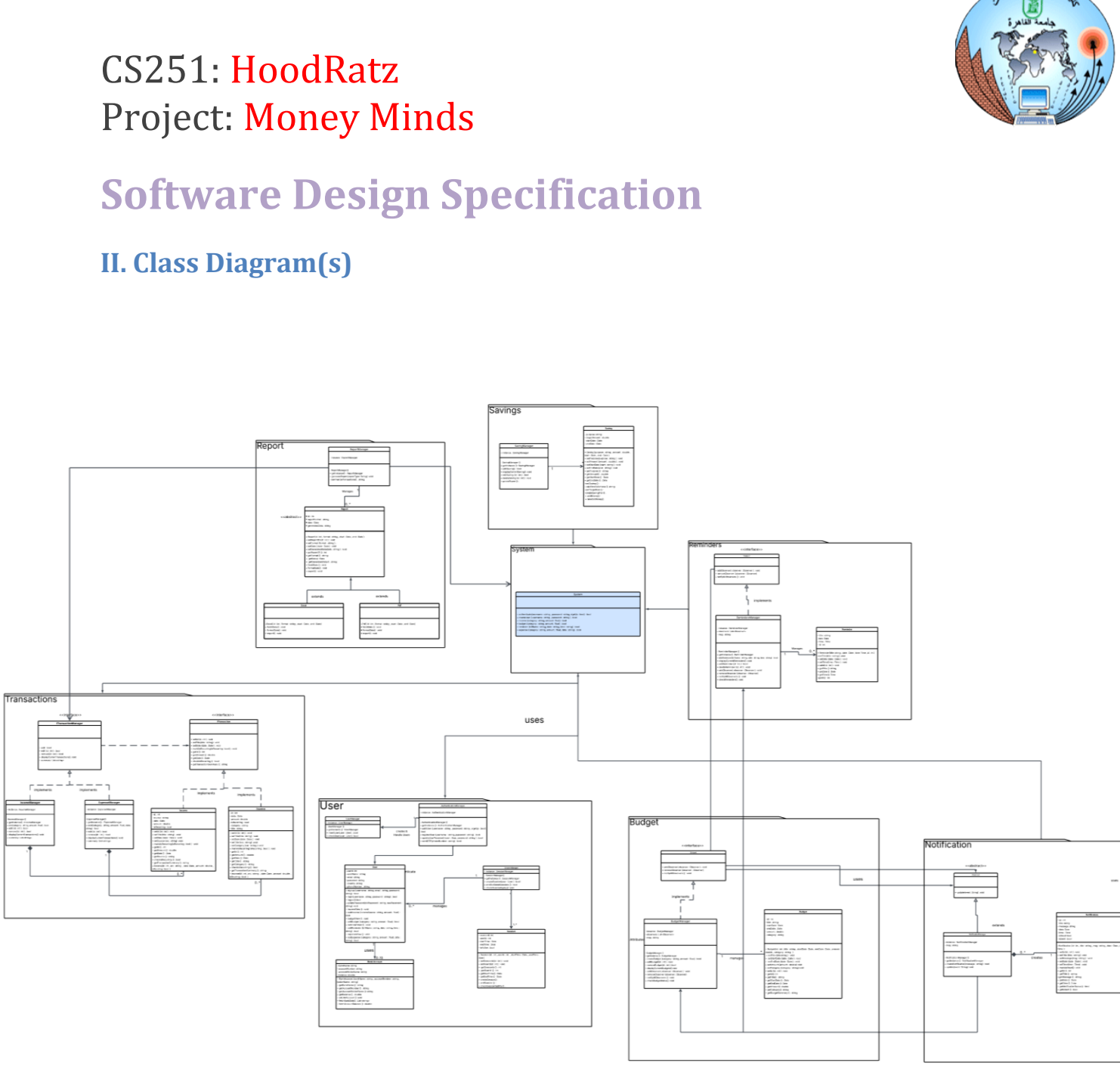
- **Users**: Individuals who interact with the application to manage their budgets.
- **Front End (Application)**: The graphical user interface that users interact with. It sends and receives data via APIs.
- **API**: Facilitates communication between the Front End and the Back End.
- **Back End**: Processes requests, applies business logic, manages authentication, reporting, and communicates with the database.
- **Authentication Service**: Handles user login, registration, and secure access management.
- **Database (SQL)**: Stores persistent data, including users' incomes, expenses, and transaction history.
- **Analytics & Reporting**: Generates financial reports and visual insights based on user data.

