# Semester Project Documentation

### Semester Project Title: To-do application

## GROUP MATES

| | Student Name | Student Reg # | Student Degree |
|---|---|---|---|
| Student-1 | HAMNA RASHEED | 2024199 | MGS |
| Student-2 | ATTA UR REHMAN | 2024717 | MGS |
| Student-3 | SOBIA | 2024594 | MGS |
| Student-4 | HUZAIFA ABBASI | 2024233 | MGS |

## Main Features

- **Task Management**:
- Users can add tasks to the list, providing a description for each task.
- Tasks are stored in a structured format (`description` and `status`).
- **View Tasks**:
- Displays all tasks in the list.
- Each task is shown with its index, description, and completion status (`[X]` for completed or `[ ]` for pending).
- **Delete Tasks**:
- Users can delete a specific task by entering its number.
- The list updates dynamically by shifting remaining tasks to fill the gap.
- **Mark Tasks as Completed**:
- Users can mark a specific task as completed by entering its number.
- The completion status of the task is updated to `true`.
- **Menu-Based Interaction**:
- A simple, user-friendly menu guides users through the available options.
- Options include adding, viewing, deleting, marking tasks, and exiting the application.
- **Fixed-Size Task Storage**:
- Supports up to a fixed number of tasks (`MAX_TASKS`), ensuring predictable memory usage.
- **Input Validation**:
- Ensures valid inputs for operations such as selecting tasks to delete or mark as completed.
- Displays appropriate error messages for invalid choices.
- **Persistent Loop**:
- The application runs continuously in a loop until the user explicitly chooses to exit.
- **Beginner-Friendly Design**:
- Uses fundamental C++ concepts like arrays, functions, structures, and loops.
- Code is modular and easy to follow, making it suitable for educational purposes.

## Types of Users

- **Students**
- **Teachers**
- **Professionals**
- **Project Managers**
- **Freelancers**
- **Home Makers**
- **Developers**
- **Event Planners**
- **Content Creators**
- **Business Owners**

## Requirements Breakdown

### *Feature #1: Add Task*

1.1 The user must be able to input a task description.

1.2 Each task must be stored with a completion status set to "not completed" by default.

1.3 The system should prevent adding tasks if the maximum limit is reached.

1.4 A confirmation message should be displayed upon successful addition of a task.

### *Feature #2: View Tasks*

2.1 The system must display all tasks in the list.

2.2 Each task should show its index number, description, and completion status.

2.3 If no tasks are available, the system should display an appropriate message.

### *Feature #3: Delete Task*

3.1 The user must be able to select a task to delete using its index number.

3.2 The system should validate the user input to ensure it corresponds to a valid task.

3.3 Upon deletion, the system should shift remaining tasks to fill the gap.

3.4 A confirmation message should be displayed upon successful deletion.

3.5 If no tasks are available, the system should prevent deletion and display a message.

### *Feature #4: Mark Task as Completed*

4.1 The user must be able to select a task to mark as completed using its index number.

4.2 The system should validate the user input to ensure it corresponds to a valid task.

4.3 The task's completion status should be updated to "completed."

4.4 A confirmation message should be displayed upon successful status update.

4.5 If no tasks are available, the system should prevent marking and display a message.

### *Feature #5: Menu-Based Interaction*

5.1 The system must display a menu with all available options (add, view, delete, mark completed, exit).

5.2 The user must be able to input their choice from the menu.

5.3 The system should validate the user's input and display an error message for invalid choices.

5.4 The application must loop continuously until the user selects the exit option.

# Features to Coding Matrix

| Sr No. | Feature Name | Concept Used | Functions Created | Variables/Obj Created | Line of Code Written |
|---|---|---|---|---|---|
| 1 | Add Task | Input Handling, Arrays | `addTask()` | `tasks[]`, `count` | 13 |
| 2 | View Tasks | Loops, Conditional Logic | `viewTasks()` | `tasks[]`, `count` | 14 |
| 3 | Delete Task | Loops, Array Shifting | `deleteTask()` | `tasks[]`, `count`, `index` | 21 |
| 4 | Mark Task as Completed | Loops, Conditional Logic | `markTaskAsCompleted()` | `tasks[]`, `count`, `index` | 18 |
| 5 | Menu-Based Interaction | Loops, Switch Statement | `displayMenu()` (Menu Logic) | `choice`, `taskCount` | 25 |
| 6 | Exit Application | Conditional Logic | Controlled by `main()` | `choice` | 6 |