Program in C++ using Threads : Simulation of Elevator Program with 9 floors

in
Parallel and Distributed Computing (CS0051)

Instructor:
Doc. Hadji Tejuco

Members:
Gian Ijy Clemente
Jabenn P. Tinio

27/01/2025

TABLE OF CONTENTS

III.Introduction

A. Purpose

The purpose of the program aims to enhance our comprehension of multithreading ideas in C++ through the simulation of a real-world application: an elevator system including nine stories. This simulation aims to demonstrate how threads can be effectively used to manage concurrent tasks, such as handling multiple elevator requests, managing floor-level operations, and optimizing responsiveness.

This program aims to:

1. Explore the practical use of C++ threads and synchronization mechanisms

2. Enhance our problem-solving skills in designing systems that require parallel and distributed computing principles.

This activity functions as both a practical exercise and a framework for implementing theoretical principles in a real-world context, along with the course objectives of promoting hands-on experience with parallelism and concurrency.

B. Objectives

We are asked to provide a simulation of an elevator program based on our own specifications. The program has several objectives, which can be pointed out as follows:

1. To create a simulation of real-world elevator functionality. To the best of the developers' skills, they replicate a near-real-world elevator function that can ascend or descend floors of a nine-story building.
2. To demonstrate multithreading in C++ by a simulation of an elevator program. Handling user requests and controlling elevator movements.

C. Scope

The scope of the program is a single user simulation of a real-world elevator with 9 floors that can handle multiple requests from the users.

IV. Project Overview

A. Problem Statement

To simulate an Elevator Program, with 9 floors.

B. Key Features

1. **Visualization :** Shows the user a simple text visualisation of the floor where the simulated elevator is, when it is moving, and whether it has arrived at its destination.

2. **Multiple Request Handling :** Can handle multiple floor requests from the user.

V . Requirements Analysis

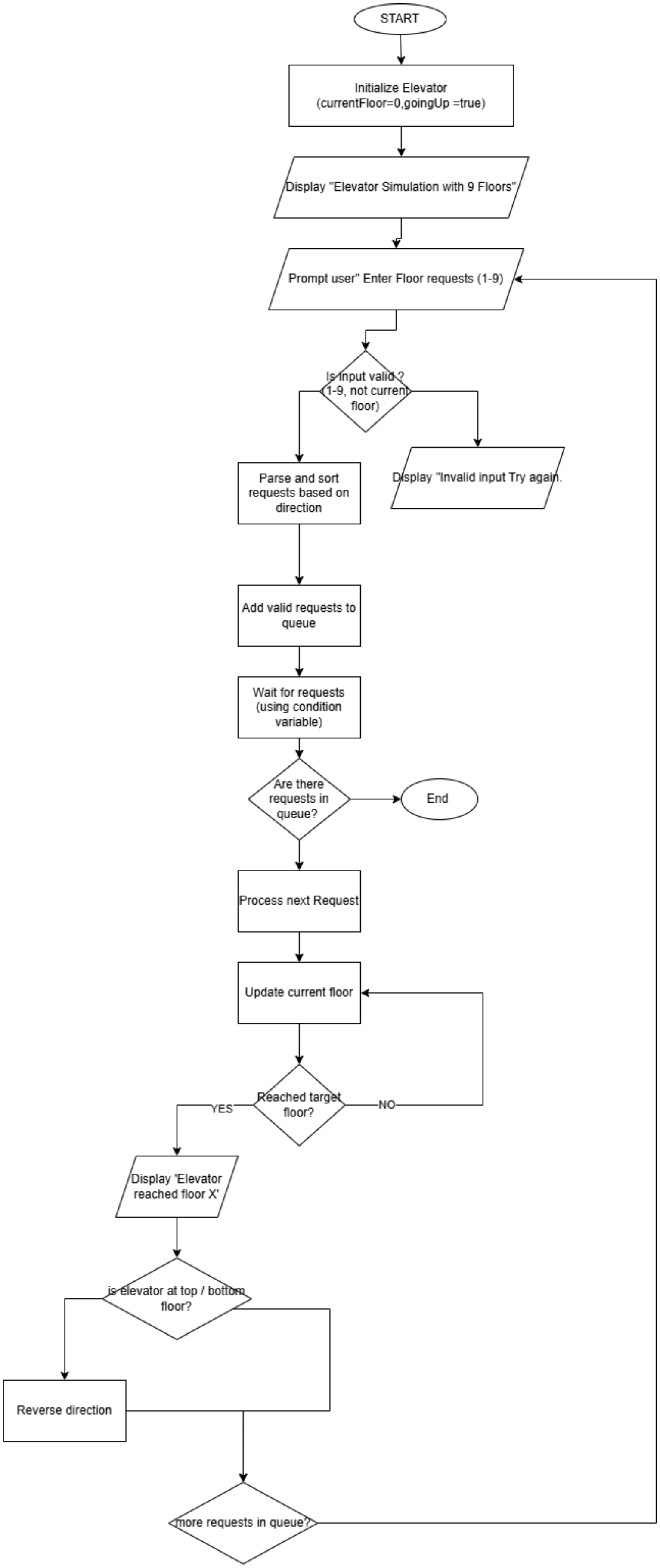
A. Functional Requirements

1. **Elevator Simulation**: Enable simulation for moving between floors (1-9).
2. **Request Processing**: Accept and process user-inputted floor requests.
3. **Request Sorting**: Sort floor requests in ascending or descending order based on the elevator's direction.
4. **Direction Control**: Reverse direction at top (9th floor) or bottom (ground floor).
5. **Real-time Status**: Display current floor during movement.
6. **Input Validation**: Validate user input for floor requests

B. Non-Functional Requirements

1. **Performance:** Minimal response time for processing and moving between floors.
2. **Concurrency:** Efficient handling of requests using multithreading.
3. **Security:** Thread-safe access to shared resources
4. **Usability:** Clear prompts and instructions for user input, handling invalid inputs gracefully.
5. **Scalability:** Extendable to support more floors or multiple elevators.

VI . System Design

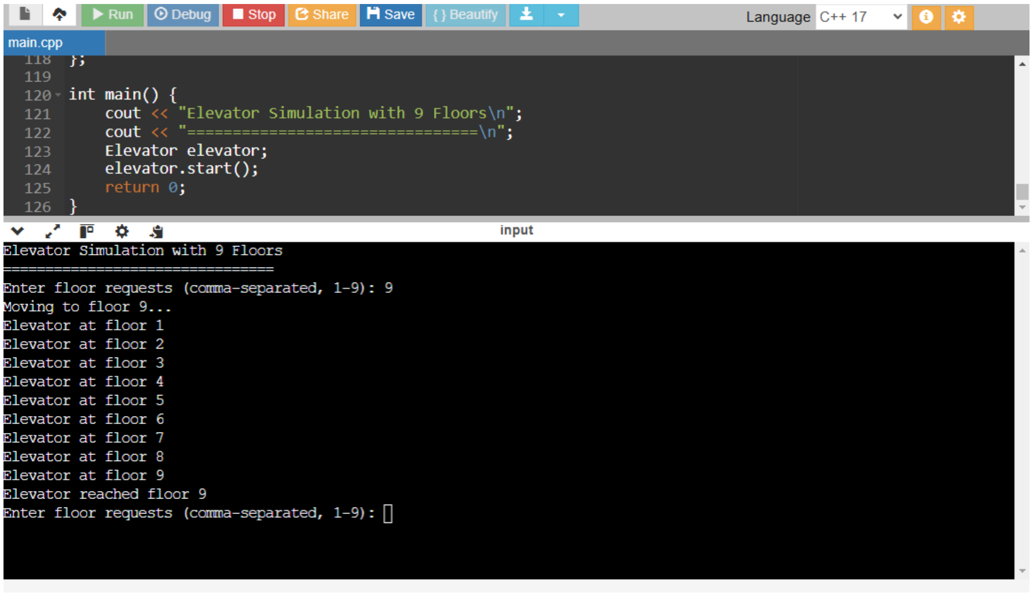

START
Initialize Elevator (currentFloor=0,goingUp =true)
Display "Elevator Simulation with 9 Floors"
Prompt user" Enter Floor requests (1-9)
Is input valid ? (1-9, not current floor)
Parse and sort requests based on direction
Display "Invalid input Try again.
Add valid requests to queue
Wait for requests (using condition variable)
Are there requests in queue?
End
Process next Request
Update current floor
Reached target floor?
YES
NO
Display 'Elevator reached floor X'
is elevator at top / bottom floor?
Reverse direction
more requests in queue?

