
VIT
BHOPAL
TO-DO
NAME : GAURI NANDANA M
REGISTRATION NUMBER : 25BAI10053
COURSE: INTRODUCTION TO PROBLEM SOLVING AND PROGRAMMING

LIST

INTRODUCTION

IT IS A SMALL AND INTERACTIVE PYTHON PROJECT WHICH WILL HELP ALL USERS MANAGE THEIR TASK EFFICIENTLY.IT ALLOWS ADDITION,REMOVAL,VIEWING OF TASKS ALONG WITH SETTING PRIORITIES AND TRACKING ACHIEVEMENTS THROUGH BADGES WHILE PROVIDING MOTIVATIONAL QUOTES THAT ARE GENERATED RANDOMLY. IT SOLVES THE PROBLEM BY HELPING PEOPLE MANAGE TASK EFFICIENTLY THUS HELPING IN TIME MANAGEMENT SKILLS.

Problem Statement:

People often struggle to track daily tasks, leading to missed deadlines and lower productivity. Manual methods like notebooks are inefficient and error-prone. A solution that organizes, prioritizes, and motivates task completion helps manage workloads effectively and improve productivity.

Objectives :

The objective of this project is to provide a simple, interactive system that helps users organize and manage their daily tasks efficiently. It aims to enable users to add, remove, and view tasks, set priorities, and track achievements to stay motivated and improve productivity.

FUNCTIONAL REQUIREMENTS

I . THREE MAJOR FUNCTIONAL MODULES:

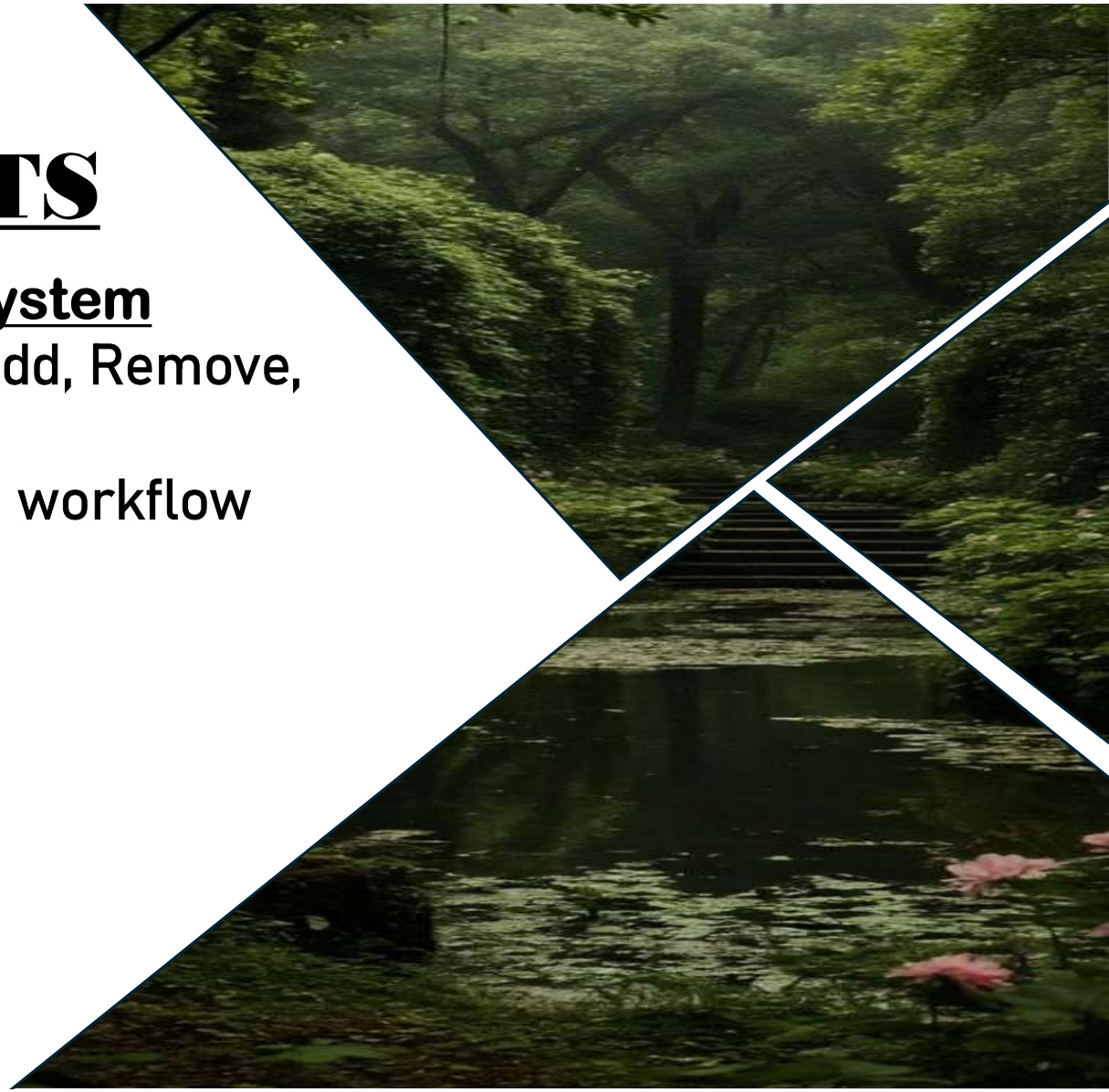
1. **Task Management (CRUD Operations)**

- **Add new tasks**
- **Remove completed tasks**
- **View tasks with priority sorting**

2. **Achievement & Motivation**

- **Track number of completed tasks**
- **Assign achievement badges (Beginner, Rising Prodigy,Master the Task)**
- **Show random motivational quotes**

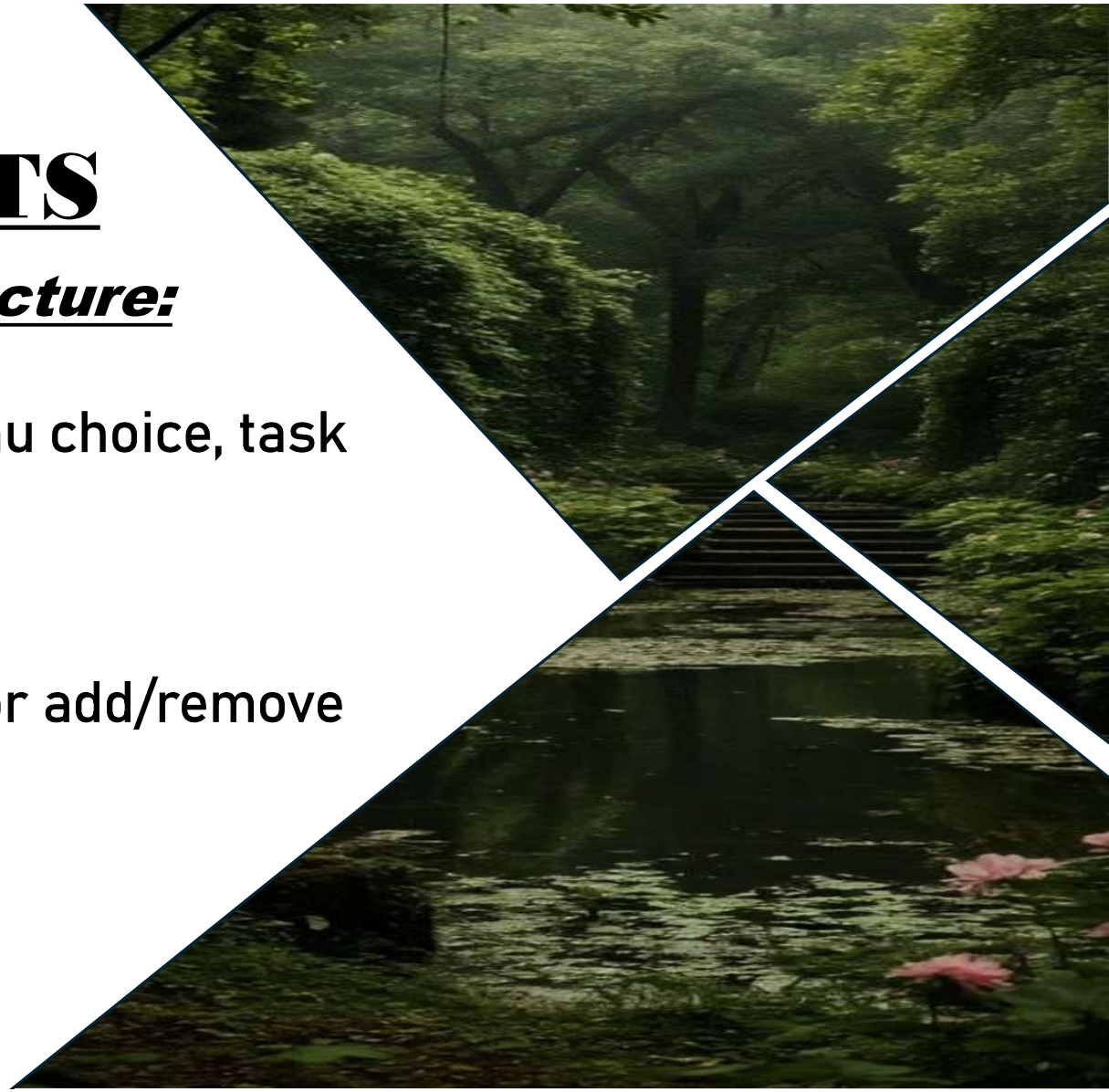
FUNCTIONAL REQUIREMENTS



3. User Interaction & Menu System

- Display menu with options (Add, Remove, View, Achievements, Exit)
- Handle user inputs and guide workflow

FUNCTIONAL REQUIREMENTS



II. Clear Input/Output Structure:

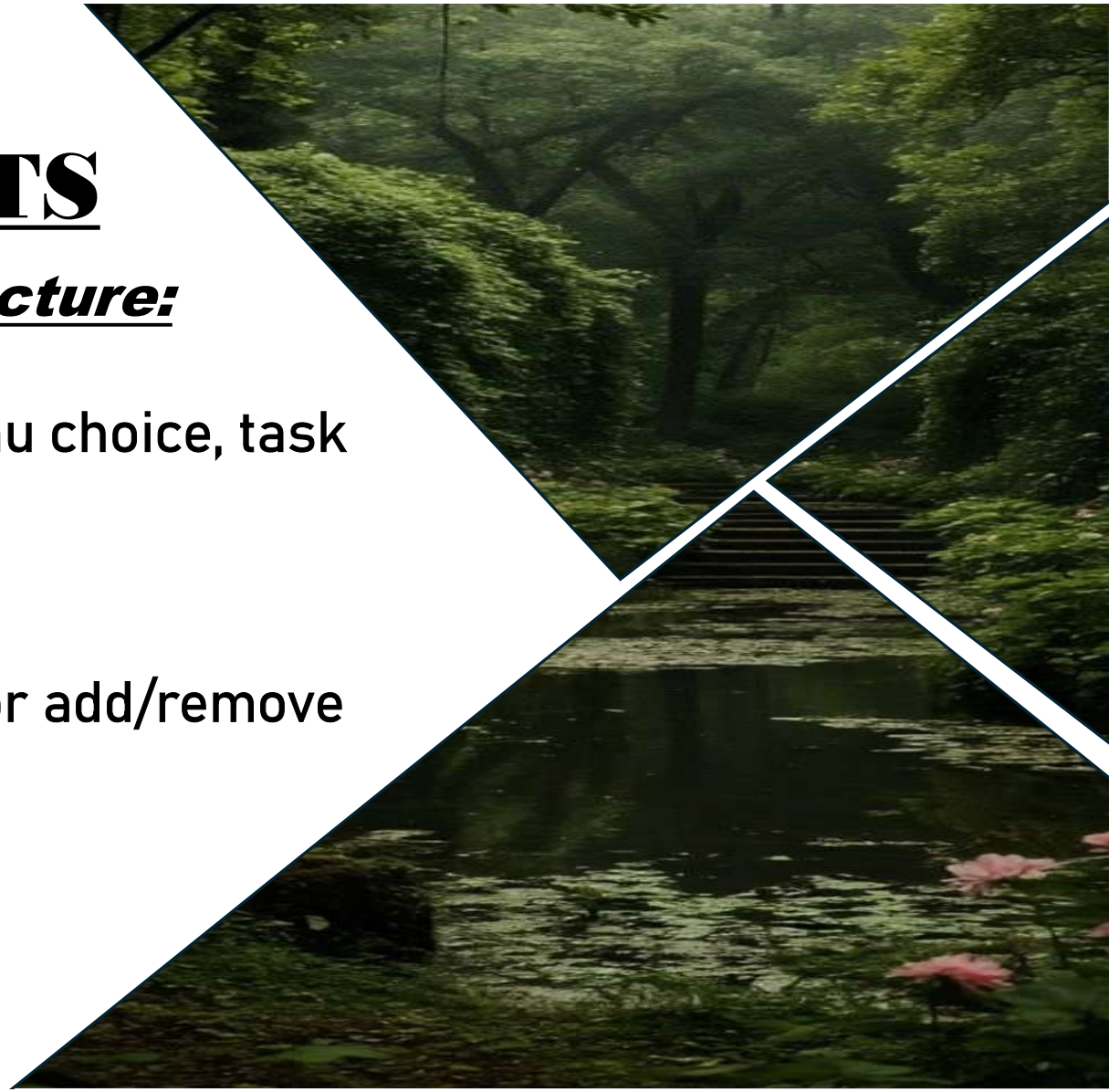
1. Inputs:

