# Chapter 2: Design

## 1. System Architecture

The Budget Management App follows a Model–View–Controller (MVC) architecture. This separation improves modularity, simplifies maintenance, and allows for efficient scalability.

### Technologies Used

|  |  |
| --- | --- |
| Layer | Description |
| Model | MySQL database accessed through SQL queries for storing users, budgets, and expenses |
| View | EJS templates styled with Bootstrap to create responsive, dynamic web pages |
| Controller | Node.js with Express handles routes, logic, session management, and data flow |

### Data Flow

1. The user interacts with the frontend (EJS + Bootstrap) via the browser.  
2. Requests are handled by Express routes in the controller layer.  
3. The controller accesses the MySQL database through parameterized SQL queries.  
4. Data is passed back to the frontend for rendering.

### Architecture Diagram (Text Representation)

User (Browser)  
 ↓  
Frontend (EJS Templates + Bootstrap)  
 ↓  
Controller (Express.js Routes)  
 ↓  
Model (MySQL via db.js)  
 ↓  
MySQL Database

### Why MVC?

MVC was chosen because it separates responsibilities clearly:  
- Views are decoupled from logic  
- Routes/controllers handle interactions  
- Models ensure secure and structured data handling  
  
This structure reflects good design practices and matches the complexity of the implemented system.

## 2. Use Case Modeling

Below are five representative use cases, directly mapped to features implemented in the system. These exclude Log In, Log Out, and Register as per assessment brief.

### Use Case 1: Add Expense

Actor: User

Description: The user enters an amount, selects a category and date, writes a note, and submits the form.

Main Flow:

* - User opens the Add Expense form.
* - Inputs amount, category, note, and date.
* - Submits the form.
* - The system saves the expense to the database.

Alternate Flow:

* - If validation fails (empty or invalid fields), the user is prompted to correct them.

### Use Case 2: Set Monthly Budget

Actor: User

Description: The user sets or updates their monthly budget.

Main Flow:

* - User navigates to the Set Budget page.
* - Inputs a month and a numeric budget value.
* - Submits the form.
* - The system saves or updates the budget in the database.

Alternate Flow:

* - If the input is not a valid number, the user is shown an error.

### Use Case 3: View Monthly Summary

Actor: User

Description: The user views categorized expenses and total spending against the monthly budget.

Main Flow:

* - User clicks on the Summary page.
* - Applies optional filters (date range, category).
* - System fetches all matched records from the database.
* - System calculates total spent and remaining budget and displays the breakdown.

### Use Case 4: Edit or Delete Expense

Actor: User

Description: Modify or remove an existing expense.

Main Flow:

* - User views list of past expenses.
* - Clicks Edit or Delete on a specific item.
* - Form is shown (Edit) or confirmation is requested (Delete).
* - The system updates or deletes the entry in the database.

### Use Case 5: Export Report

Actor: User

Description: Download a filtered monthly report in CSV or PDF format.

Main Flow:

* - User selects export option on the Summary page.
* - System generates the report.
* - File is downloaded automatically.