BALLARI INSTITUTE OF
TECHNOLOGY AND MANAGEMENT

Project Scheduling Assistant POC

->MOHAMMED IRFAN
->MOHAMMED MUSADDIQ.K
->SIDDARTH SHASTRI

MENTOR : Mrs.NAGA PRATYUSHA
GUIDED BY : Mr.KADAVATI MANOHAR

CRUD: Schedule Data

1 **Create**

Adding new schedules, milestones, and tasks with their associated details.

2 **Read**

Retrieving existing schedule information for analysis and planning.

3 **Update**

Modifying existing schedule entries with changes in deadlines, tasks, or dependencies.

4 **Delete**

Removing outdated or unnecessary schedule entries.

Create Project Timelines

1. **Define Milestones**

   Key events or deliverables within a project, with their corresponding due dates.

2. **Task Dependencies**

   Relationships between tasks, where the completion of one task depends on another.

3. **Resource Allocation**

   Assign personnel or resources to specific tasks and milestones.

4. **Duration Estimates**

   Assess the time required to complete each task and milestone.

Adjust Schedules

Progress Tracking

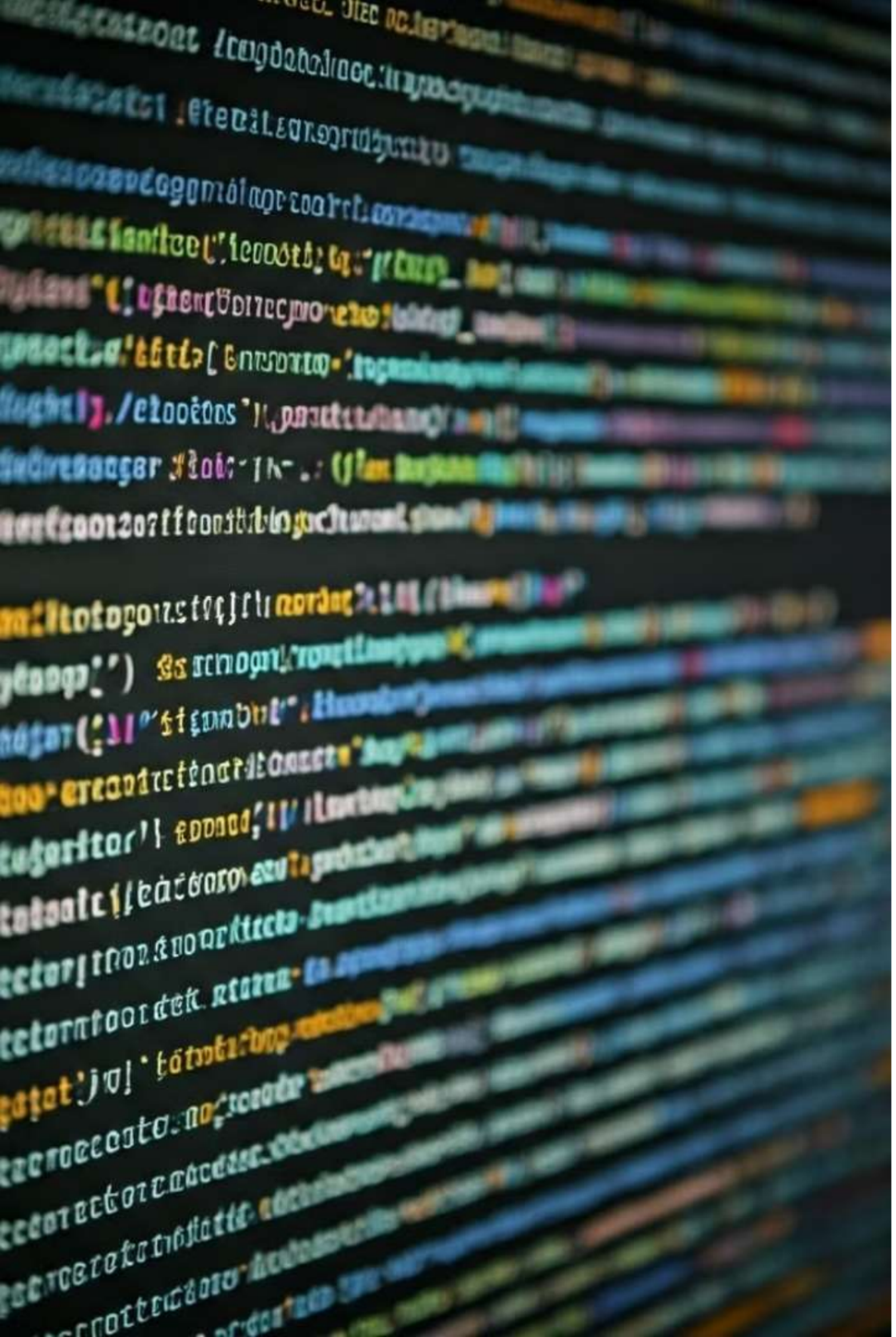
Monitor actual progress against planned schedules.

