

# Elevator Simulator

## **Team 6:**

Venkat Aashish Mathsvaraja

Shreelakshmi Alur

Aishwarya Dang

## Contents:

1.	Introduction	3
2.	Components	4
3.	Requirements	5
4.	Specifications	6
5.	Design	7
	5.1 Structure	7
	5.2 Comprises Relation	10
	5.3 Uses Relation	11
6.	Implementation	11
7.	Testing	13
8.	Project management	14
9.	References	15

## **1. Introduction:**

An elevator or lift is a type of vertical transportation generally powered by electric motors that moves people or goods between floors of a building. Because of wheelchair access laws, elevators are often a legal requirement in new multistory buildings, especially where wheelchair ramps would be impractical.

Simulation is a way of designing and developing applications to represent the real world. Elevator system simulation software can be used to model complex traffic patterns and elevator arrangements efficiently.

In this project, we have implemented an elevator simulator and control mechanism system that shows elevators moving based on scheduling algorithm. The system consists of internal and external buttons that can be controlled by the user and doors that open and close. The scheduling algorithm keeps track of the elevators and sends the nearest one if nearest car algorithm is used when a call is made, hence saving time. It takes number of floors and elevators as an input from the user and simulates the real world elevator system. We have used Java Swing to build the GUI of the elevator simulator and simulate the real world system. As testing is a vital part of any software system, we have used EvoSuite for the purpose. This process has helped us identifying the errors and we were able to fix them effectively.

As in the present world elevators serve multiple floors, In this complex system an algorithm is required to decide which elevator serves which floor. We can't test the algorithm on a real elevator system after construction, It should be tested before. So an Elevator simulator is used to simulate the real world type scenario where multiple elevators serve different floors in a building.

## **2. Components:**

Components of a real world elevator system that we are simulating are

- Elevator car: The elevator system consists of a car / carriage mounted on a platform within an enclosed space called shaft. This is primarily used to carry the passenger or goods to different floors.
- Doors: Elevator doors protect passengers from falling into the shaft. The most common way in which the elevator doors are built are two panels that meet in the middle. To open, they slide laterally.
- External buttons: Elevators are typically controlled from the outside by a set of up and down buttons at each floor. When pressed at a certain floor, the button calls the elevator to pick up the passengers. If the particular elevator is currently serving traffic in a certain direction, it will only answer calls in the same direction unless there are no more calls beyond that floor.

- Internal buttons: Each elevator has a set of internal buttons numbered for each floor. When a particular floor button is pressed, it takes the passenger to the corresponding floor.
- Elevator Lights: Whenever an elevator button is pressed (internal or external), the button lights up. The light turns off when the elevator reaches the destination.

### **3. Requirements:**

The project we developed has these functional requirements:

- To design and simulate an elevator system to control the elevators and pick up the elevator calls
- To design a simulator where user can simulate the real world elevator system and test different algorithms.
- Perform manual simulation with number of elevators and floors taken as inputs from the user
- Integrate a Scheduling Algorithm to serve the elevator calls in an efficient manner by selecting the nearest available elevator
- Design the GUI to simulate the elevator system which resembles the real world system
- Each elevator should have a set of inner elevator buttons, one for each floor.

- Each floor (except first and last) should have two external buttons. One to request an up-elevator and other for down-elevator.
- All calls must be served by an elevator without fail.

## **4. Specifications:**

The project we developed has these program specifications:

- Once the button is pressed (internal or external), the lights are turned on and cause the elevator to visit the corresponding floor. The lights are turned off once the elevator reaches the floor.
- Elevator should travel in same direction while there are remaining requests in that direction.
- When an external up or down button is pressed at a floor, the nearest elevator should reach the floor(In nearestcar algorithm). If all the elevators are on the same floor, random elevator should be selected to serve the call.
- When an external elevator button is pressed, if the particular elevator is currently travelling in a certain direction, it will only answer calls in the same direction unless there are no more calls beyond that floor.
- If an external button is pressed at a floor and an elevator is present at the same floor, the elevator should just open and close its doors for the passenger.