- Task name, task priority, menu choice, task number/name to remove

2. Outputs:

- Task lists with priority
- Confirmation messages for add/remove actions
- Motivational quotes
- Achievement badges

FUNCTIONAL REQUIREMENTS



II. Clear Input/Output Structure:

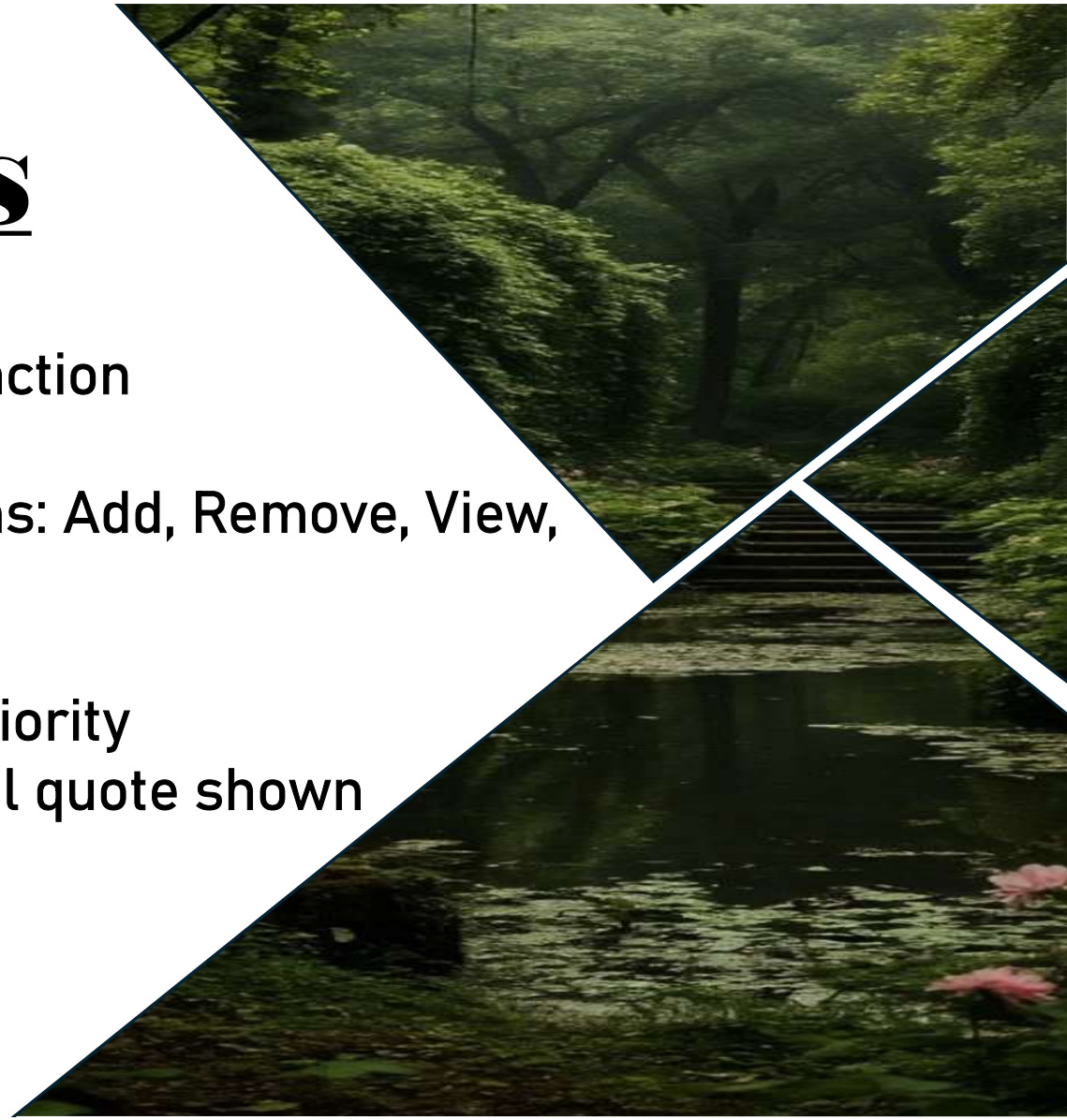
1. Inputs:

- Task name, task priority, menu choice, task number/name to remove

2. Outputs:

- Task lists with priority
- Confirmation messages for add/remove actions
- Motivational quotes
- Achievement badges

FUNCTIONAL REQUIREMENTS

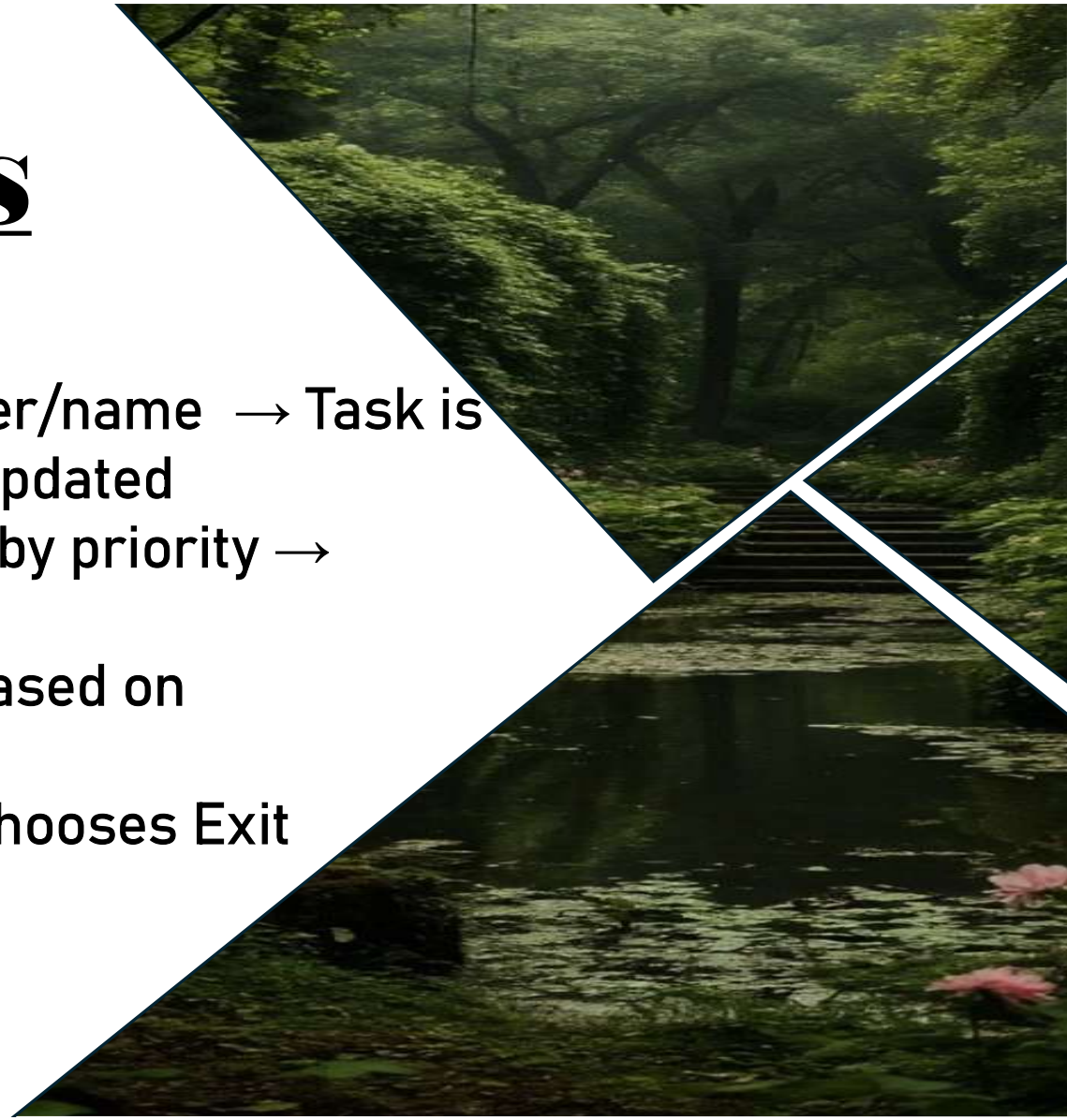


III. LOGICAL WORKFLOW:

1. Logical Workflow of User Interaction

- User opens the app
- 2. Menu is displayed with options: Add, Remove, View, Achievements, Exit
- 3. User selects an option:
- Add Task: Enter task name & priority
  → Task is added → Motivational quote shown

