Modelling and Design of a Finite State Machine Based Elevator Control System for Efficient Vertical Transportation

Shanmukha Reddy Anreddy, K. Sriram Charan, N. Naga Sai, Niharika Panda
Department of Computer Science and Engineering,
Amrita School of Computing, Bengaluru
Amrita Vishwa Vidyapeetham, India
bl.en.u4cse22105@bl.students.amrita.edu, bl.en.u4cse22123@bl.students.amrita.edu, bl.en.u4cse22135@bl.students.amrita.edu, p_niharika@blr.amrita.edu

***Abstract*—The elevator control system serves as a crucial element in modern infrastructures, facilitating the vertical movement of individuals within buildings efficiently and securely. Finite State Machines (FSMs) offer a method to model the behavior of elevators, capturing transitions between distinct states such as stationary, ascending, descending, and door operations. This paper presents the design of a simplified elevator control system employing FSM principles, accompanied by python code for intuitive visualization of the elevator's status and functionalities. The FSM model encompasses states including Stopped, Moving Up, Moving Down, Opening Doors, and Closing Doors, each characterized by specific transitions triggered by user commands or operational requirements. Through this approach, the system achieves robustness, responsiveness, and adaptability to varying usage scenarios, ensuring safe and efficient elevator operations in diverse building environment.**

Keywords - Elevator Control System, Finite State Machine, Python, Vertical Transportation, Object-Oriented Programming, Modular Design, User Experience, Efficiency, Safety, Adaptability.

I. Introduction

One of the most representative vertical transportation systems, typified by elevators, stand as the veins of modern buildings, nonchalantly transporting people from one floor to another and thereby making the functioning of skyscrapers feasible. Elevators are much more than just means of transport: they are profoundly fascinating items of engineering, loaded with commands destined for complex algorithms and controls. This project will penetrate into the core of elevator control system design, uncovering the intricate and multifaceted factors that are the ultimate flowchart of their operation. At the core of this process there are Finite State Machines that express subtle process difficult behaviors with their algorithmic approach. Elevator states' maze describes FSM as our compass, showing us the way through all this maze's complexity, going from stand to the elevator's upward and downward motions, until the harmony of door opening and closing are accomplished. Taking advantage of precise planning and thorough designing, we want to apply our solution which is based on unique patented technology that ensures the smooth and considerate overall experience for the users as well as compliance with energy efficiency, safety, and other applicable standards.

While in today's modern city scene the skyscrapers and complex buildings typically come with shiny birth and sprawling extension, the only thing that allows people to move between floors and reach their destination in a safety and comfort is the elevator, which is the link to the floors' basement. This project is based on the technical and systems principles of IT And it is going to support the ramified requirements for vertical traffic in the modern facilities. The development of the future elevator control system, which highly accurate, safe and progressive, is going to be prepared through a detailed study of elevator dynamics and behaviors, which we set out to lead the way in. Our dedication to the job is boundless, as through the process of thorough research, modeling, and simulation, we aim to erase any sort of mystery of the process of elevator operation and rather to show the way that this operation can bring a value both for the passengers and for all the safety and efficiency parameters popular among vertical transport systems.

FSMs become the building blocks of our design, providing a way to represent the complex stages of a elevator action through a simple modeling. Enclosing the different states and transitions of the elevators within the FSM structure we use the mighty FSMs in the control of the elevators to deal with intricate commune breaking situations. Every speed from being stationary to being in both positive and negative direction will be in the FSM and there are scheduled ways to achieve efficiency, user experience and safety. Intelligent vertical transportation is our leading partner in this quest. We strive to redefine the paradigms of high-tier transport system, where capabilities, resilience, and compatibility all coalesce into an experience of traveling to unimaginable heights. The great work proposed by the Chou and Rajamani et.al., presented the elevator vertical vibration model, which was based on the theoretical and experimental research. Their study helped to increase the understanding of how to provide dependable elevator systems [1].