- When there are no requests, elevator remains at its current floor with its doors closed.

## **5. Design:**

### **5.1. Structure:**

Detailed architecture of the system is described below:

#### **Main Module:**

**Classes:** Main module has three classes. Namely, 'Controller', 'ElevatorCalls' and 'Movement'. This module is like a control unit of the elevator system.

##### **1. Controller:** It has the following methods:

- **main():** Creates an instance of elevator class and adds it to the container. It also listens to the elevator calls and creates UI according to the input.
- **readInput():** Reads the input from the user through the console. i.e. number of floors and number of elevators.
- **getNumberOfFloors():** Returns the number of floors
- **getNumberOfElevators():** Returns the number of elevators
- **getElevatorContainer():** Returns the list of elevator objects

**2. ElevatorCalls:** Uses threads to process multiple elevator calls and has following methods:

- listen(): Initiates the listener for elevator calls
- run(): Sorts the requests based on internal and external button calls.

Adds the requests to the queue and serves them accordingly

**3. Movement:** Responsible for the movement of the elevator (car)

- doMovement(): Obtains the elevator to be moved from the elevator container and initiates its movement

### **Algorithm Module:**

**Classes:** It has a single class called SchedulingAlgorithm. It is responsible for serving the elevator calls efficiently by finding the nearest elevator for a floor and hence saving time and energy.

- randomElevator(): Selects a random elevator amongst a list of elevators
- getNearestElevator(): Calculates and finds the nearest elevator among a list of elevators corresponding to the floor (destination)

### **Model Module:**

**Classes:** It has one class called Elevators. It saves the list of elevators so that the data can be accessed by other classes at different times. It has the following methods:

- getId(): Returns the id of the elevator



- `setCurrentFloor()`: Sets the given floor as the current floor
- `getCurrentFloor()`: Returns the current floor
- `getQueue()`: Returns the queue of elevator calls
- `addQueue()`: Adds the floor number to be served to the priority queue
- `serveFloor()`: Elevator reaches the destined floor to serve the request and sets it as the current floor for that elevator
- `open()`: Opens the elevator doors
- `close()`: Closes the elevator doors

