A stack of books on a desk

AI-generated content may be incorrect.C

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 |  |

Contents

[Instructions [To be removed] 3](#_Toc17187)

[Team 3](#_Toc17879)

[Document Purpose and Audience 3](#_Toc22739)

[System Models 3](#_Toc5252)

[I. Architecture Diagram 3](#_Toc29105)

[II. Class Diagram(s) 4](#_Toc7901)

[III. Class Descriptions 5](#_Toc10477)

[IV. Sequence diagrams 6](#_Toc30452)

[Class - Sequence Usage Table 7](#_Toc9583)

[V. State Diagram 8](#_Toc30911)

[VI. SOLID Principles 8](#_Toc10551)

[VII. Design Patterns 8](#_Toc1153)

[Tools 8](#_Toc10433)

[Ownership Report 8](#_Toc25884)

# Instructions [To be removed]

* **IMPORTANT. Rename this document to**

**CS251-2025-SectionNumber-TAFirstName-Topic-IDs-DraftSDS.pdf // For draft version**

**CS251-2025-SectionNumber-TAFirstName-Topic-IDs-SDS.pdf**

* **Include it in a zip file with the code of the project and project presnetation**
* **Remove the following notes and any red notes**
* **This document is the template document for your Software Design.**
* **For further guidelines and information, READ homework 2 document, and sample SRS.**

# Team

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Email** |
| 20230553 | Loai Hataba | [20230553@stud.fci-cu.edu.eg](mailto:20230553@stud.fci-cu.edu.eg) |
| 20230231 | Abdullah Mohamed | [20230231@stud.fci-cu.edu.eg](mailto:20230231@stud.fci-cu.edu.eg) |
| 20230121 | Hossam Abdelaziz | [20230121@stud.fci-cu.edu.eg](mailto:20230121@stud.fci-cu.edu.eg) |

# 

# Document Purpose and Audience

# Purpose

# This document describes the design, structure, and 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.

# 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.

**Architecture Diagram**

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

A screenshot of a computer

AI-generated content may be incorrect.

## 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

* **List down your classes and describe them**
* **A class with very small responsibility should be deleted and one with very big responsibility should be divided.**

| **Class ID** | **Class Name** | **Description & Responsibility** |
| --- | --- | --- |
| 1. |  |  |
| 2. |  |  |

## 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

* **For the ONE MOST IMPORTANT object, draw a state diagram to show the developer the different states it can be in. (for example it is initially created, then it can be shipped, cancelled (if cancelling is possible), …., etc.)**

## VI. SOLID Principles

* **Explain how you applied THREE OF THE SOLID PRINCIPLES in your design and show the part that the principles where applied in.**

## VII. Design Patterns

* **Use at least THREE DESIGN PATTERNS, any ones from the 23 patterns, not just ones explained in lecture. Explain where you used it and what was the benefit of using it in this place.**

# Tools

* **Write a list of all tools used to develop the design (e.g., PlanetUML, ArgoUML, Visual-Paradigm, etc.)**

# Ownership Report

* **Remove the following notes and any red notes**
* **For every item in this document, write the owners. If someone is owner of something, s/he understands it 100%.**
* **Team leader must verify the table with the team members.**

|  |  |
| --- | --- |
| **Item** | **Owners** |
|  | **Part of class diagram and sequence diagrams 2 and 3.** |
|  |  |
|  |  |