ELEVATOR CONTROL SYSTEM

PROJECT REPORT

18CSC202J- OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY

(2018 Regulation)

II Year/ III Semester

Academic Year: 2022 -2023

By

SHAURYA SINGH SRINET (RA2111032010006)

SHOUNAK CHANDRA (RA2111032010026)

Under the guidance of

Dr. GOUTHAMAN. P

Assistant Professor

Department of Networking and Communications

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Chengalpattu

NOVEMBER 2022

BONAFIDE

This is to certify that **18CSC202J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY project report** titled "**ELEVATOR CONTROL SYSTEM"** is the bonafide work of **SHAURYA SINGH SRINET (RA2111032010006), SHOUNAK CHANDRA (RA2111032010026)** who undertook the task of completing the project within the allotted time.

Signature of the Guide

Dr. Gouthaman. P

Assistant Professor

Department of NWC

SRM Institute of Science and Technology

Signature of the II Year Academic Advisor

Professor and Head

Department of NWC

SRM Institute of Science and Technology

About the course: -

18CSC202J/ 8AIC203J - Object Oriented Design and Programming are 4 credit courses with **L T P C as 3-0-2-4** (Tutorial modified as Practical from 2018 Curriculum onwards)

Objectives:

The student should be made to:

- Learn the basics of OOP concepts in C++
- Learn the basics of OOP analysis and design skills.
- Be exposed to the UML design diagrams.
- Be familiar with the various testing techniques

Course Learning Rationale (CLR): The purpose of learning this course is to:

1. Utilize class and build domain model for real-time programs
2. Utilize method overloading and operator overloading for real-time application development programs
3. Utilize inline, friend and virtual functions and create application development programs
4. Utilize exceptional handling and collections for real-time object-oriented programming applications
5. Construct UML component diagram and deployment diagram for design of applications
6. Create programs using object-oriented approach and design methodologies for real-time application development

Course Learning Outcomes (CLO): At the end of this course, learners will be able to:

1. Identify the class and build domain model
2. Construct programs using method overloading and operator overloading
3. Create programs using inline, friend and virtual functions, construct programs using standard templates
4. Construct programs using exceptional handling and collections
5. Create UML component diagram and deployment diagram
6. Create programs using object-oriented approach and design methodologies

Table 1: Rubrics for Laboratory Exercises

(Internal Mark Split up: - As per Curriculum)

CLAP-1	5= (2(E-lab Completion) + 2(Simple Exercises) (from CodeZinger, and any other coding platform) + 1(HackerRank/Code chef/LeetCode Weekend Challenge)	Elab test
CLAP-2	7.5= (2.0(E-lab Completion) + 2.0 (Simple Exercises) (from CodeZinger, and any other coding platform) + 3.5 (HackerRank/Code chef/LeetCode Weekend Challenge)	Elab test
CLAP-3	7.5= (2.0(E-lab Completion (80 Pgms) + 2.0 (Simple Exercises) (from CodeZinger, and any other coding platform) + 3.5 (HackerRank/Code chef/LeetCode Weekend Challenge)	**2 Mark -** E-lab Completion **80 Program** Completion from 10 Session (Each session min 8 program) **2 Mark -** Code to UML conversion GCR Exercises **3.5 Mark - Hacker Rank** Coding challenge completion
CLAP-4	5= 3 (Model Practical) + 2 (Oral Viva)	• **3 Mark** – Model Test • **2 Mark** – Oral Viva
Total	25	

COURSE ASSESSMENT PLAN FOR OODP LAB

S. No	List of Experiments	Course Learning Outcomes (CLO)	Blooms Level	PI	No of Programs in each session
1.	Implementation of I/O Operations in C++	CLO-1	Understand	2.8.1	10
2.	Implementation of Classes and Objects in C++	CLO-1	Apply	2.6.1	10
3,	To develop a problem statement. 1. From the problem statement, Identify Use Cases and develop the Use Case model. 2. From the problem statement, Identify the conceptual classes and develop a domain model with a UML Class diagram.	CLO-1	Analysis	4.6.1	Mini Project Given
4.	Implementation of Constructor Overloading and Method Overloading in C++	CLO-2	Apply	2.6.1	10
5.	Implementation of Operator Overloading in C++	CLO-2	Apply	2.6.1	10
6.	Using the identified scenarios, find the interaction between objects and represent them using UML Sequence diagrams and Collaboration diagrams	CLO-2	Analysis	4.6.1	Mini Project Given
7.	Implementation of Inheritance concepts in C++	CLO-3	Apply	2.6.1	10
8.	Implementation of Virtual function & interface concepts in C++	CLO-3	Apply	2.6.1	10
9.	Using the identified scenarios in your project, draw relevant state charts and activity diagrams.	CLO-3	Analysis	4.6.1	Mini Project Given
10.	Implementation of Templates in C++	CLO-3	Apply	2.6.1	10
11.	Implementation of Exception of Handling in C++	CLO-4	Apply	2.6.1	10
12.	Identify the User Interface, Domain objects, and Technical Services. Draw the partial layered, logical architecture diagram with UML package diagram notation such as Component Diagram, Deployment Diagram.	CLO-5	Analysis	4.6.1	Mini Project Given
13.	Implementation of STL Containers in C++	CLO-6	Apply	2.6.1	10
14.	Implementation of STL associate containers and algorithms in C++	CLO-6	Apply	2.6.1	10
15.	Implementation of Streams and File Handling in C++	CLO-6	Apply	2.6.1	10

LIST OF EXPERIMENTS FOR UML DESIGN AND MODELLING:

To develop a mini-project by following the exercises listed below.

1. To develop a problem statement.
2. Identify Use Cases and develop the Use Case model.
3. Identify the conceptual classes and develop a domain model with UML Class diagram.
4. Using the identified scenarios, find the interaction between objects and represent them using UML Sequence diagrams.
5. Draw relevant state charts and activity diagrams.
6. Identify the User Interface, Domain objects, and technical services. Draw the partial layered, logical architecture diagram with UML package diagram notation.