VII . Implementation

A. Technologies Used

1. Programming Languages :
    a. C++ 17
2. Frameworks and Libraries:
    a. C++ Standard Library (STL)
    b. <thread> -
    c. <mutex>
    d. <condition_variable>
    e. <queue>
    f. <vector>
    g. <algorithm>
    h. <string>
    i. <chrono>
    j. <iostream>
3. Tools:
    a. OnlineGDB

B. Screenshot:



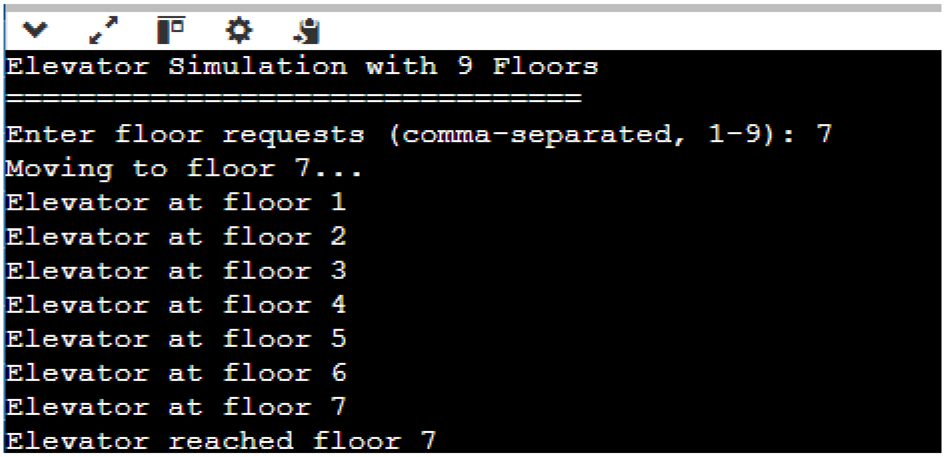
Language C++ 17

```
};

int main() {
    cout << "Elevator Simulation with 9 Floors\n";
    cout << "================================\n";
    Elevator elevator;
    elevator.start();
    return 0;
}
```

```
Elevator Simulation with 9 Floors
================================
Enter floor requests (comma-separated, 1-9): 9
Moving to floor 9...
Elevator at floor 1
Elevator at floor 2
Elevator at floor 3
Elevator at floor 4
Elevator at floor 5
Elevator at floor 6
Elevator at floor 7
Elevator at floor 8
Elevator at floor 9
Elevator reached floor 9
Enter floor requests (comma-separated, 1-9): 
```

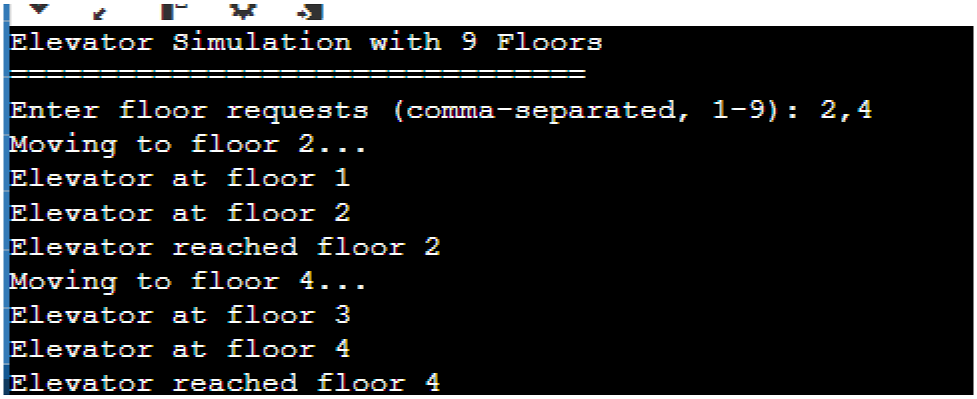
VIII . Testing

a. Single Input - choosing 1 floor



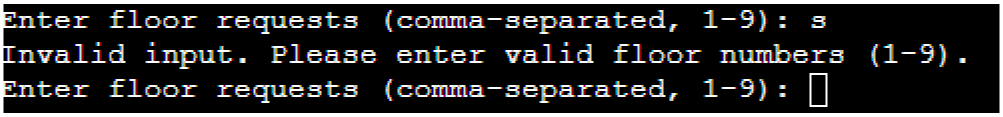
```
Elevator Simulation with 9 Floors
==================================
Enter floor requests (comma-separated, 1-9): 7
Moving to floor 7...
Elevator at floor 1
Elevator at floor 2
Elevator at floor 3
Elevator at floor 4
Elevator at floor 5
Elevator at floor 6
Elevator at floor 7
Elevator reached floor 7
```

b. **Multiple inputs -** choosing multiple floors



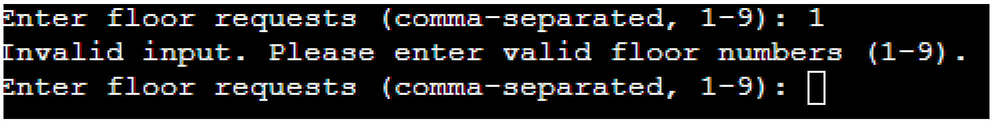
```
Elevator Simulation with 9 Floors
==================================
Enter floor requests (comma-separated, 1-9): 2,4
Moving to floor 2...
Elevator at floor 1
Elevator at floor 2
Elevator reached floor 2
Moving to floor 4...
Elevator at floor 3
Elevator at floor 4
Elevator reached floor 4
```

c. Inputting a non-numerical value



```
Enter floor requests (comma-separated, 1-9): s
Invalid input. Please enter valid floor numbers (1-9).
Enter floor requests (comma-separated, 1-9):
```

d. Inputting the same floor value

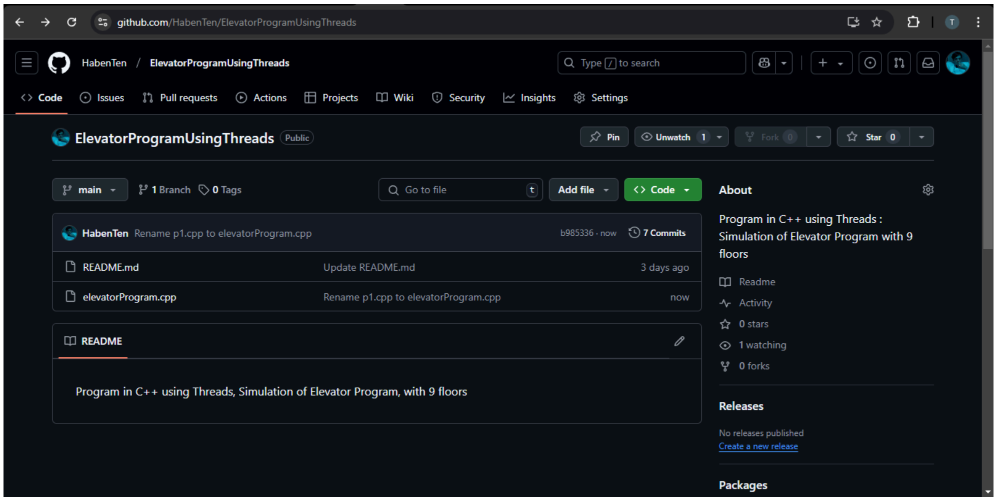


```
Enter floor requests (comma-separated, 1-9): 1
Invalid input. Please enter valid floor numbers (1-9).
Enter floor requests (comma-separated, 1-9):
```

IX . User Manual

A. Installation Guide

a. Visit our git repository at https://github.com/HabenTen/ElevatorProgramUsingThreads.git



HabenTen Rename p1.cpp to elevatorProgram.cpp b985336 · now 7 Commits

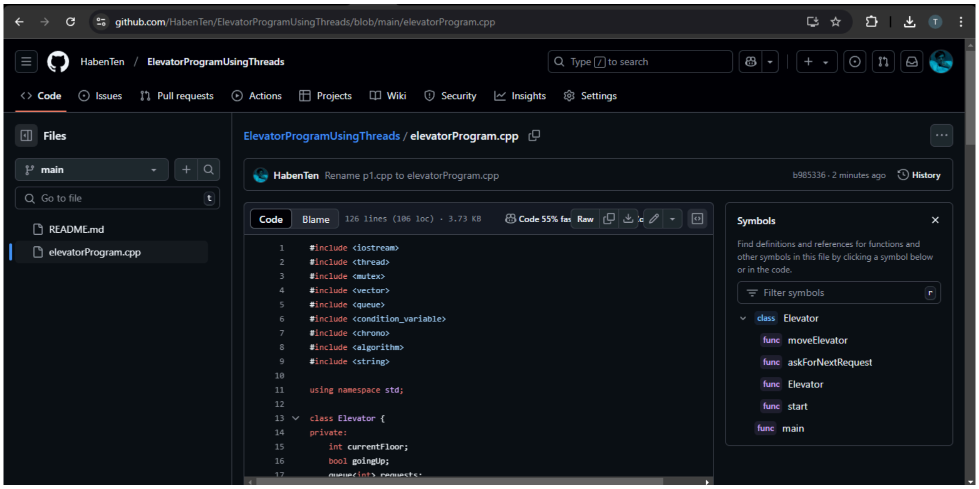
README.md	Update README.md	3 days ago
elevatorProgram.cpp	Rename p1.cpp to elevatorProgram.cpp	now

README

Program in C++ using Threads, Simulation of Elevator Program, with 9 floors

About: Program in C++ using Threads : Simulation of Elevator Program with 9 floors

b. Click elevatorProgram.cpp, either download it or copy the code.However for this tutorial we will be using OnlineGDB for ease of use, so we recommend to copy the code.

