StudyBuddy- Personal Study Planner

Name: Zainab Hasan

Registration Number : 25MIB10050

Subject: Introduction to Problem Solving and Programming Sem-1

Program: Integrated M.Tech in Artificial Intelligence and Bioinformatics

University: VIT Bhopal University

1. INTRODUCTION

StudyBuddy is a simple, functional, and efficient study-planning tool built using Python.
The project demonstrates how programming fundamentals like modular design, file handling, data structures, and algorithms can be used to solve a real-world problem faced by students: organizing academic tasks effectively.

The system maintains a structured study task list, enabling users to add tasks, view them, update their status, sort and search through them, and delete tasks. Data is stored persistently using file handling concepts learned in class.

2. PROBLEM STATEMENT

Students often struggle with managing assignments, deadlines, and subject-wise study schedules.
Without a centralized system, tasks are forgotten or completed late, reducing productivity and academic performance.

There is a need for a simple tool that organizes study tasks, tracks progress, and helps students stay consistent.

3. FUNCTIONAL REQUIREMENTS

The system includes seven major functional modules:

1. Add Task – Create new tasks with name, subject, and deadline.

2. View Tasks – Display all tasks neatly.

3. Search by Subject – View tasks related to a particular subject.

4. Sort by Deadline – Arrange tasks from nearest to farthest due date.

5. Mark as Completed – Update the status of selected tasks.

6. Check Completion Status - Check the number of completed versus pending tasks.

7. Delete Tasks – Remove tasks permanently.

8. Save and Exit - Exit the main menu

Each module has a clear input/output structure and follows a logical menu-driven workflow.

4. NON-FUNCTIONAL REQUIREMENTS

1. Usability – Easy menu-driven interface; clear instructions.

2. Performance – Fast file read/write operations using text files.

3. Maintainability – Fully modular code with separate functions.

4. Reliability – Validations to prevent crashes during user input.

5. Error Handling – Graceful handling of invalid choices, missing files, and empty task lists.

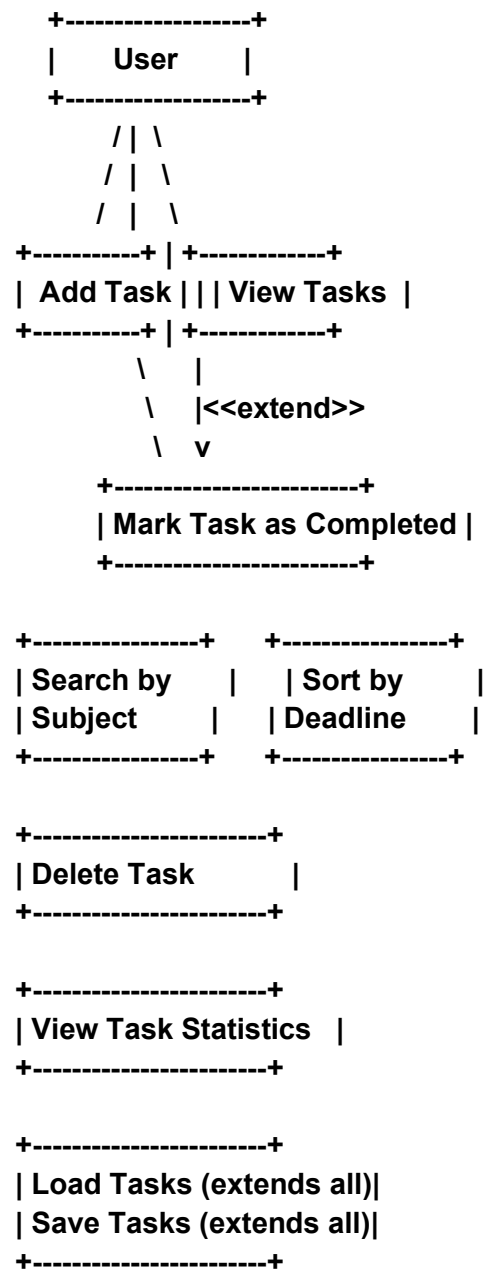
5. . SYSTEM ARCHITECTURE

A simple three-layer architecture is used:

User Interface (Menu)
↓
Application Logic (Task functions)
↓
Data Layer (Text file storage)

6. DESIGN DIAGRAMS

6.1 USE CASE DIAGRAM



Actors:

- User (student using the planner)