Suggested Software Tools for UML:

StarUML, Rational Suite, Argo UML (or) equivalent, Eclipse IDE and Junit

ABSTRACT

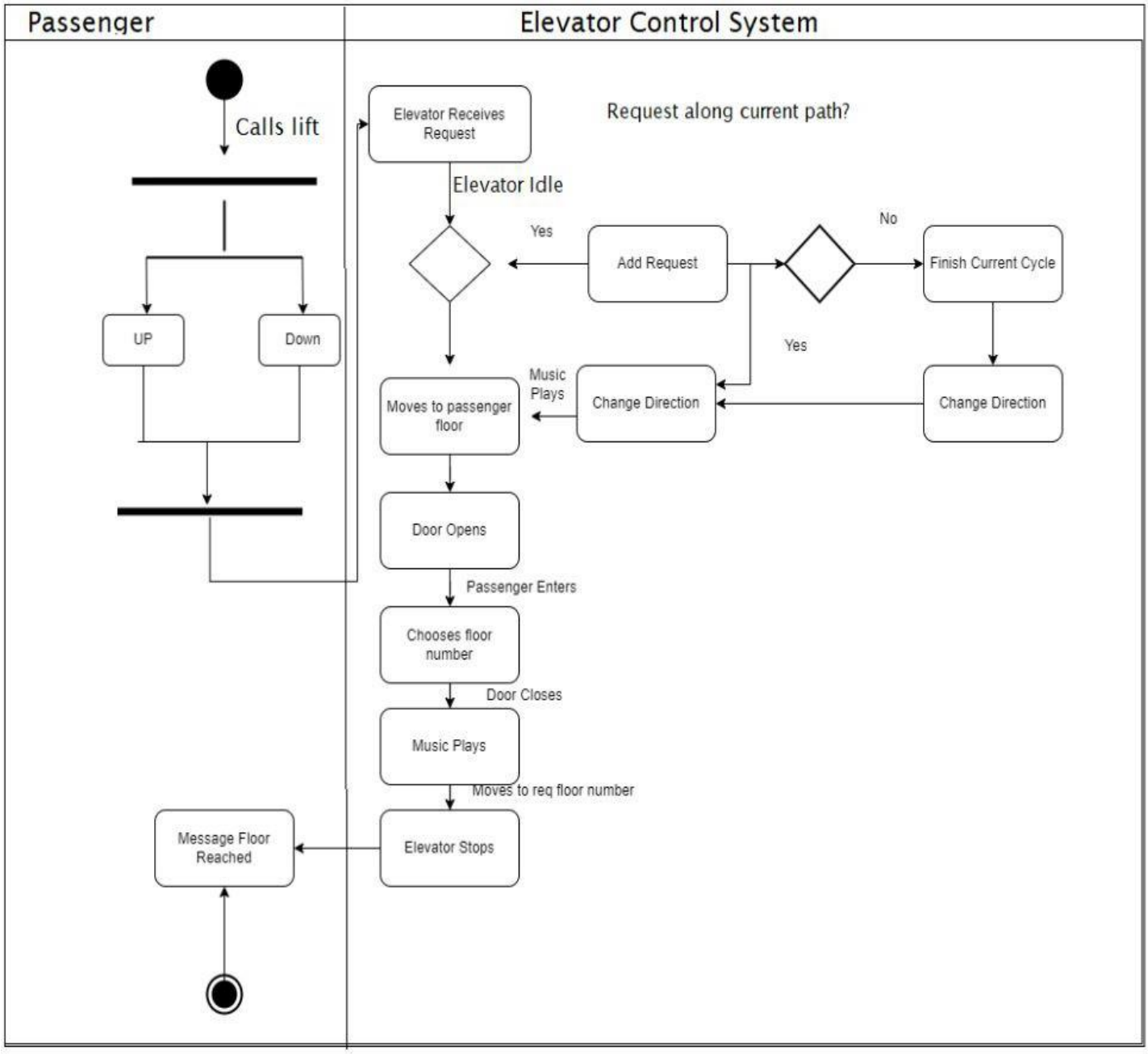
In the design procedure of the elevator control circuit, the controller- Datapath approach was used. In this approach, all the functional and memory are concentrated in the Datapath while the control signals are generated by a simpler sequence diagram. To specify the control sequence and data processing tasks of the designed digital system, a hardware algorithm has been adopted and the corresponding State chart has been used. The components have been designed with Component Diagrams and Medium Scale Integration (MMI) logic components. In this report we have implemented the elevator control system for industrial buildings like office IT spaces or a very secured government building. With the help of UML diagrams like Component diagram we have demonstrated how can we make the building more secured. Often elevators are the point of contact for many access points. Diagrams like Activity Diagrams demonstrate the basic working of the elevator. The case diagram demonstrates the total functioning of the elevator. The total system can be practically implemented in real life.

MODULE DESCRIPTION

Elevator Control System is the system responsible for coordinating all aspects of elevator service such as travel, speed, and accelerating, decelerating, door opening speed and delay, leveling and hall lantern signals. It accepts inputs (e.g., button signals) and produces outputs (elevator cars moving, doors opening, etc.). The main aims of the elevator control system are:

- To bring the lift car to the correct floor.
- To minimize travel time.
- To maximize passenger comfort by providing a smooth ride.
- To accelerate, decelerate and travel within safe speed limits.