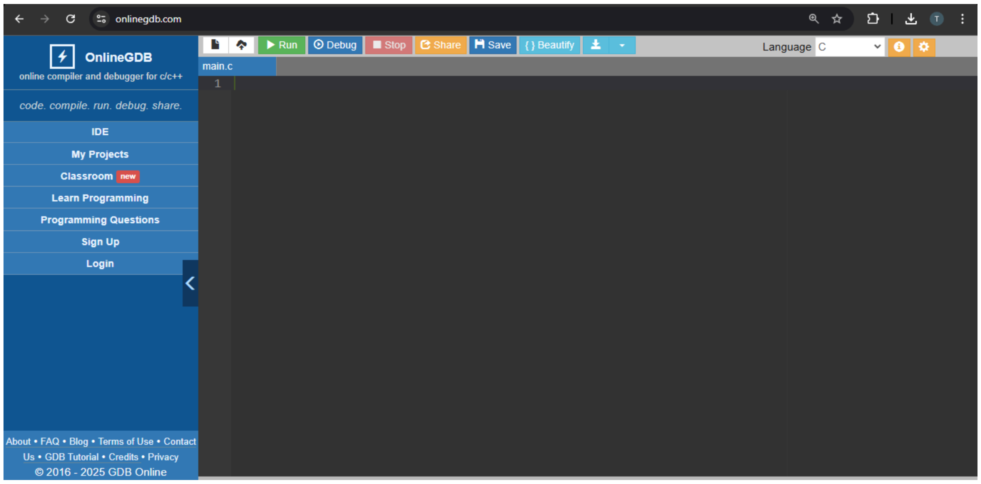


HabenTen Rename p1.cpp to elevatorProgram.cpp b985336 · 2 minutes ago

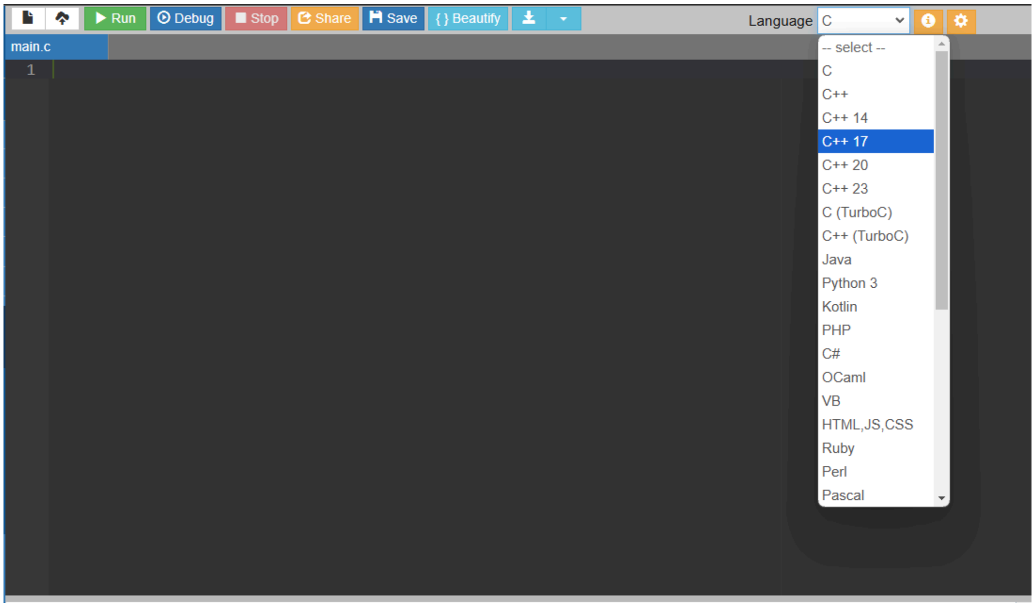
Find definitions and references for functions and other symbols in this file by clicking a symbol below or in the code.

```
#include <iostream>
#include <thread>
#include <mutex>
#include <vector>
#include <queue>
#include <condition_variable>
#include <chrono>
#include <algorithm>
#include <string>

using namespace std;

class Elevator {
private:
    int currentFloor;
    bool goingUp;
    queue<int> requests;
```

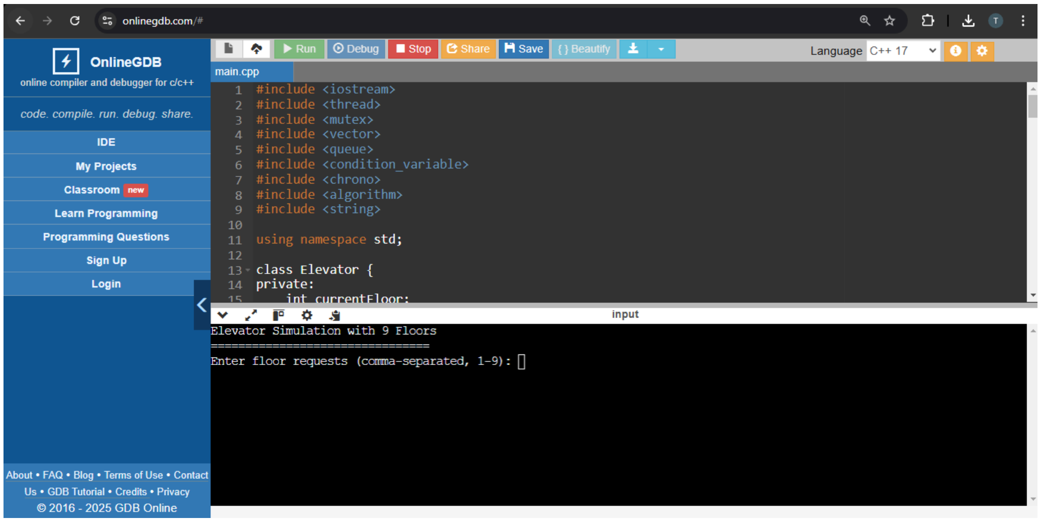
c. Once you copied the code go to onlinegdb.com



d. On the top left you will see Language with a drop down selection box, click on it and then choose C++ 17.



e. Simply paste the code on the working environment then run.



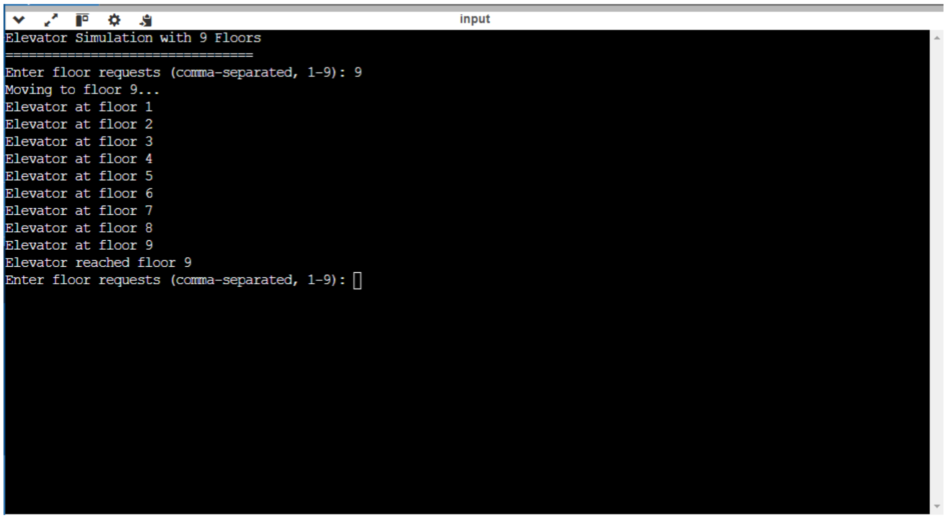
```cpp
#include <iostream>
#include <thread>
#include <mutex>
#include <vector>
#include <queue>
#include <condition_variable>
#include <chrono>
#include <algorithm>
#include <string>

using namespace std;

class Elevator {
private:
    int currentFloor;
```

```
Elevator Simulation with 9 Floors
================================
Enter floor requests (comma-separated, 1-9): 
```