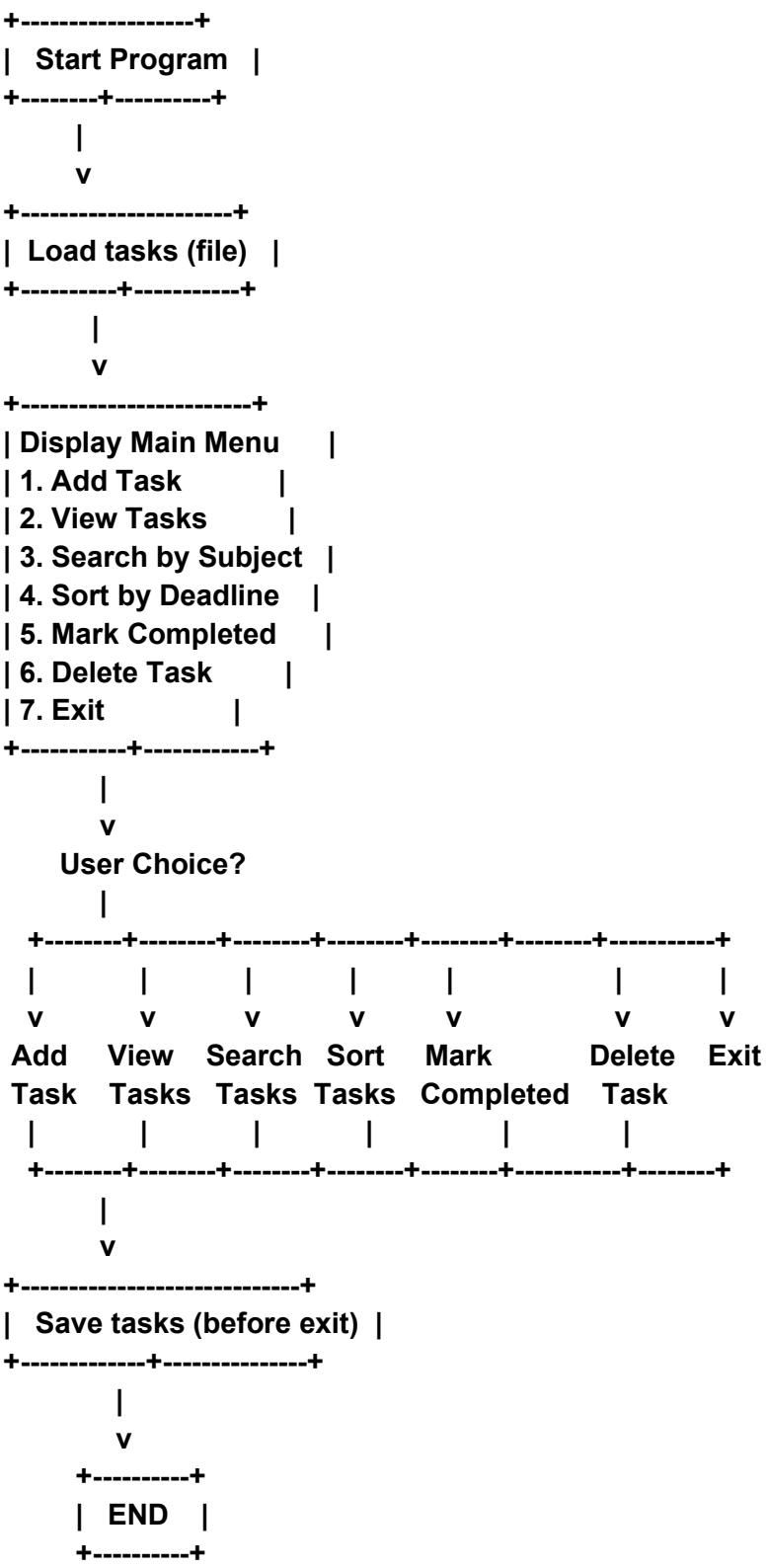
Use Cases:

1. Add Task
2. View Tasks
3. Search Tasks by Subject
4. Sort Tasks by Deadline
5. Mark Task as Completed
6. Delete Task
7. Load Tasks
8. Save Tasks
9. View Task Statistics (Completed vs Pending)

Relationships:

- User → interacts with all use cases
- “Load Tasks” and “Save Tasks” extend all CRUD operations
- “Mark Completed” extends “View Tasks” (requires selection)
- “Delete Task” extends “View Tasks”

6.2 WORKFLOW DIAGRAM



6.3 Sequence Diagram (Example: Add Task)

User → Program: Open Application
Program → File System: Load tasks

User → Program: Select "Add Task"
Program → User: Prompt for input
User → Program: Enter name, subject, deadline

Program → Task Manager: Create new task object
Task Manager → Program: Task added to list

Program → File System: Save updated task list
Program → User: Show "Task added successfully"