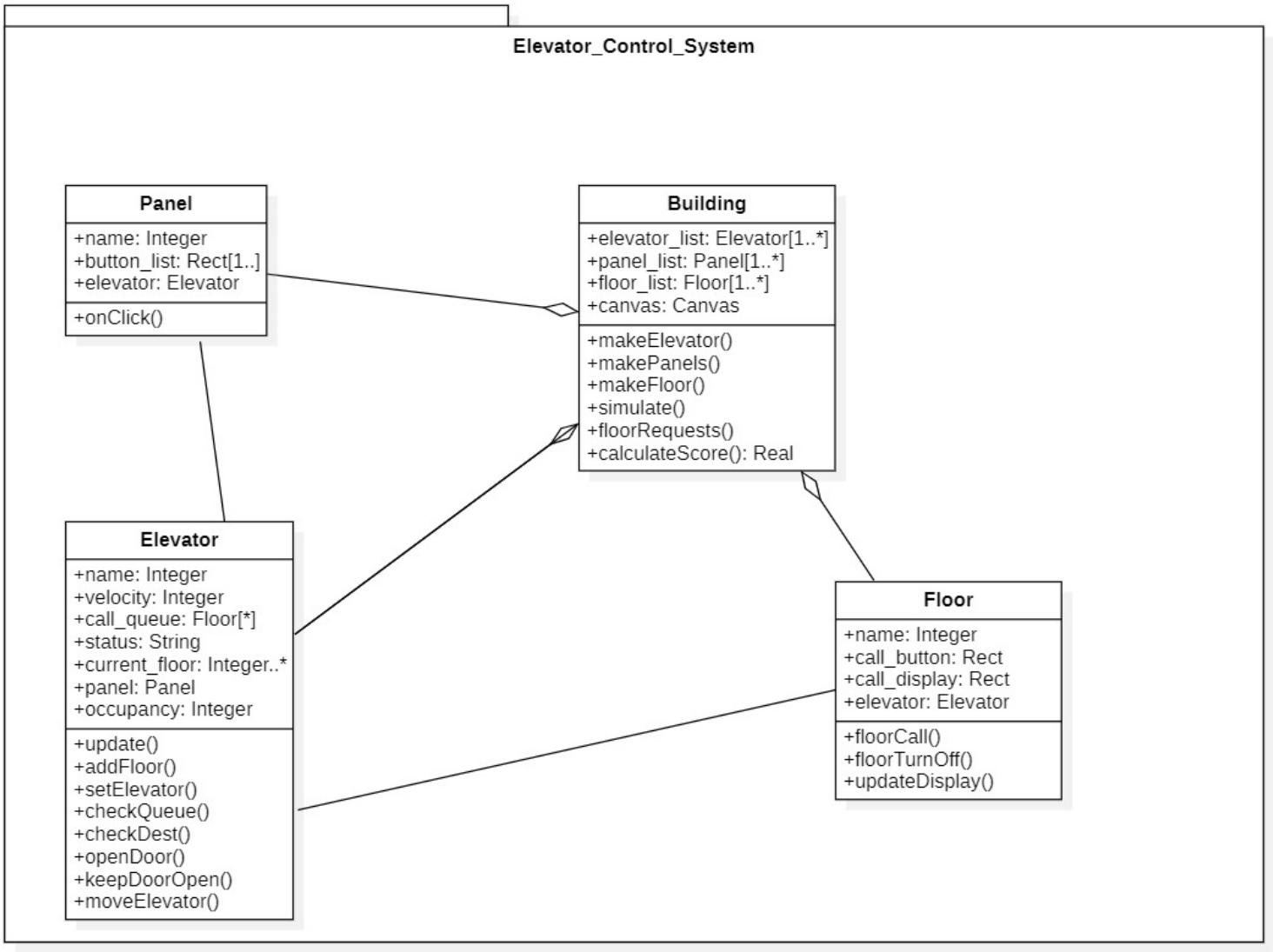
Use Case Diagram with Explanation



The state machine diagram depicts the following states of an elevator: Idle, Moving Down, Moving Up, Stopping, Door Opening, Door Closing, Next Stop Processing, Open Door. The diagram contains transitions between particular states, that may fire only when their condition is met:

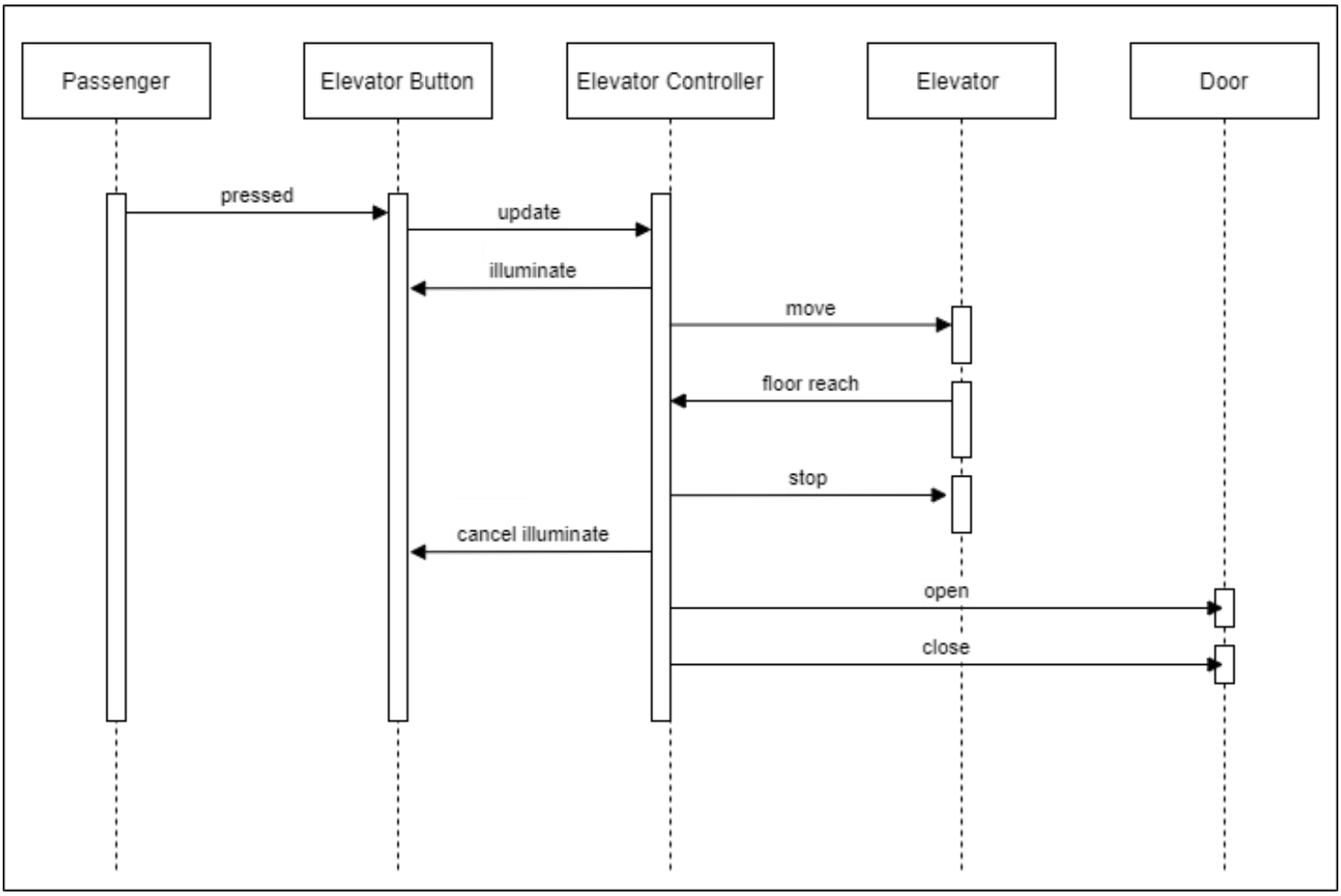
- Idle to Decision - floor chosen
- Decision to Moving Down - elevator is above the current floor
- Decision to Moving Up - elevator is below the current floor
- Moving Down to Stopping - elevator approached the destination floor
- Moving Up to Stopping - elevator approached the destination floor
- Stopping to Door Opening - elevator stopped on the destination floor
- Door Opening to Open Door - door fully open
- Door Open to Door Closing - open door timer elapsed
- Door Closing to Door Opening - someone stepped into the door
- Door Closing to Next Stop Processing - door closed
- Next Stop Processing to Moving Down - elevator is above the destination floor
- Next Stop Processing to Moving Up - elevator is below the destination floor
- Next Stop Processing to Idle - no other destination available

Class Diagram with Explanation



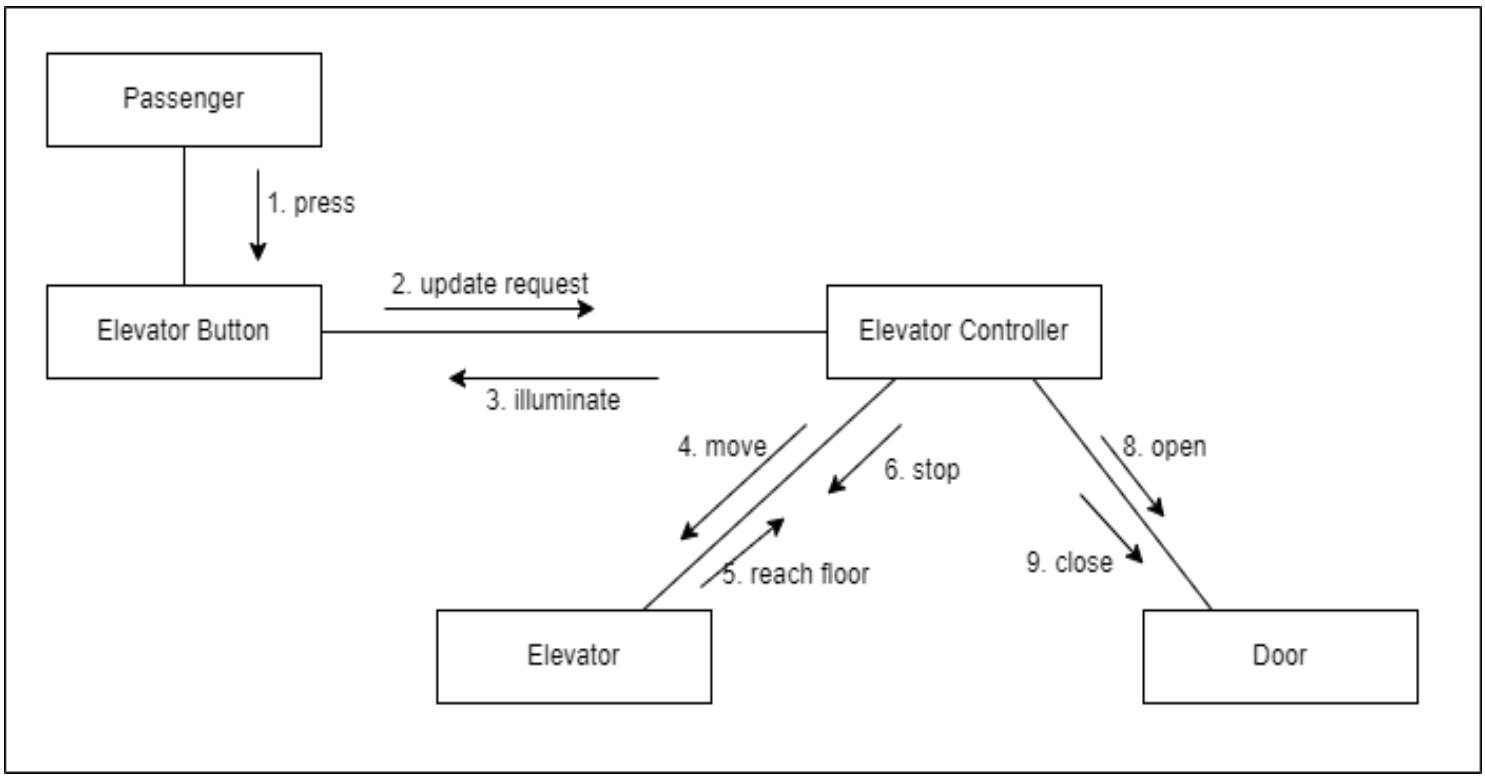
Class diagram captures the basic the basic functional aspects of the elevator button, floor button, sensor, door, floor light. The panel class is the most interacted part of the elevator. It is the panel through which the inputs for the floor are given. The building receives the request from the elevator and invokes such as make elevator(), makePanels(), makefloors(),simulate().

