Part D

D1 – System / Software Requirements Specification (SRS)

D1.1 Purpose

The purpose of this SRS is to explain the requirements of our Personal Budget and Expense Management System in a clear and simple way.
In this document, we describe what the system should do, the main features we want to include, and what we aim to achieve in the project.
Writing the SRS helps our team have the same understanding of the system before we start working on the design.

D1.2 Scope

The scope of this system is to provide a simple and clear application that helps users manage their personal budget and track their daily expenses.
The system allows the user to enter income, record different types of expenses, view a summary of their spending, and follow their saving goals.
The focus of the project is on organizing financial information in an easy way that supports better decision-making and helps the user understand their financial situation.

D1.3 System Overview

The system provides a set of basic features that help users manage their personal finances easily.
It allows the user to add income, record expenses with categories, and view a dashboard that shows total income, total expenses, and the remaining budget.
The system also supports setting saving goals and tracking progress over time.
Overall, the system focuses on giving the user a clear and organized view of their financial status.

D1.4 Stakeholders and Users

The main stakeholders of this system are the end users who want to organize their personal budget and monitor their spending.

D1.5 Constraints and Assumptions

The system is developed as a student project, so it is designed as a simple prototype rather than a full commercial application.
It assumes that users will enter their financial information correctly, since the system does not include advanced validation or bank integration.
The system will not connect to real financial services, and all data will be stored and handled within the application only.
The design focuses on clarity and easy use, without requiring complex features or external dependencies.

D1.6 Relationship to Other Documents

This SRS is connected to the other documents in the project.
The ethical guidelines in Part B helped shape the requirements by focusing on user

privacy, accuracy, and fairness.
Part C explains the Agile (Scrum) process, which affects how the requirements are updated and refined during the project.
The design documents—such as UML diagrams and the SDD—will be based on the requirements written in this SRS, and they will show how the system will be built according to these requirements.

D2 – Plan for Requirements Engineering

We gathered all the requirements we needed, whether the need was in financial matters such as expenses, costs, transportation and their management, or the needs were in the method of interviews and what resulted from them, and we explained them clearly and accurately to everyone, before we moved on to the design stage.

D2.1 Requirements Elicitation Methods

Requirements Collection Methods: After deliberating among ourselves on how to collect requirements, the choice was made to use three main techniques because they are more effective and clear, and also conform to the approved requirements engineering.

1. **Stakeholder Interviews**

We selected different and varied groups so that the group requirements would be natural and expressed by each of them without prior restrictions. For example, we chose (students, employees, high-spending people). The goal of the interview was to understand the required features and the difficulties in managing their expenses.

2. **Scenario Development**

After considering how we understand the sequence of tasks, their arrangement, missing requirements, expected errors, and alternatives, we developed a scenario that reflects the daily reality of using the system, such as: adding income, recording expenses, displaying summaries, and tracking savings goals.

3. **Use Case Modeling**

We prepared cases that illustrate the interactions in basic uses between the user and the system, which are:
* Adding income.
* Adding expenses.
* Viewing the control panel.
* Viewing the summary.
* Setting the savings goal.

This helped us a lot in defining the basic goals of the user as well as the system's boundaries and responsibilities.

D2.2 Requirements Analysis and Classification

We analyzed and organized the requirements after collecting them, to avoid duplication and conflict and to ensure clarity.

Functional Requirements

These include basic system capabilities such as income management, expense tracking, dashboard display, and savings goal tracking. We will detail these requirements in section D3.

Non-Functional Requirements

These refer to aspects of system quality such as ease of use, privacy, and performance. We will elaborate on these in section D4.

Constraints and Assumptions

We assumed in the system that the user would input correct data, that it would not be linked to external financial matters, and that it would be characterized by ease and avoid complexity.

D2.3 Requirements Prioritization and Negotiation

After reviewing the requirements, we ranked them from highest to lowest priority based on their importance to the user and their alignment with the project:

* High priority: Expense tracking, income entry, dashboard.
* Medium priority: Summaries, savings goals.
* Low priority: Advanced reports.

D2.4 Requirements Documentation

We documented all requirements in the SRS document mentioned in section D1. This document is the primary reference for the design process, such as UML diagrams, the EER model, and the SDD system design document. We did this to ensure consistency between the analysis and design phases.

D2.5 Requirements Validation

We verified that the written requirements were correct, complete, realistic, and testable, by implementing a set of activities that confirmed this.