The new proposed elevator control system incorporates new scheduling algorithms and optimal destination dispatching to improve efficacy and minimize passenger's waiting time. By embedding the Internet of Things solutions, it is possible to achieve anticipation of problems and control of performance to ensure greater availability. Furthermore, using speech recognition and touchless interfaces also increases accessibility and users' satisfaction and provides enhanced improvements in vertical transportation.

II. LITERATURE SURVEY

The identification of Gupta is concerned about the modeling of the elevator system particularly with vertical motion and lift velocities as well as numerical computation of free vibrations in the longitudinal dynamic analysis [2]. Ilcea et.al., pointed out the advantages of PLC and SCADA technologies in mining elevators for measurement and control, also, safety systems are crucial for the functioning of the elevators. They also discussed the factors of thread management and engine speed in their work [3]. The study by Atef Gharbi focused on elevator group control systems, discussing basic methods alongside with modern ones – genetic algorithms and simulated annealing [4]. The proposed work of Wu and Yang is an adaptive and energy-aware regulation strategy for elevator with traffic sensitivity as well as customized methods where necessary [5]. Examined new control algorithms for better exploitation of the mining elevator operation mechanisms via discrete event simulations was given by Kouvakas et al. (2023). Their research focuses on aspects such as safe and efficient improvements that result from parametric optimization [6].

The design of Ming et al., (2018) is on internet of things-based safety analysis system of elevator, which provided a real time fault detection and client-server communication to reduce vibration risk [7]. The work Yao et al., (2022) proposed an IoT-based elevator breakdown monitoring system that provides safety and reliability along with the economical services to handle failure [8]. It can be related as, Liu et al. (2023) applied big data technology to strategic management of elevators by analyzing potential safety issues thus increasing efficiency and reliability of the elevators [9]. As for the multiple level manufacturing, Koumboulis suggested the use of supervisors dedicated to the vertical and horizontal control, as it decreases the process correlation [10]. The theory of Pranave et.al., put forward a lexical-searching model for helping literature review authors with argumentative narrative writing difficulties for first-time authors. It revisits human activity recognition and API research, including the Insolvency and Bankruptcy Board of India for corporate law and resolution plans [11].

In the other study, FSMs are used in the pursuit of proposing a Super Mario game AI simulation and outlining the transitions of Mario agents' states in deepening game AI design [12]. The work proposed an FSM-based control system to address the behaviour of autonomous vehicles such as lane keeping, lane changing, and emergency braking. It improves driving safety by means of the concept of state hierarchy and transition rules at the node level of the car; thus, it is helpful in the development of AV systems [13]. An advance implementation by Nagarajan er.al., utilizes CFGs and the pyformlang library to identify the legitimacy of the English statements, suggesting a technique for statement preprocessing and parsing. It covers characteristics in NLP and mentions prior work in speech recognition using the W FSM [14]. The paper shows how blackhole attacks harm the efficiency of the LLNs in IoT by reducing performance through the Cooja simulator. Their research also stresses on stronger security strategies deployment for LLNs, and evaluating routing protocol objective functions to counter such risks [15].

III. METHODOLOGY

The proposed development plan of the elevator control system in Python will adopt an elaborate, yet a tuple and systematic approach aimed at employing object-oriented programming (OOP) principles and modularization design as shown in Fig.1. So that it can easy for the implementation and maintenance.