B. Usage Instructions

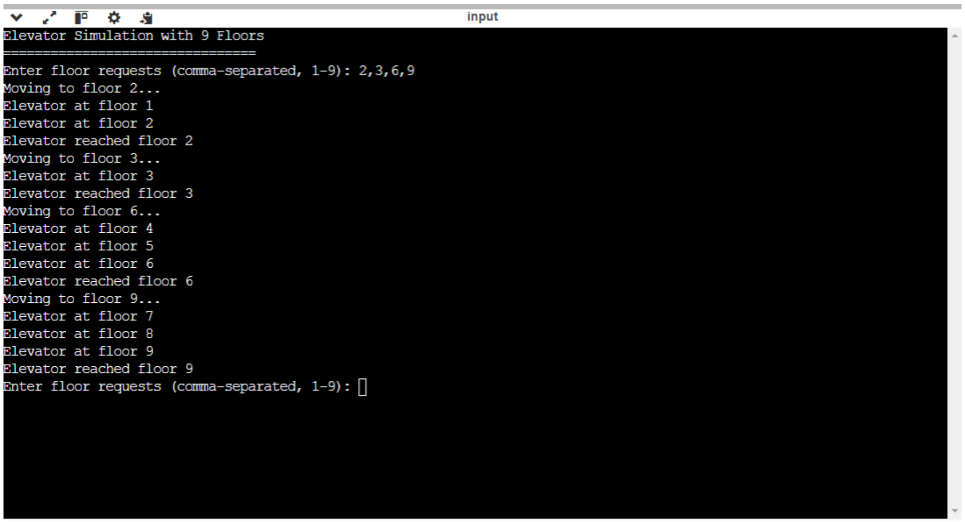
Once you run the program you can now use the program.

a. **Choosing a single floor:** when choosing a single floor just type in the number of the floor that you want to go to (only floors available 1-9).



```
Elevator Simulation with 9 Floors
================================
Enter floor requests (comma-separated, 1-9): 9
Moving to floor 9...
Elevator at floor 1
Elevator at floor 2
Elevator at floor 3
Elevator at floor 4
Elevator at floor 5
Elevator at floor 6
Elevator at floor 7
Elevator at floor 8
Elevator at floor 9
Elevator reached floor 9
Enter floor requests (comma-separated, 1-9): 
```

b. **Choosing multiple floors :** when choosing multiple floors just type in the numbers of the floors that you want to go to, it should be separated by comma sign (,)(only floors available 1-9).



```
Elevator Simulation with 9 Floors
================================
Enter floor requests (comma-separated, 1-9): 2,3,6,9
Moving to floor 2...
Elevator at floor 1
Elevator at floor 2
Elevator reached floor 2
Moving to floor 3...
Elevator at floor 3
Elevator reached floor 3
Moving to floor 6...
Elevator at floor 4
Elevator at floor 5
Elevator at floor 6
Elevator reached floor 6
Moving to floor 9...
Elevator at floor 7
Elevator at floor 8
Elevator at floor 9
Elevator reached floor 9
Enter floor requests (comma-separated, 1-9): 
```

X . Challenges and Solutions

The developers struggled with managing multiple users in an elevator system. Creating a program for just one user simplifies things and reduces synchronization issues. Future enhancements could concentrate on optimizing thread management and improving the system's ability to handle multiple users.

XI . Future Enhancements

a. UI and Visualization

Based on our user feedback and the developers' own decision that we can improve this area of our program. We can further improve the simulation presentation or visualization by adding a non text elements (eg. shapes or other things that can represent an elevator). Since we already finish a course about implementing visual graphics into C++ (eg. glut or glew).

b. Speed

We can further improve the speed of the program by changing the structure of our program or making it handle multiple threads greater. With this we improve the efficiency of the program.

c. Reliability

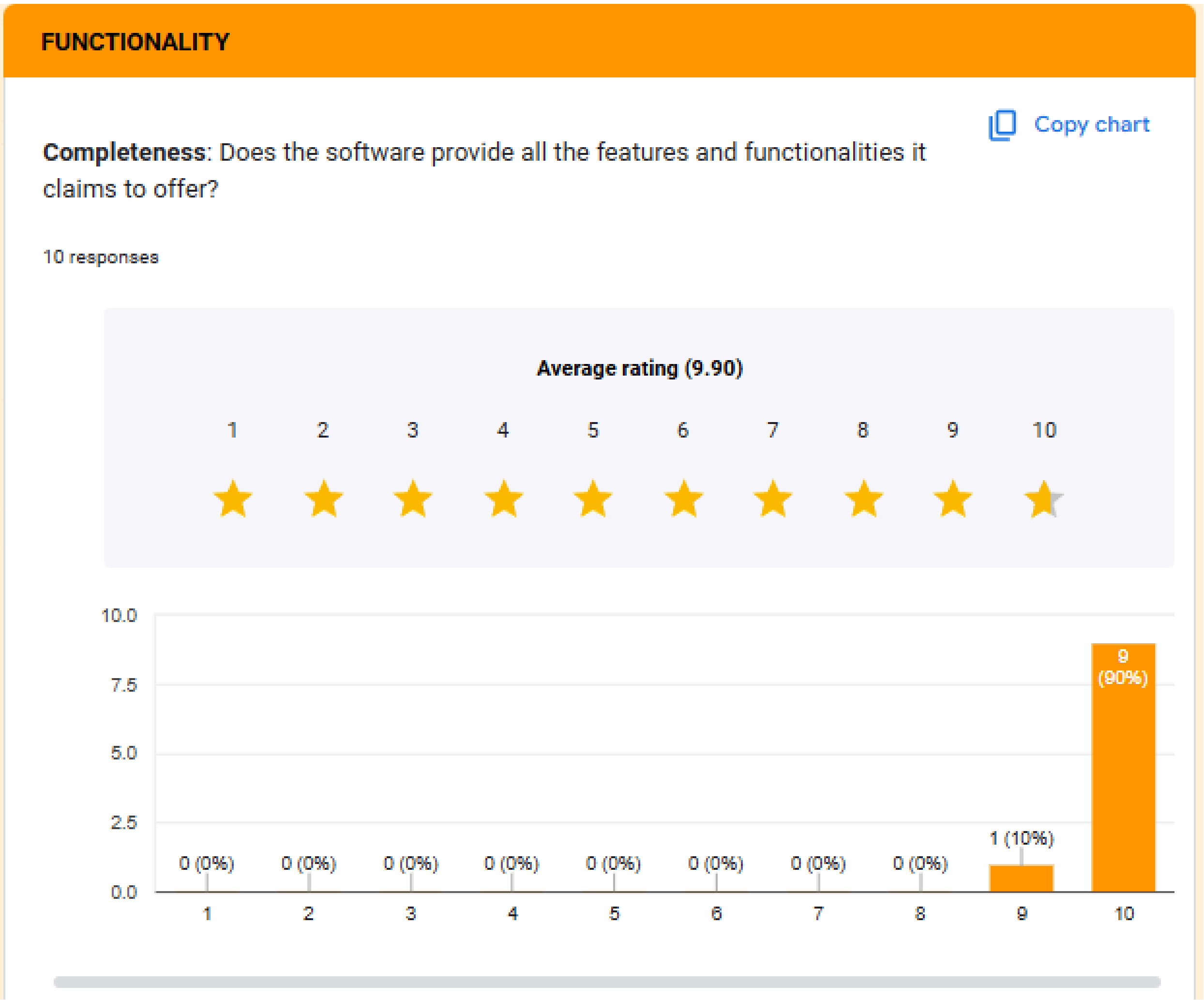
Because of our program inconsistency that sometimes it shows bugs to the user that are supposed to not happen during run time. It is the developers fault in their end, however this can further be improved by fixing the bugs and flaws of the program making it more consistent.

d. Accepts Multiple Users

In the future the developers want to implement a multi-user elevator system. With this it will be an accurate simulation of an elevator where there are different peoples in different floors and tracking how many people are in the elevator currently.