Software Design Specification

Architecture Diagram

The architecture diagram below shows the relationship between different components using a simple arrow-and-box notation:



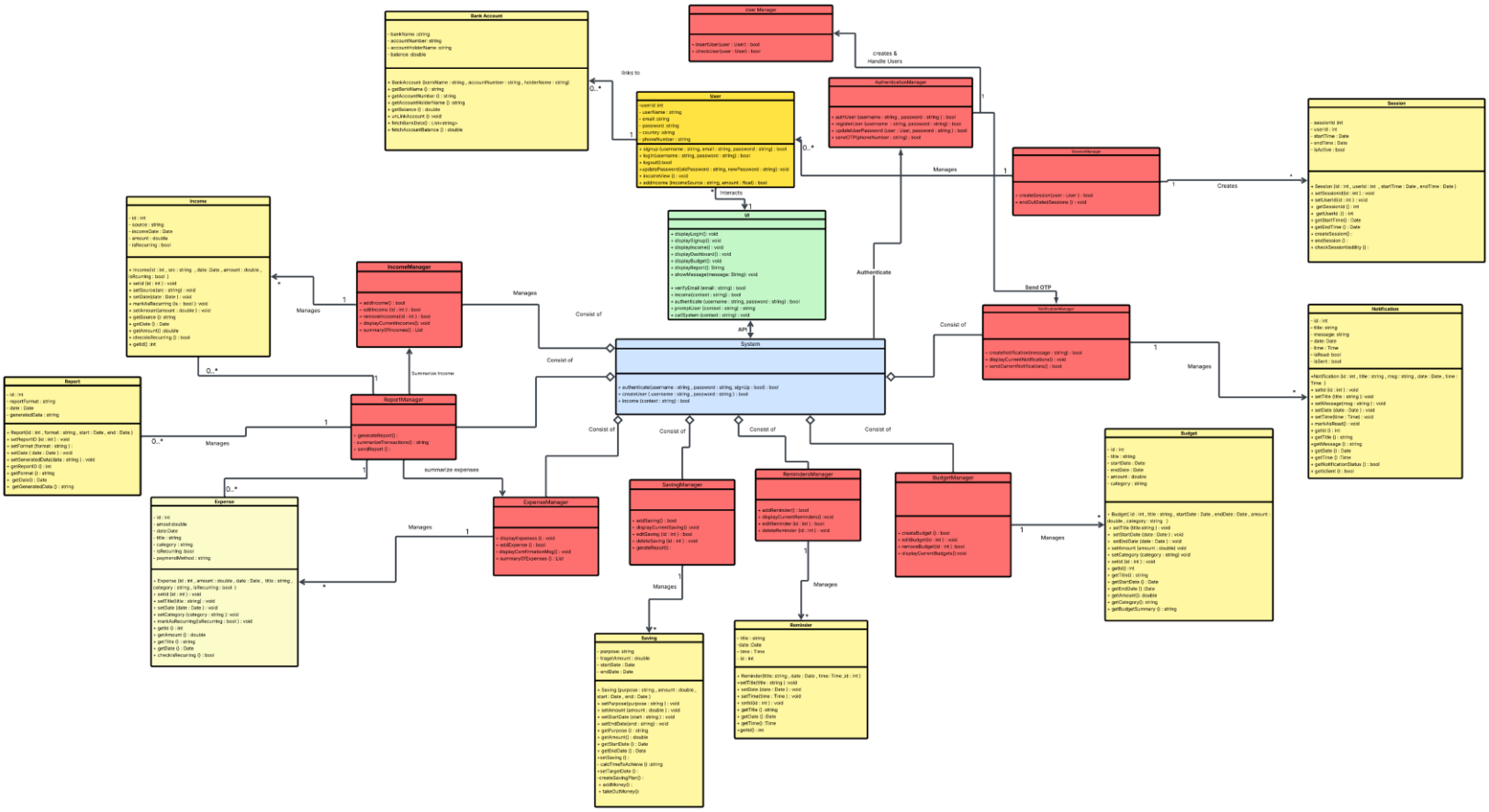


CS251: HoodRatz
Project: Money Minds

Software Design Specification

II. Class Diagram(s)

Software Design Specification



III. Class Descriptions

Class ID	Class Name	Description & Responsibility
1	Income	Represents an income entry with properties like source, amount, and date; responsible for managing income-related operations.
2	IncomeManager	Manages multiple Income objects; responsible for adding, deleting, retrieving, and summarizing incomes.
3	BankAccount	Represents a user's bank account details; responsible for storing account number, balance, and bank name.
4	Report	Represents financial reports; responsible for summarizing income and expenses over a time period.
5	ReportManager	Manages creation and retrieval of financial reports based on user data.
6	Expense	Represents an expense entry with properties like type, amount, and description; manages individual expense records.