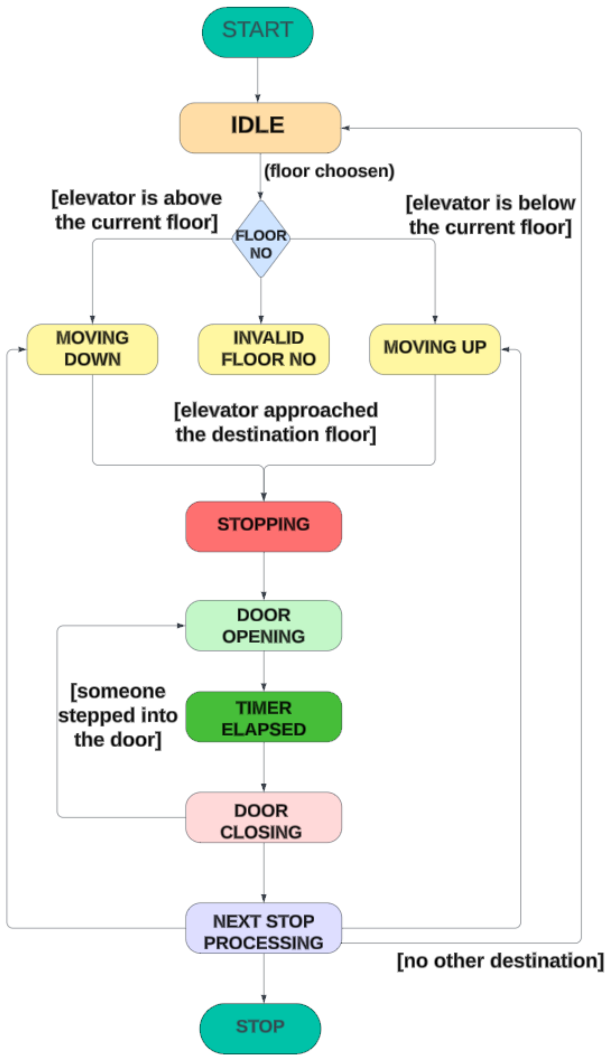


Fig.1. The proposed Methodology represented in the flow chart

To begin, we will define two main classes: A modal switch of Elevator and Elevator Control System have been created. The Elevator class will capture all the behavior and attributes of a particular elevator unit such as the current floor, movements direction (up, down), and lift doors status. The course will cover techniques for both the elevation and lowering of the elevator, stopping its motion, and also control of the doors.

Fig.2. tells about the Elevator Control System class will be a mastermind of elevator operations and order the movements requests to floors while arranging simultaneous actions of multiple elevators if available.

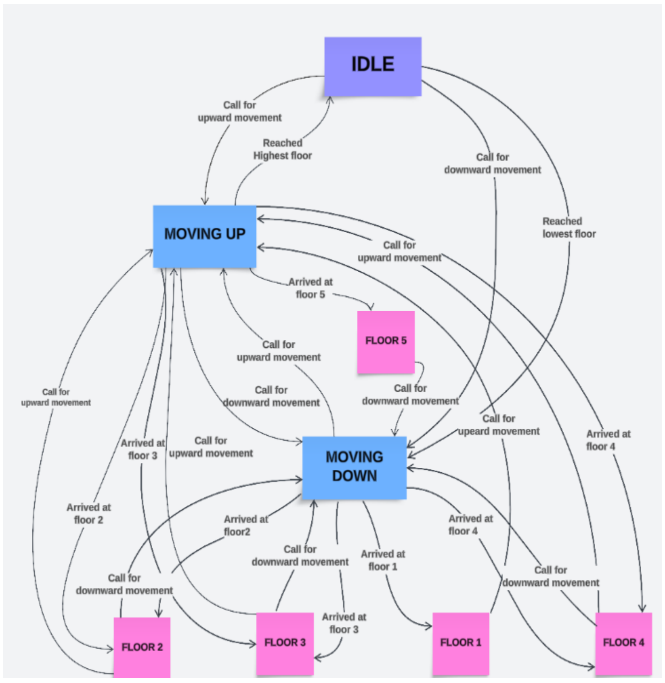


Fig.2. The class breakdown of the Elevator Control System which gives a complete class diagram

In class there will be an example of how the floor request processing method is implemented, whether the elevator is going up or down and how the elevator instance take up the tasks. Our entire process of development will be adhering to the Python best practices, so we will use PEP 8 style guides, meaningful variables and function names, and docstrings providing the clear understanding of what is being programmed. Further, we will do a validating using a full-fledged testing to prove the systems function and robustness to operate smoothly and true to itself under various eventualities and edge cases.

Using the principle of modularity and object orientation would aim at creating a system that is simple to scale and flexible; it can be extended for increasing the functionality or adding new features. With the help of intensive optimization and test runs, we will try to bring into reality a dependable and simple-to-operate solution- vertical movement in buildings which experiences speedy growth.

