To-Do List Implementation in C++

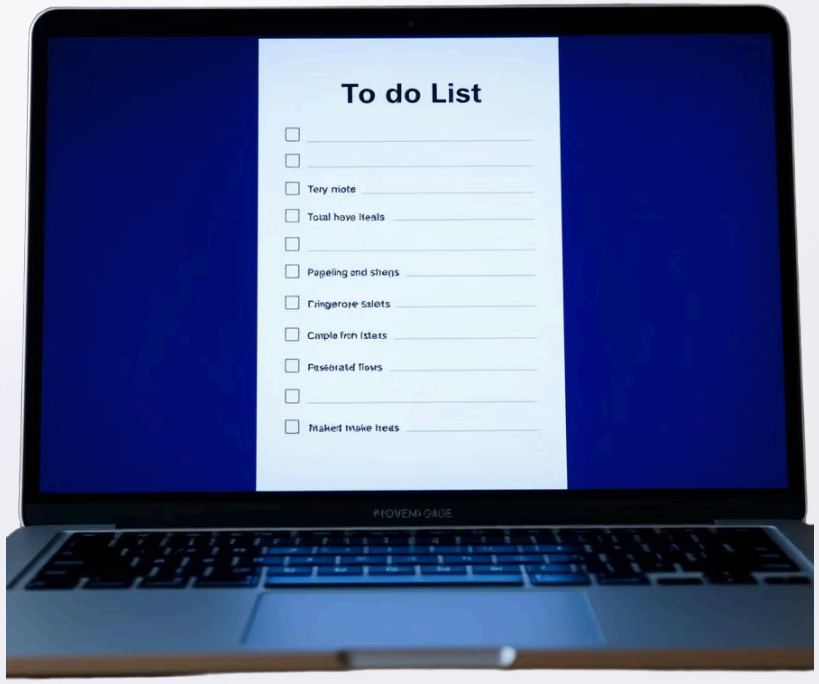
Presented by:

S. Saniya Haseen - AP23110010517

D. Jasmitha - AP23110010518

K. Chandini - AP23110010523

Guided by: Kavitha Rani Karnena Madam



Objective:

To design a console-based To-Do List application for efficient task management. Motivation: Simplify daily task organization. Learn and apply C++ concepts like file handling and modular programming. Outcome: A functional application with task addition, updating, deletion, and persistent data storage.

Key Components

Structure:

`ToDoList` structure: Contains unique task IDs and descriptions.

File Handling: Persistent storage in `todo.txt` using `ifstream` and `ofstream`. Temporary files for secure updates and deletions.

Modular Design: Functions like `addTask()`, `deleteTask()`, and `updateTask()` ensure reusability.

Core Functionalities

Add Tasks: Assigns unique IDs and stores tasks persistently.

Show All Tasks: Displays tasks in a user-friendly format.

Search Tasks: Finds tasks based on unique IDs.

Update Tasks: Modifies task descriptions.

Delete Tasks: Removes completed or irrelevant tasks.

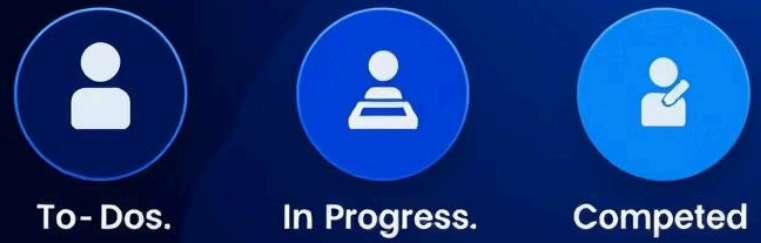
Methodology

Approach:

Modular functions: `addTask()`, `showTask()`, `searchTask()`, etc. File handling ensures persistence across sessions.

Task Structure: Each task has a unique ID and description.

Development Highlights: Error handling for file operations. Temporary files for updates and deletions.



Code Highlights

Structure of To-Do List:

`struct ToDoList { int id; string task; }`

Key Functions:

