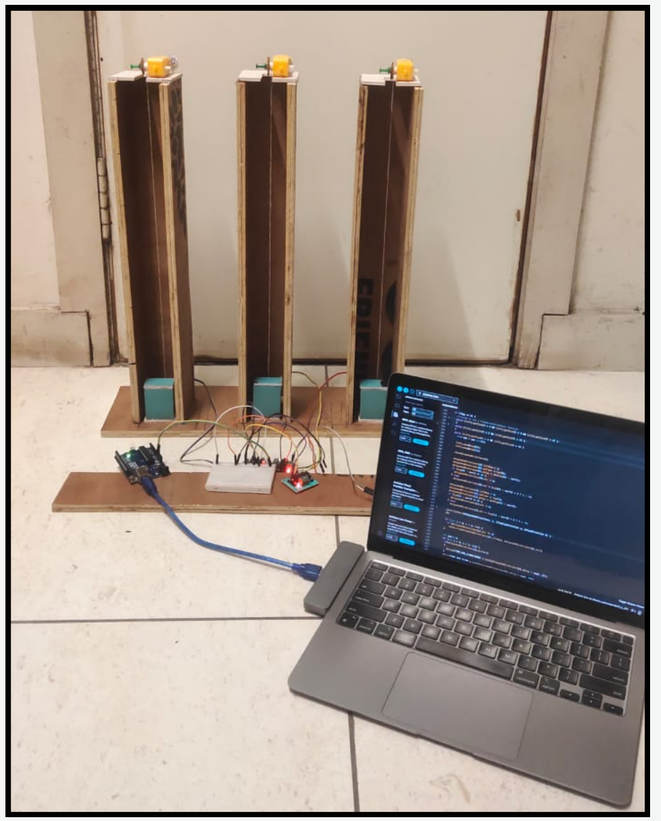
# ELEVATE: Enhancing Elevator Group Management Systems

CPG: 48

UEC 797: Capstone Project

CSED Department



**Team Mentor**

**Dr. Rajiv Kumar**

Associate Professor


**Jahnvi Gangwar**
Team Leader

**Mukul Singhal**
Team Member

**Veer Daksh Agarwal**
Team Member

**Shamayla Jindal**
Team Member

**Chahat Joneja**
Team Member

# Table of contents

# OVERVIEW

This proposal outlines an innovative approach to elevator systems in a world marked by towering skyscrapers and a fast-paced lifestyle, aiming to transform the urban experience.

The project's primary objective is to balance user experience optimization with environmental responsibility. Prioritizing reduced wait and travel times signifies a commitment to minimizing carbon footprints, thereby contributing to environmental preservation. This dual commitment underscores a dedication to sustainable goals, fostering a more environmentally conscious and responsible future.

# SCOPE AND UTILITY

The implemented algorithms intelligently respond to occupancy levels, user inputs, and traffic patterns, optimizing elevator scheduling and enhancing energy efficiency.The system's scalability and potential for future enhancements promise to usher in new possibilities, making vertical transportation an even more efficient and intelligent aspect of modern building management.

1. Predictive and Proactive Maintenance
2. Smart Building Integration- control, lighting, and security systems
3. Energy-saving analytics
4. User Feedback Mechanism
5. Accessibility features for visually impaired and handicapped users
6. Techniques like federation learning to increase security

# OBJECTIVES

- To understand the existing elevator group structure.
- To explore the best machine learning and deep models to optimize elevator group performance.
- To increase the system's efficiency by developing algorithms responding to occupancy levels, user inputs, and traffic patterns routinely used in elevator lobbies.
- To reduce the system's power consumption by incorporating various optimization algorithms.