A. *States:*

Stopped: The elevator is stationary and waiting for a command.
Moving Up: The elevator is in motion and moving upwards to reach the requested floor.
Moving Down: The elevator is in motion and moving downwards to reach a requested floor.
Opening Doors: The elevator has reached the desired floor and is opening its doors.
Closing Doors: The elevator doors close after passengers have entered/exited.

B. *Elevator Transition Diagram:*

In the transition diagram let four floors as 4 states and actions will be up, down and start state will be first floor. Fig.3. gives us a detailed working of the Elevator transition through the different floors. We can see that all states are marked as final state because each individual floor acts like a final state. That means elevator is going to stop at any floor as per the request.

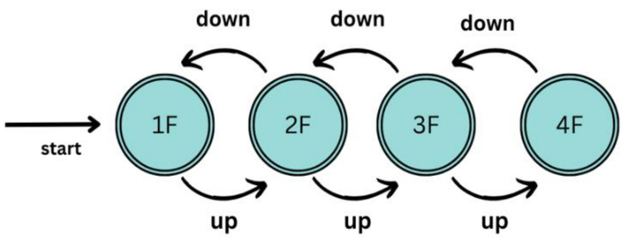


Fig.3. The working procedure of the Elevator Transition Diagram

C. *Algorithm:*

The algorithm for the proposed work goes in the following way with detailed explanation of all the classes and features.

a) Initialization

1. Initialize the Elevator class with:
    - current_floor = 0
    - direction = None
    - door_open = False
    - requested_floor = None
2. Initialize the ElevatorControlSystem class with:
    - An instance of Elevator.

b) Elevator Movement

1. **Move Up**:
    - Set direction = "up".
    - Increment current_floor by 1.
    - Display the message: "Moving Up, Floor <current_floor> ▲".

- If current_floor equals requested_floor:
    - Open doors.
    - Wait for 2 seconds.
    - Close doors.
- Otherwise, continue moving.

2. **Move Down**:
    - Set direction = "down".
    - Decrement current_floor by 1.
    - Display the message: "Moving Down, Floor <current_floor> ▼".
    - If current_floor equals requested_floor:
        - Open doors.
        - Wait for 2 seconds.
        - Close doors.
    - Otherwise, continue moving.
3. **Stop**:
    - Set direction = “stop”.
    - Display the message: "Stopping".
4. **Open Doors**:
    - Set door_open = True.
    - Display the message: "Opening Doors".
5. **Close Doors**:
    - Set door_open = False.
    - Display the message: "Closing Doors".

c) Floor Request Handling

1. **Request Floor**:
    - Set requested_floor in the elevator object.
    - If requested_floor is greater than current_floor:
        - Move up until current_floor equals requested_floor.
    - Else if requested_floor is less than current_floor:
        - Move down until current_floor equals requested_floor.
    - Else:
        - Open doors.
        - Wait for 2 seconds.
        - Close doors.
        - Display the message: "Elevator Arrived at Floor <requested_floor>".

d) Main Control Loop

1. **Main Function**:
    - Initialize an instance of ElevatorControlSystem.
    - Display the welcome message: "Welcome to the Elevator Control System".
    - Continuously prompt the user to enter a floor number or 'Exit' to quit:
        - If input is 'Exit', display "Exiting..." and terminate.
        - If input is a valid floor number (0 to 10), request the specified floor.
        - If input is invalid, display an error message.

D. *Transition Table:*

The transition table of the Elevator Control System as shown in the Table-I. The control of the elevator is done using a state transition table that regulates the working of the elevator under different states. Here is a brief overview of the states that need to be implemented: At the start, the state of the system will be Idle. After holding a floor notice, then it moves to Moving up/down, where it directs it to the floor that has requested for the service. In the case the elevator arrives on the requested level, the doors are opened. In all states, arriving at a new floor request changes the system to the according moving state or unlocks the door if it is already on the proper floor. The Stop state takes the system back to the Idle state so that various changes and actions carry out cohesively.