1. **Requirements Review**

We have reviewed all the requirements and made sure that they are clear and comprehensive

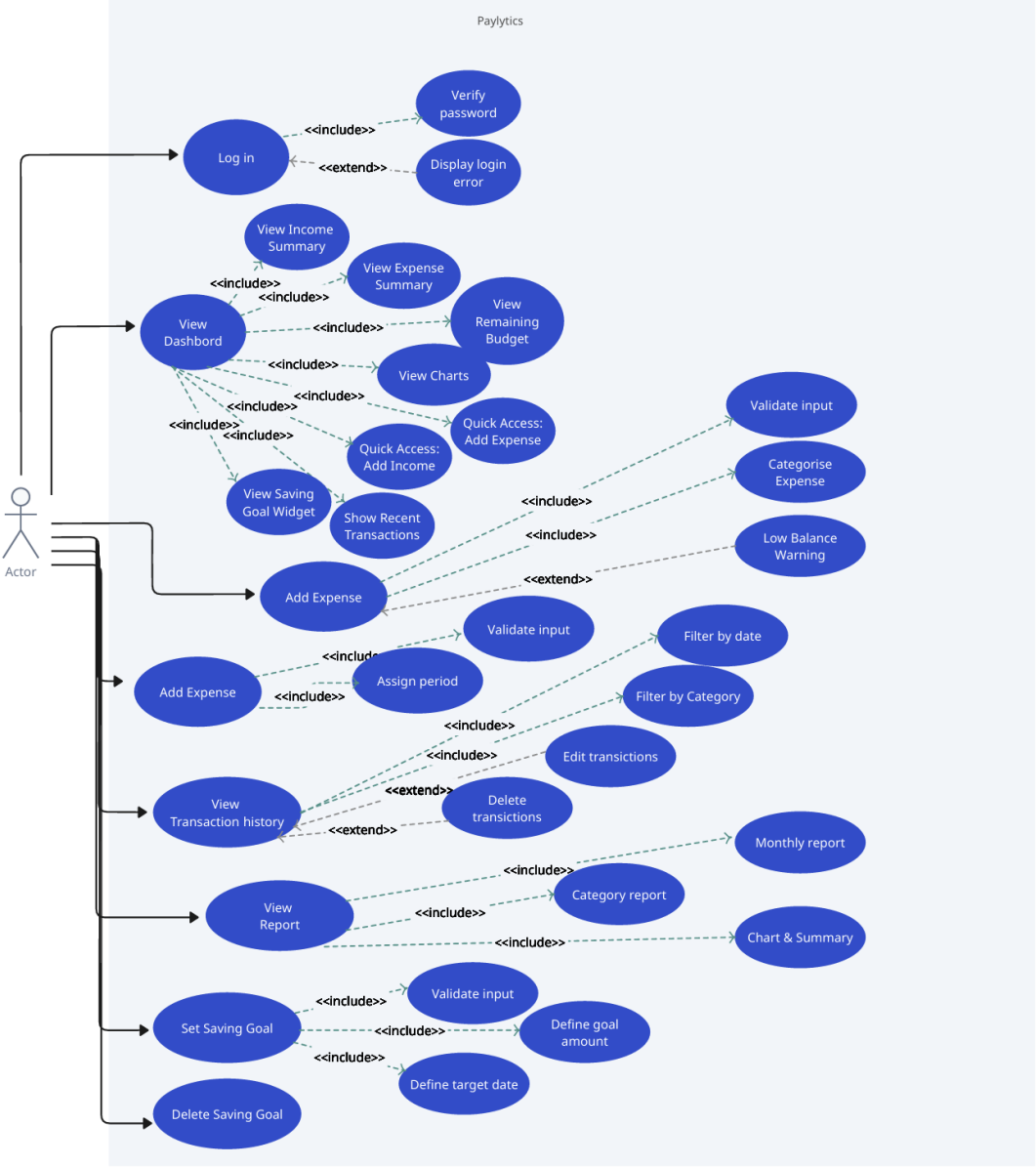
2. **Scenario Evaluation**

We applied the previous scenario to ensure that all steps supported the requirements.

3. **Testability Check**

We verified each requirement by creating a simple test during system implementation. After testing, we modified any requirement that was unclear to the user or not measurable. For this verification, we adopted the methods recommended in the lecture, such as: reviews, building test cases, and using mental prototypes.