FUNCTIONAL REQUIREMENTS

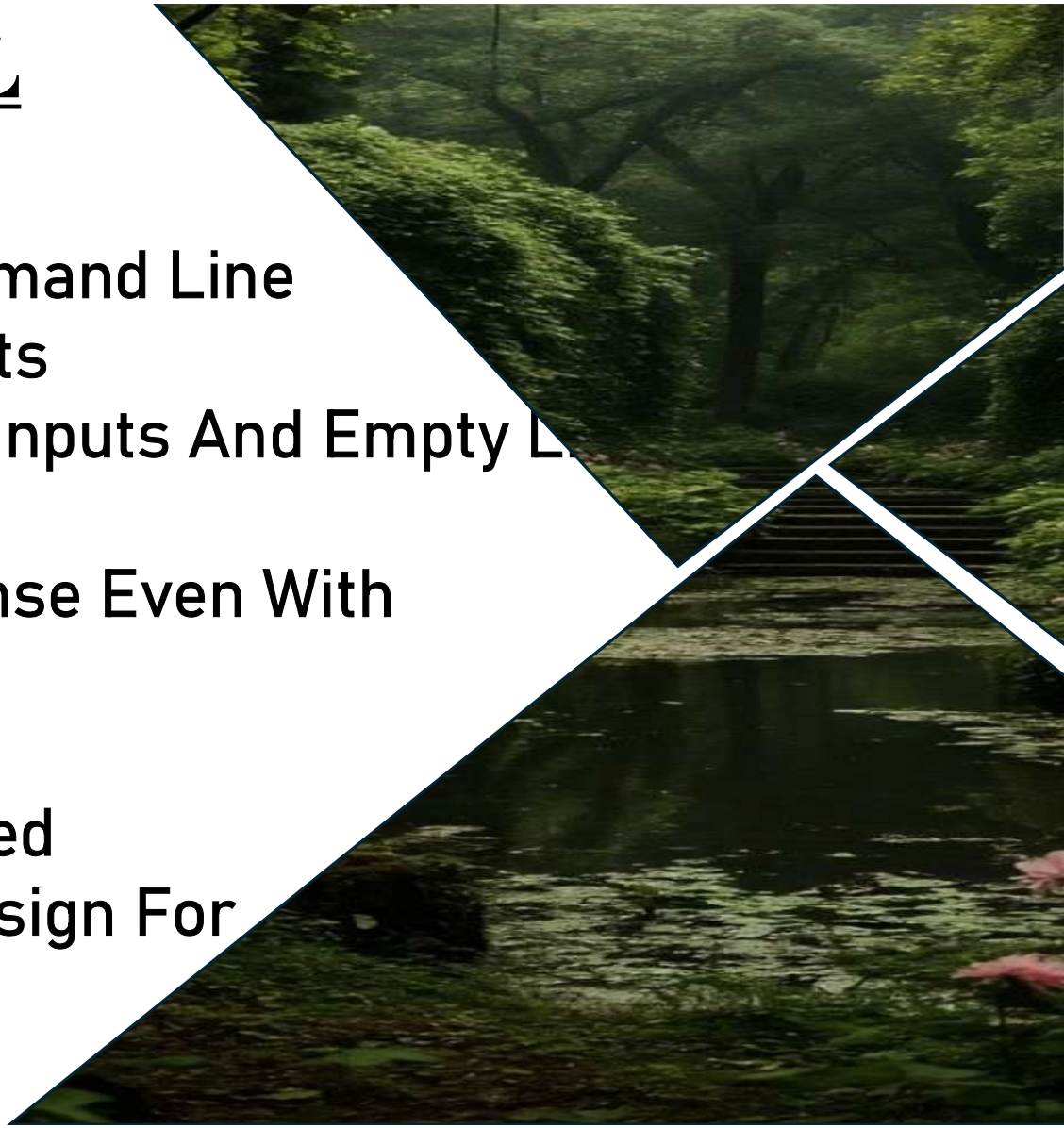


III. LOGICAL WORKFLOW:

- Remove Task: Enter task number/name → Task is removed → Completed count updated
- View Tasks: Show tasks sorted by priority → Suggest next task
- Achievements: Display badge based on completed tasks
- Loop back to menu until user chooses Exit

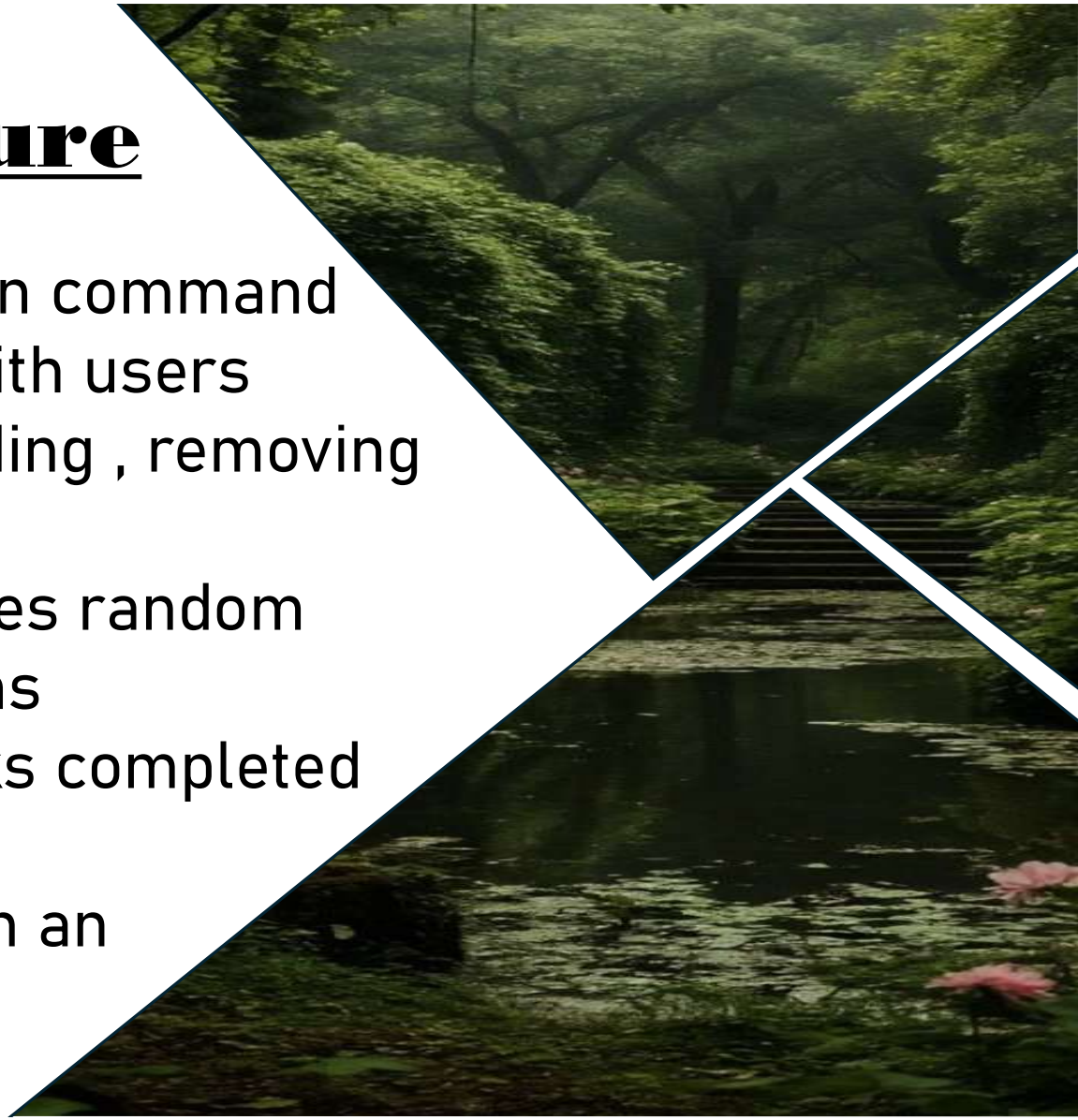
NON-FUNCTIONAL REQUIREMENTS

- Usability : Easy To Use Command Line Interface With Clear Prompts
- Reliability : Handles Invalid Inputs And Empty L… Gracefully
- Performance : Quick Response Even With A Large Number Of Tasks
- Portability : Can Run On Any System With Python Installed
- Maintainability : Modular Design For Easy Updates And Future Enhancements

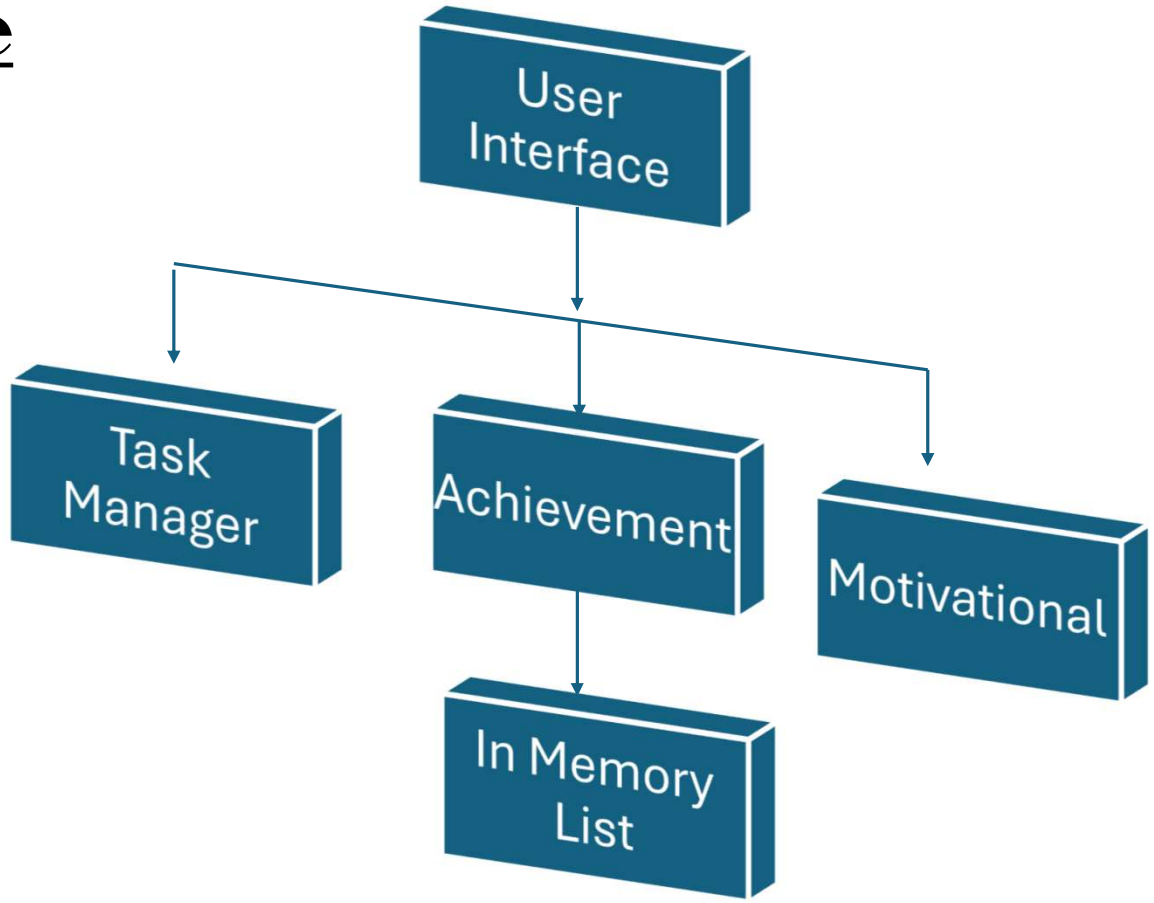


System Architecture

- User interface : Menu driven command line interface to interact with users
- Task manager: Handles adding , removing and viewing tasks
- Motivational engine: Provides random quotes and task suggestions
- Achievement tracker: Tracks completed tasks and award badges
- Data storage: Stores task in an in-memory list