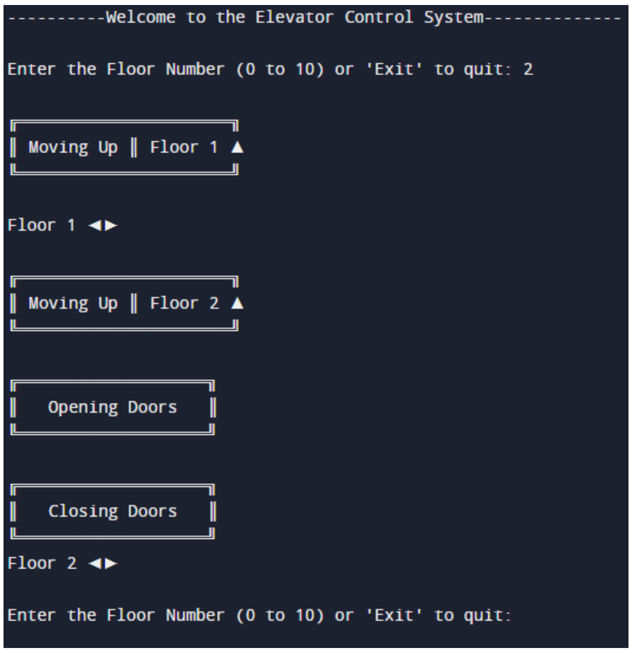
TABLE-I
TRANSITION TABLE OF THE ELEVATOR CONTROL SYSTEM

State	Input	Next State	Output
Idle	Request Floor	Moving Up/ Down	Elevator Moving Up/ Down to Floor X
Moving Up	-	Moving Up/ Stop	Elevator Moving Up to Floor X
Moving Down	-	Moving Down/ Stop	Elevator Moving Down to Floor X
Stop	-	Idle	Elevator Stopped
Any State	Request Floor	Same Floor/ Open Door	Elevator Arrived at Floor X/ Elevator Doors Opening
Any State	-	Idle	-

IV. Results And Discussion

The Python code implements an elevator control system with two classes: Elevator and Elevator Control System. The Elevator class manages the behavior of an individual elevator unit, including movement, door operations, and current status. The Elevator Control System class orchestrates the operation of the elevators, processing floor requests and directing elevator movement accordingly. The program prompts the user to input floor numbers, moves the elevator to the requested floor, and opens/closes the doors as necessary. The system ensures efficient and reliable vertical transportation within buildings, prioritizing safety and user experience.

This example demonstrates the elevator moving to floors 2 and 3 in response to user input. After reaching each floor, the elevator opens its doors, waits for a brief moment, and then closes the doors before proceeding. Finally, the program exits when the user types 'Exit'. Fig.4. tells us about the interface of the Elevator Control System with all the details in the command-line environment.

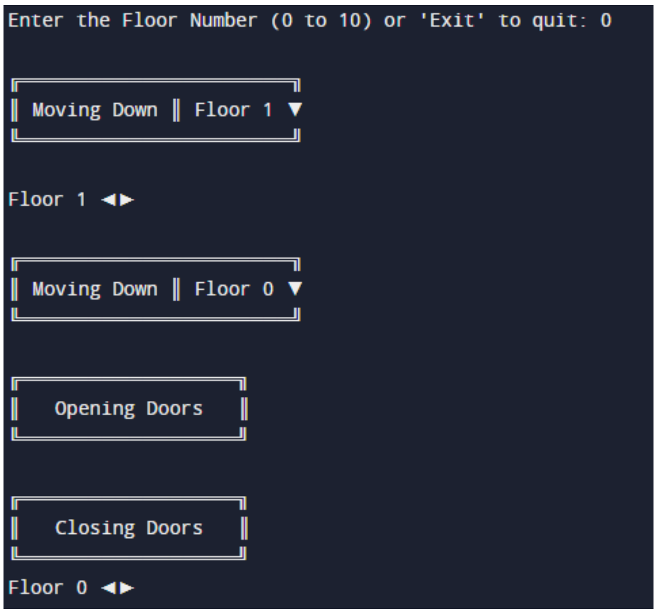


```
----------Welcome to the Elevator Control System-------------

Enter the Floor Number (0 to 10) or 'Exit' to quit: 2

╔════════════════════════╗
║ Moving Up ║ Floor 1 ▲
╚════════════════════════╝

Floor 1 ◄►

╔════════════════════════╗
║ Moving Up ║ Floor 2 ▲
╚════════════════════════╝

╔═════════════════════╗
║    Opening Doors    ║
╚═════════════════════╝

╔═════════════════════╗
║    Closing Doors    ║
╚═════════════════════╝
Floor 2 ◄►

Enter the Floor Number (0 to 10) or 'Exit' to quit:
```