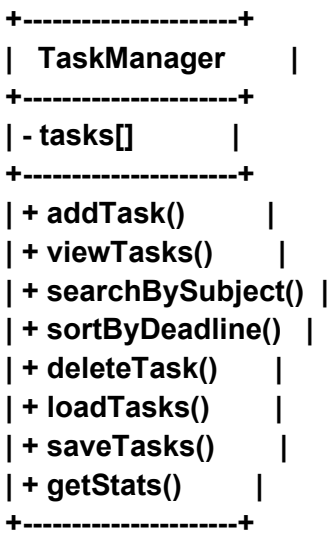
User → Menu → add_task() → write to file → confirmation message

6.4 Component Diagram

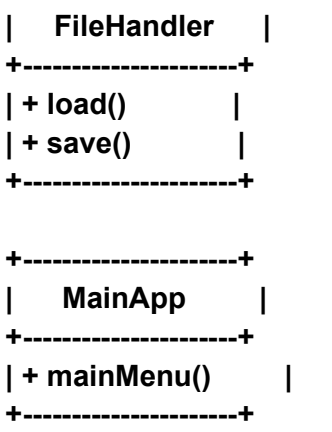
Component 1: study_planner.py

```
+--------------------+
|      Task          |
+--------------------+
| - name             |
| - subject          |
| - deadline         |
| - status           |
+--------------------+
| + markCompleted()  |
| + toString()       |
+--------------------+
```

Component 2: study_tasks.txt

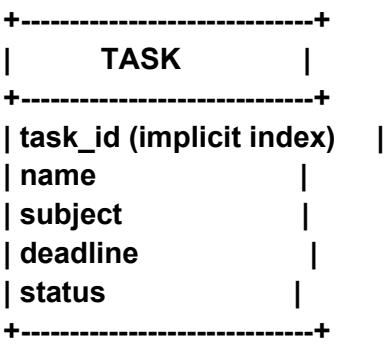


Component 3: input/output console



6.5 Storage Design (ER Diagram equivalent)

Since file storage is used:
Task File Structure:
name | subject | deadline | status



7. DESIGN DECISIONS & RATIONALE

- Text file storage was chosen because SQL is not yet part of the syllabus.
- Separate functions maintain readability and modularity.
- String-based data format ensures compatibility with simple file operations.
- Menu-driven design aligns with beginner-level user interaction requirements.

8. IMPLEMENTATION DETAILS

Language: Python

Concepts Used:

- File handling (open, read/write)
- Dictionaries & lists
- Loops and conditionals
- Modular functions
- Input validation
- Sorting (using Python's sorted())