Sequence Diagram with Explanation



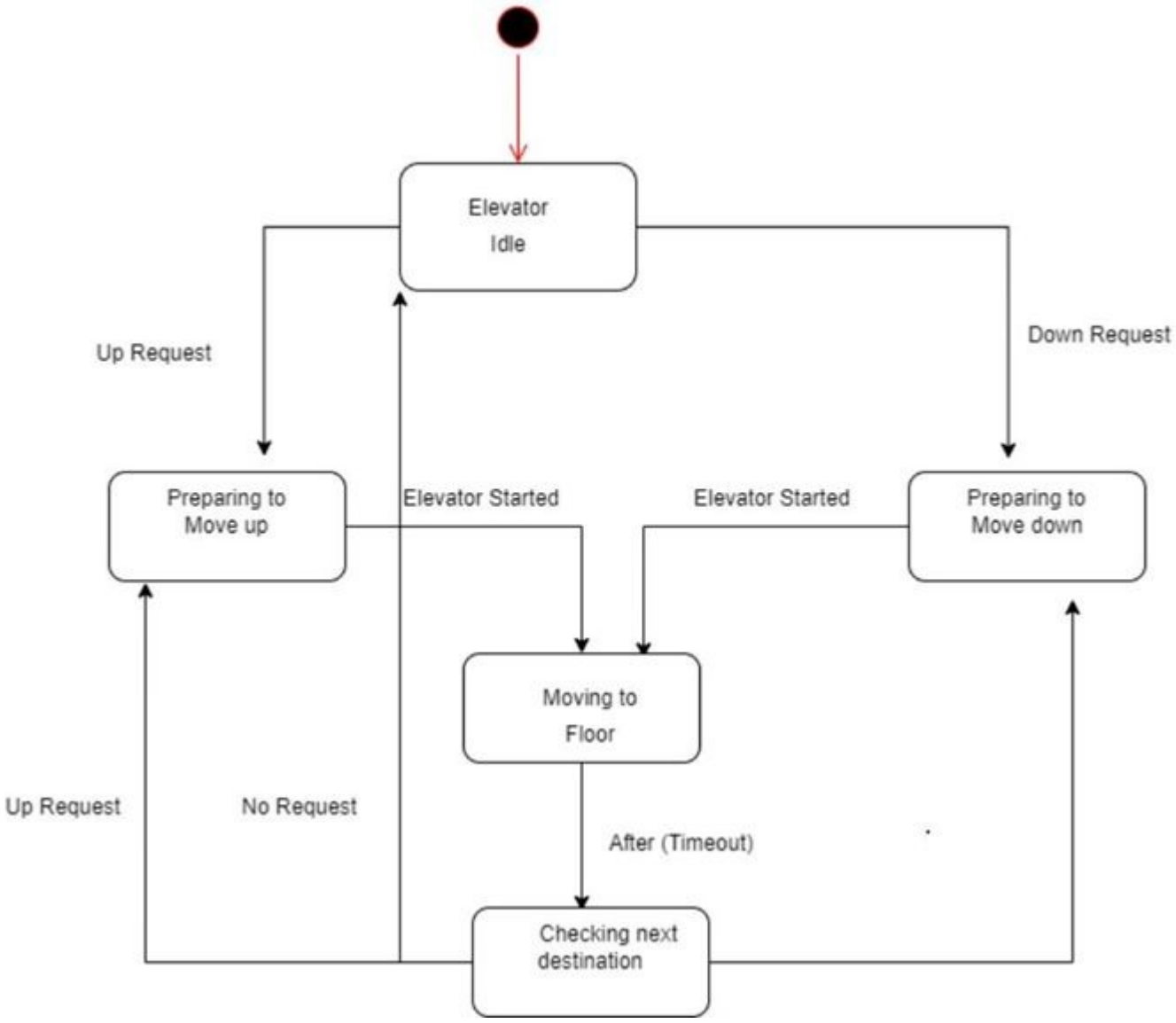
A sequence diagram and collaboration diagram convey similar information but expressed in different ways. A Sequence diagram shows the explicit sequence of messages suitable for modeling a real-time system, whereas a collaboration diagram shows the relationships between objects. Over here the relationship between Passenger, Elevator Button, Elevator Controller, Elevator, and the Door. Like when the passenger is passed Elevator button is invoked. Upon which the elevator control is invoked. The control thereby reinvokes instructions for the elevator and thereby asks the elevator to move. When the elevator reaches its destination door is invoked and the passenger steps out.