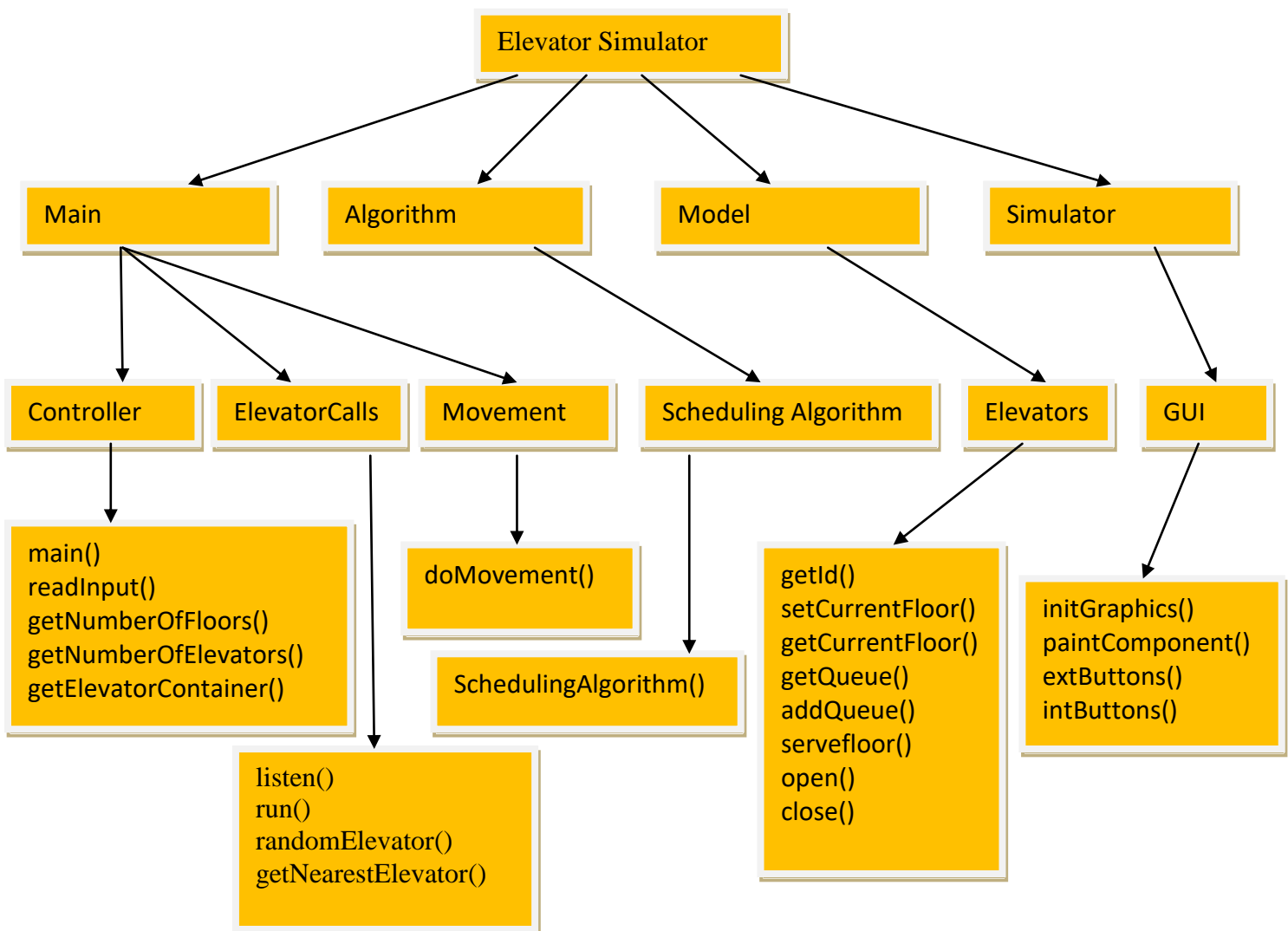
### **Simulator Module:**

**Classes:** We have used Java Swing to develop the UI of the system. The Simulator module has one class called GUI class. It is responsible for the visual representation of the elevator system. It has the following methods:

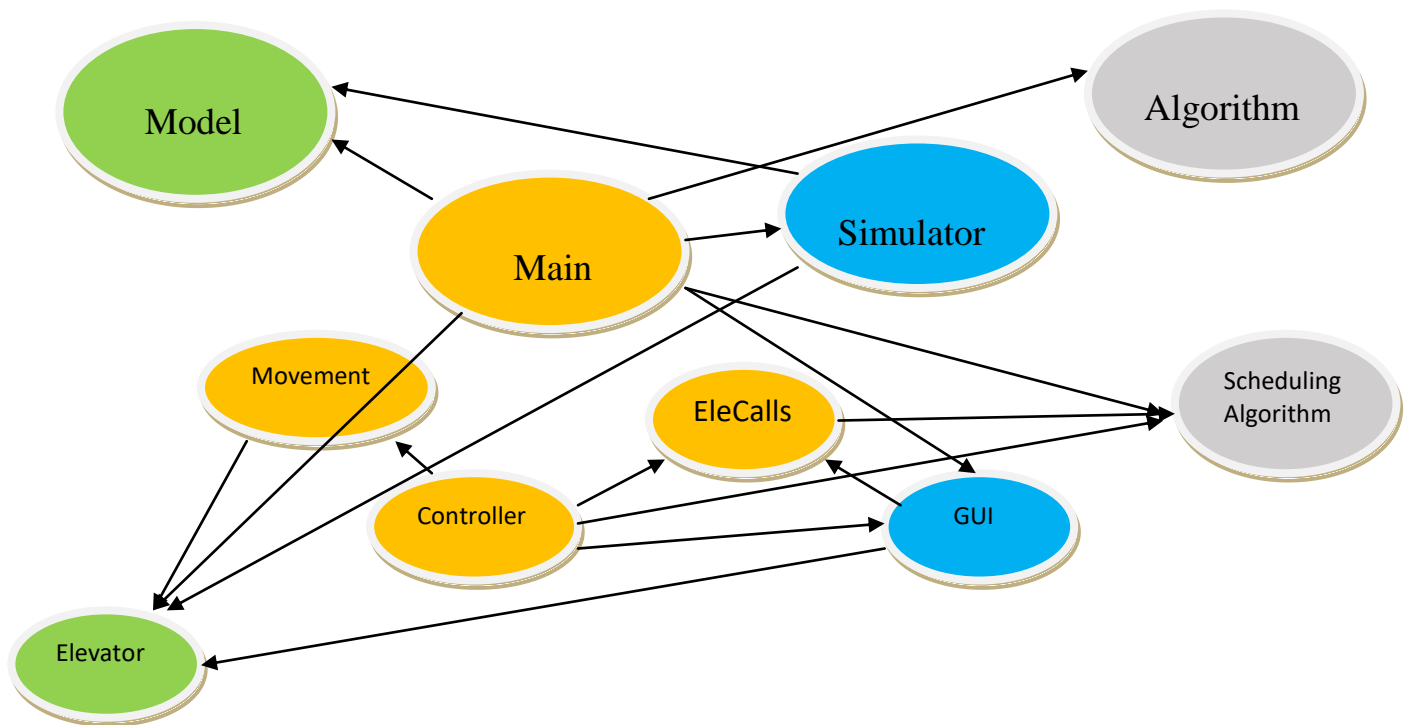
- `createGraphics()`: It initializes graphics of the system by building the layouts of all the components
- `paintComponent()`: It draws and paints the components of UI like buttons and lights
- `extButtons()`: Draws the external buttons of the elevator and handles their click events