Dependency Updates

Adjust schedules based on changes in task dependencies.

Resource Changes

Modify schedules based on changes in available resources or personnel.



Object-Oriented Python Solution

Project Class

Encapsulates project information, including timelines, tasks, and dependencies.

Timeline Class

Represents a project timeline, with a collection of milestones and tasks.

Task Class

Represents an individual task, including duration, dependencies, and resource allocation.

Milestone Class

Represents a milestone, with a deadline and a set of associated tasks.

Manage Project Schedules

1

Schedule Creation

Users can create new project schedules, defining milestones, tasks, and dependencies.

2

Schedule Viewing

Users can view existing schedules, displaying timelines, task lists, and progress reports.

3

Schedule Updates

Users can update schedules by adjusting task durations, deadlines, or dependencies.

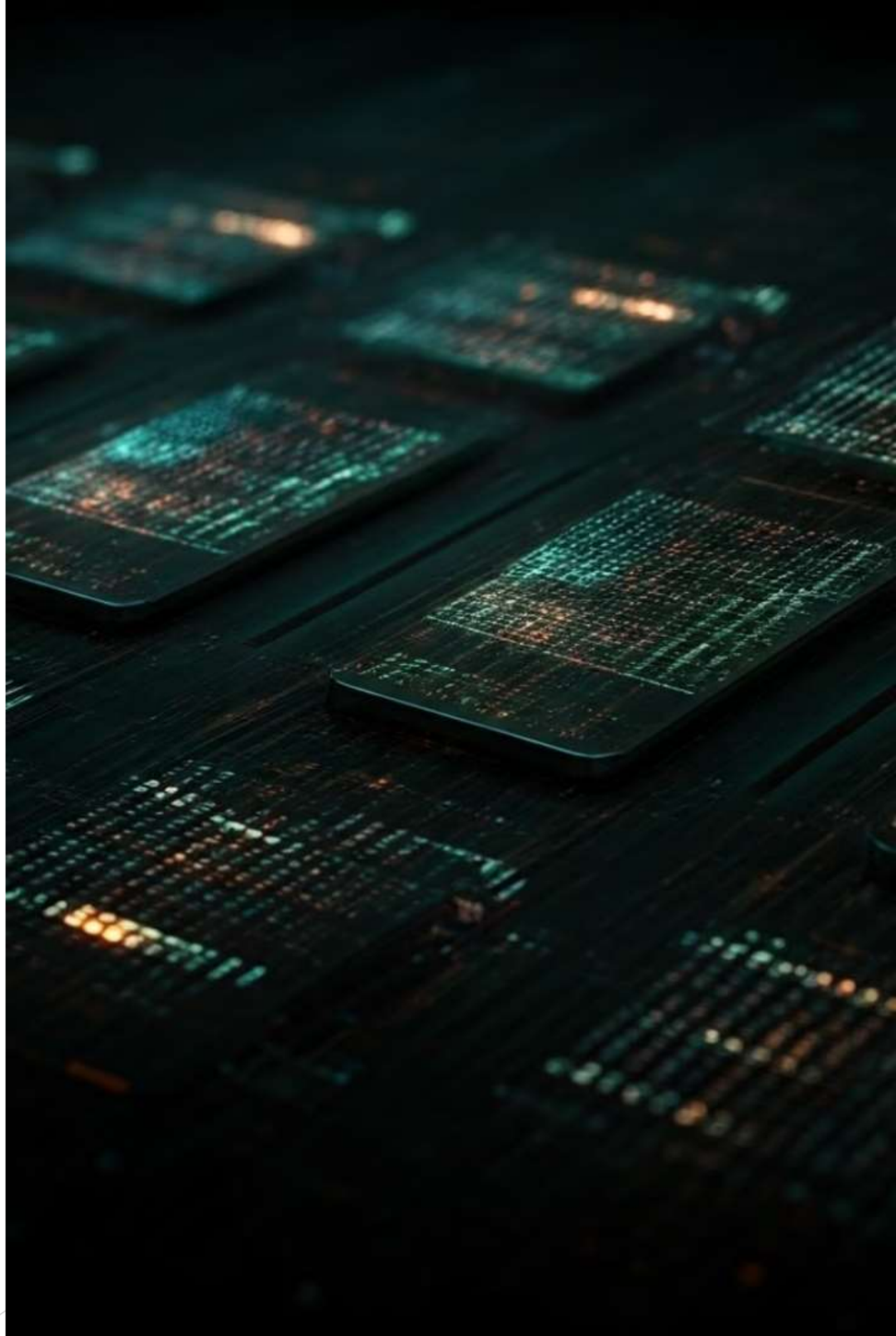
4

Schedule Reporting

The system generates reports summarizing progress, highlighting critical paths, and identifying potential issues.