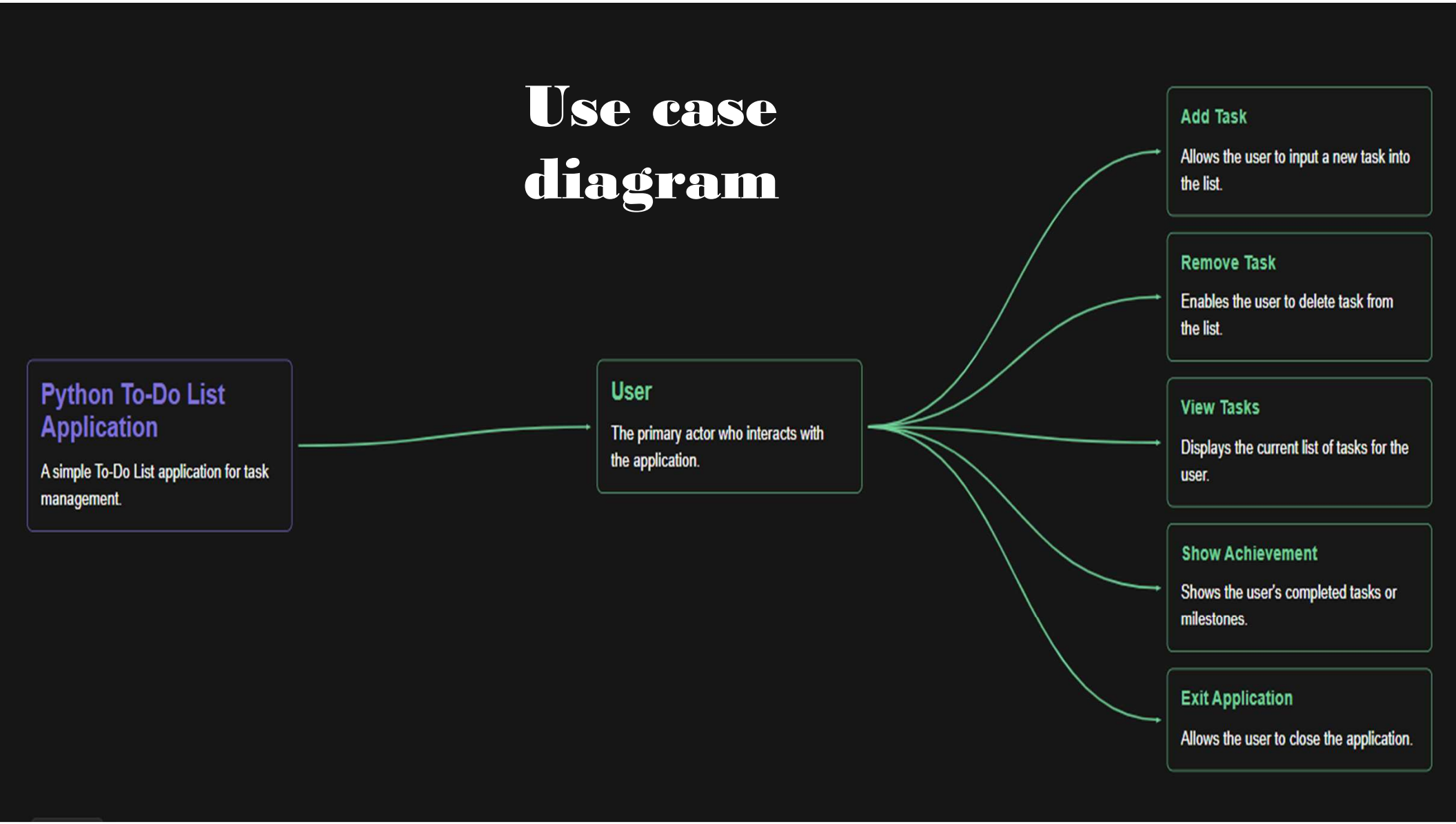


Architecture Diagram

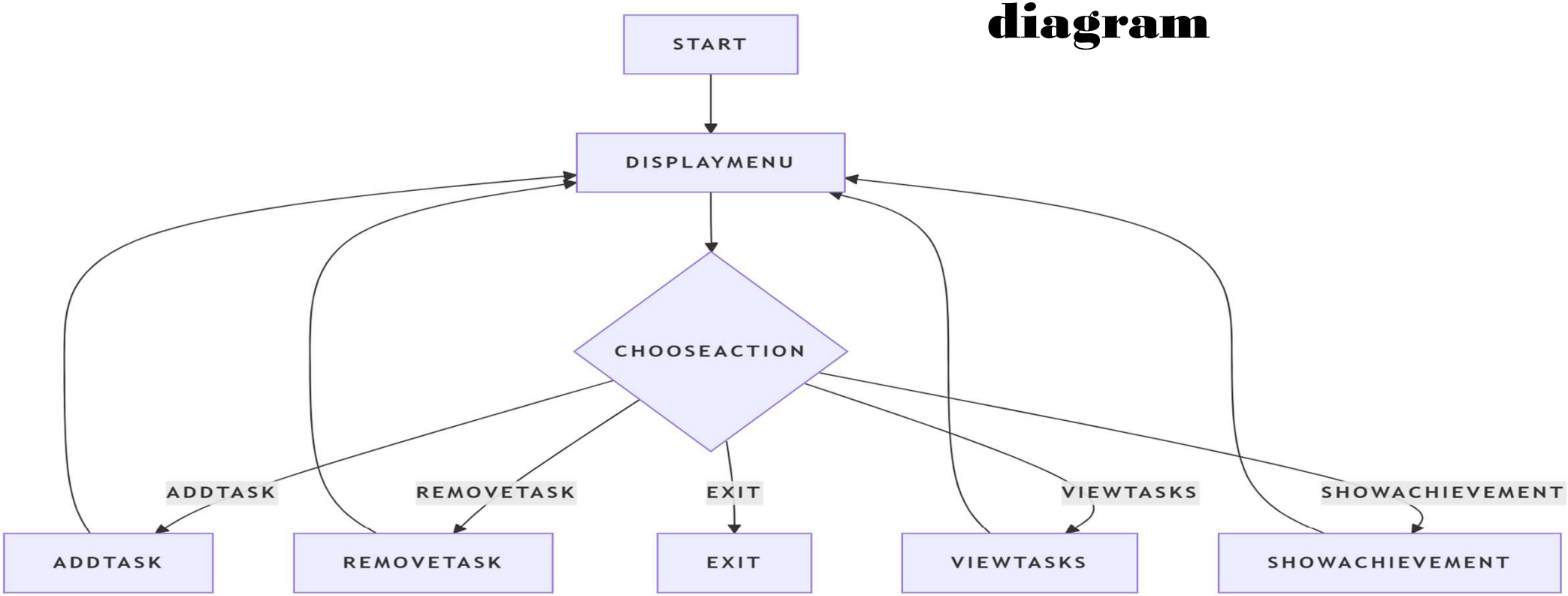


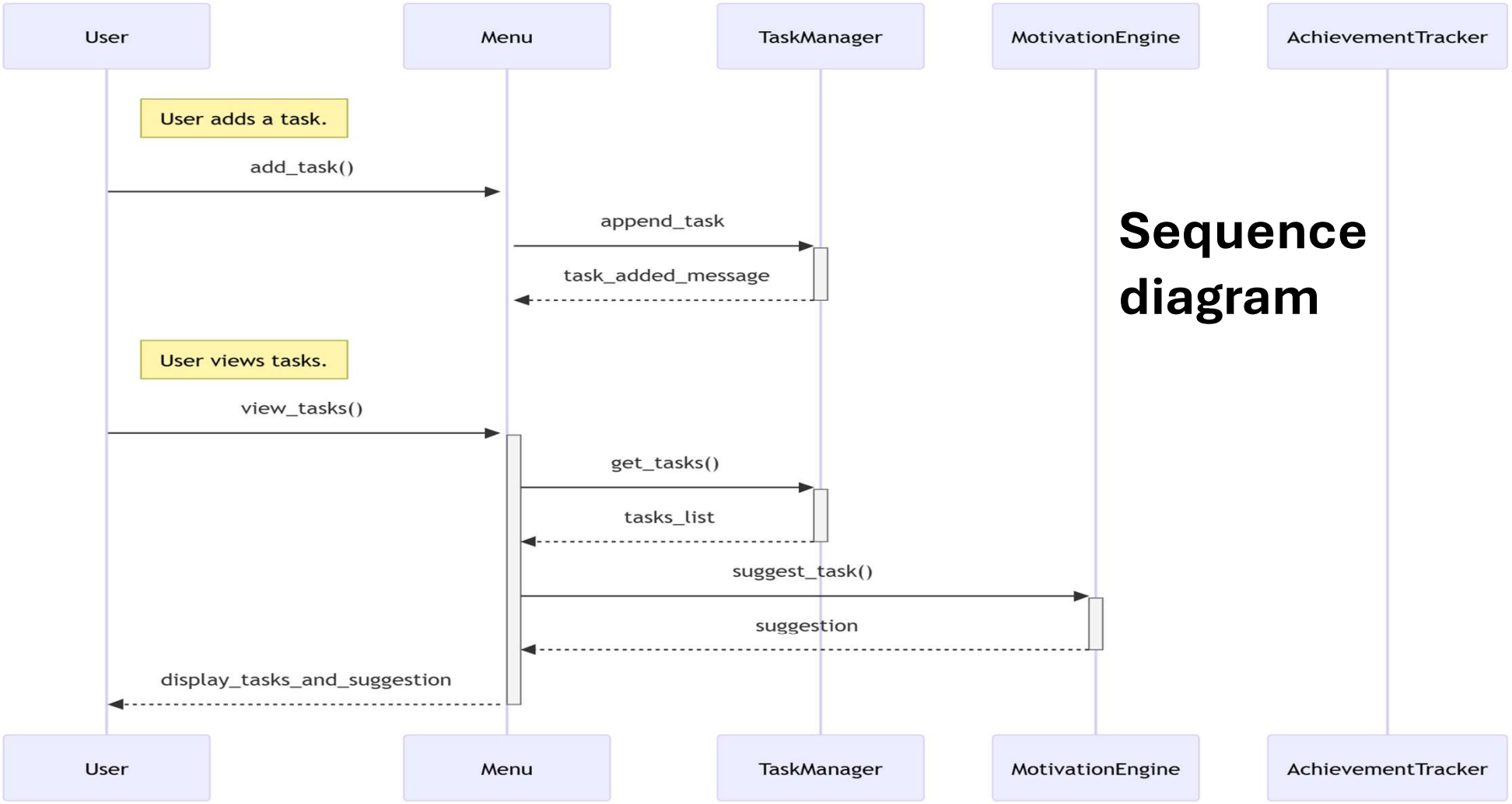
Design Diagrams

Use case diagram



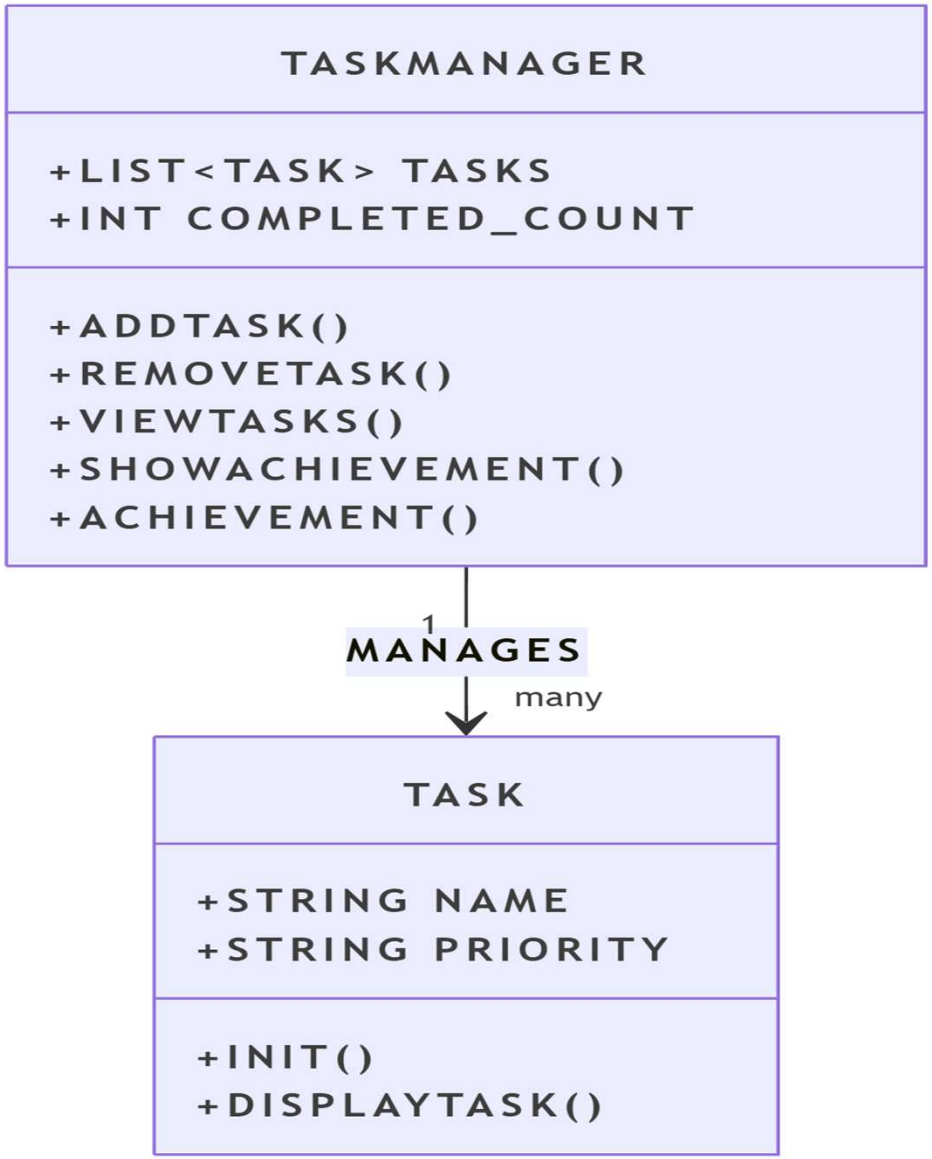
Workflow diagram



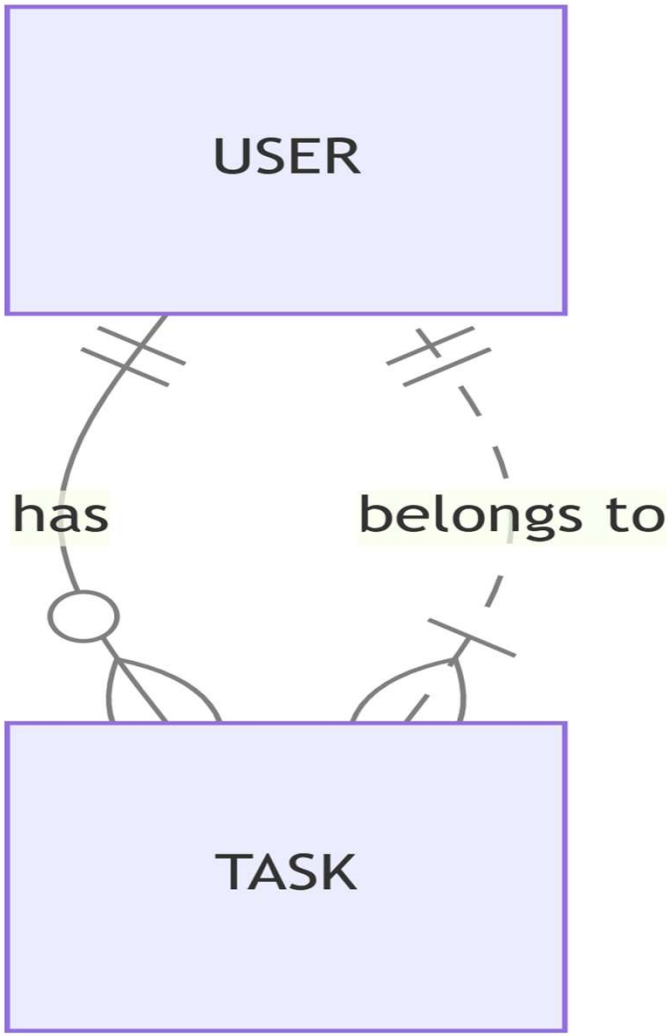


Sequence diagram

CLASS DIAGRAM

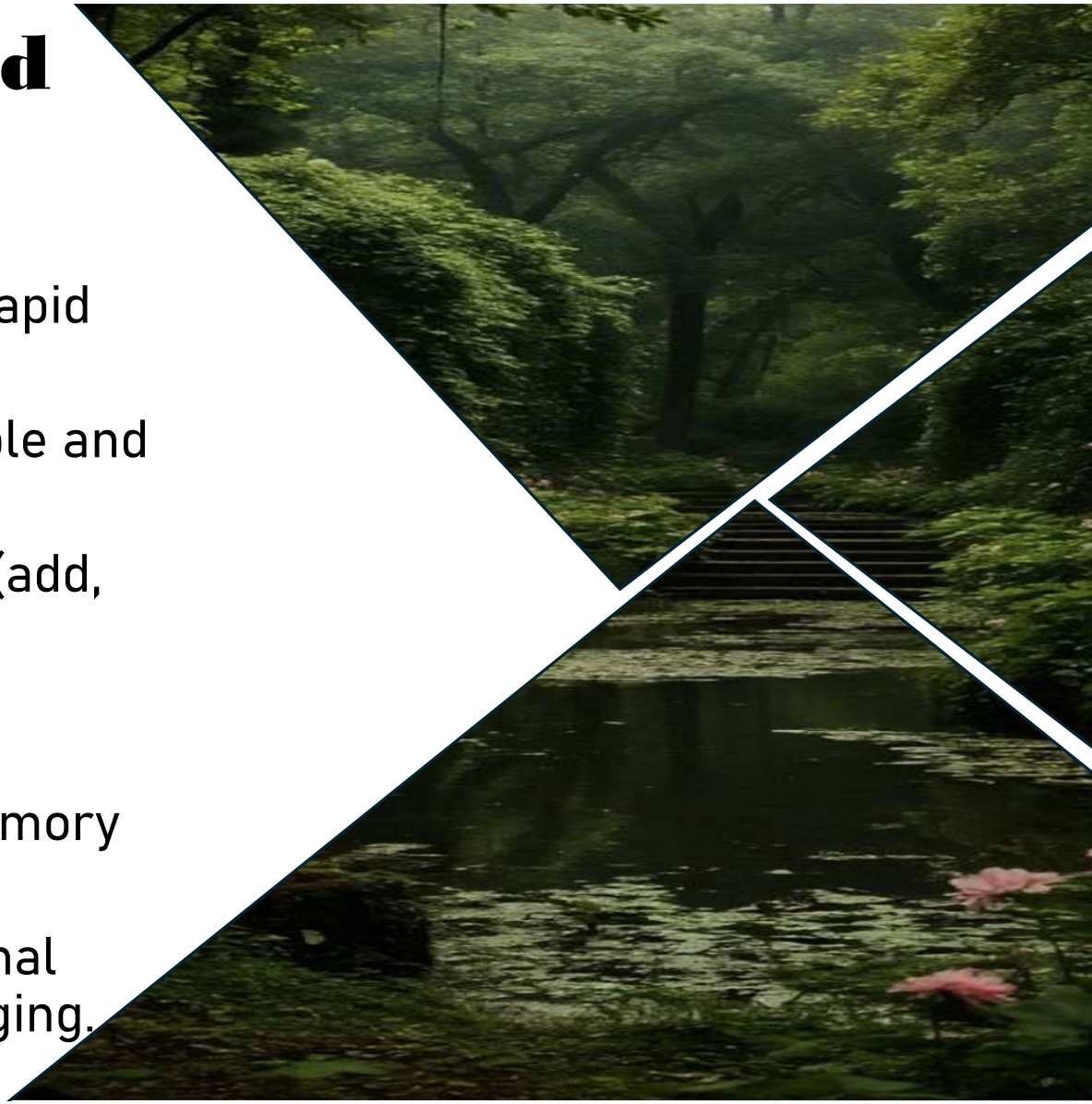


ER diagram



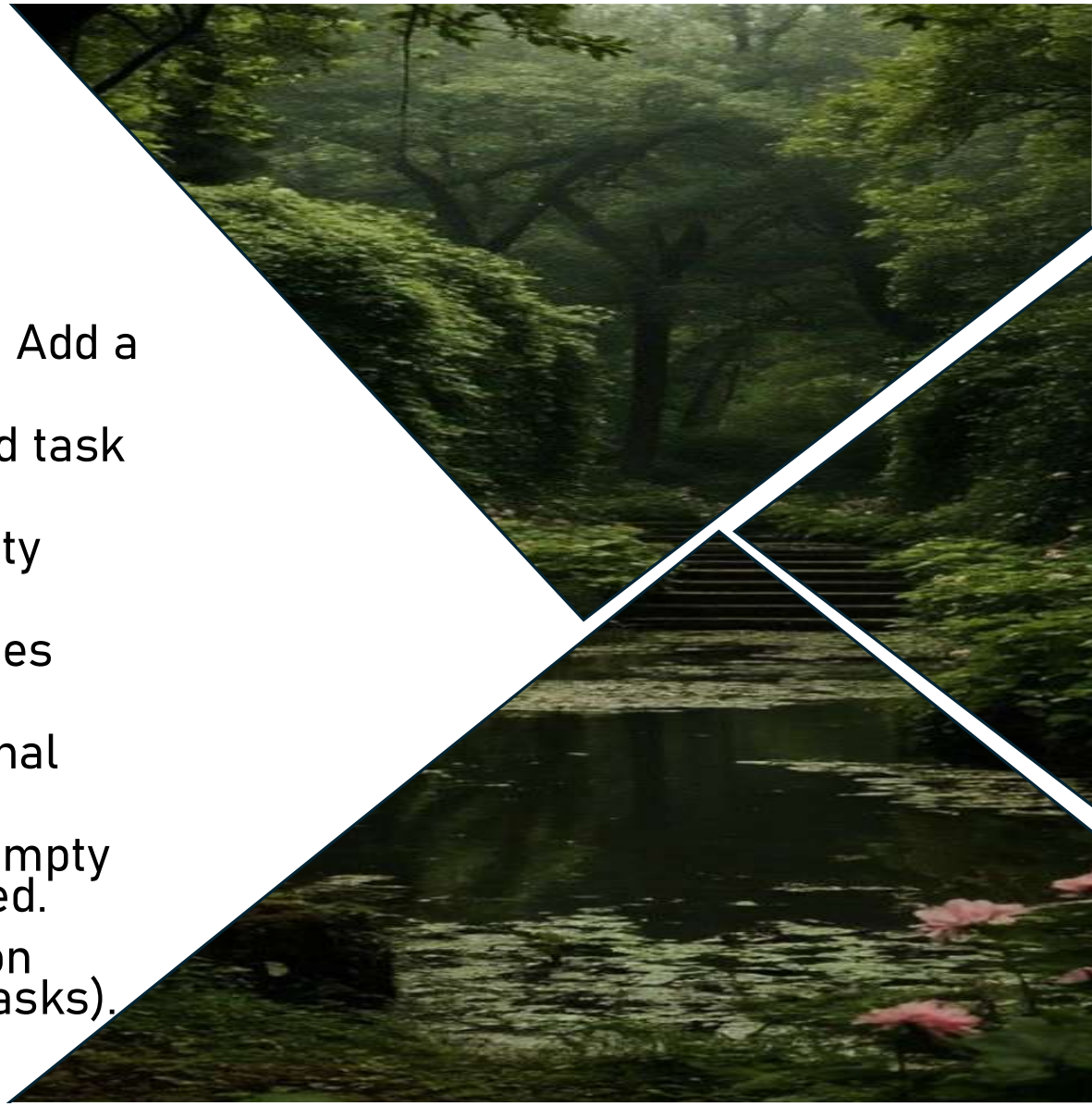
Design Decisions and Rationale

- Python: Chosen for its simplicity, readability, and ability to handle rapid prototyping.
- Menu-driven CLI: Provides a simple and accessible interface.
- Modular Functions: Each feature (add, remove, view, achievements) implemented separately for maintainability.
- Lists for storage: Stored in In-memory lists
- Random suggestions & motivational quotes: Makes the program engaging.