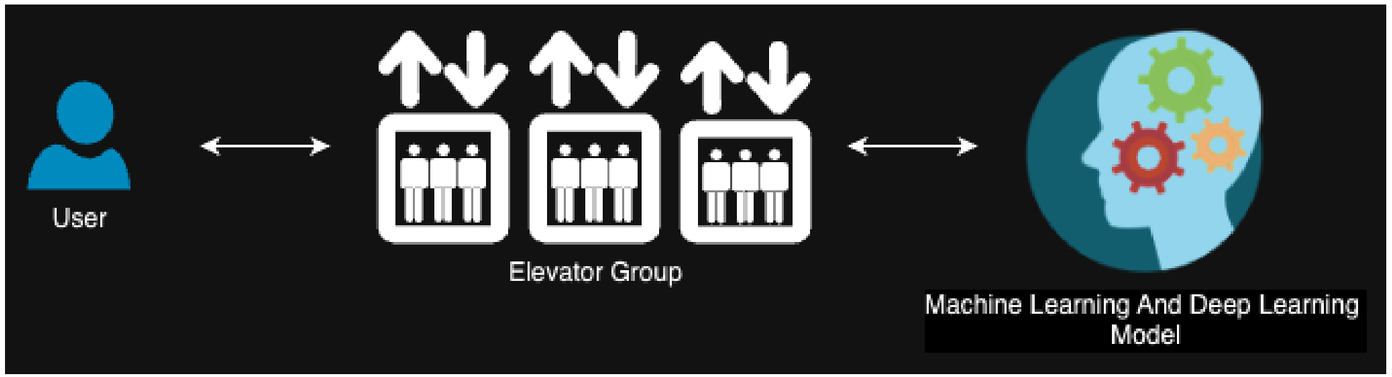
# LITERATURE SURVEY

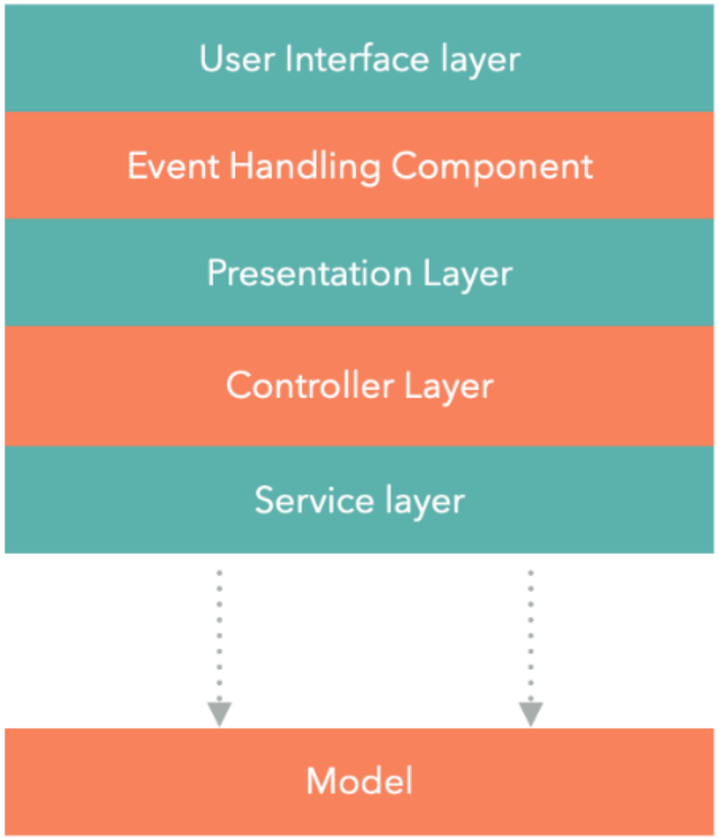
## Existing Smart Lifts

1. Regenerative Drive Systems (2000s): Captures and reuses energy during descent, enhancing elevator energy efficiency.
2. Mitsubishi Destination Control (2000s): Optimizes elevator routes by grouping passengers with similar destinations for efficient travel.
3. Otis Gen2 Elevator (2000s): Flat-belted steel hoisting system for smoother rides, reduced energy consumption, and space efficiency.
4. Schindler Ahead (2016): IoT-based predictive maintenance system enhancing elevator reliability and minimizing downtime.
5. Thyssenkrupp MULTI (2017): Ropeless elevator using magnetic levitation for vertical and horizontal movement, redefining building design.

# ARCHITECTURE USED

The project considers user interaction and requests and then processes the requests, allocating the best possible elevator based on the ML models for serving the request.