Software Design Specification

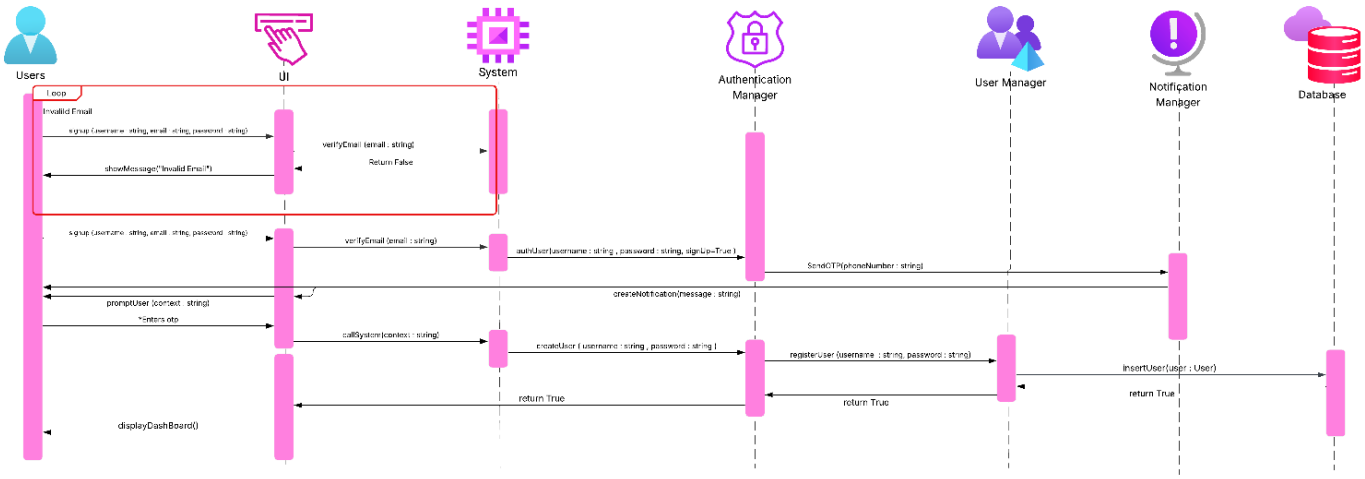
Class ID	Class Name	Description & Responsibility
7	ExpenseManager	Manages multiple Expense objects; responsible for adding, deleting, and retrieving expenses.
8	Saving	Represents a saving goal or entry; manages target amounts and current savings status.
9	SavingManager	Manages user savings; responsible for adding savings and generating saving reports.
10	User	Represents a system user with authentication credentials; manages personal user details.
11	UserManager	Manages creating and checking for users in the database.
12	Budget	Represents a budget plan for a category or time period; manages allocation and spending tracking.
13	BudgetManager	Manages user budgets; responsible for creating and managing budget plans.
14	Notification	*Represents a message or alert sent to users; responsible for delivering real-time updates, reminders, or warnings based on system events or user actions.*
15	Notification Manager	Represents a notification message; manages sending alerts to users.
16	AuthenticationManager	Responsible for verifying and managing user authentication (login/signup).
17	Reminder	*Represents a scheduled alert for important financial activities or goals; responsible for setting, updating, and managing reminders triggered at specific times or conditions.*
18	Reminder Manager	Represents a reminder entity; manages notification scheduling.
19	UI	Represents the front end of the application where the user would interact with the system.
20	System	Central class represents the entire system; that coordinates between managers and entities.
21	Session	Represents a user's active interaction period with the system; responsible for temporarily storing user data (such as login state…) during usage, until the session ends or expires.
22	Session Manager	*Responsible for creating, maintaining, and terminating user sessions; manages session-related data like active users, timeouts, and session validation to ensure continuous and secure user interaction.*