XII .Conclusion

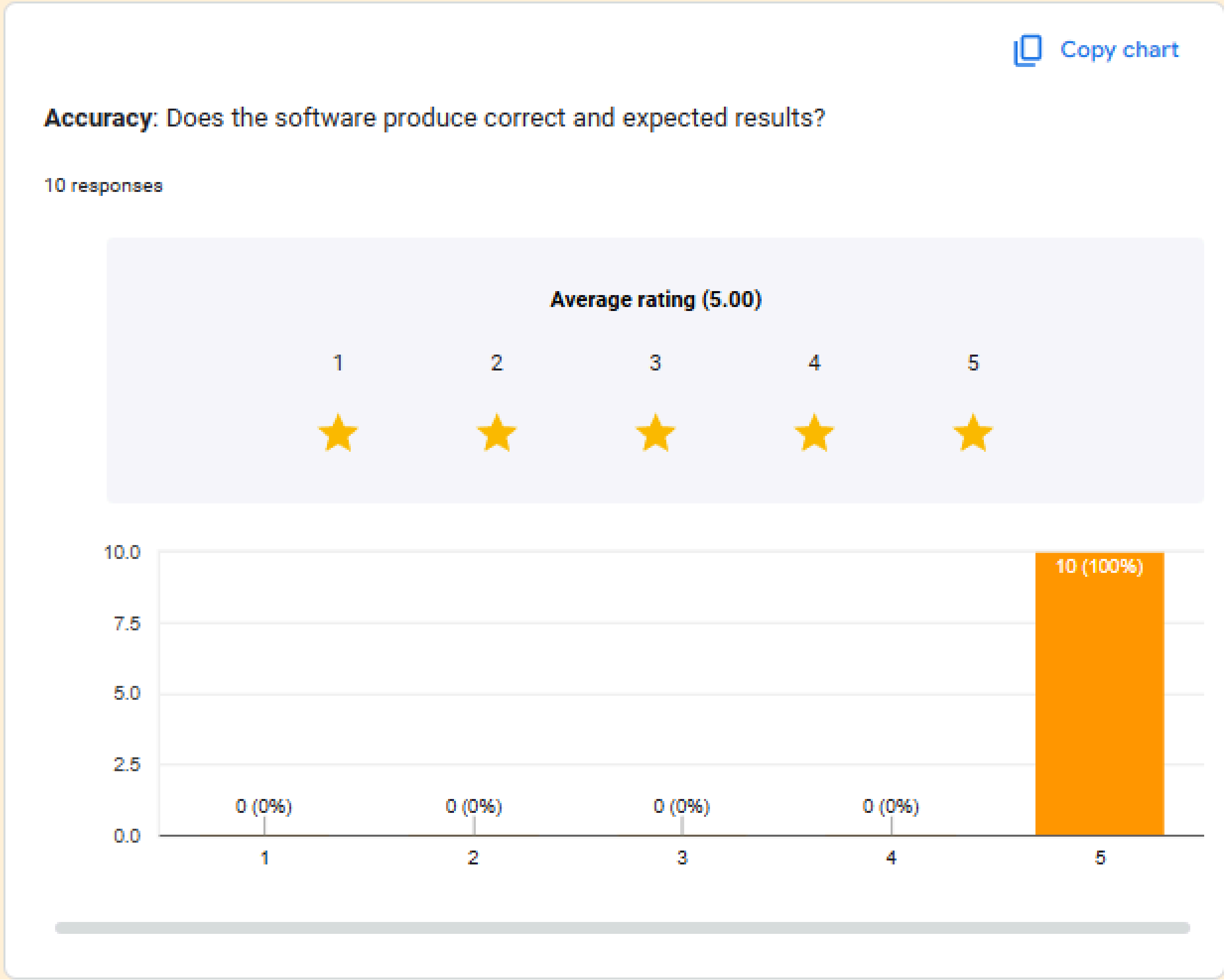
A. Summary



FUNCTIONALITY

Completeness: Does the software provide all the features and functionalities it claims to offer?

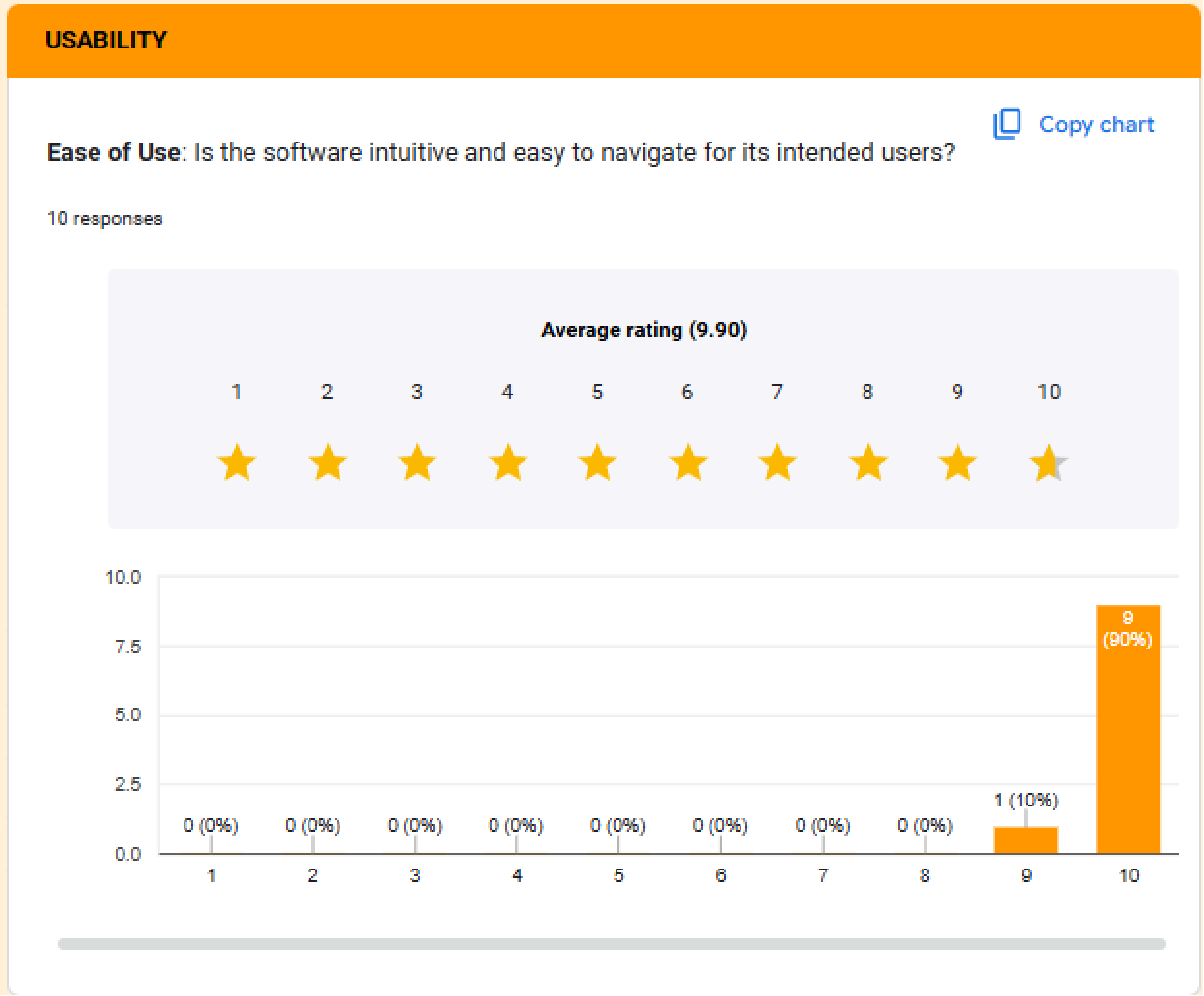
10 responses



Copy chart

Accuracy: Does the software produce correct and expected results?