Fig.4. The interface of the Elevator Control System case-I

This displaying the movement and doors of the elevator and how they move as the elevator goes from one floor to another. It contains dialog boxes for floor input, signals for moving up or down, and messages for the doors open and close state.

The Fig.5. demonstrates how a command-line based elevator control system works, showing how the elevator moves from the third floor down to the ground floor.

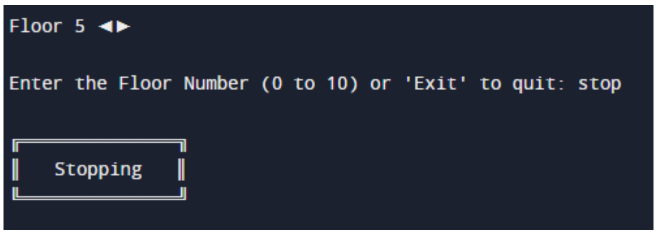


```
Enter the Floor Number (0 to 10) or 'Exit' to quit: 0

╔══════════════════════════╗
║ Moving Down ║ Floor 1 ▼
╚══════════════════════════╝

Floor 1 ◄►

╔══════════════════════════╗
║ Moving Down ║ Floor 0 ▼
╚══════════════════════════╝

╔═════════════════════╗
║    Opening Doors    ║
╚═════════════════════╝

╔═════════════════════╗
║    Closing Doors    ║
╚═════════════════════╝
Floor 0 ◄►
```

Fig.5. The command-line Elevator Control System Interface for case-I.

It offers users visual feedback as to the motion of the elevator with messages such as those indicating movement downwards alongside the current floor level. On reaching the ground floor, the system announces 'doors open' and 'doors close' depending on the ones involved. This visualization shows that the system is capable of interpreting user input of requests, stressing out the transition between states of an elevator and producing correct responses with the right amount of delay in a simulated environment.

Fig.6. shows the test case where the input is 'stop' and the elevator is stopping.

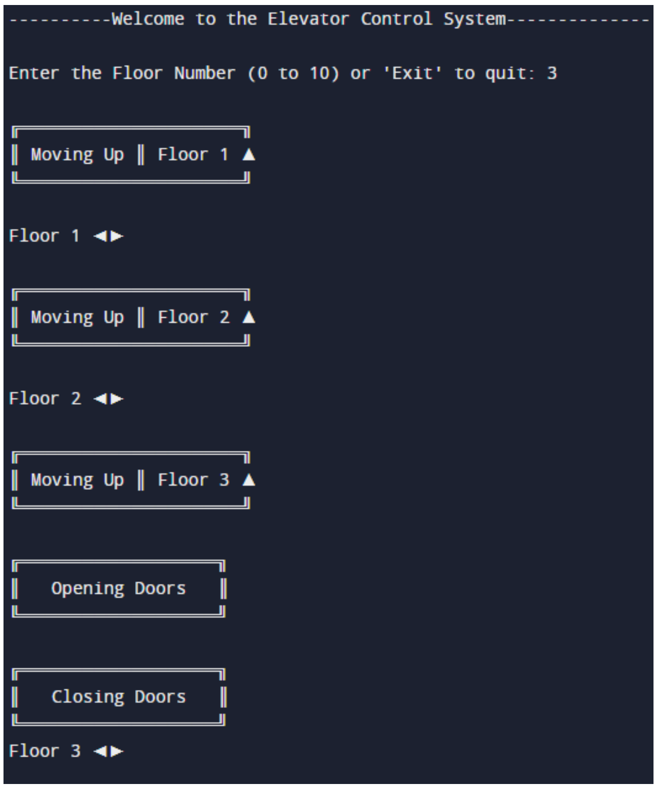


```
Floor 5 ◄►

Enter the Floor Number (0 to 10) or 'Exit' to quit: stop

╔════════════════╗
║    Stopping    ║
╚════════════════╝
```

Fig.6. The test case showing that the Elevator System is stopping

It acts like emergency stop. Where it is more useful whenever there occurs a interrupts like fire accident or power failure. That can be considered as safety matters more.