CS251: HoodRatz
Project: Money Minds

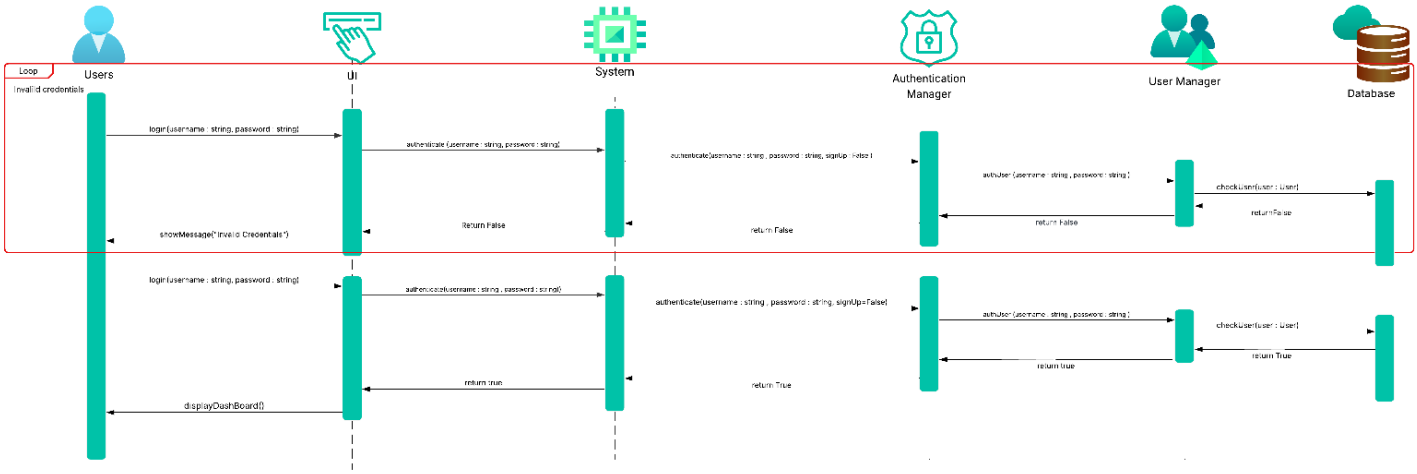
Software Design Specification

IV. Sequence diagrams

Sign Up

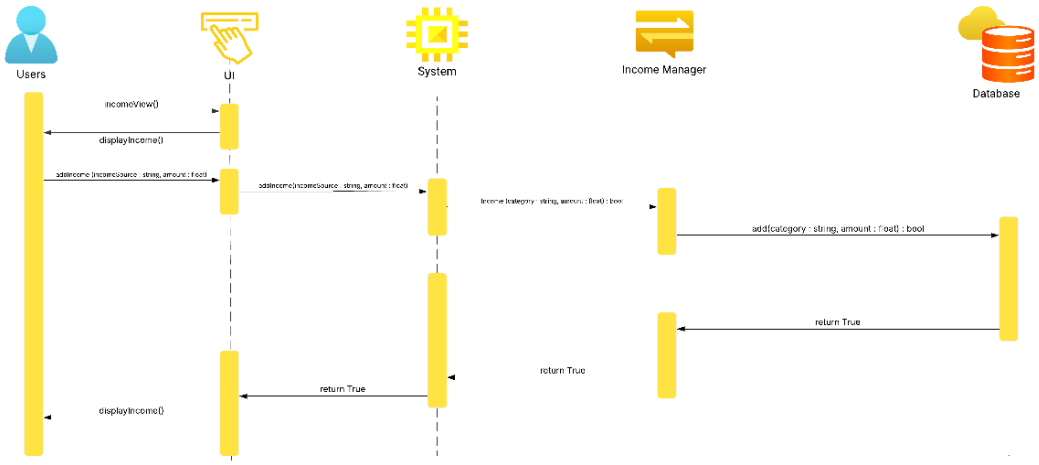


Log in

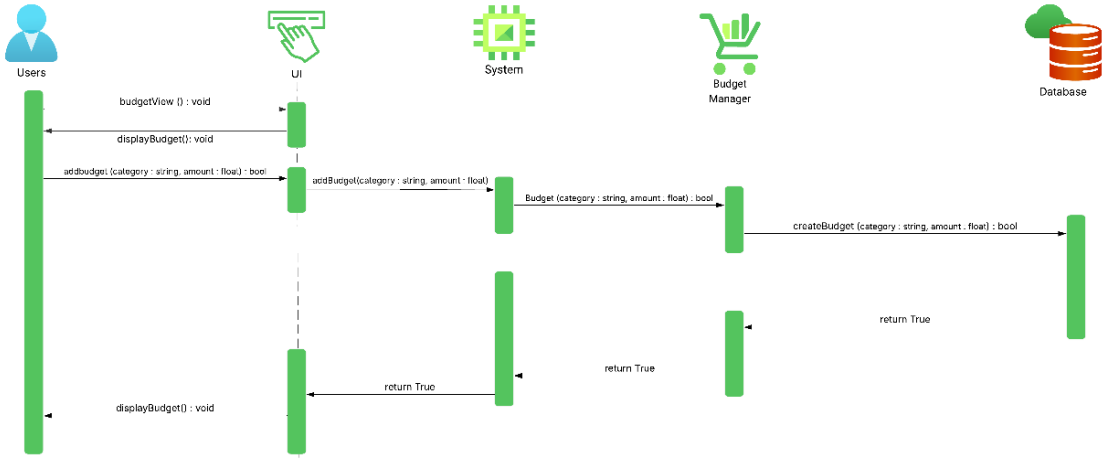


Software Design Specification

Tracking Income

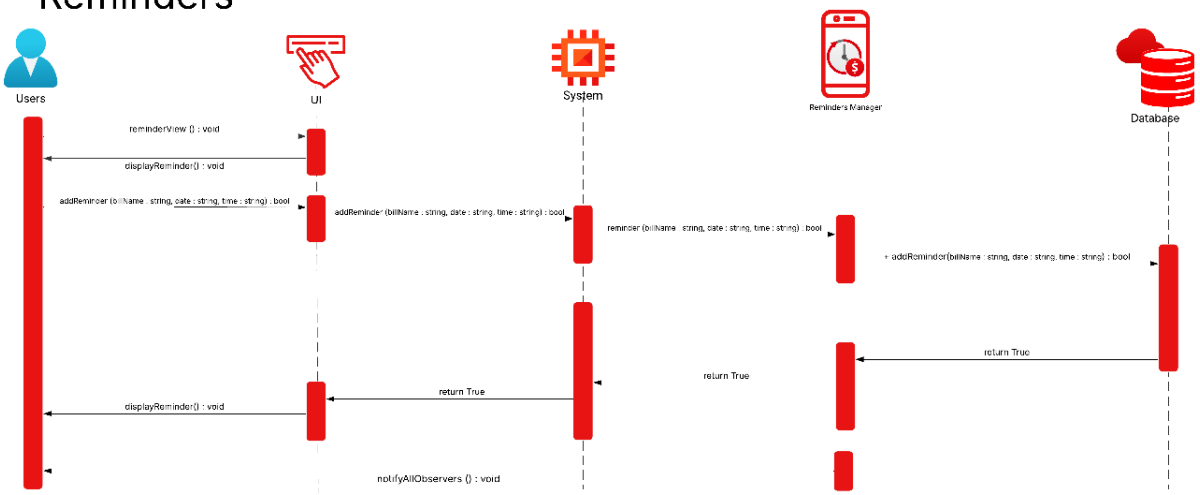


Budgeting & Analysis

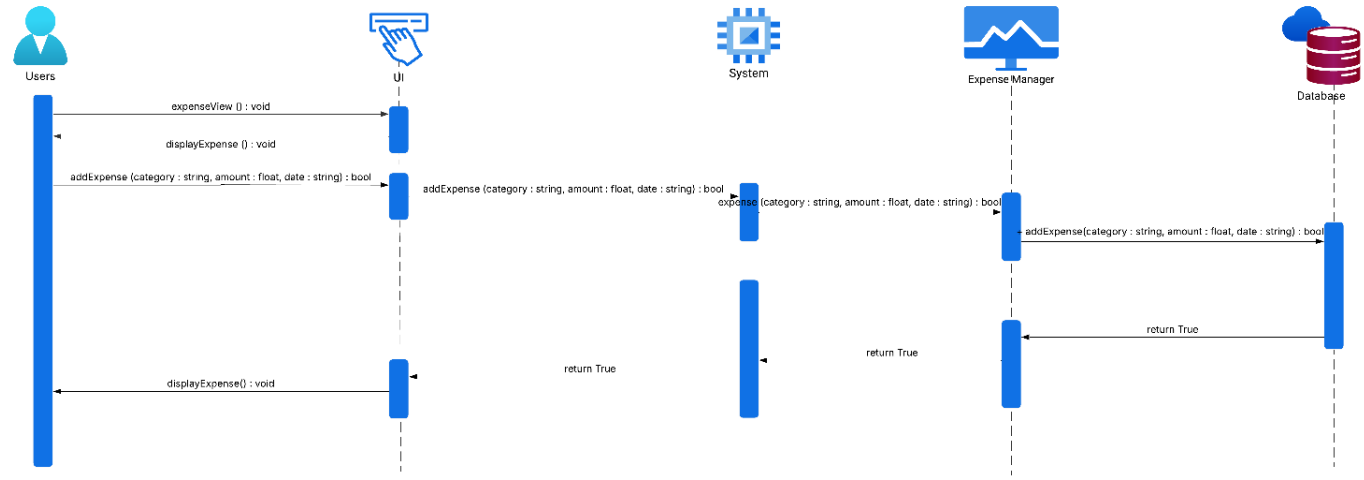


Software Design Specification

Reminders



Expense Tracking



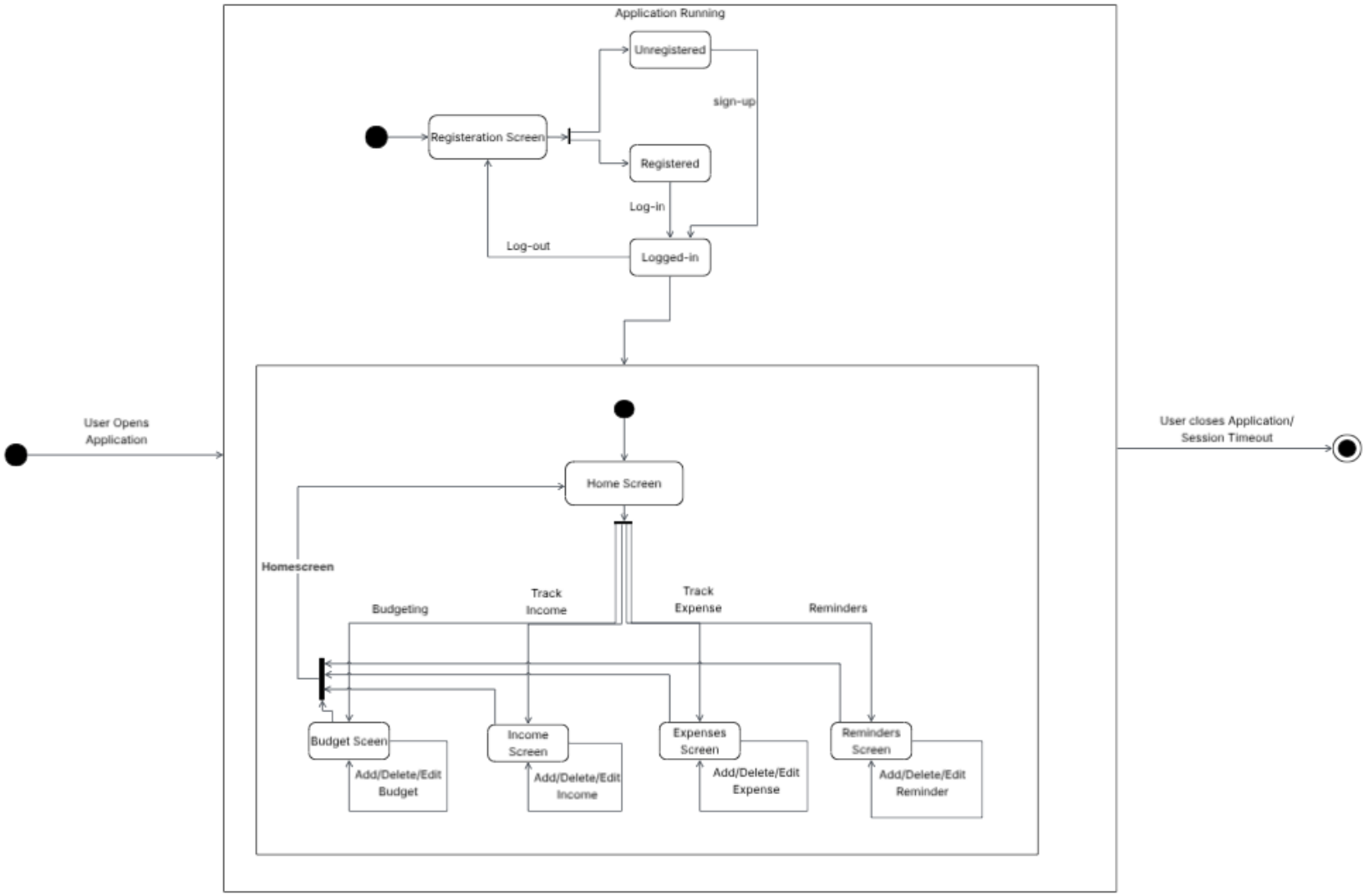
Software Design Specification

Class - Sequence Usage Table

Sequence Diagram	Classes Used	All Methods Used
1. Sign Up	Users UI System Authentication Manager User Manager Notification Manager	signup (username : string, email : string, password : string) showMessage("Invalid Email") verifyEmail (email : string) authenticate(username : string , password : string, signUp : bool) SendOTP(phoneNumber : string) SendCurrentNotifications() createNotification(message : string) promptUser (context : string) callSystem(context : string) createUser (username : string , password : string) registerUser (username : string, password : string) insertUser(user : User) displayDashBoard()