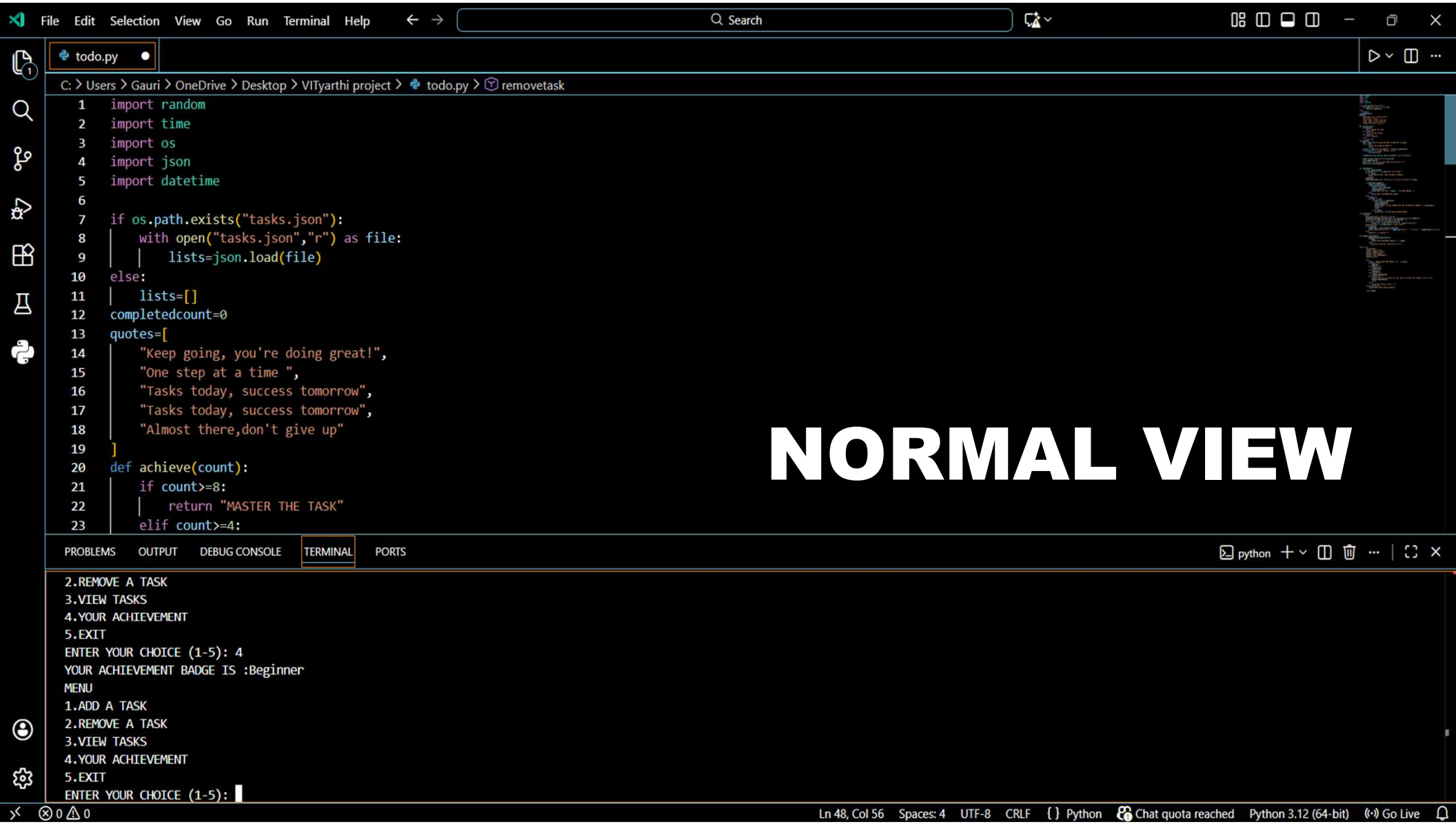


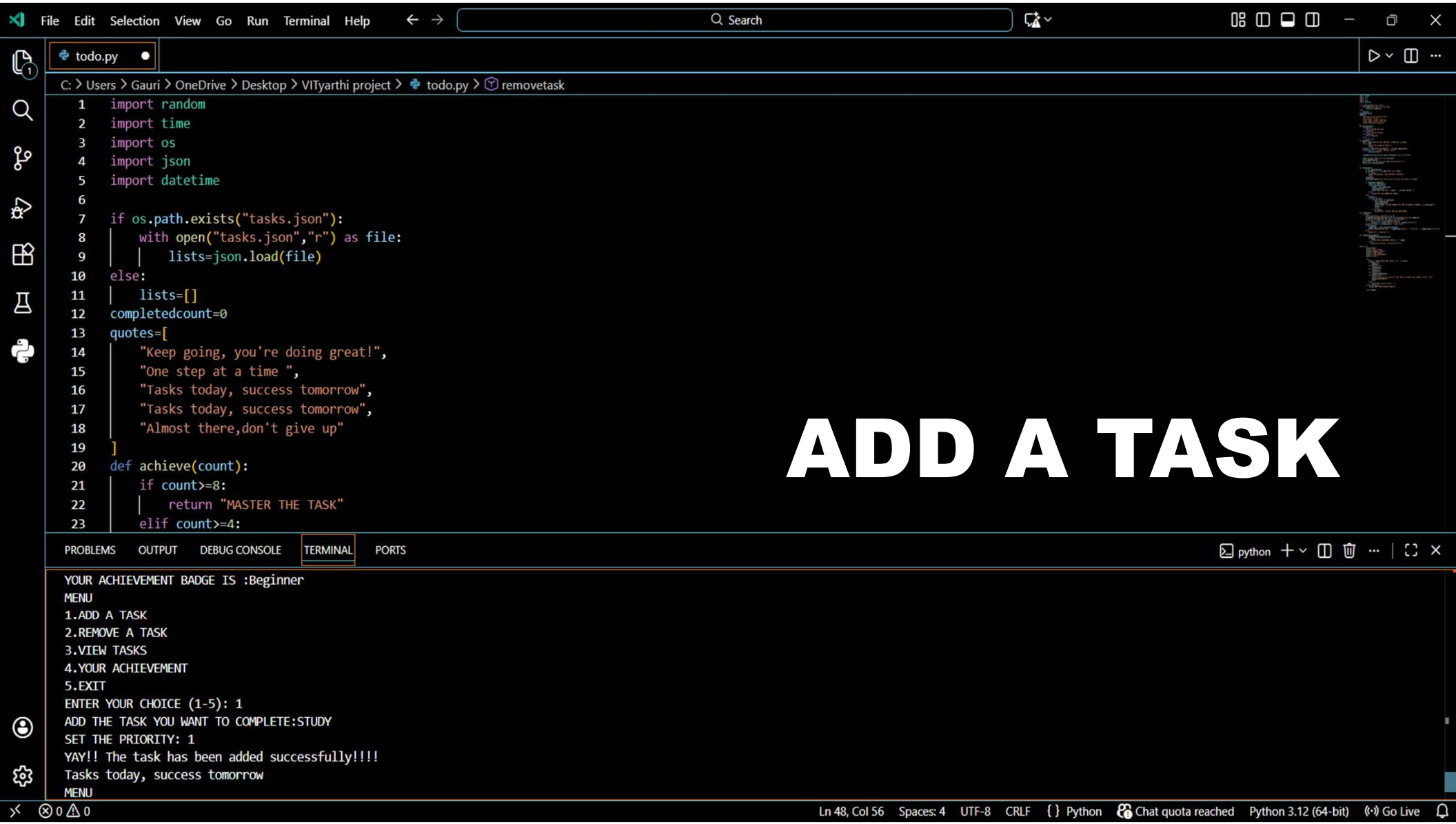
Implementation Details

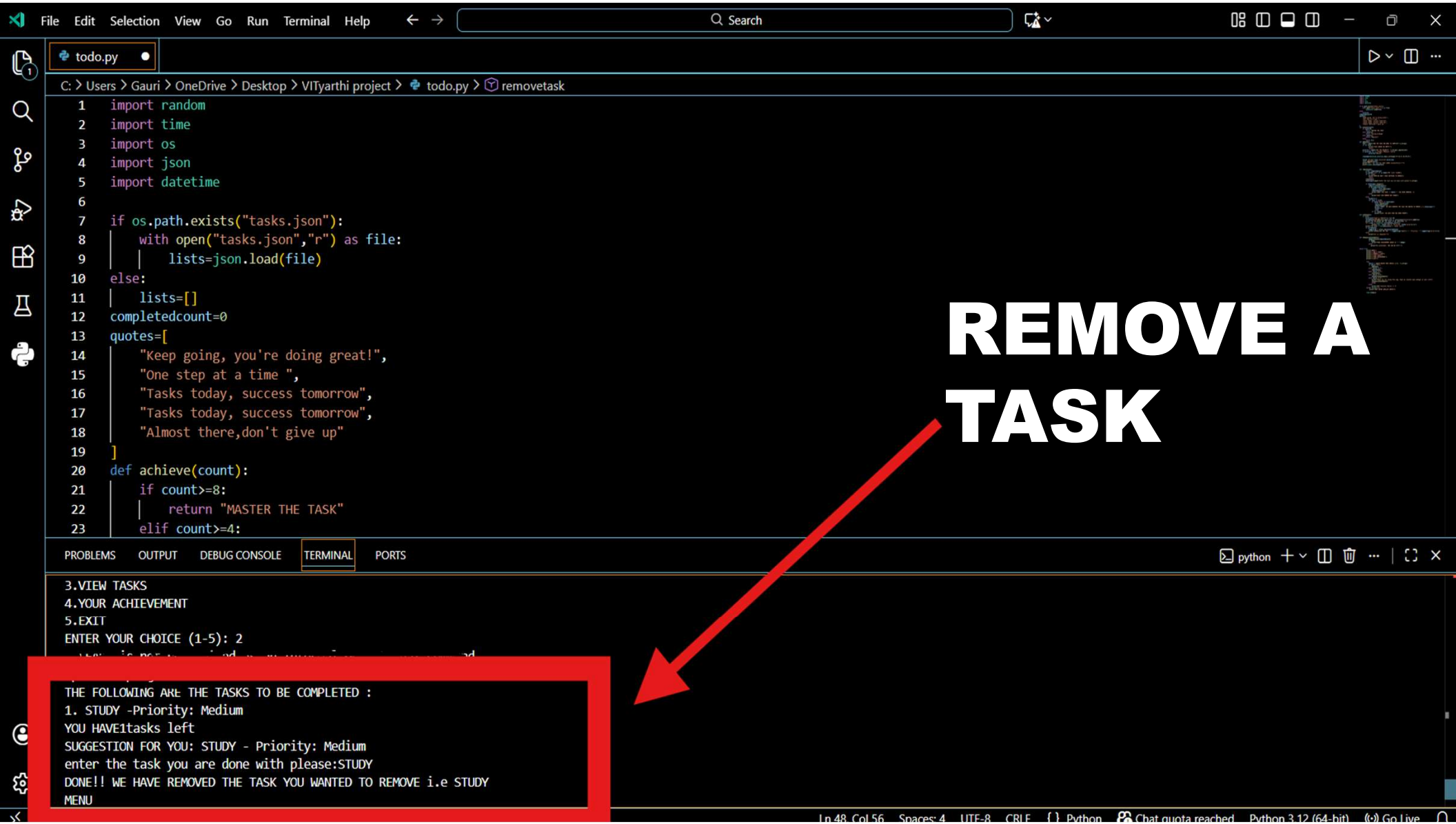
- Developed in Python 3.12.4
- Functions Implemented: add_task(): Add a task with name and priority.
- remove_task(): Remove a completed task safely.
- view_tasks(): Display tasks in priority order with suggestions.
- show_achievements(): Display badges based on completed tasks.
- Random module used for motivational messages and task suggestions.
- Input validation ensures only non-empty tasks with valid priority are accepted.
- Achievements are awarded based on milestones (e.g., 1, 4, 8 completed tasks).