# TECHNIQUES AND TOOLS USED

## Tools

- Arduino UNO
- DC Motors
- L-293D

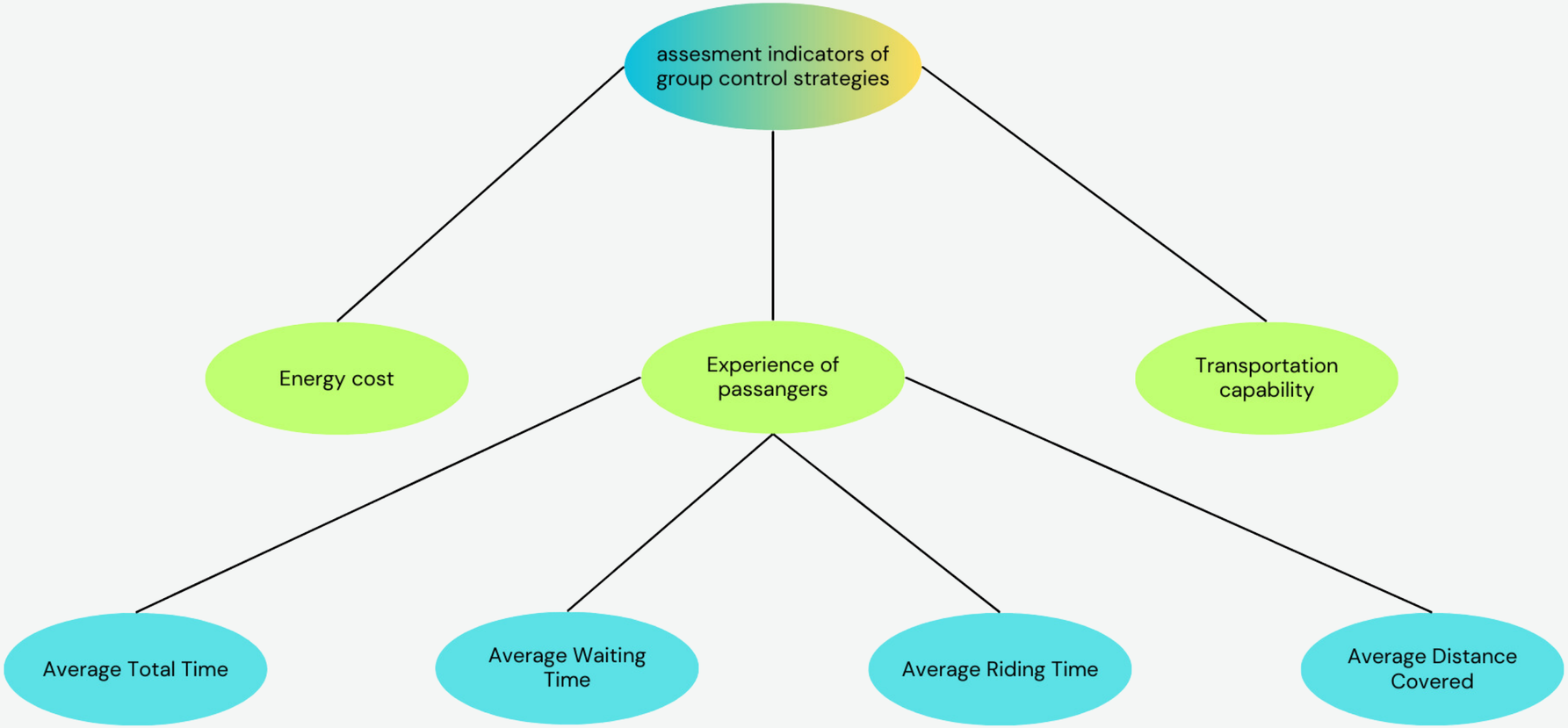
## Algorithms

- Zoning
- First Come, First Serve
- ACO algorithm
- Predictive Dispatching

## Software

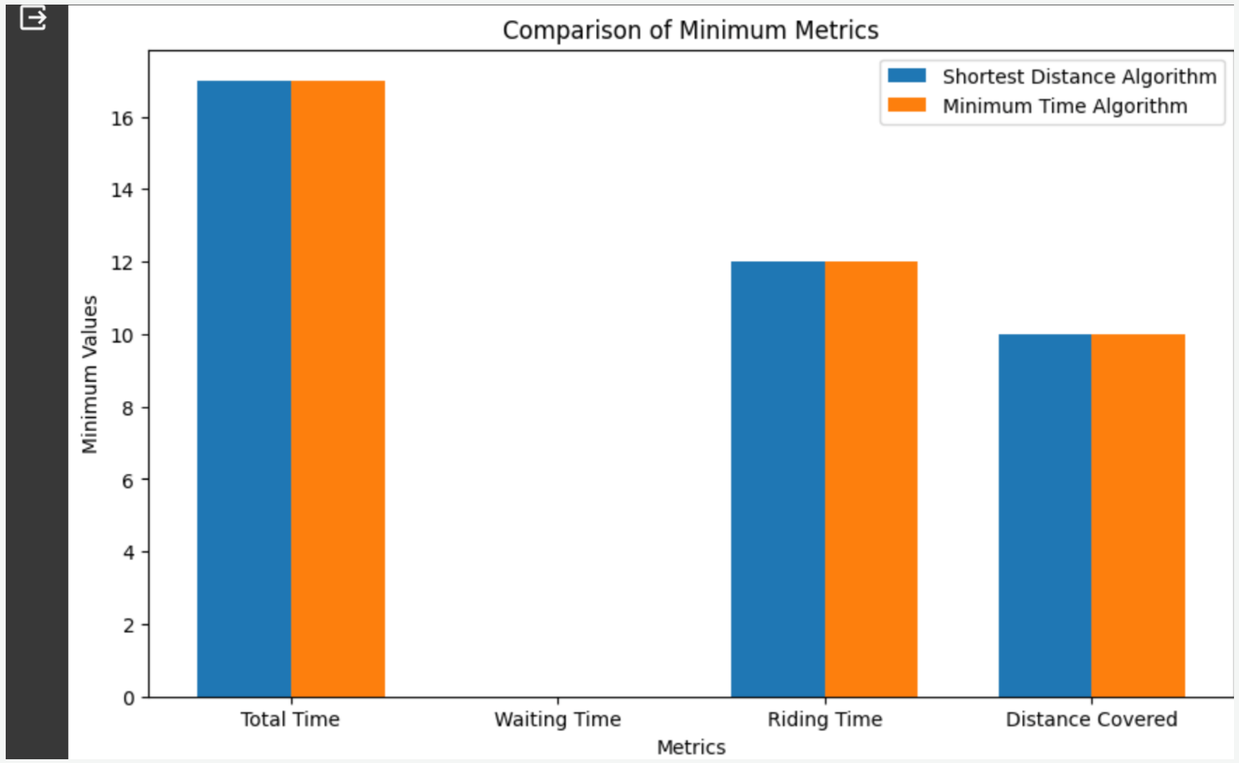
- Arduino IDE
- Google Colab
- Python

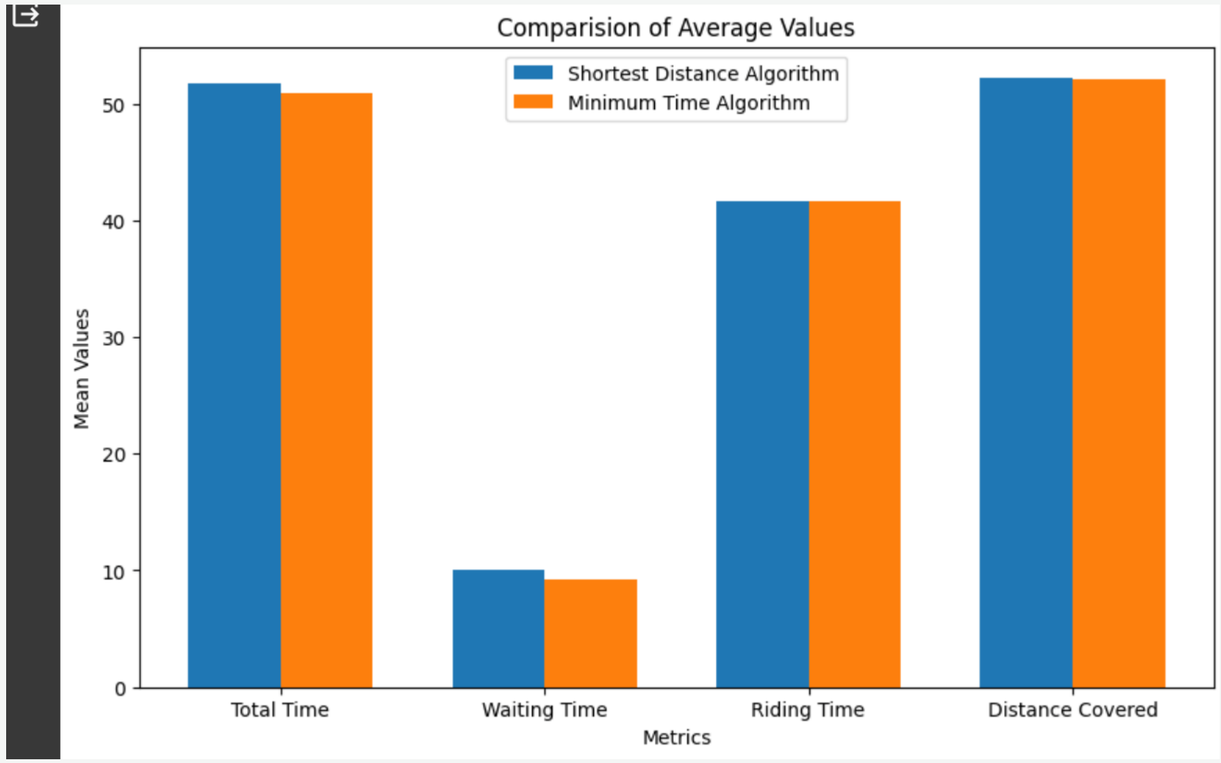
# Parameters to Work on



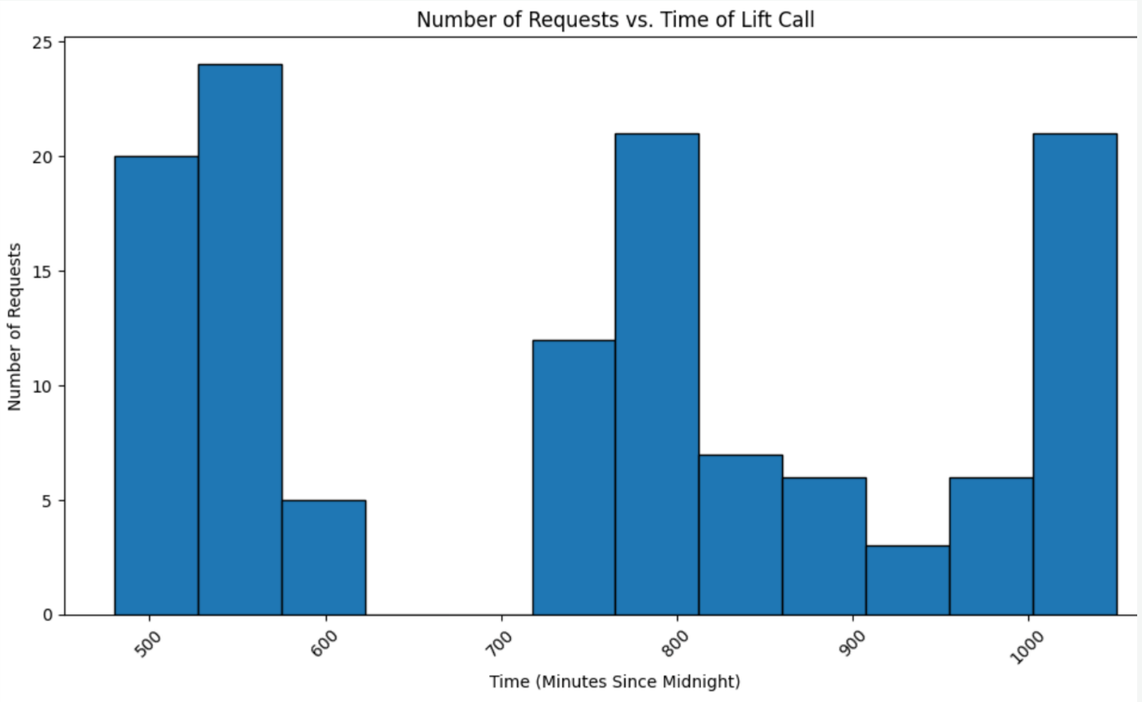
# SNAPSHOTS OF THE PROJECT

# Comparision of Metrics



# Peak Traffic Patterns




Assessment result of Mornign Peak mode:

| | Algorithms | Average Total Time | Average Waiting Time | Average Riding Time | Average distance covered | Total Stops |
|---|---|---|---|---|---|---|
| 0 | Shortest Distance | 48.500000 | 20.833333 | 27.666667 | 46.666667 | 15 |
| 1 | Minimum Time | 40.166667 | 12.500000 | 27.666667 | 38.333333 | 16 |

Assessment result of Morning Normal mode:

| | Algorithms | Average Total Time | Average Waiting Time | Average Riding Time | Average distance covered | Total Stops |
|---|---|---|---|---|---|---|
| 0 | Shortest Distance | 48.500000 | 20.833333 | 27.666667 | 46.666667 | 15 |
| 1 | Minimum Time | 40.166667 | 12.500000 | 27.666667 | 38.333333 | 16 |

Assessment result of Lunch mode:

| | Algorithms | Average Total Time | Average Waiting Time | Average Riding Time | Average distance covered | Total Stops |