Communication Diagram with Explanation



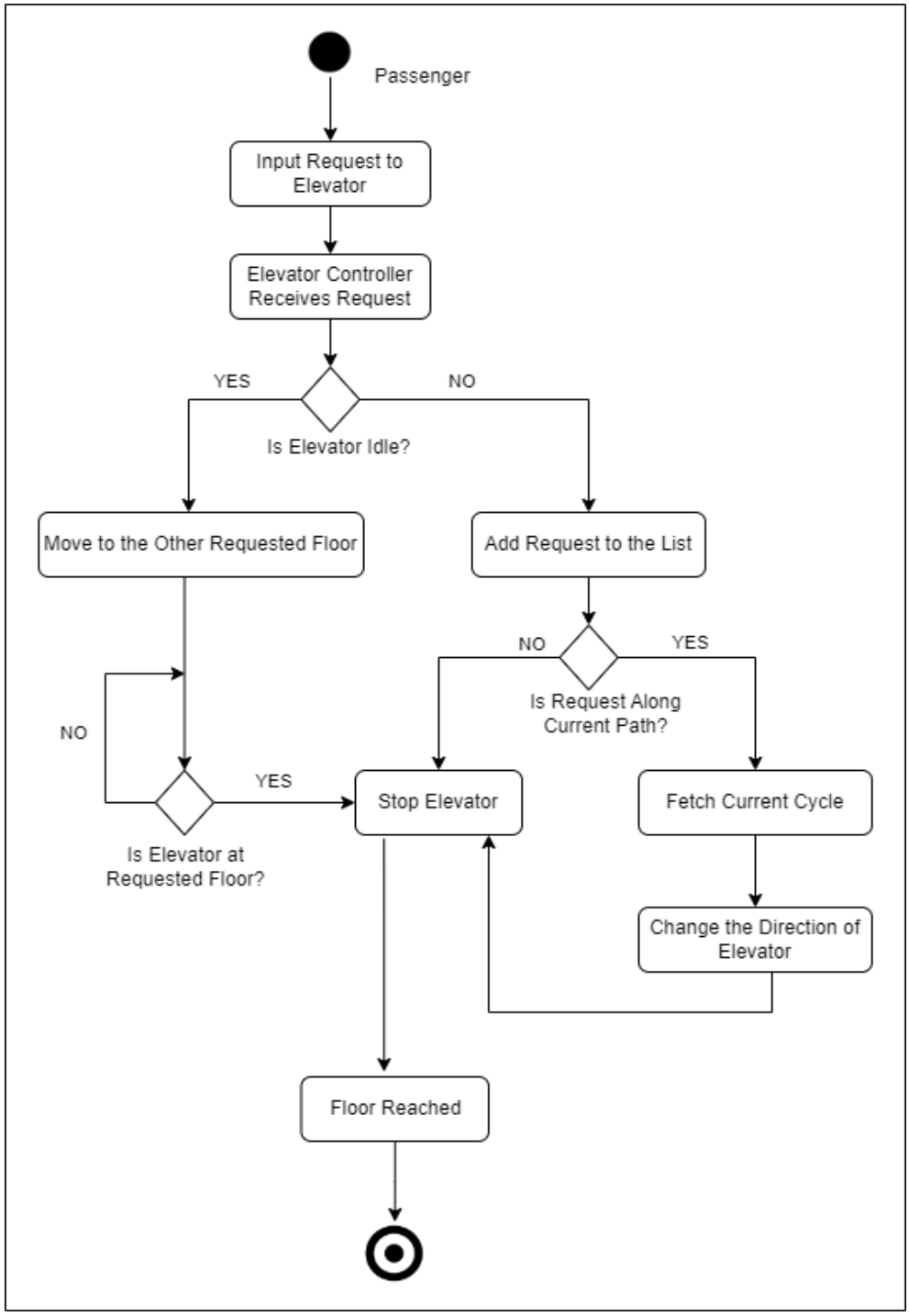
It mainly focusses on communicating between various components of a system. The same way by which a passenger communicates to the elevator button. The Button communicates the request to the elevator controller. The controller thereby moves the elevator or opens or closes the door.

State Chart Diagram with Explanation



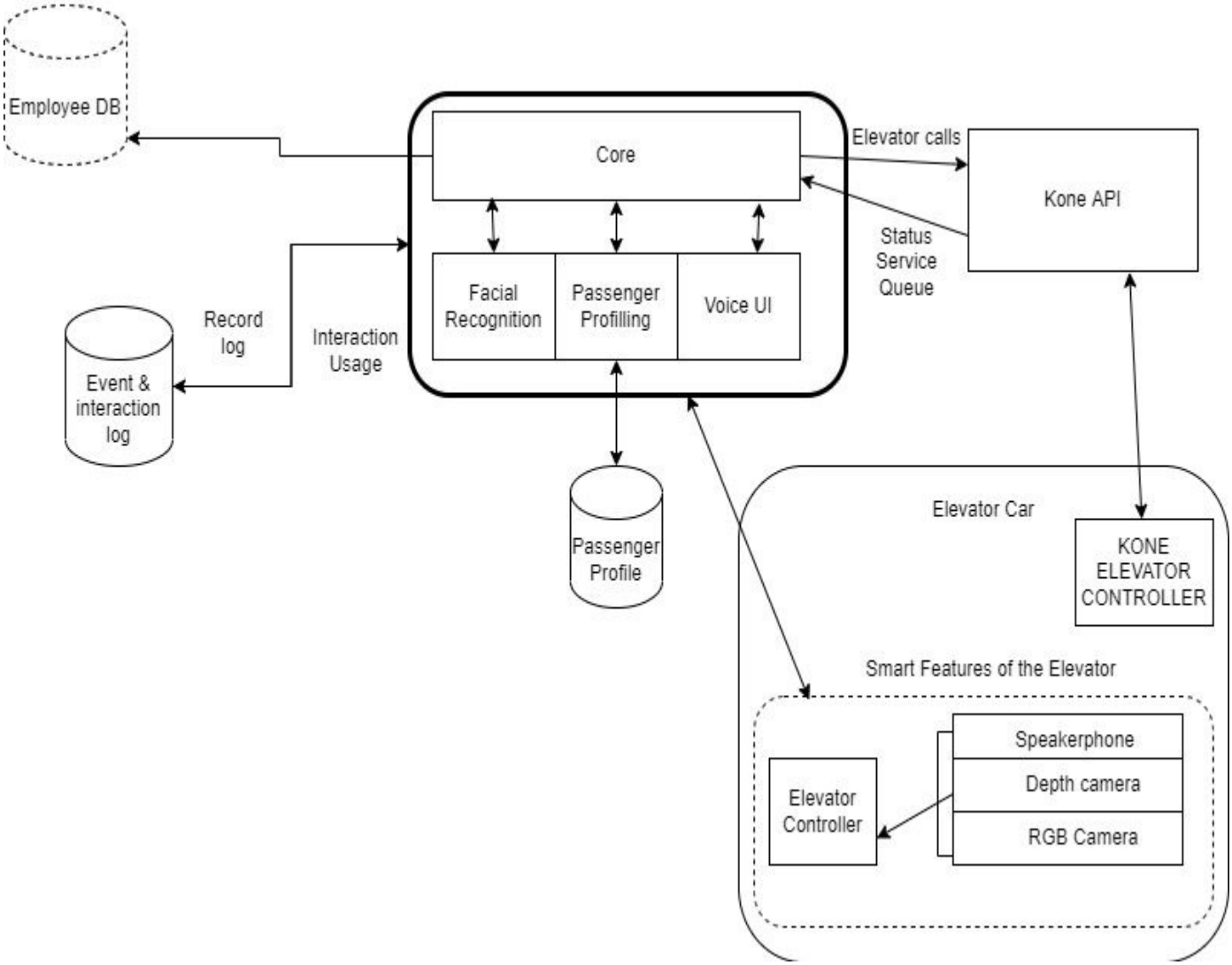
The state chart diagram is responsible for describing the various state of the elevator. It has a module Elevator Idle which can send 2 requests one to move the elevator up & the other to move to move the elevator down. Once the elevator starts another module called moving to floor is invoked which helps the elevator thereby to move to its destination.