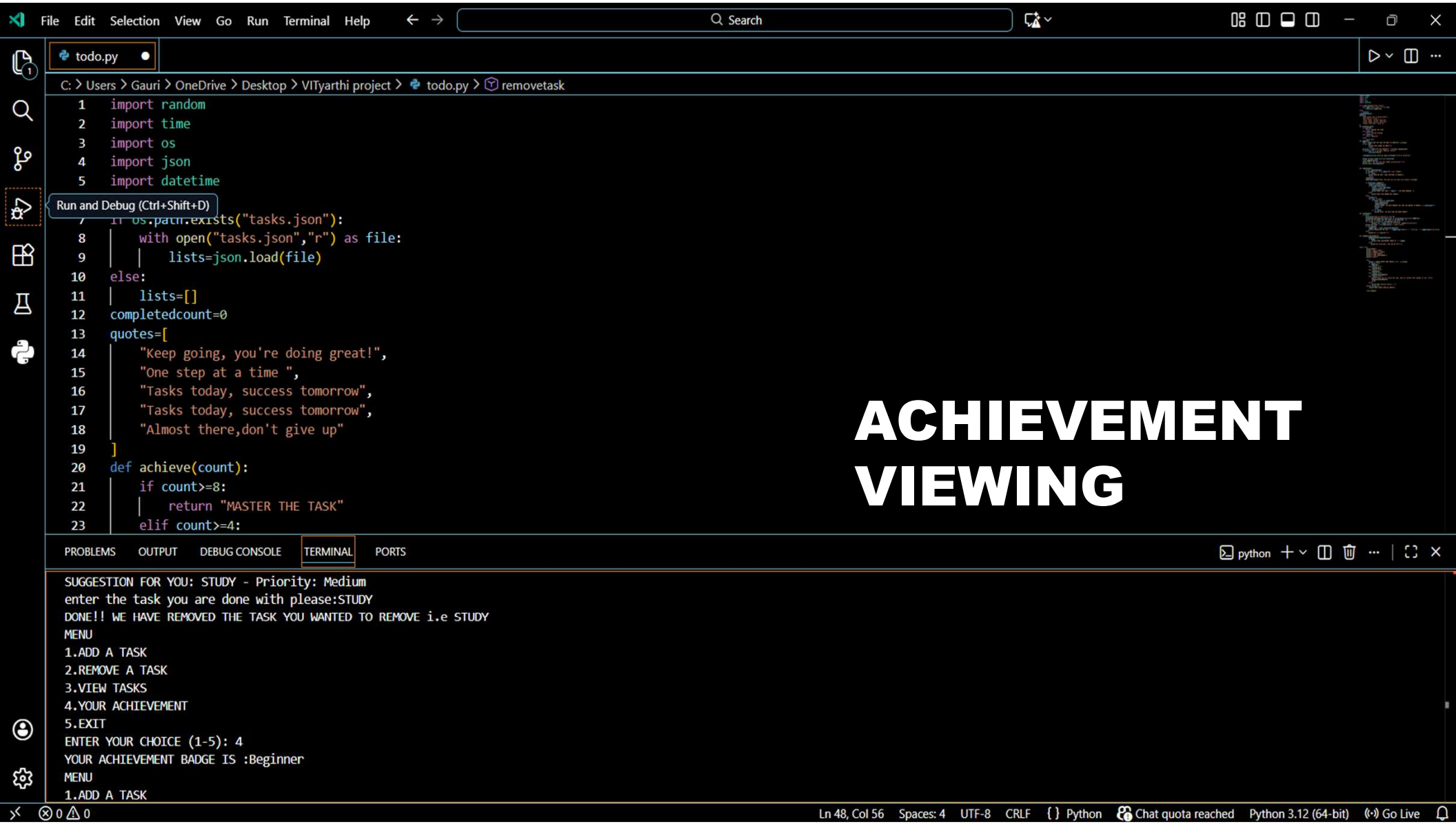


PICTURES OF RESULTS


File Edit Selection View Go Run Terminal Help
Search
todo.py
C: > Users > Gauri > OneDrive > Desktop > VITyarthi project > todo.py > removetask
import random
import time
import os
import json
import datetime
if os.path.exists("tasks.json"):
with open("tasks.json","r") as file:
lists=json.load(file)
else:
lists=[]
completedcount=0
quotes=[
"Keep going, you're doing great!",
"One step at a time ",
"Tasks today, success tomorrow",
"Tasks today, success tomorrow",
"Almost there,don't give up"
]
def achieve(count):
if count>=8:
return "MASTER THE TASK"
elif count>=4:
NORMAL VIEW
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
python
2.REMOVE A TASK
3.VIEW TASKS
4.YOUR ACHIEVEMENT
5.EXIT
ENTER YOUR CHOICE (1-5): 4
YOUR ACHIEVEMENT BADGE IS :Beginner
MENU
1.ADD A TASK
2.REMOVE A TASK
3.VIEW TASKS
4.YOUR ACHIEVEMENT
5.EXIT
ENTER YOUR CHOICE (1-5):
Ln 48, Col 56 Spaces: 4 UTF-8 CRLF { } Python Chat quota reached Python 3.12 (64-bit) Go Live


File Edit Selection View Go Run Terminal Help
Search
todo.py
C: > Users > Gauri > OneDrive > Desktop > VITyarthi project > todo.py > removetask
import random
import time
import os
import json
import datetime
if os.path.exists("tasks.json"):
with open("tasks.json","r") as file:
lists=json.load(file)
else:
lists=[]
completedcount=0
quotes=[
"Keep going, you're doing great!",
"One step at a time ",
"Tasks today, success tomorrow",
"Tasks today, success tomorrow",
"Almost there,don't give up"
]
def achieve(count):
if count>=8:
return "MASTER THE TASK"
elif count>=4:
ADD A TASK
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
python
YOUR ACHIEVEMENT BADGE IS :Beginner
MENU
1.ADD A TASK
2.REMOVE A TASK
3.VIEW TASKS
4.YOUR ACHIEVEMENT
5.EXIT
ENTER YOUR CHOICE (1-5): 1
ADD THE TASK YOU WANT TO COMPLETE:STUDY
SET THE PRIORITY: 1
YAY!! The task has been added successfully!!!!
Tasks today, success tomorrow
MENU
Ln 48, Col 56 Spaces: 4 UTF-8 CRLF { } Python Chat quota reached Python 3.12 (64-bit) Go Live

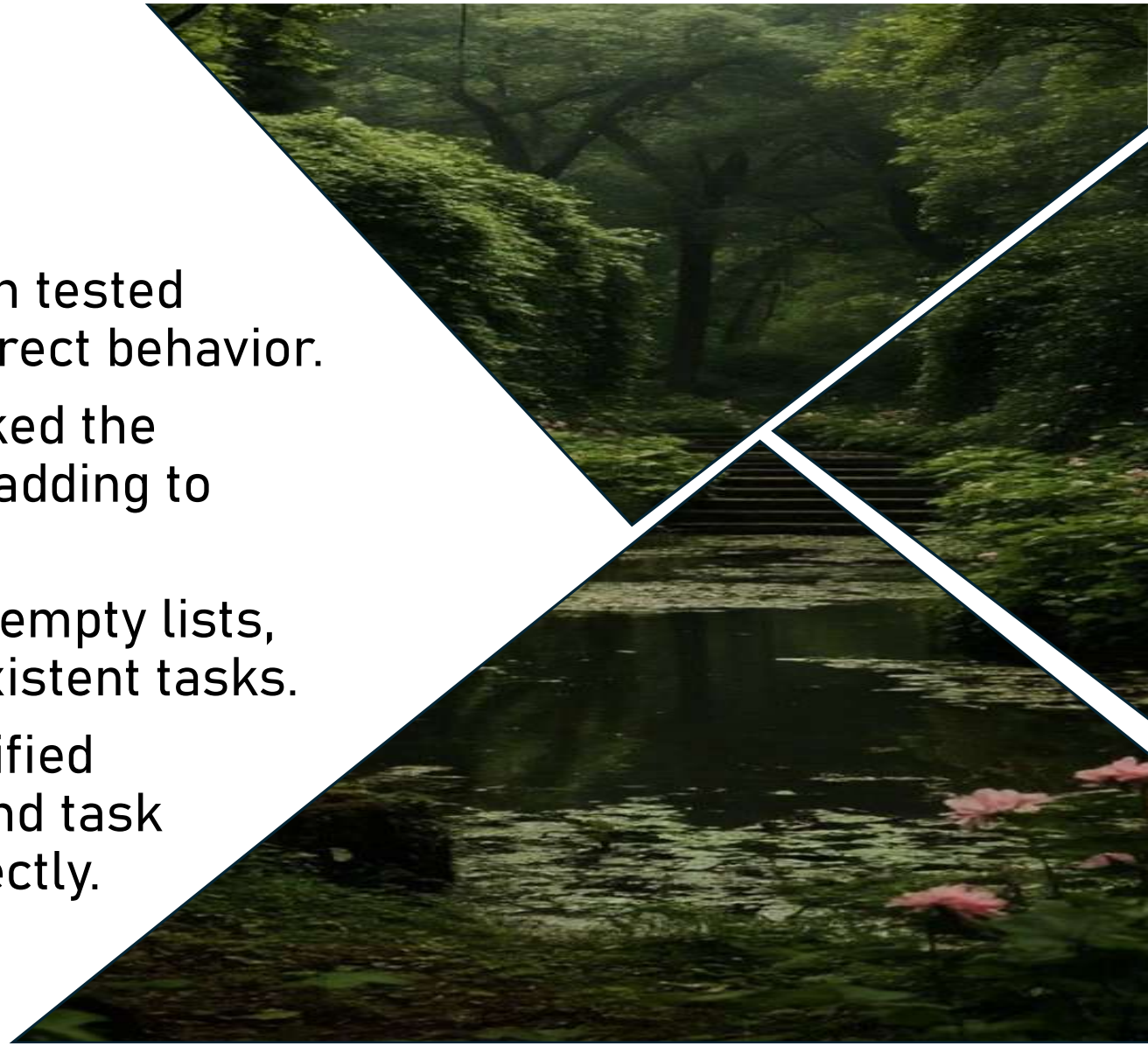

File Edit Selection View Go Run Terminal Help
Search
todo.py
C: > Users > Gauri > OneDrive > Desktop > VITyarthi project > todo.py > removetask
import random
import time
import os
import json
import datetime
if os.path.exists("tasks.json"):
with open("tasks.json","r") as file:
lists=json.load(file)
else:
lists=[]
completedcount=0
quotes=[
"Keep going, you're doing great!",
"One step at a time ",
"Tasks today, success tomorrow",
"Tasks today, success tomorrow",
"Almost there,don't give up"
]
def achieve(count):
if count>=8:
return "MASTER THE TASK"
elif count>=4:
REMOVE A TASK
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
python
3.VIEW TASKS
4.YOUR ACHIEVEMENT
5.EXIT
ENTER YOUR CHOICE (1-5): 2
THE FOLLOWING ARE THE TASKS TO BE COMPLETED :
1. STUDY -Priority: Medium
YOU HAVE1tasks left
SUGGESTION FOR YOU: STUDY - Priority: Medium
enter the task you are done with please:STUDY
DONE!! WE HAVE REMOVED THE TASK YOU WANTED TO REMOVE i.e STUDY
MENU
Ln 48, Col 56 Spaces: 4 UTF-8 CRLF Python Chat quota reached Python 3.12 (64-bit) Go Live