2. Log in	Users UI System Authentication Manager User Manager Notification Manager	login(username : string, password : string) authenticate (username : string, password : string) authUser (username : string , password : string) checkUser(user : User) showMessage("Invalid Credentials") displayDashBoard()
3. Track Income	Users UI System Income Manager	incomeView() displayIncome() addIncome (incomeSource : string, amount : float) income(context : string) addIncome() : bool
4. Budgeting & Analysis	Users UI System Budget Manager	budgetView () : void displayBudget(): void addbudget (category : string, amount : float) : bool Budget (category : string, amount : float) : bool createBudget (category : string, amount : float) : bool
5. Reminders	Users UI System Notification Manager	reminderView () : void displayReminder() : void addReminder (billName : string, date : string, time : string) : bool reminder (billName : string, date : string, time : string) : bool
6. Expense Tracking	Users UI System Expense Manager	expenseView () : void displayExpense () : void addExpense (category : string, amount : float, date : string) : bool expense (category : string, amount : float, date : string) : bool

Software Design Specification

V. State Diagram



Software Design Specification

VI. SOLID Principles

- In our design, we applied three SOLID principles to enhance modularity and maintainability. First, the Single Responsibility Principle was followed by ensuring each manager class—such as the BudgetManager, ExpenseManager, IncomeManager, ReminderManager, UserManager, and AuthenticationManager—handled only its specific domain logic, separating concerns cleanly and making each class easier to manage and test.

- We implemented the Open/Closed Principle by designing our classes to be open for extension but closed for modification; for example, we could add new types of reminders or income sources by extending base classes without altering existing code.

- The Liskov Substitution Principle was upheld by ensuring that any subclasses used could replace their base classes without affecting the functionality of the program, such as using different types of accounts or budget strategies that still conformed to a common interface or abstract class, maintaining consistent behavior throughout the system.

VII. Design Patterns

- We incorporated three design patterns to improve structure, reusability, and communication between components in our application.
- We used the singleton pattern for all our manager classes (such as BudgetManager, ExpenseManager, etc.) to ensure that only one instance of each manager exists throughout the program. This helped maintain consistent data access and state management across different parts of the system.
- We applied the template method pattern within the report package, where we defined a general report generation structure in a base class and allowed subclasses to implement specific formatting or content logic. This promoted code reuse and made it easier to add new report types in the future.
- We implemented the observer pattern to establish communication between the budget, notification, and reminder components. When a budget change occurred, relevant observers like the notification or reminder systems were automatically updated, enabling real-time responses and reducing tight coupling between these parts.

CS251: HoodRatz
Project: Money Minds

Software Design Specification

Tools

- LucidChart

Ownership Report

Item	Owners
System Architecture & Sequence Diagrams	Loai Hataba
Class Diagram, SOLID, & Design Patterns	*Abdullah Mohammed*
Class Responsibilities & State Diagram	Hossam Abdelaziz