- `intButtons()`: Draws the internal buttons of the elevator and handles their click events

## 5.2 Comprises Hierarchy:



### 5.3 Uses Relation:



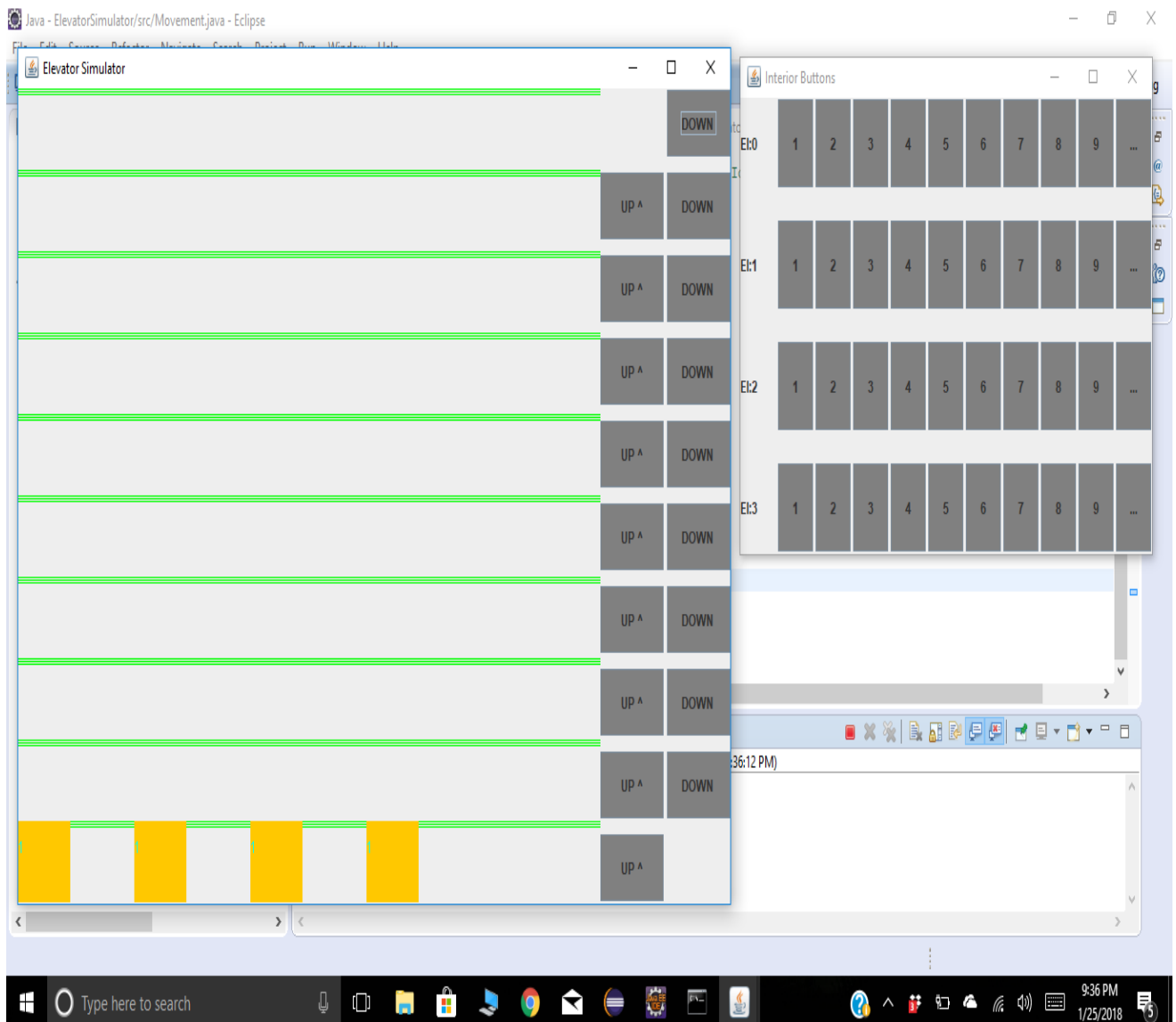
## 6. Implementation:

Firstly, the users inputs the number of elevators and number of floors to be simulated through the console.

```
Controller (1) [Java Application] C:\Program Files\Java\jdk1.8.0_144\bin\javaw.exe (C
Enter number of floors : 5
Enter number of elevators: 2|
```

After that, the simulation window will appear. The user can press internal buttons for each elevator, and can press the external buttons for each floor, after that the elevator will arrive at the desired floor and open the door.

```
Console
Controller (1) [Java Application] C:\Program Files\Java\jdk1.8.0_144\bin\javaw.exe (J
Enter number of floors : 10
Enter number of elevators: 4
```



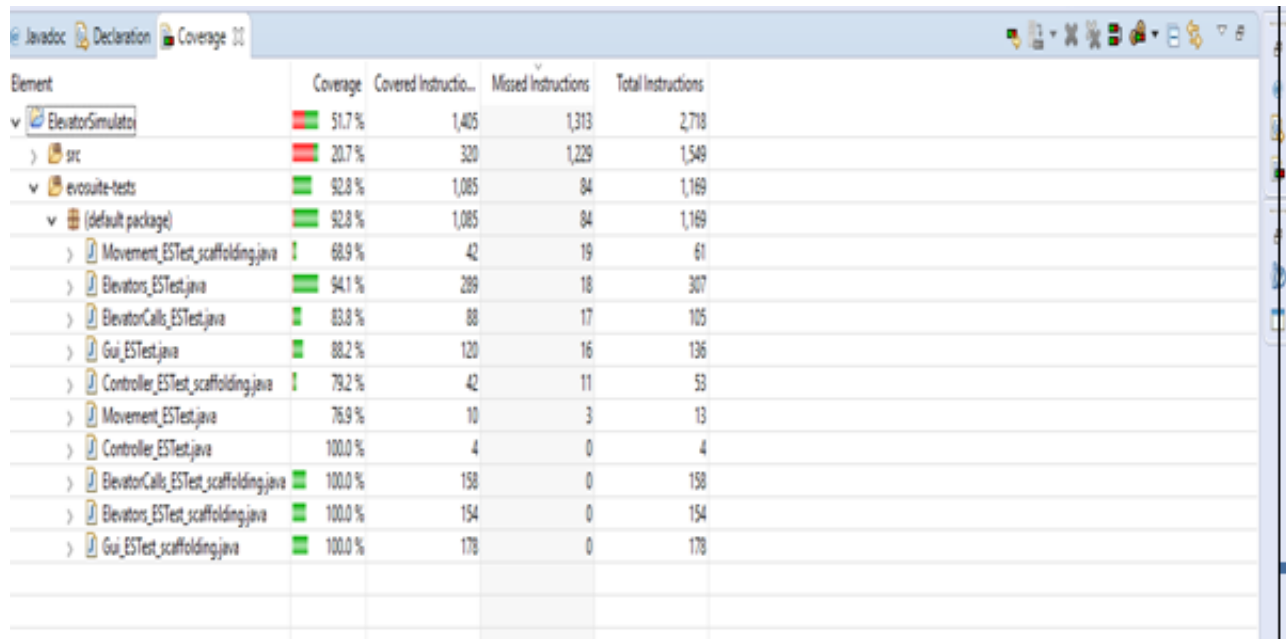
```
59  
60  
61     Elevators chosen1 = getNearestElevator(goToFloorEnteredByGui,Elevators.motion_direction);  
62  
63     // Elevators chosen1= randomElevator();  
64  
65  
66
```