File Edit Selection View Go Run Terminal Help
Search
todo.py
C: > Users > Gauri > OneDrive > Desktop > VITyarthi project > todo.py > removetask
Run and Debug (Ctrl+Shift+D)
import random
import time
import os
import json
import datetime
if os.path.exists("tasks.json"):
with open("tasks.json","r") as file:
lists=json.load(file)
else:
lists=[]
completedcount=0
quotes=[
"Keep going, you're doing great!",
"One step at a time ",
"Tasks today, success tomorrow",
"Tasks today, success tomorrow",
"Almost there,don't give up"
]
def achieve(count):
if count>=8:
return "MASTER THE TASK"
elif count>=4:
ACHIEVEMENT VIEWING
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
python
SUGGESTION FOR YOU: STUDY - Priority: Medium
enter the task you are done with please:STUDY
DONE!! WE HAVE REMOVED THE TASK YOU WANTED TO REMOVE i.e STUDY
MENU
1.ADD A TASK
2.REMOVE A TASK
3.VIEW TASKS
4.YOUR ACHIEVEMENT
5.EXIT
ENTER YOUR CHOICE (1-5): 4
YOUR ACHIEVEMENT BADGE IS :Beginner
MENU
1.ADD A TASK
Ln 48, Col 56 Spaces: 4 UTF-8 CRLF { } Python Chat quota reached Python 3.12 (64-bit) Go Live


File Edit Selection View Go Run Terminal Help
Search
todo.py
C: > Users > Gauri > OneDrive > Desktop > VITyarthi project > todo.py > removetask
import random
import time
import os
import json
import datetime
if os.path.exists("tasks.json"):
with open("tasks.json","r") as file:
lists=json.load(file)
else:
lists=[]
completedcount=0
quotes=[
"Keep going, you're doing great!",
"One step at a time ",
"Tasks today, success tomorrow",
"Tasks today, success tomorrow",
"Almost there,don't give up"
]
def achieve(count):
if count>=8:
return "MASTER THE TASK"
elif count>=4:
VIEWING TASK
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
python
2.REMOVE A TASK
3.VIEW TASKS
4.YOUR ACHIEVEMENT
5.EXIT
ENTER YOUR CHOICE (1-5): 3
THE FOLLOWING ARE THE TASKS TO BE COMPLETED :
1. STUDY -Priority: Medium
2. CSE PROJECT -Priority: Medium
3. HOMEWORK -Priority: Medium
YOU HAVE3tasks left
SUGGESTION FOR YOU: STUDY - Priority: Medium
MENU
1.ADD A TASK
Ln 48, Col 56 Spaces: 4 UTF-8 CRLF Python Chat quota reached Python 3.12 (64-bit) Go Live

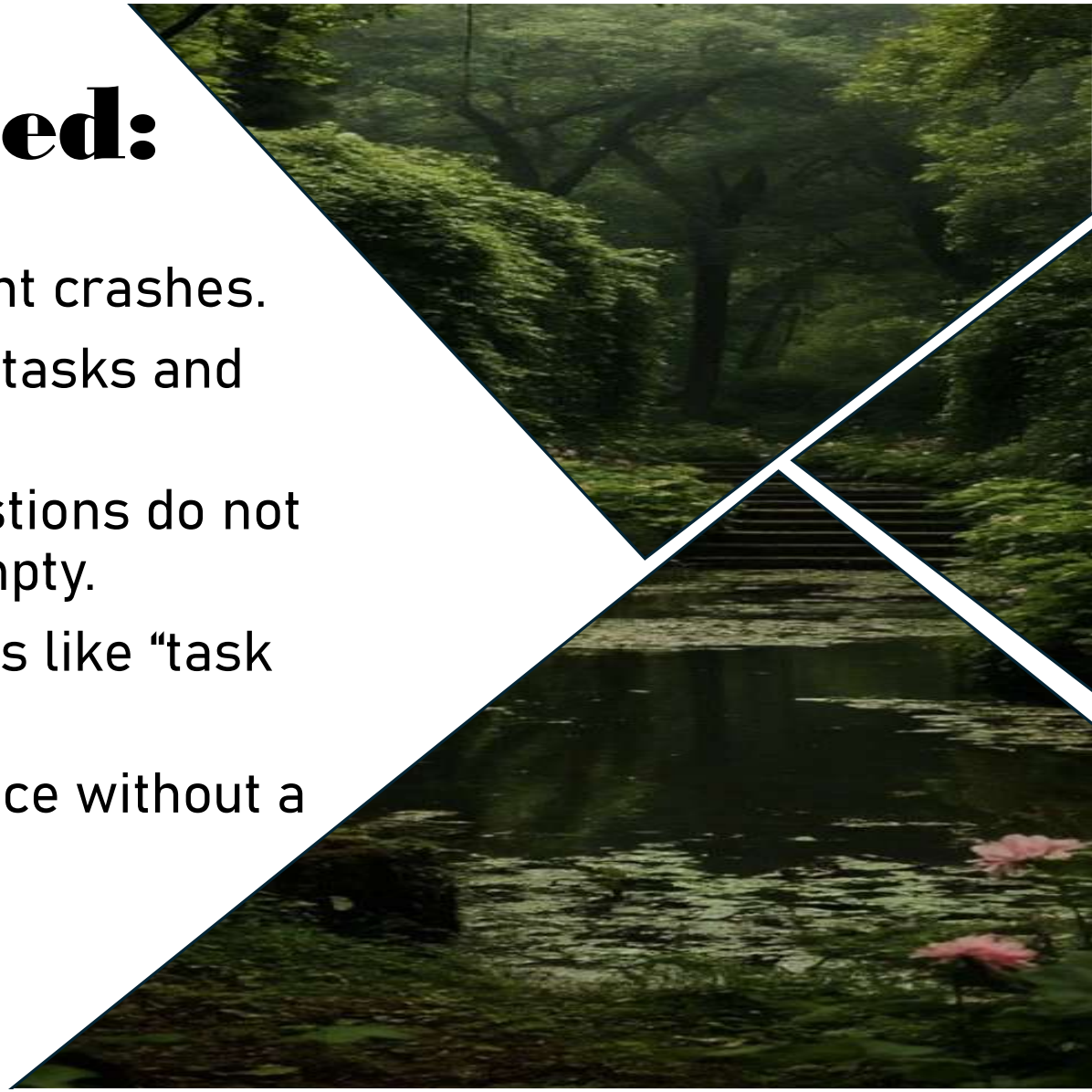
Testing Approach

- Unit Testing: Each function tested individually to ensure correct behavior.
- Integration Testing: Checked the complete workflow from adding to removing tasks.
- Boundary Testing: Tested empty lists, invalid inputs, and non-existent tasks.
- Randomness Testing: Verified motivational messages and task suggestions appear correctly.



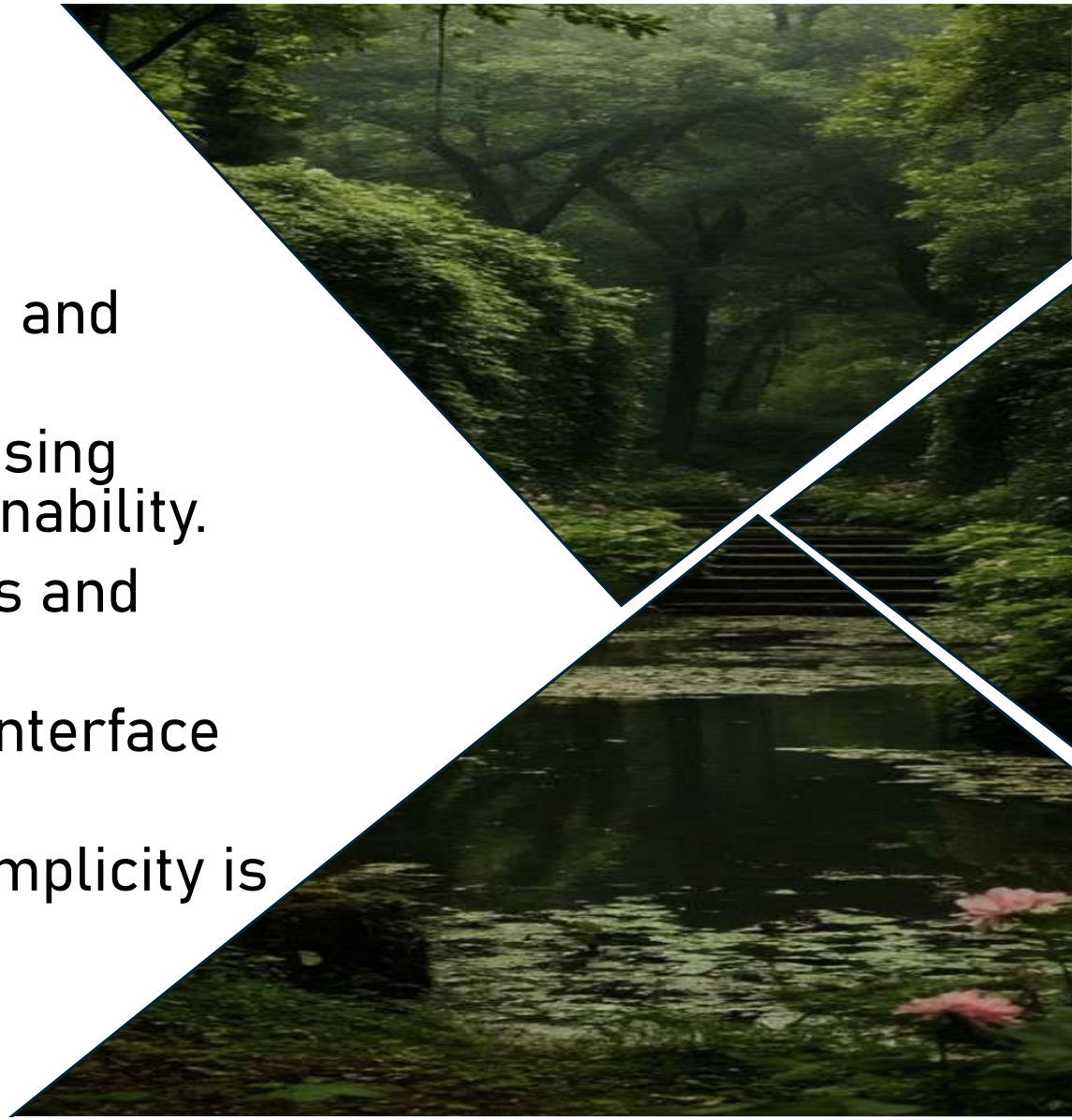
Challenges faced:

- Handling empty lists to prevent crashes.
- Correctly tracking completed tasks and updating achievements.
- Ensuring random task suggestions do not break when the task list is empty.
- Avoiding misleading messages like “task completed” for added tasks.
- Making an engaging experience without a GUI.



Learnings and Key Takeaways

- Importance of input validation and handling edge cases.
- How to structure a program using modular functions for maintainability.
- Using Python's data structures and random module effectively.
- Designing a userfriendly CLI interface improves engagement.
- Balancing functionality and simplicity is crucial in small projects.



Future Enhancements

- Implement persistent storage (file/database) to save tasks permanently.
- Develop a GUI version using Tkinter or PyQt.
- Add reminders or notifications for pending tasks.
- Enable task categories or tags for better organization.
- Integrate analytics or dashboard to track completed tasks over time.
- Integrate with calendar apps or email notifications.

References

1. Python Official Documentation – https://docs.python.org/3/Python Random Module
2. Documentation – https://docs.python.org/3/library/random.html
3. Stack Overflow – For programming solutions and debugging help- https://stackoverflow.com/
4. “Python Essentials” – VITyarthi
5. Modules used : random and time (Built in modules)
6. Github