D2.6 UML Use Case Diagram



D3 - Functional Requirements (FR)

Functional requirements: explain the actions and tasks that the system should be capable of doing in regard to its users or other systems. They are directly connected with the business purpose of the system:

1. Add Monthly Income: The system should have an option of letting the user key in and record the form of money that he/she earns regularly.
2. Expenses with Classification: It must have a functionality to record separate purchases and classify each purchase by a certain category (such as Groceries or Utilities) to organize them.
3. Display Control Panel (Total Income & Expenses): The primary display should present the summary or dashboard view with the things that have been computed, total income, and the total expenses during a specific period.
4. Show Monthly Expenses Summary: The system should create a report that would contain the expenses grouped by their types within a month.
5. Create a Saving Target: It should enable the user to set a particular financial target that they are intending to reach such as the amount of money they want to save.
6. Keep Records of the Progress to Saving Goal: The system should automatically keep track and show the user the distance between him/her and the target set in requirement 5.
7. Show List of All Recorded Expenses: The system has to show a detailed, frequently searchable or filterable, ledger of all the transactions recorded.
8. Deletion or Modification of Past Expense: It should enable one to edit or delete an expense he/she had recorded in the past to fix errors.

D4 - Non-Functional requirements (NFR)

Non-functional requirements define conditions under which the functioning of a system should be assessed, but not an actual behavior. They place restrictions upon the design and implementation and are commonly referred to as quality attributes or "ilities". They make the system reliable, usable and efficient:

1. The system should be user friendly and need no technical skills (Usability): The interface and navigation should be user friendly, that is, a non technical user should be able to use the software immediately.
2. The system should not compromise the privacy of the user and disclose his data (Security/Privacy): The application should use security provisions to ensure that sensitive financial information is not accessed or disclosed by unauthorized people.
3. Response time must be fast and the time to respond to each operation must not take more than 3 seconds (Performance): All major operations (such as saving an expense or loading a report) are expected to be completed speedily with a defined time limit that can be measured to guarantee an enjoyable experience.

4. The system interface should be straightforward and easy to navigate and display the information without crowding (Aesthetics/Clarity): The visual side should be well-structured and clear and display information in a logical manner without cluttering or overloading the visual memory.

D5:External interface requirements

D5.1- User interfaces:

- The system shall maintain a consistent layout, colors, fonts, and icon styles.
- The interface shall be designed for mobile use and adapt to different smartphone screen sizes.
- The system shall avoid visual clutter and present only essential information .

Login Screen

- The system shall provide a login screen containing fields for email and password, along with a “Login” button.
- The system shall verify the password when login and display an error message if it is wrong.

Main Dashboard

- The system shall provide a main dashboard that includes:

- A summary of total income.
- A summary of total expenses.
- Remaining budget.
- A simple chart showing income and expenses.
- Quick access buttons for adding income and adding expense.
- A widget showing recent transactions.
- A widget showing the savings goal.

Add Expense Screen

- Users shall be able to enter an expense with amount, category, date, and optional notes.
- The system shall provide basic input feedback if required fields are empty or values are negative .
- The system shall warn users if the expense exceeds the remaining budget.
- Users shall select a category for each expense (e.g., food, shopping).

Add Income Screen

- Users shall enter income amount and optional notes.
- The system shall check that the amount is not empty or negative.
- Users can assign the income to a time period (e.g., monthly, weekly).

Transaction History Screen

- Users can view all transactions, with the ability to filter by date or category.
- Users should be able to edit or delete transactions.

Reports Screen

- Users can view reports by month or category.
- Reports shall include charts and summaries.

Savings Goal Screens

- Users can set a saving goal by defining the amount and target date.
- The system shall provide basic input feedback to ensure values are non-empty .
- users shall be able to delete an existing goal saving goal .

D5.2- Hardware interfaces:

As stated in the project constraints and assumptions, the system is designed as a simple student prototype, therefore, the hardware requirements are minimal.

- The system shall operate on mobile devices such as smartphones.
- The system shall support different screen sizes without breaking the layout.
- The system shall require only basic mobile hardware features such as touchscreen interaction and internal storage, No additional hardware capabilities are needed.

D5.3- Software interfaces:

According to the project constraints, the system does not integrate with any external financial services and all user data is stored locally within the application.

- The system shall use local storage to save income, expenses, and budget information.
- The system shall use basic graphical components to display summarise and charts.
- The system shall not require connection to external databases or web services.
- The system shall support operation on standard mobile operating systems (iOS/Android).

contributions	
Nourah alawadh	D1 and D2
Ramah alhamdan	D3 and D4
Reema alrsheed	D5