Activity Diagram with Explanation



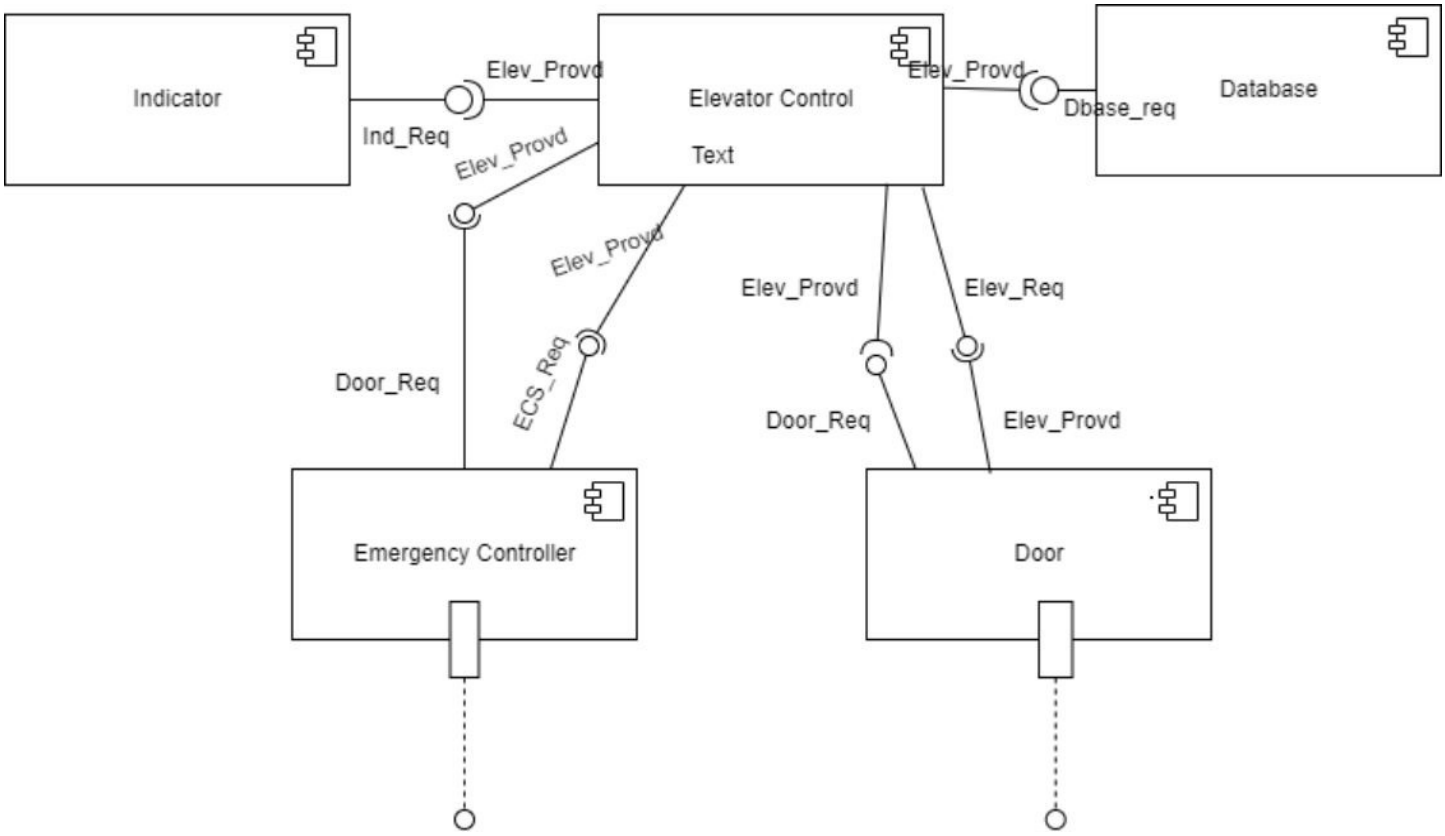
The activity diagram is regarding the various activities of the elevator. responds in Boolean. Whenever the elevator receives a request like a request from the controller it responds to it. The first request it processes is that whether the elevator is idle or not. It thereby requests its current path.