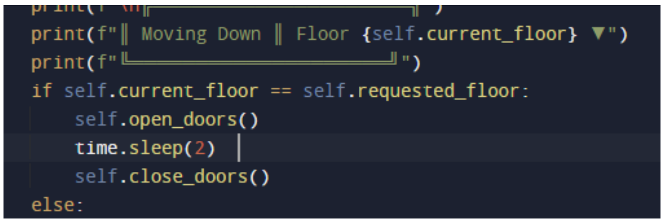
Fig.7. tells how the code works in command line interface of how the elevator control system works in the different floor number inputs.



```
----------Welcome to the Elevator Control System-------------

Enter the Floor Number (0 to 10) or 'Exit' to quit: 3

╔════════════════════════╗
║ Moving Up ║ Floor 1 ▲
╚════════════════════════╝

Floor 1 ◄►

╔════════════════════════╗
║ Moving Up ║ Floor 2 ▲
╚════════════════════════╝

Floor 2 ◄►

╔════════════════════════╗
║ Moving Up ║ Floor 3 ▲
╚════════════════════════╝

╔═════════════════════╗
║    Opening Doors    ║
╚═════════════════════╝

╔═════════════════════╗
║    Closing Doors    ║
╚═════════════════════╝
Floor 3 ◄►
```

Fig.7. The command-line Elevator Control System Interface for case-II.

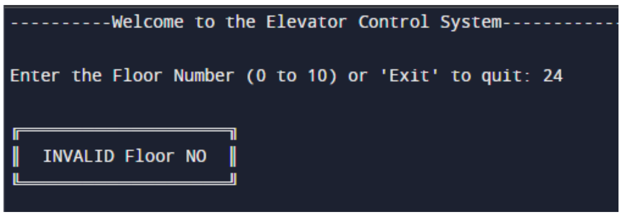
Fig.8. tells the time lapse between each function that will be implemented. We can infer that the time lapse is 2 seconds. Time lapse was involved in the code that stimulates the real life which can be related as someone stepped into the door.



```
print(f"║ Moving Down ║ Floor {self.current_floor} ▼")
print(f"╚══════════════════════════════╝")
if self.current_floor == self.requested_floor:
    self.open_doors()
    time.sleep(2)
    self.close_doors()
else:
```

Fig.8. The code snippet showing the time lapse involved between opening and closing of the elevator door.

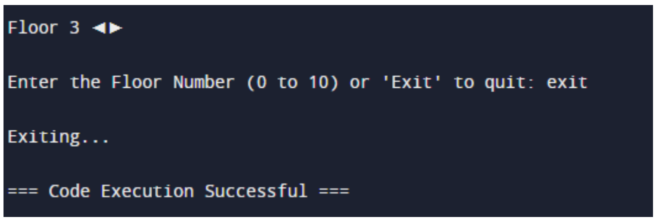
Fig.9. shows the test case that talks about the invalid floor number input because the input floor is limited and also if there is no inputted floor number is accessible.



```
----------Welcome to the Elevator Control System-----------

Enter the Floor Number (0 to 10) or 'Exit' to quit: 24

╔════════════════╗
║  INVALID Floor NO  ║
╚════════════════╝
```

Fig.9. The test case showing the Invalid floor number.

Fig.10. shows the command-line interface of how the elevator control system. Where the input is 'exit', so that it is getting exited from the code and also displays that code executed successfully.



```
Floor 3 ◄►

Enter the Floor Number (0 to 10) or 'Exit' to quit: exit

Exiting...

=== Code Execution Successful ===
```

Fig.10. The command-line Elevator Control System Interface for case-II

V. Conclusion And FutureScope

In conclusion, the developed elevator control system in Python effectively demonstrates the principles of object-oriented programming and modular design in addressing the complexities of vertical transportation within buildings. The system enables dependable and user-friendly coordination of elevator movement, floor request processing, and door operation management by utilizing the Elevator and Elevator Control System classes. The technology demonstrates efficiency, safety, and adaptability through iterative testing and development, highlighting its potential to improve the vertical mobility experience in contemporary infrastructures.