10 responses

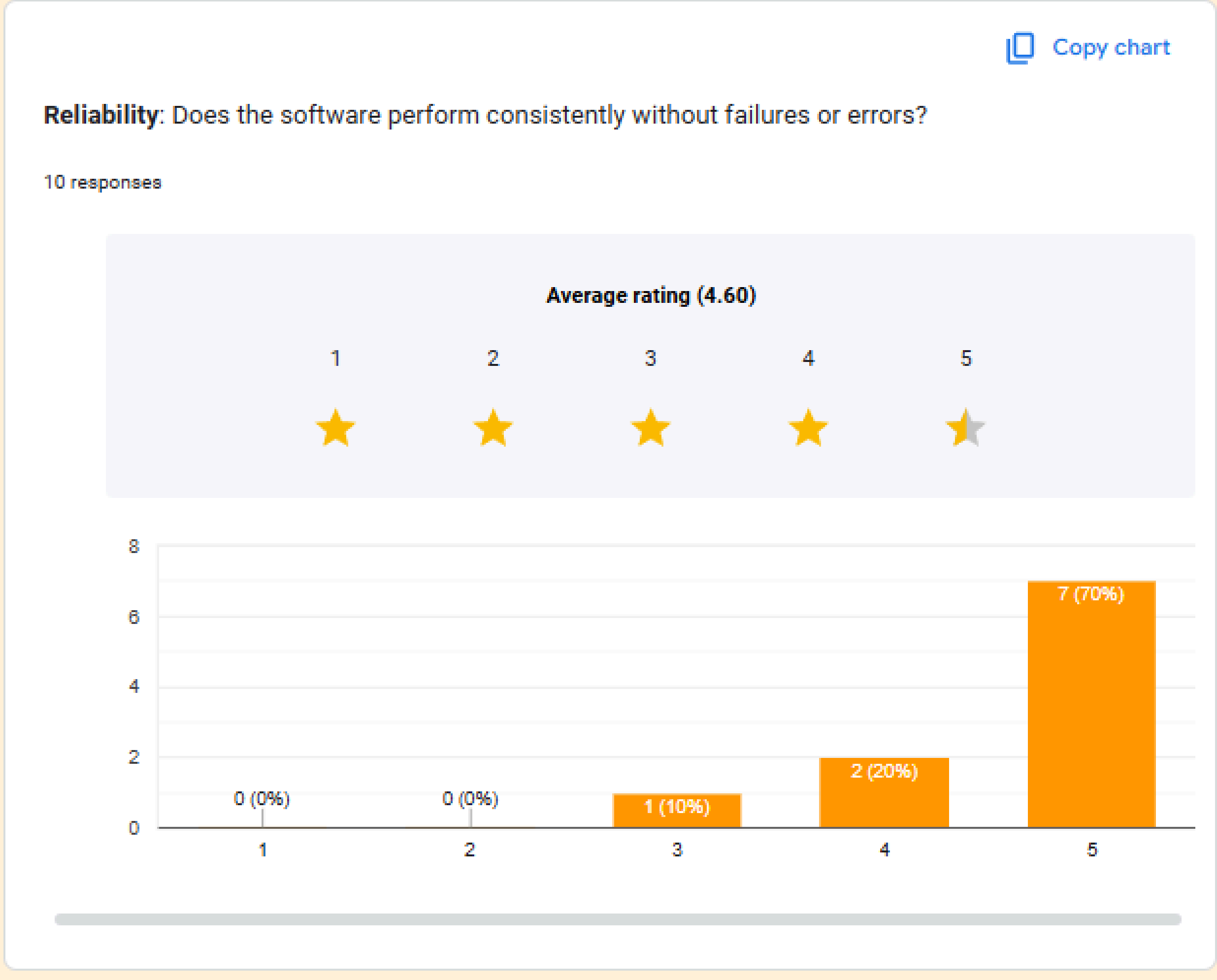


USABILITY

Ease of Use: Is the software intuitive and easy to navigate for its intended users?

10 responses

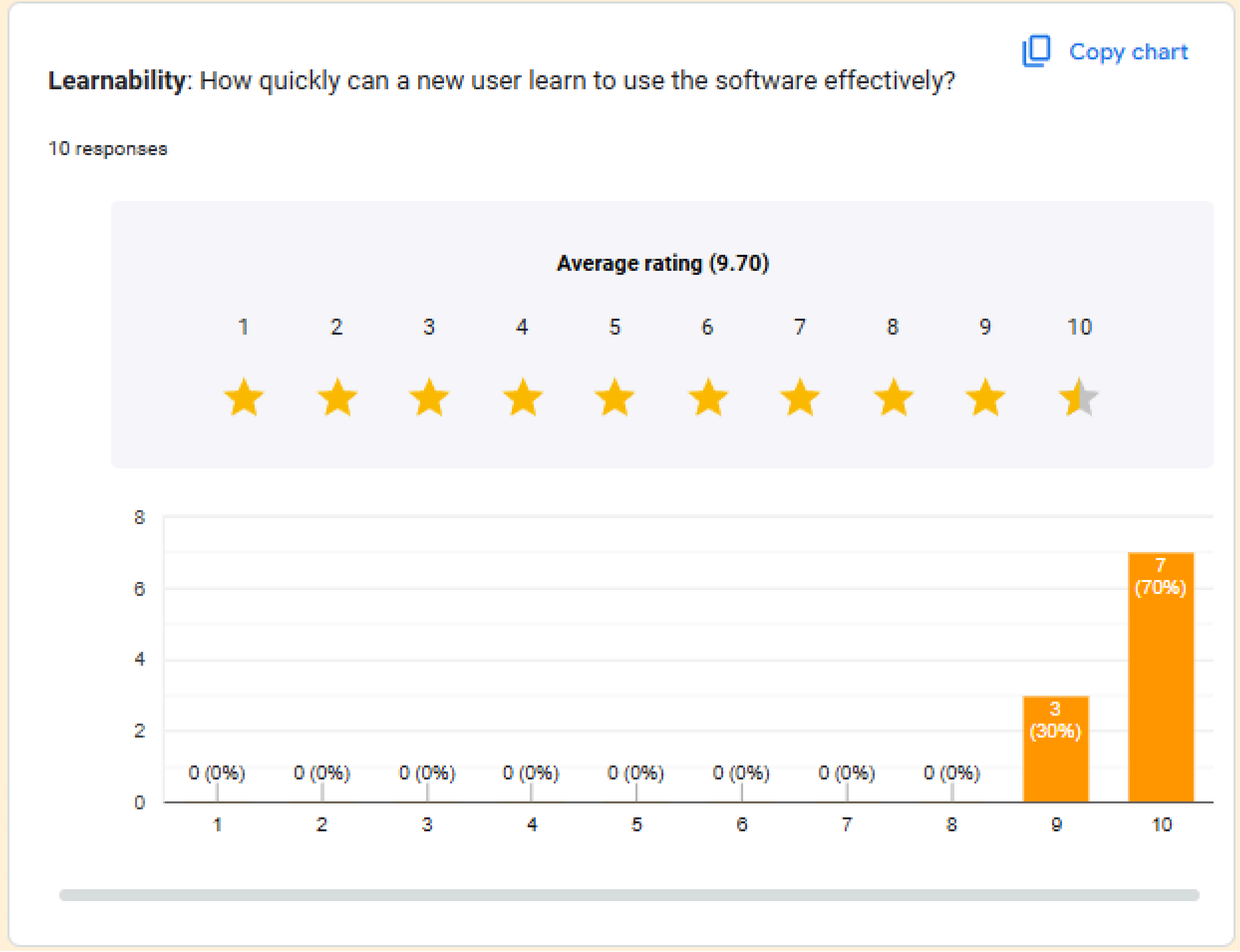
Average rating (9.90)



Reliability: Does the software perform consistently without failures or errors?

10 responses

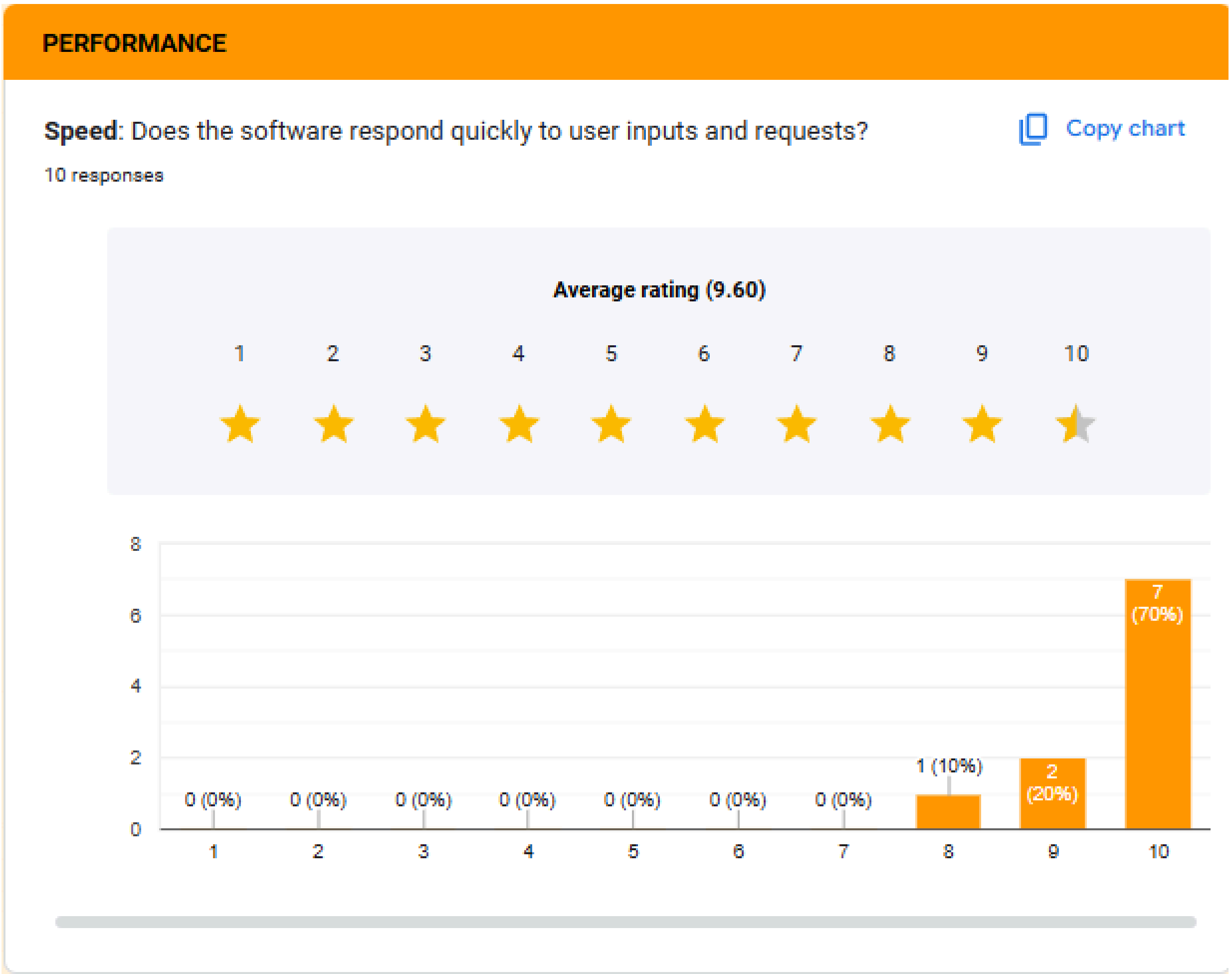
Average rating (4.60)



Learnability: How quickly can a new user learn to use the software effectively?