9. SCREENSHOTS / RESULTS

```
        choice = input("Enter your choice: ")

        if choice == "1":
            add_task(tasks)
        elif choice == "2":
            view_tasks(tasks)
        elif choice == "3":
            search_by_subject(tasks)
        elif choice == "4":
            sort_by_deadline(tasks)
        elif choice == "5":
            mark_completed(tasks)
        elif choice == "6":
            task_summary(tasks)
        elif choice == "7":
            delete_task(tasks)
        elif choice == "8":
            save_tasks(tasks)
            print("Goodbye! All tasks saved.")
            break
        else:
            print("Invalid choice. Please try again.")

main()
```

```
# Menu
def main():
    tasks = load_tasks()

    while True:
        print("============================")
        print("     STUDY PLANNER     ")
        print("============================")
        print("1. Add Task")
        print("2. View Tasks")
        print("3. Search Tasks by Subject")
        print("4. Sort Tasks by Deadline")
        print("5. Mark Task Completed")
        print("6. Count Completed vs Pending Tasks")
        print("7. Delete Task")
        print("8. Exit")
```

```
============================
     STUDY PLANNER
============================
1. Add Task
2. View Tasks
3. Search Tasks by Subject
4. Sort Tasks by Deadline
5. Mark Task Completed
6. Count Completed vs Pending Tasks
7. Delete Task
8. Exit
Enter your choice:
```

```
============================
     STUDY PLANNER
============================
1. Add Task
2. View Tasks
3. Search Tasks by Subject
4. Sort Tasks by Deadline
5. Mark Task Completed
6. Count Completed vs Pending Tasks
7. Delete Task
8. Exit
Enter your choice: 1

--- Add New Study Task ---
Task Name: Lab Report
Subject: Chemistry
Deadline (DD/MM/YYYY): 25/11/2025
Task added successfully!
============================
```

```
==========================
     STUDY PLANNER
==========================
1. Add Task
2. View Tasks
3. Search Tasks by Subject
4. Sort Tasks by Deadline
5. Mark Task Completed
6. Count Completed vs Pending Tasks
7. Delete Task
8. Exit
Enter your choice: 2

--- Your Study Tasks ---
1. Lab Report | Chemistry | 25/11/2025 | Pending
2. Project | Programming | 25/11/2025 | Pending
3. Presentation | Chemistry | 26/11/2025 | Pending
```

```
==========================
     STUDY PLANNER
==========================
1. Add Task
2. View Tasks
3. Search Tasks by Subject
4. Sort Tasks by Deadline
5. Mark Task Completed
6. Count Completed vs Pending Tasks
7. Delete Task
8. Exit
Enter your choice: 3
Enter subject to search: Chemistry

--- Search Results ---
Lab Report | 25/11/2025 | Pending
Presentation | 26/11/2025 | Pending
==========================
```

```
==========================
     STUDY PLANNER
==========================
1. Add Task
2. View Tasks
3. Search Tasks by Subject
4. Sort Tasks by Deadline
5. Mark Task Completed
6. Count Completed vs Pending Tasks
7. Delete Task
8. Exit
Enter your choice: 4

--- Tasks Sorted by Deadline ---
Lab Report | Chemistry | 25/11/2025 | Pending
Project | Programming | 25/11/2025 | Pending
Presentation | Chemistry | 26/11/2025 | Pending
==========================
```

```
==========================
     STUDY PLANNER
==========================
1. Add Task
2. View Tasks
3. Search Tasks by Subject
4. Sort Tasks by Deadline
5. Mark Task Completed
6. Count Completed vs Pending Tasks
7. Delete Task
8. Exit
Enter your choice: 5

--- Your Study Tasks ---
1. Lab Report | Chemistry | 25/11/2025 | Pending
2. Project | Programming | 25/11/2025 | Pending
3. Presentation | Chemistry | 26/11/2025 | Pending

Enter task number to mark as completed: 3
Congratulations! Task marked as completed! (^¬^)
==========================
```

```
============================
      STUDY PLANNER
============================
1. Add Task
2. View Tasks
3. Search Tasks by Subject
4. Sort Tasks by Deadline
5. Mark Task Completed
6. Count Completed vs Pending Tasks
7. Delete Task
8. Exit
Enter your choice: 6

Completed Tasks: 1
Pending Tasks: 2
============================
```

```
============================
     STUDY PLANNER
============================
1. Add Task
2. View Tasks
3. Search Tasks by Subject
4. Sort Tasks by Deadline
5. Mark Task Completed
6. Count Completed vs Pending Tasks
7. Delete Task
8. Exit
Enter your choice: 7

--- Your Study Tasks ---
1. Lab Report | Chemistry | 25/11/2025 | Pending
2. Project | Programming | 25/11/2025 | Pending
3. Presentation | Chemistry | 26/11/2025 | Completed

Enter task number to delete: 3
Task deleted!
============================
```

```
============================
     STUDY PLANNER
============================
1. Add Task
2. View Tasks
3. Search Tasks by Subject
4. Sort Tasks by Deadline
5. Mark Task Completed
6. Count Completed vs Pending Tasks
7. Delete Task
8. Exit
Enter your choice: 2

--- Your Study Tasks ---
1. Lab Report | Chemistry | 25/11/2025 | Pending
2. Project | Programming | 25/11/2025 | Pending
============================
```

```
============================
      STUDY PLANNER
============================
1. Add Task
2. View Tasks
3. Search Tasks by Subject
4. Sort Tasks by Deadline
5. Mark Task Completed
6. Count Completed vs Pending Tasks
7. Delete Task
8. Exit
Enter your choice: 8
Goodbye! All tasks saved.
```

10. TESTING APPROACH

- Manual testing for each function
- Boundary testing for task numbers
- Validation for empty lists and missing files
- Incorrect input handling (non-numeric choices)

11. CHALLENGES FACED

- Implementing persistent storage without SQL
- Designing proper validation to avoid program crashes
- Ensuring file structure remained consistent after multiple edits

12. LEARNINGS & KEY TAKEAWAYS

- Practical understanding of file handling
- Importance of modular code
- How basic data structures form the backbone of larger systems
- Mapping real-world needs software solutions
- Designing simple architecture for scalable systems

13. FUTURE ENHANCEMENTS

- Add calendar integration
- Add notifications or reminders
- GUI using Tkinter
- Switch storage to JSON or SQLite
- Add priority levels
- Export tasks to PDF or CSV

14. REFERENCES

- Python Documentation
- Class Notes (Introduction to Problem Solving & Programming)
- VITyarthi Python Essentials Course Material