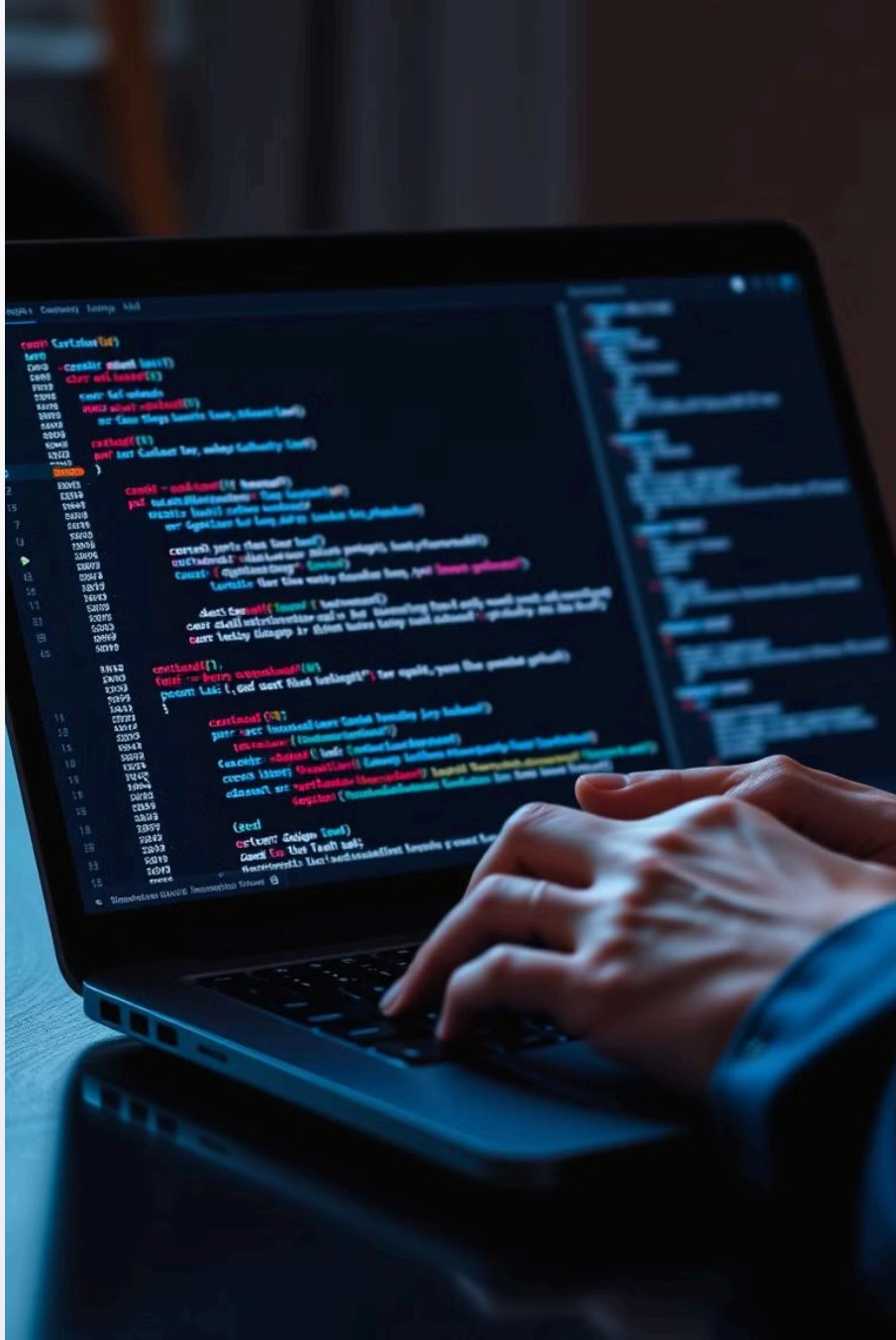
`addTask()`: Adds a new task.

`showTask()`: Displays all tasks.

`deleteTask()`: Deletes tasks using temporary files.

`updateTask()`: Updates task descriptions.

Persistence: Tasks stored in todo.txt.



Recommendation Logic

Priority Management: Users can focus on high-priority tasks first.

Task Categorization: Group tasks by type (e.g., Work, Personal, Urgent).

Future Enhancements: AI-based suggestions for deadlines and prioritization.

Example Usage

Task Addition:

Input: "Complete homework"

Output: Task added successfully!

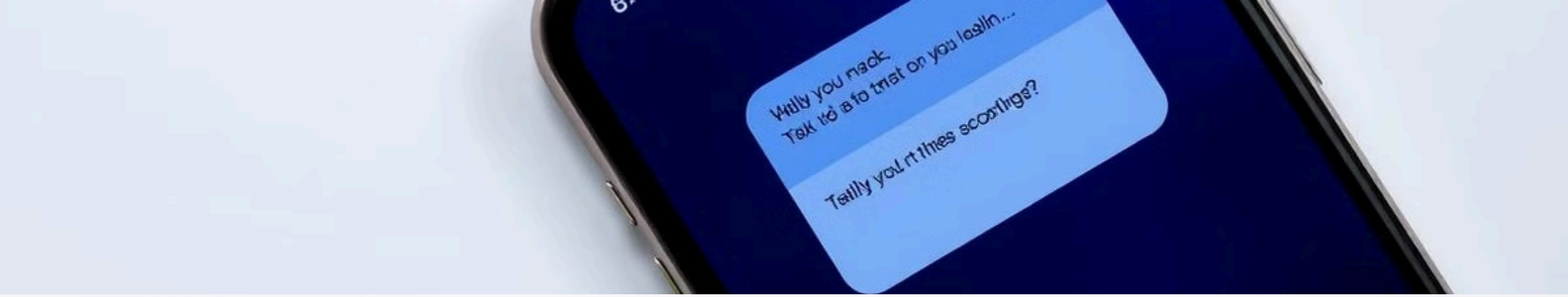
Task Deletion: Input: Task ID = 1

Output: Task deleted successfully!

Task Update:

Input: Update Task ID = 1 to "Submit project report"

Output: Task updated successfully!



Console Menu

Options:

Add Task

Show Tasks

Search Tasks

Update Task

Delete Task

Exit

Example Flow:

Select Option 1 → Add a task.

Select Option 3 → Search for a task by ID.

Key Takeaways

Practical Application:

Demonstrates the use of C++ concepts like file handling and modular programming.

Skill Development: Strengthened error handling and UI design skills.

Future Scope: Potential for integrating reminders, user authentication, and a GUI.