Package Diagram with Explanation



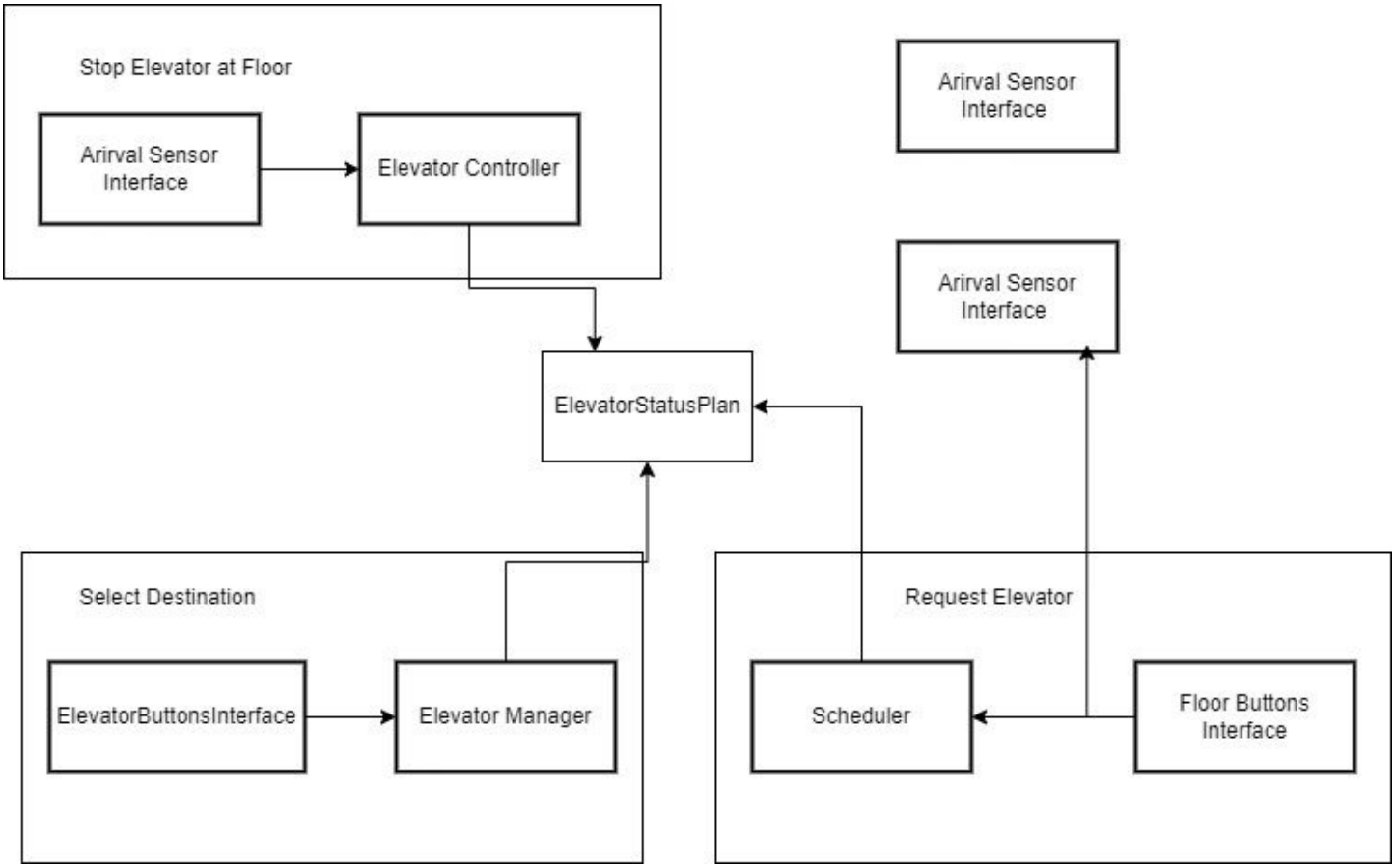
The package diagram is responsible for bringing the various packages of a diagram together. It starts with an Employee Data base which has the biometrics record of all the employees who would be using the elevator. It is kept at the core. Whenever an employee gets inside the elevator, he is first authenticated to make sure whether he really fits in the elevator. He puts his biometrics which is authenticated using KONE API and the database. There is a surveillance camera inside the elevator which records every moment inside the elevator.

Component Diagram with Explanation



Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system. Over here the various components are the indicator, Elevator Control, Database and Emergency Controller and door. As the elevator moves the indicator moves the elevator controls are also invoked. The various components like the Emergency Controller and the door are called upon. The emergency control is invoked to call upon an emergency stop when there is an emergency. The door is called upon when it arrives at its destination floor.

Deployment Diagram with Explanation



Deployment diagrams are used to visualize the hardware processors/ nodes/ devices of a system, the links of communication between them and the placement of software files on that hardware. Whenever the elevator stops at a floor the arrival sensor interface and the elevator controller are invoked due to which elevator status plan is also invoked. When a request is passed the scheduler and the floor buttons interfaces are called upon.

Conclusion

The report on Elevator Control System has been made. The various stages of the project have been shown using various diagrams like Use Case, Class Case, Sequence, Communication Diagram, State Chart, Activity Diagram, Package Diagram, Component Diagram, Deployment Diagram. The project corresponds to a real-life elevator system for a building having multiple floors with high security at its core.

References

- https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/
- https://www.guru99.com/state-machine-transition-diagram.html
- https://www.web-feats.com/classes/dj/lessons/uml/elevator.html
- https://www.electrical-knowhow.com/2012/04/elevator-controlsystem.html#:~:text=Elevator%20Control%20System%20is%20the,leveling%20and%20hall%20lantern%20signals.