By Integrating sophisticated algorithms and IoT technology could boost productivity, reliability, and user experience. Further research and development could revolutionize vertical transportation, making it safer and more responsive to user needs. And also, in further we can be prosed by using Flip flops (D and JK) and the logic gates circuit.

VI. References

[1] Cho, Y. M., & Rajamani, R. (2001). Identification and experimental validation of a scalable elevator vertical dynamic model. Control Engineering Practice, 9(2), 181-187.

[2] Gupta, M. K. (2021). Simulation of vertical people transportation systems.

[3] Ilcea, G. I., Molnar, R. N., Pasculescu, D., Marioane, A. C., Lazar, T., Andreica, M., & Furdui, D. (2023, October). Device for Controlling the Mining Elevator Transportation Process. In International Conference Interdisciplinarity in Engineering (pp. 204-213). Cham: Springer Nature Switzerland.

[4] Gharbi, A. (2024). Exploring Heuristic and Optimization Approaches for Elevator Group Control Systems. Applied Sciences, 14(3), 995.

[5] Wu, Y., & Yang, J. (2024). Directional optimization of elevator scheduling algorithms in complex traffic patterns. Applied Soft Computing, 158, 111567.

[6] Kouvakas, N. D., Koumboulis, F. N., Fragkoulis, D. G., Tzamtzi, M. P., Panagiotakis, G. E., & Tsatsanias, A. (2023, September). A Reconfigurable Supervisory Control Algorithm for the Parametric Model of Multi-elevator Systems in Mines. In International Conference on Frontiers of Artificial Intelligence, Ethics, and Multidisciplinary Applications (pp. 215-227). Singapore: Springer Nature Singapore.

[7] Ming, Z., Han, S., Zhang, Z., & Xia, S. (2018). Elevator Safety Monitoring System Based on Internet of Things. International Journal of Online Engineering, 14(8)..

[8] Yao, W., Jagota, V., Kumar, R., Ather, D., Jain, V., Quraishi, S. J., & Osei-Owusu, J. (2022). Study and application of an elevator failure monitoring system based on the internet of things technology. Scientific Programming, 2022.

[9] Liu, P., Xiong, J., Yu, W., Cheng, H., & Wang, X. (2023). Research on Elevator Safety Detection Management based on Big Data. Advances in Engineering Technology Research, 5(1), 61-61.

[10] Koumboulis, F. N., Fragkoulis, D. G., & Michos, A. A. (2023). Modular supervisory control for multi-floor manufacturing processes. Control Theory and Technology, 21(2), 148-160.

[11] Pranave, K. C., Challa, V. K. R., & Panda, N. (2024). System Search Service Implementation Based on a Custom Lexical Search. Procedia Computer Science, 235, 1548-1557.

[12] Nambiar, A. S., Likhita, K., Sri Pujya, K. V. S., & Supriya, M. (2022). Design of Super Mario Game Using Finite State Machines. In Computer Networks and Inventive Communication Technologies: Proceedings of Fifth ICCNCT 2022 (pp. 739-752). Singapore: Springer Nature Singapore.

[13] Jaswanth, M., Narayana, N. K., Rahul, S., & Supriya, M. (2022, April). Autonomous car controller using behaviour planning based on finite state machine. In 2022 6th International Conference on Trends in Electronics and Informatics (ICOEI) (pp. 296-302). IEEE.

[14] Nagarajan, H., Vancha, P., & Supriya, M. (2022, July). Recognising the English language using context free grammar with pyformlang. In 2022 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT) (pp. 1-6). IEEE.

[15] Panda, N., & Supriya, M. (2022, October). Blackhole attack impact analysis on low power lossy networks. In 2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT) (pp. 1-5). IEEE.