10 responses

Average rating (9.70)

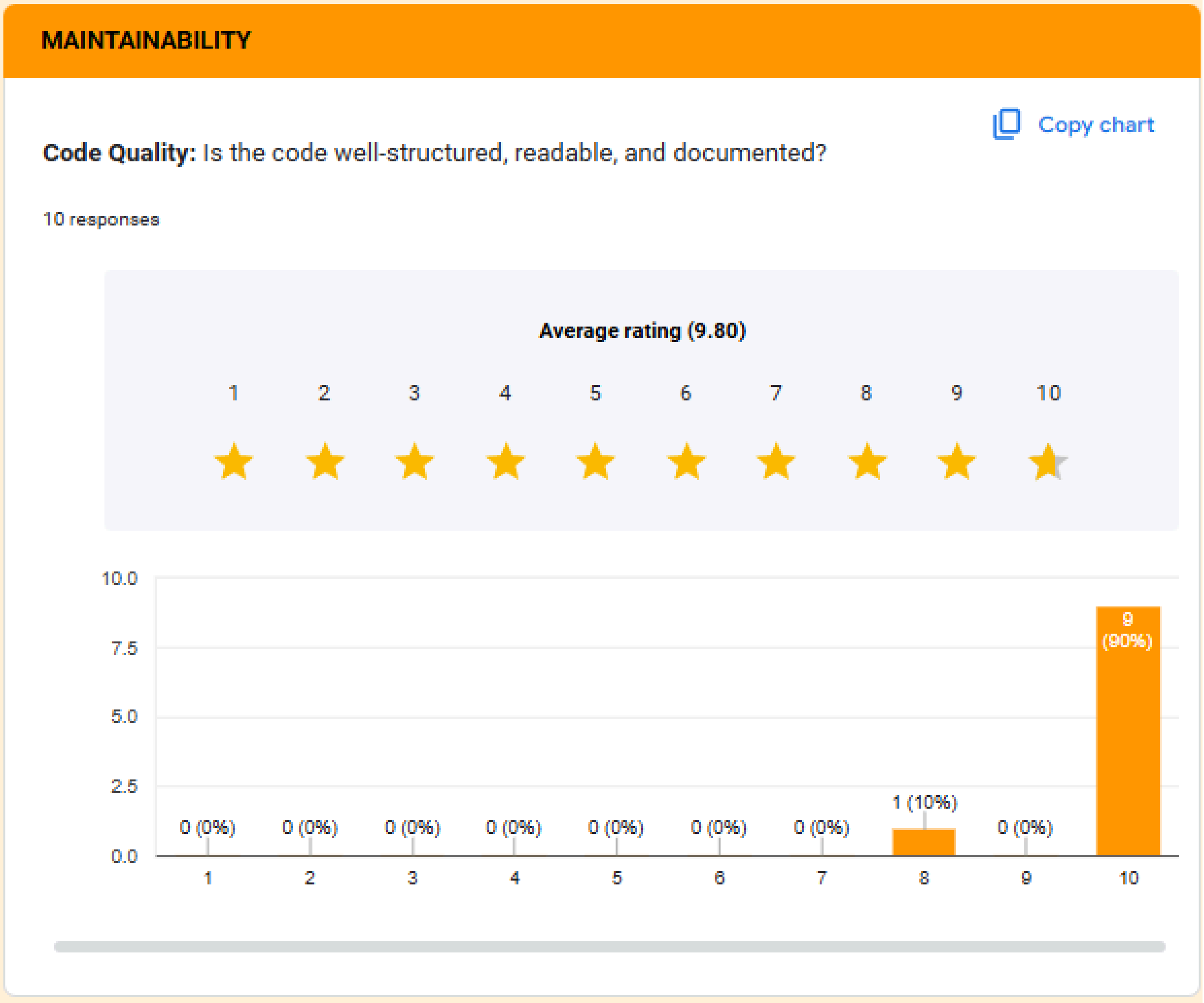


PERFORMANCE

Speed: Does the software respond quickly to user inputs and requests?

10 responses

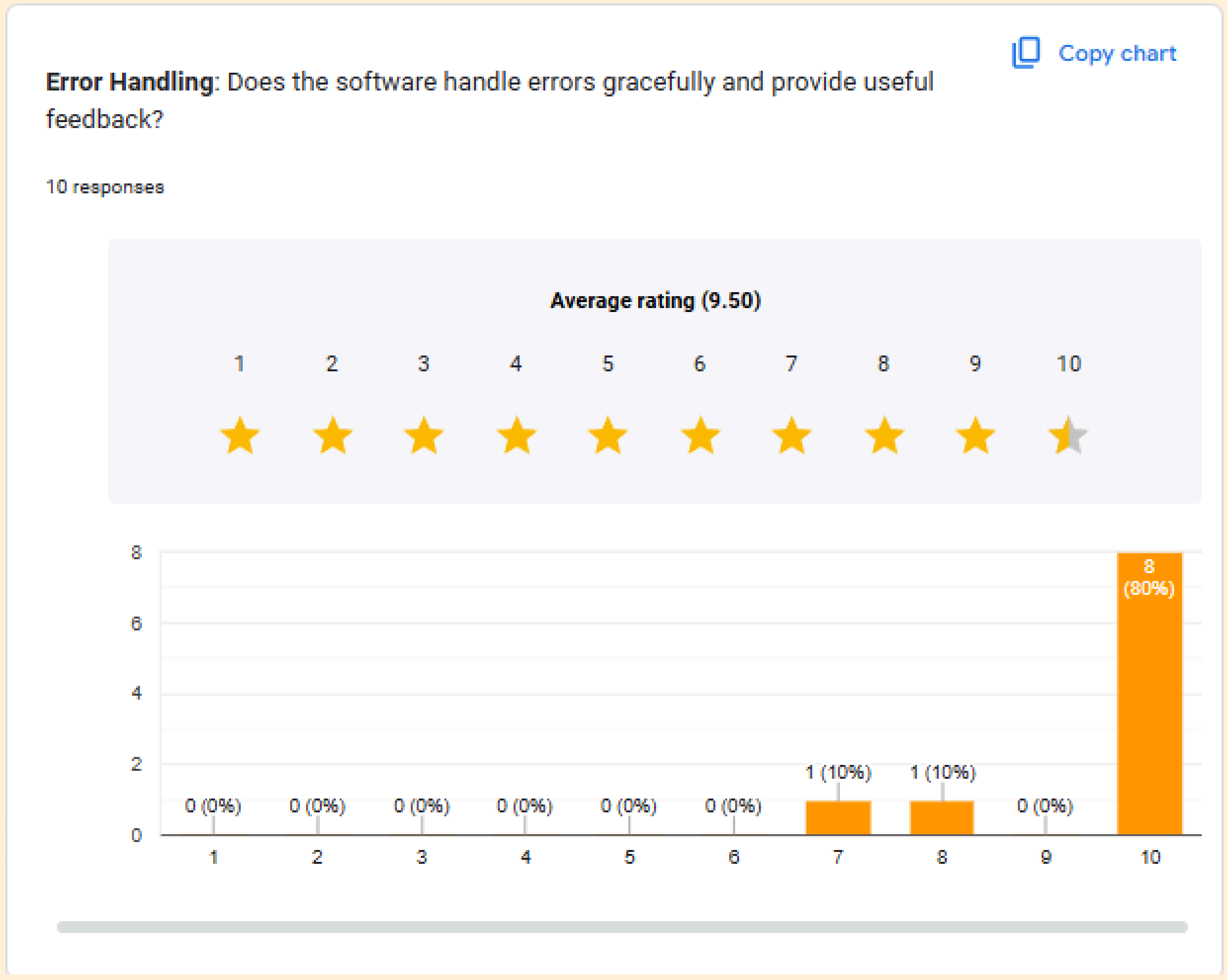
Average rating (9.60)



MAINTAINABILITY

Code Quality: Is the code well-structured, readable, and documented?