We have implemented two algorithms, Random elevator and nearest car algorithm. To change the algorithm just comment or uncomment the code shown above in ElevatorCalls class.

## 7. Testing:

Software testing is an essential part of any successful software development process. We have used EvoSuite for our program verification. EvoSuite is a tool that automatically generates JUnit tests for Java software using evolutionary algorithm.

The evolutionary technique evolves all the test cases in a test suite at the same time, and the fitness function considers all the testing goals simultaneously. The technique starts with an initial population of randomly generated test suites, and then uses a Genetic Algorithm to optimize for the particular coverage area. At the end, it gives us the best minimized test suite as a result. With this approach, most of the complications and downsides of the one target at a time approach either disappear or become significantly reduced.



Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
▼ ElevatorSimulator	51.7 %	1,405	1,313	2,718
> src	20.7 %	320	1,229	1,549
▼ evosuite-tests	92.8 %	1,085	84	1,169
▼ (default package)	92.8 %	1,085	84	1,169
> Movement_ESTest_scaffolding.java	68.9 %	42	19	61
> Elevators_ESTest.java	94.1 %	289	18	307
> ElevatorCalls_ESTest.java	83.8 %	88	17	105
> Gui_ESTest.java	88.2 %	120	16	136
> Controller_ESTest_scaffolding.java	79.2 %	42	11	53
> Movement_ESTest.java	76.9 %	10	3	13
> Controller_ESTest.java	100.0 %	4	0	4
> ElevatorCalls_ESTest_scaffolding.java	100.0 %	158	0	158
> Elevators_ESTest_scaffolding.java	100.0 %	154	0	154
> Gui_ESTest_scaffolding.java	100.0 %	178	0	178

## 8. Project management:

Project Management is a crucial part in the software development process as it helps in building an efficient product with minimum risks. In our project, we used several techniques to manage our work and to achieve our goals; we provide the list of communication ways, and the techniques used in our project.

### Communication:

- **Group Meetings:** Meetings started with the updates of previous work, planning for work to be done and the process in which the work can be carried out.
- **Emails:** We frequently communicated through email to share some files and docs and also to update each other
- **WhatsApp:** It was used for quick communication mainly to discuss non-time consuming queries and doubts.

- Google Hangout: We used Hangout for video calls when we were unable to conduct group meetings.

### **Techniques Used:**

- Pair programming: We used pair programming most of the times as it helped us to prevent the bugs in real-time. It also helped us to approach the problem in a better way by brainstorming.
- Scrum Methodology: We divided the work in terms of sprints for each module and completed them one by one.

### **References:**

- <https://www.pmi.org/about/learn-about-pmi/what-is-project-management>
- fraser-arcuri~2013~evosuite.pdf
- <https://www.evosuite.org>
- <https://softwareengineering.stackexchange.com/>