VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

J Component

Object Oriented Analysis and Design (ITE – 1007)

Slot – E2

Faculty: Dr.Prabadevi B

By

Omkar Kulkarni – 19BIT0196

Vaibhav Agrawal – 19BIT0185

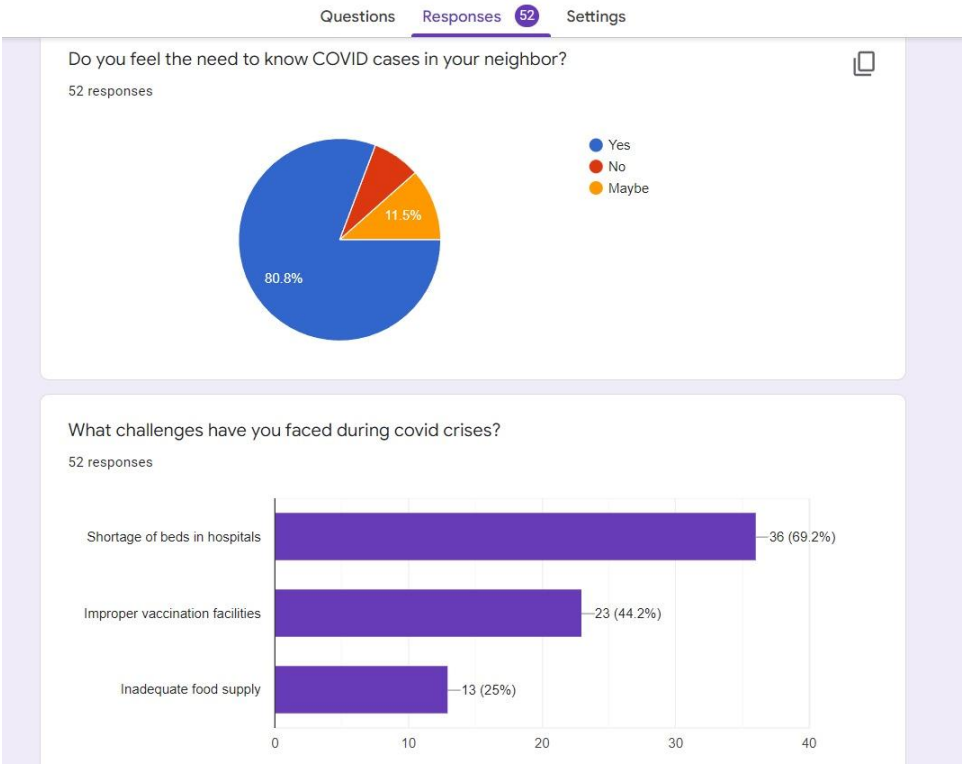
Gairik Das – 19BIT0188

Project Title –

Covid Care System

Google Form:

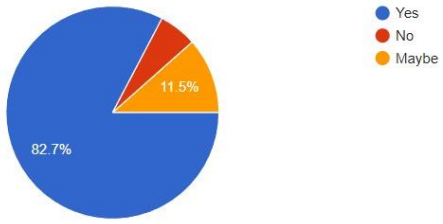
Spreadsheet link:
https://docs.google.com/spreadsheets/d/1qb55YE7PqkjLAnm102GyGbCuDuWfVT0KGgCi0aaczUM/edit?resourcekey#gid=1146946519



Questions Responses 52 Settings

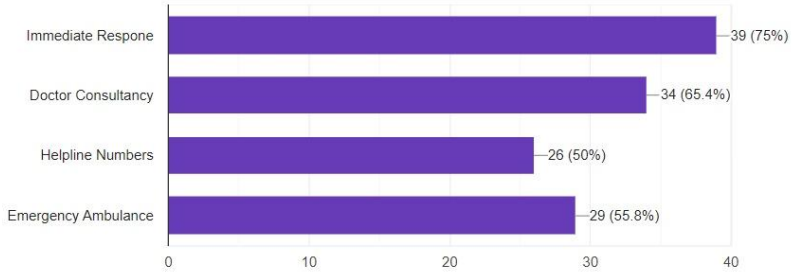
Do you want a COVID tracking system?

52 responses



What do you expect from a COVID care system?

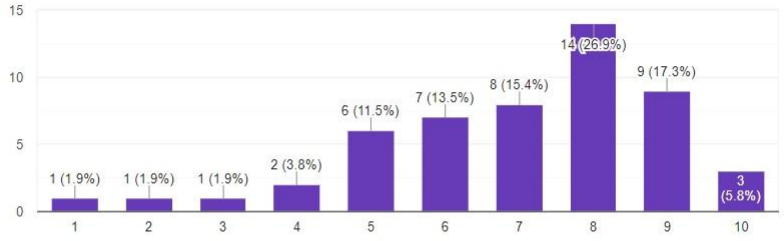
52 responses



Questions Responses 52 Settings

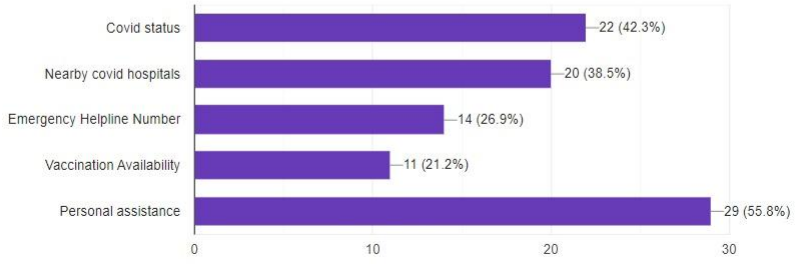
On a scale, how much would you rate the other such platforms based on covid tracking?