10 responses

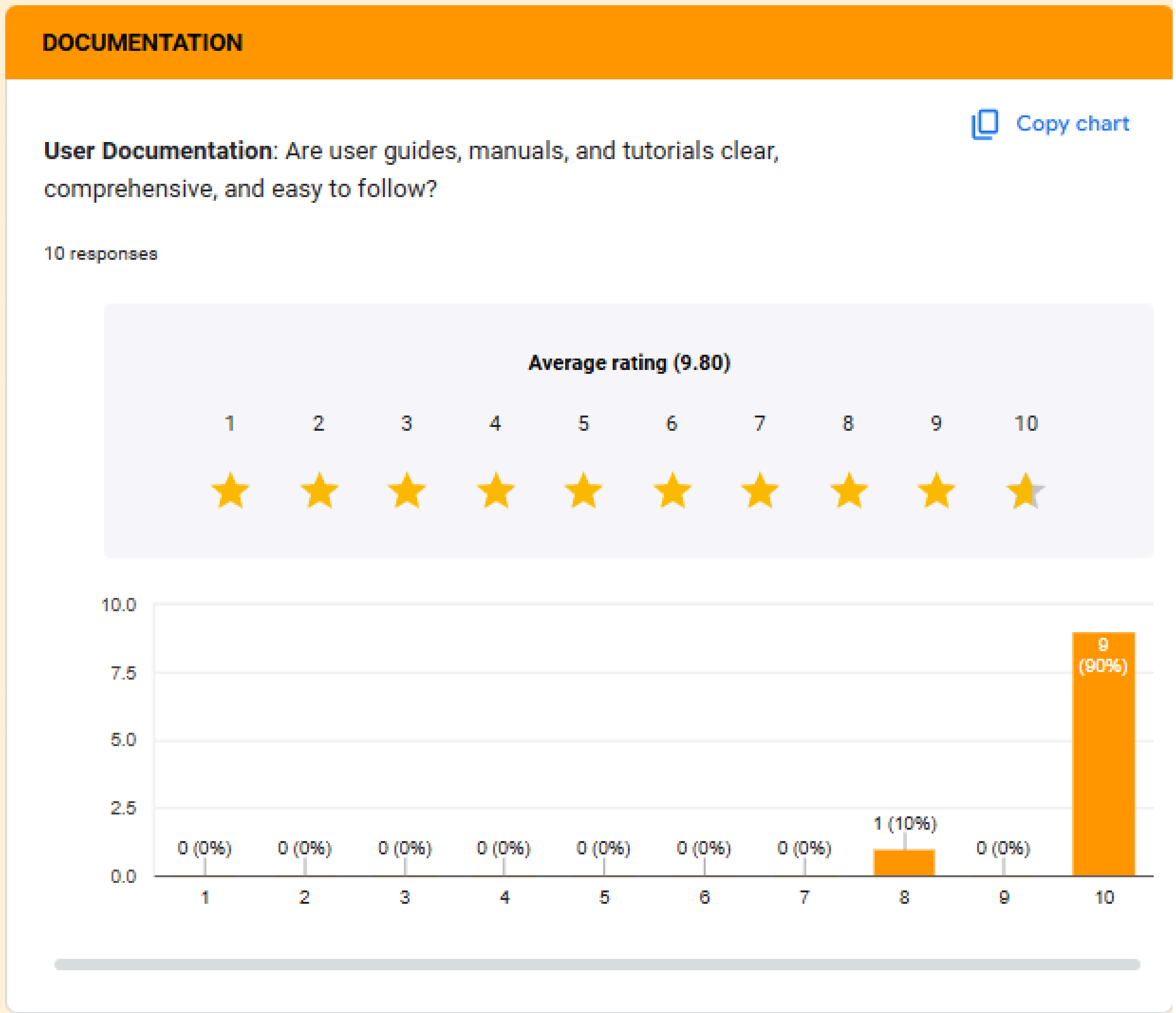


Copy chart

Error Handling: Does the software handle errors gracefully and provide useful feedback?

10 responses

Average rating (9.50)



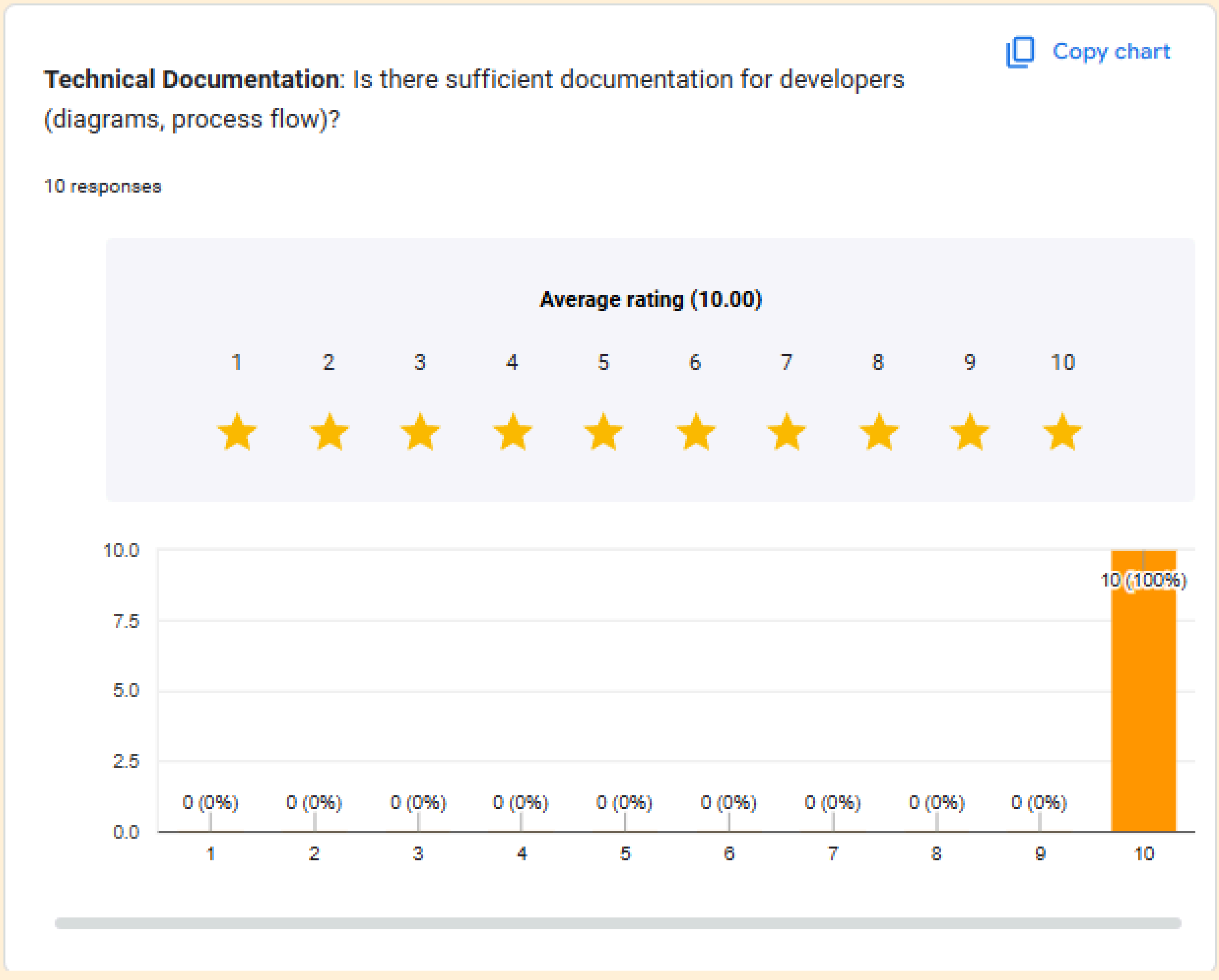
DOCUMENTATION

Copy chart

User Documentation: Are user guides, manuals, and tutorials clear, comprehensive, and easy to follow?

10 responses

Average rating (9.80)



Technical Documentation: Is there sufficient documentation for developers (diagrams, process flow)?

10 responses

B. Conclusion

The software receives very positive ratings in key areas, but a few areas for improvement to develop.

Although the technical documentation and user guides have received high praise, there may be areas for improvement in UI and visualisation, speed, and error handling.

Furthermore, while the reliability metric remains high, it could be enhanced by addressing any remaining edge cases or infrequent failures.

Overall, this program is of decent quality, user-friendly, and achieves its goals effectively.

XIII .References

Tejuco, H. (2025). *C++ Threads* [GitHub repository]. GitHub. https://github.com/TechTutorialHub/CS0051

XIV. Appendices

https://github.com/HabenTen/ElevatorProgramUsingThreads