|---|---|---|---|---|---|---|
| 0 | Shortest Distance | 49.314286 | 9.142857 | 40.171429 | 49.428571 | 95 |
| 1 | Minimum Time | 51.200000 | 11.028571 | 40.171429 | 51.714286 | 105 |

Assessment result of Evening mode:

| | Algorithms | Average Total Time | Average Waiting Time | Average Riding Time | Average distance covered | Total Stops |
|---|---|---|---|---|---|---|
| 0 | Shortest Distance | 49.285714 | 8.428571 | 40.857143 | 50.000000 | 54 |
| 1 | Minimum Time | 46.523810 | 5.666667 | 40.857143 | 48.095238 | 54 |

Assessment result of After Hours mode:

| | Algorithms | Average Total Time | Average Waiting Time | Average Riding Time | Average distance covered | Total Stops |
|---|---|---|---|---|---|---|
| 0 | Shortest Distance | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| 1 | Minimum Time | 0.0 | 0.0 | 0.0 | 0.0 | 0 |

# Zone Control

| Zoning Method Criteria | Zone 1 | Zone 2 | Zone 3 |
|---|---|---|---|
| Numerical range of requested floors | Floors 1 to 3 | Floors 4 to 6 | Floors above 6 |
| Average waiting time for requested floor | Avg. waiting time <= 5 sec | 5 sec < Avg. waiting time <= 10 sec | Avg. waiting time > 10 sec |
| Number of occupants on a floor | 0-2 occupants | 2-5 occupants | More than 5 occupants |