Create, Read, Update, Delete

CRUD Operation	Description
Create	Add new schedule entries (projects, timelines, tasks, milestones)
Read	Retrieve schedule information for viewing or analysis
Update	Modify existing schedule entries based on project progress or changes
Delete	Remove outdated or unnecessary schedule entries



Address Problem Statement

The Project Scheduling Assistant POC addresses the need for a tool to effectively manage project schedules. By providing capabilities for creating, updating, and deleting schedules, the POC enables users to track progress, identify critical paths, and adapt to changing project dynamics.

Algorithm

- Start

Initialize Project Scheduler
User selects action:

- Create Timeline

Input: Timeline ID
Output: New ProjectTimeline created

- Add Schedule

Input: Schedule Details
Output: Schedule added to ProjectTimeline

- Adjust Schedule

Input: Schedule ID, New Dates
Output: Schedule adjusted based on dependencies

- View Schedule

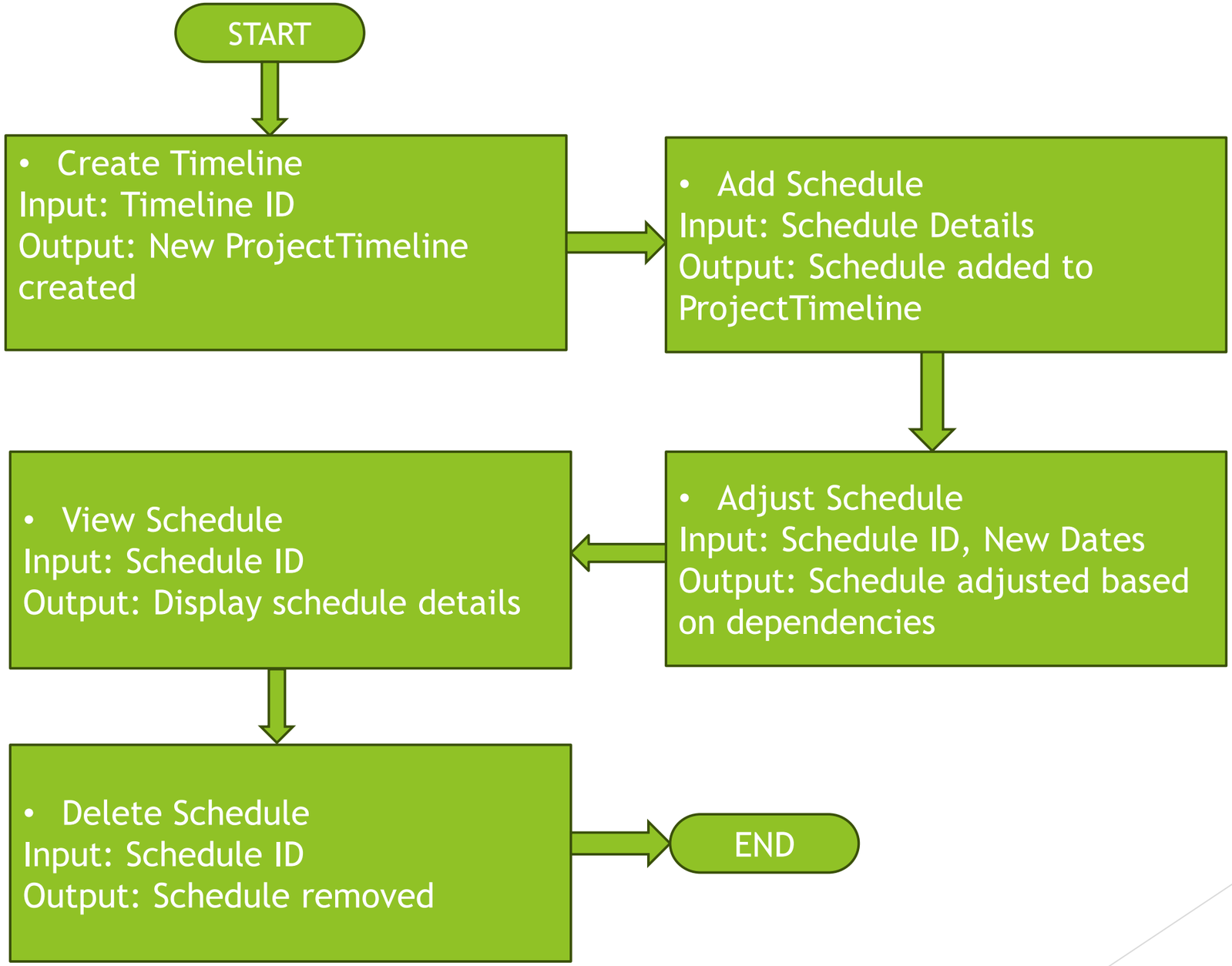
Input: Schedule ID
Output: Display schedule details