| Number of stops made by elevators | 0-2 stops | 2-4 stops | More than 4 stops |
| Joint criteria of waiting time and stops | Avg. waiting time <= 5 sec and Avg. stops <= 2 | 5 sec < Avg. waiting time <= 10 sec and 2 < Avg. stops <= 4 | Avg. waiting time > 10 sec or Avg. stops > 4 |

# Predictive Dispatching

```
[156] rfff = df['Requested From Floor'].value_counts().sort_index()
      rtff = df['Requested To Floor'].value_counts().sort_index()
```

```
Elevator 1 is moving to floor 3
Sent elevator 1 to floor 3 with 17 requests.
Elevator 1 is moving to floor 8
Sent elevator 1 to floor 8 with 17 requests.
Elevator 1 is moving to floor 6
Sent elevator 1 to floor 6 with 17 requests.
Elevator 1 is moving to floor 1
Sent elevator 1 to floor 1 with 14 requests.
Elevator 1 is moving to floor 2
Sent elevator 1 to floor 2 with 14 requests.
Elevator 1 is moving to floor 4
Sent elevator 1 to floor 4 with 13 requests.
Elevator 1 is moving to floor 7
Sent elevator 1 to floor 7 with 11 requests.
Elevator 1 is moving to floor 9
Sent elevator 1 to floor 9 with 11 requests.
Elevator 1 is moving to floor 5
Sent elevator 1 to floor 5 with 11 requests.
<ipython-input-156-8243772315c1>:6: FutureWarning: iteritems is deprecated and will be removed in a future version. Use .items instead.
  for floor, count in floor_requests.iteritems():
```

# Ant Colony Optimisation

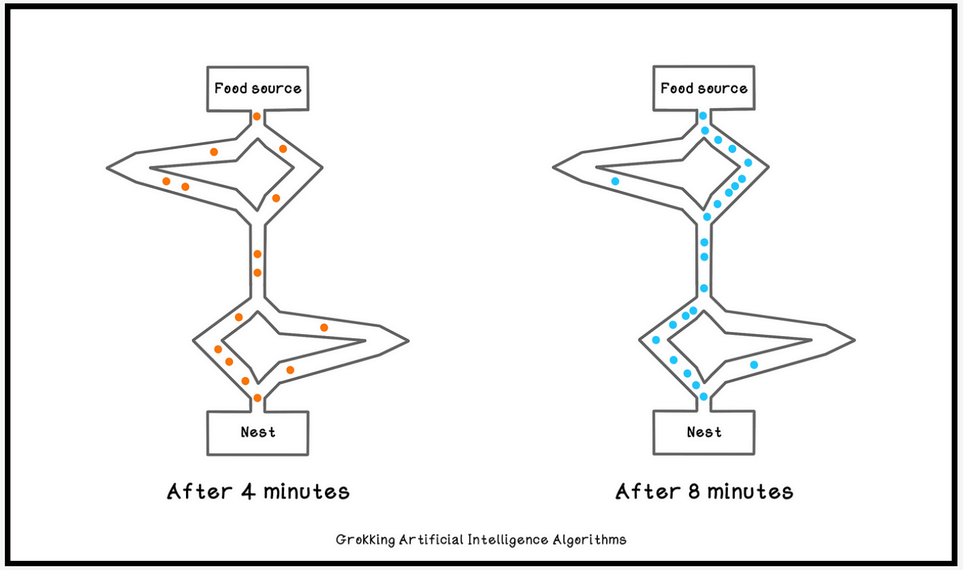
```
 AntColonyOptimization:
ef __init__(self, elevator_system, requests, iterations=10, evaporation_rate=0.5, alpha=1.0, beta=2.0):
   self.elevator_system = elevator_system
   self.requests = requests
   self.iterations = iterations
   self.evaporation_rate = evaporation_rate
   self.alpha = alpha
   self.beta = beta

ef run(self):
   for _ in range(self.iterations):
       for request in self.requests:
           floor, requesting_elevator = request
           self.elevator_system[requesting_elevator].request_floor(floor)

       for elevator in self.elevator_system.values():
           elevator.operate()

       # Update pheromones based on the efficiency of each elevator
       total_waiting_time = sum(elevator.current_floor for elevator in self.elevator_system.values())
       pheromones = np.ones((len(self.elevator_system), len(self.elevator_system)))
       for elevator_id, elevator in self.elevator_system.items():
           efficiency = 1 / (elevator.current_floor + 1)
           pheromones[:, elevator_id - 1] *= (1 - self.evaporation_rate)
           pheromones[:, elevator_id - 1] += (efficiency + 1) * self.alpha

       # Select the elevator with the highest pheromone level for each request
       for request in self.requests:
           floor, requesting_elevator = request
           probabilities = pheromones[:, requesting_elevator - 1]  # Adjust indexing
           selected_elevator = np.argmax(probabilities)
           self.elevator_system[selected_elevator + 1].request_floor(floor)  # Adjust indexing
```



# BalanceTrade Off

Assessment result based on tradeoff:

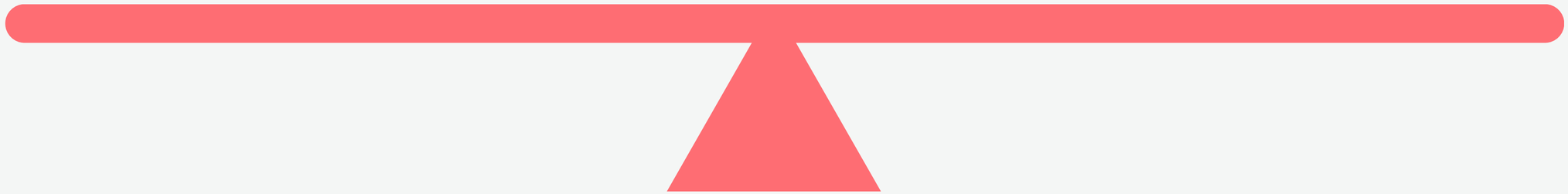
---

| | Algorithms | Average Total Time | Average Waiting Time | Average Riding Time | Average distance covered |
|---|---|---|---|---|---|
| 0 | Shortest Distance | 51.968 | 10.320 | 41.648 | 52.24 |
| 1 | Minimum Time | 50.872 | 9.224 | 41.648 | 51.76 |
| 2 | General Tradeoff | 51.968 | 10.320 | 41.648 | 51.60 |
| 3 | Tradeoff based passsanger flow | 52.280 | 10.632 | 41.648 | 53.20 |