- Delete Schedule

Input: Schedule ID
Output: Schedule removed

- Repeat or End

FLOWCHART:


START
• Create Timeline
Input: Timeline ID
Output: New ProjectTimeline created
• Add Schedule
Input: Schedule Details
Output: Schedule added to ProjectTimeline
• Adjust Schedule
Input: Schedule ID, New Dates
Output: Schedule adjusted based on dependencies
• View Schedule
Input: Schedule ID
Output: Display schedule details
• Delete Schedule
Input: Schedule ID
Output: Schedule removed
END

```
class Schedule:
    def _init_(self, schedule_id, project_name, start_date, end_date):
        self.schedule_id = schedule_id
        self.project_name = project_name
        self.start_date = start_date
        self.end_date = end_date

    def _repr_(self):
        return f"Schedule(id={self.schedule_id}, project={self.project_name}, start={self.start_date},
        end={self.end_date})“

class ProjectTimeline:
    def _init_(self, timeline_id):
        self.timeline_id = timeline_id
        self.schedules = {}

    def add_schedule(self, schedule):
        if schedule.schedule_id in self.schedules:
            raise ValueError(f"Schedule ID {schedule.schedule_id} already exists.")
        self.schedules[schedule.schedule_id] = schedule

def get_schedule(self, schedule_id):
    return self.schedules.get(schedule_id, None)
def update_schedule(self, schedule_id, start_date=None, end_date=None):
    schedule = self.get_schedule(schedule_id)
```

```
    if not schedule:
        raise ValueError("Schedule not found.")
    if start_date:
        schedule.start_date = start_date
    if end_date:
        schedule.end_date = end_date
def delete_schedule(self, schedule_id):
    if schedule_id not in self.schedules:
        raise ValueError("Schedule not found.")
    del self.schedules[schedule_id]

def _repr_(self):
    return f"ProjectTimeline(id={self.timeline_id}, schedules={self.schedules})“

class  ProjectScheduler:
    def _init_(self):
        self.timelines = {}
```

```
def create_timeline(self, timeline_id):
    if timeline_id in self.timelines:
        raise ValueError(f"Timeline ID {timeline_id} already exists.")
    self.timelines[timeline_id] = ProjectTimeline(timeline_id)

def get_timeline(self, timeline_id):
    return self.timelines.get(timeline_id, None)

def adjust_schedule(self, timeline_id, schedule_id, start_date=None,
end_date=None):
    timeline = self.get_timeline(timeline_id)
    if not timeline:
        raise ValueError("Timeline not found.")
    timeline.update_schedule(schedule_id, start_date, end_date)

def _repr_(self):
    return f"ProjectScheduler(timelines={self.timelines})"
```

```
# Example usage
if __name__ == "__main__":
    scheduler = ProjectScheduler()

    # Create a project timeline
    scheduler.create_timeline("timeline_1")

    # Create a schedule and add it to the timeline
    schedule1 = Schedule(schedule_id="schedule_1", project_name="Project A",
    start_date="2024-01-01", end_date="2024-01-31")
    scheduler.get_timeline("timeline_1").add_schedule(schedule1)

    # Retrieve and print the timeline
    print(scheduler.get_timeline("timeline_1"))

    # Adjust the schedule
scheduler.adjust_schedule("timeline_1", "schedule_1", end_date="2024-02-15")
```

```
# Retrieve and print the updated timeline
print(scheduler.get_timeline("timeline_1"))

# Delete the schedule
scheduler.get_timeline("timeline_1").delete_schedule("schedule_1")

# Print the timeline after deletion
print(scheduler.get_timeline("timeline_1"))
```

OUTPUT:

ProjectScheduler(Timelines: [ProjectTimeline(ID: Timeline_1, Schedules: [Schedule(ID: Schedule_1, Name: Task 1, Start: 2024-09-01, End: 2024-09-05), Schedule(ID: Schedule_2, Name: Task 2, Start: 2024-09-06, End: 2024-09-10)])])

Adjusted Schedule: Schedule(ID: Schedule_1, Name: Task 1, Start: 2024-09-01, End: 2024-09-06)

Deleted Schedule: Schedule(ID: Schedule_2, Name: Task 2, Start: 2024-09-06, End: 2024-09-10)

ProjectScheduler(Timelines: [ProjectTimeline(ID: Timeline_1, Schedules: [Schedule(ID: Schedule_1, Name: Task 1, Start: 2024-09-01, End: 2024-09-06)])])

Thank you