**Shortest distance optimisation**
**vs**
**Minimum wait time optimisation**

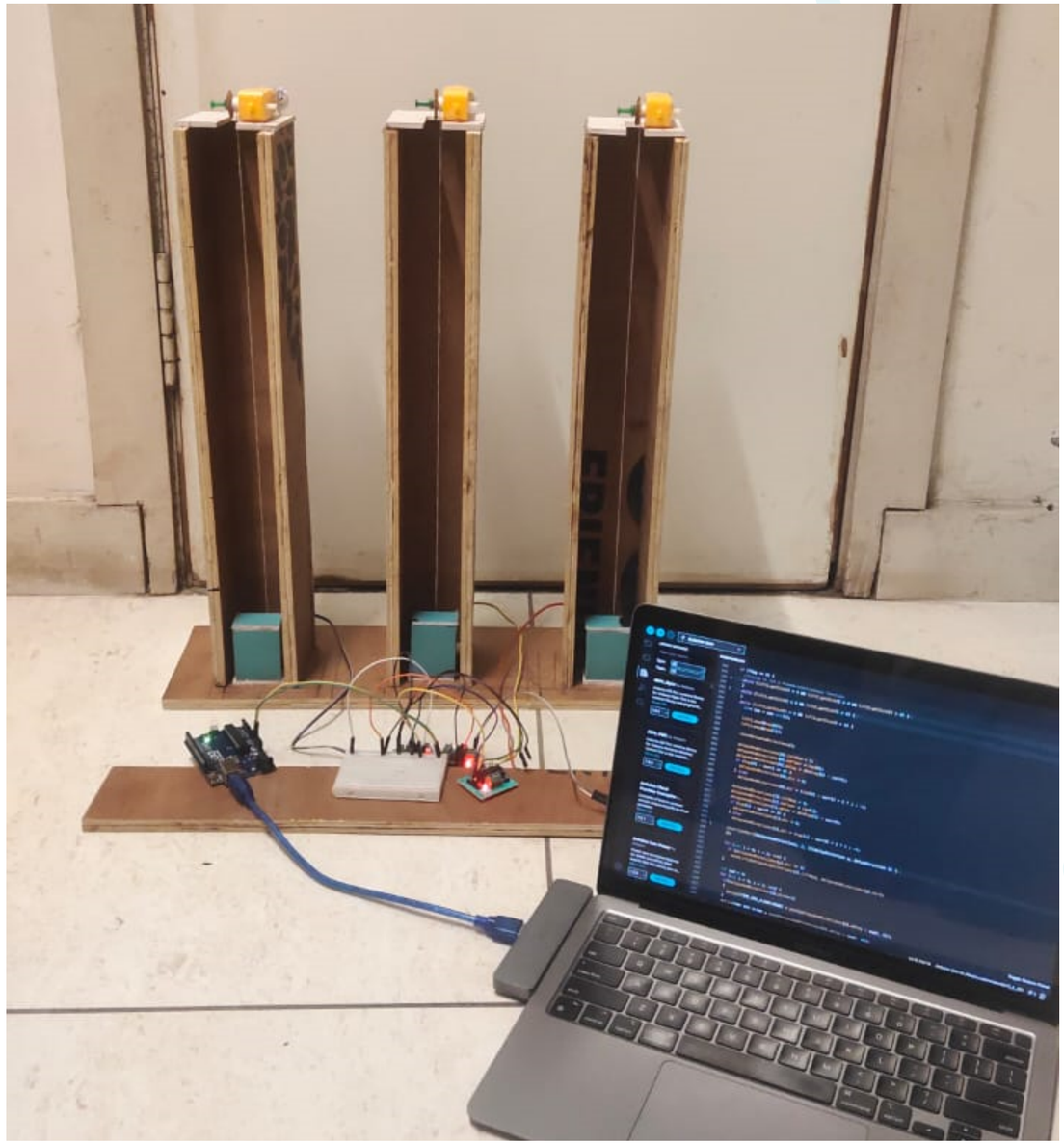
**Passenger satisfaction**

**Energy efficiency**

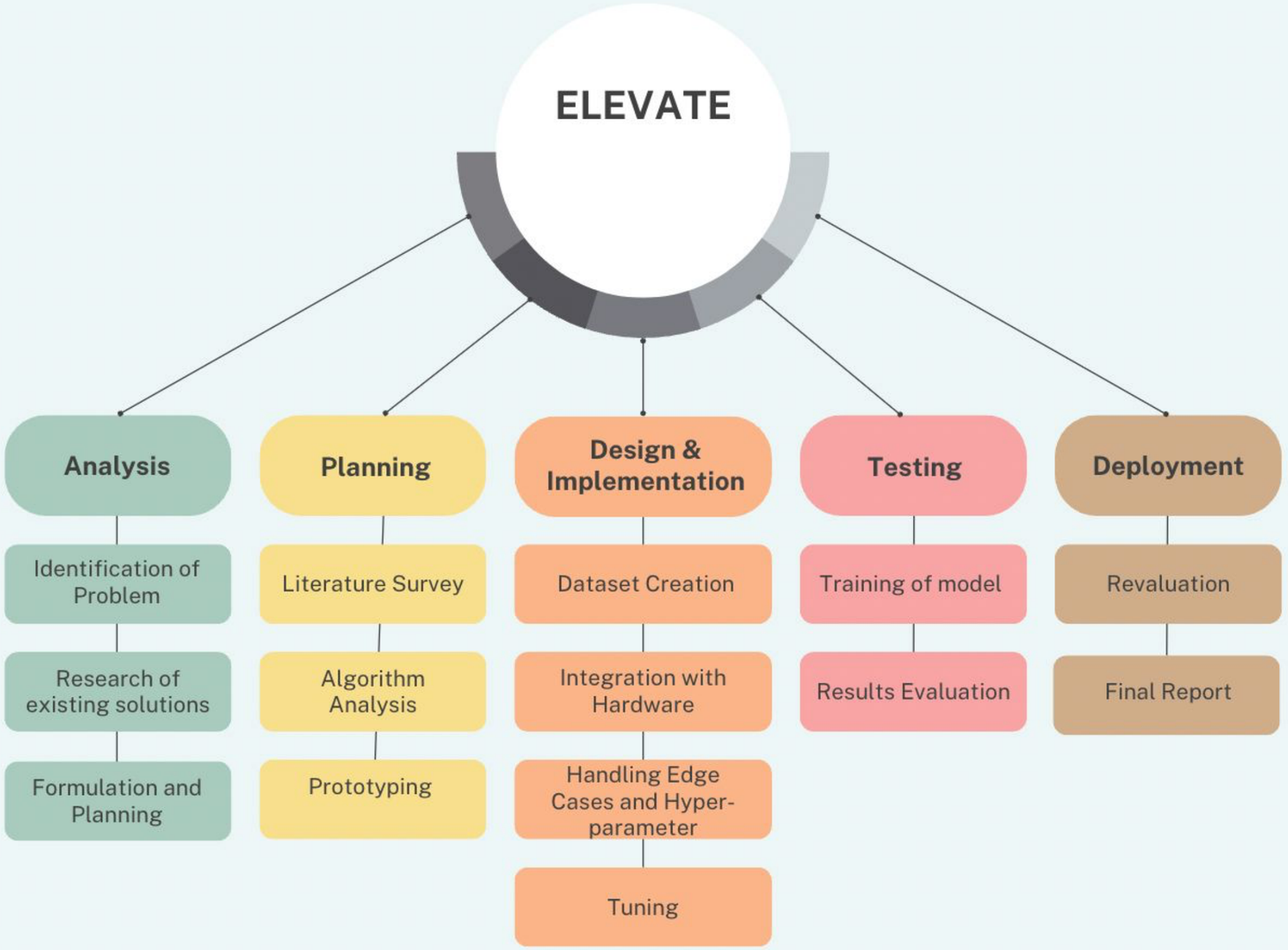


# HARDWARE

The hardware simulation uses Arduino and 3 DC motors to replicate the Lift Lobby. L239D ICs have been used to facilitate the movement of engines in both directions; efforts have been made to make the system near perfect.



# METHODOLOGY



# PROFESSIONAL AND TECHNICAL LEARNING

All the team members learned the importance of punctuality in attending the group meetings to discuss the upcoming responsibilities. The team was also regularly at the evaluation destination for panel and mentor evaluation.

Apart from professional discipline, the team learned new tools like working with IOT-related boards like Arduino and associated hardware; the group also enjoyed implementing what we learned in our lecture classes, like the H-bridge circuit.

We also enjoyed exploring domains like deep learning and industry-standard optimization technique.

# INDIVIDUAL ROLES

| Names | Software Implamentation | Research | Documenttion | Hardware implementation |
|---|---|---|---|---|
| Jahnvi Gangwar (Leader) | ✓ | ✓ | ✓ | ✓ |
| Veer Daksh Agarwal | ✓ | ✓ | ✓ | ✓ |
| Shamayla Jindal | ✓ | ✓ | ✓ | ✓ |
| Chahat Joneja | ✓ | ✓ | ✓ | ✓ |
| Mukul Singhal | ✓ | ✓ | ✓ | ✓ |

# THANK YOU!!