52 responses



What features are lacking in other covid apps?

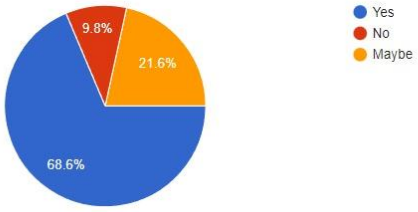
52 responses



Questions Responses 52 Settings

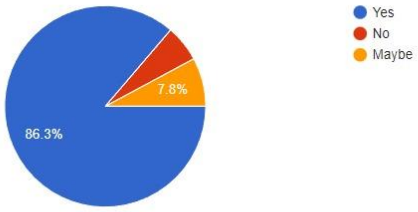
As a user, do you wants a personalized doctor to be assigned?

51 responses



Do you prefer to have emergency ambulance should have a ventilator?

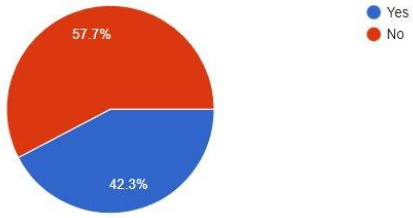
51 responses



Questions Responses 52 Settings

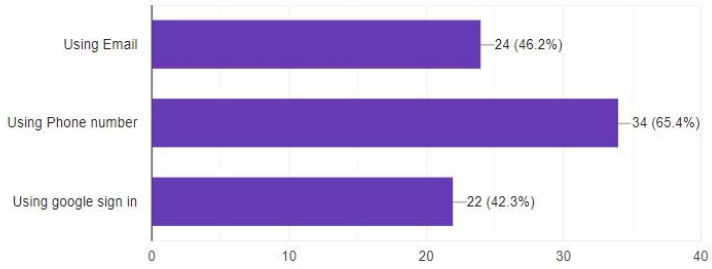
Have you seen my application based on the COVID care system?

52 responses



Which method of user authentication would you prefer according to your convenience?

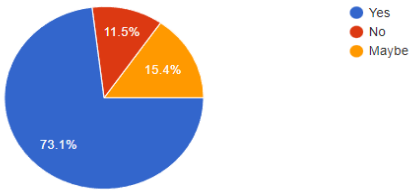
52 responses



Questions Responses 52 Settings

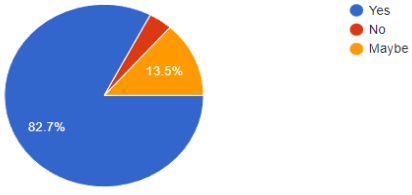
Would you like to have the photos of the respective active hospitals?

52 responses



Would you like to see the rating of the respective available doctors?

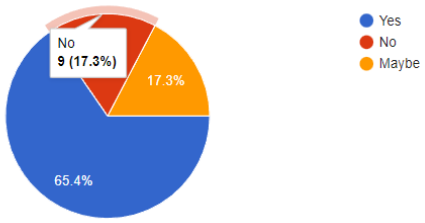
52 responses



Questions Responses 52 Settings

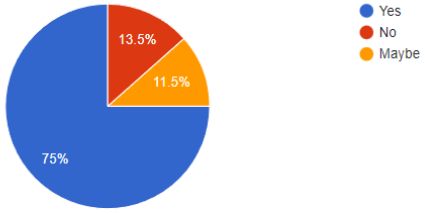
As a user, do you want your data to be analyzed and give you suggestions for future results?

52 responses



Would it be more helpful if this kind of platform was available before the pandemic?

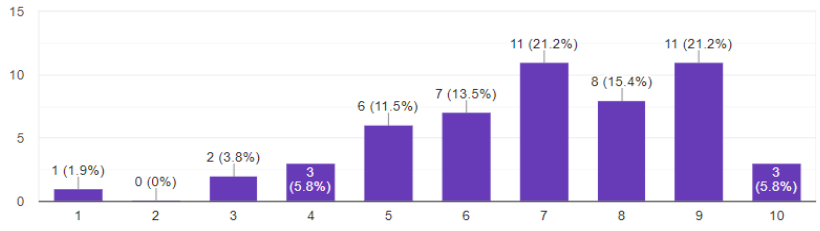
52 responses



Questions Responses 52 Settings

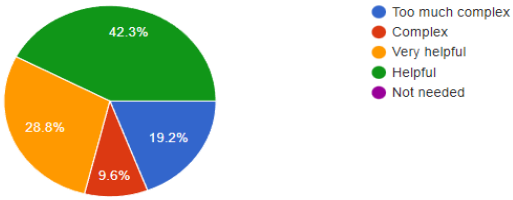
How would you rate other apps based on features?

52 responses



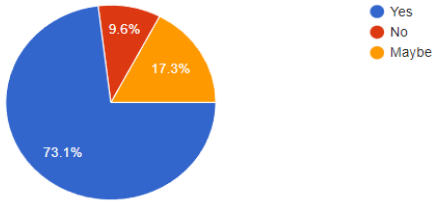
How convenient would it be if the website for the apps are also available?

52 responses



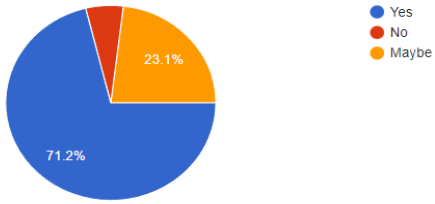
Would you prefer the feature related to pharmacy availability for medical attention?

52 responses



Do you feel the need for vaccinations and medicinal services at your home?

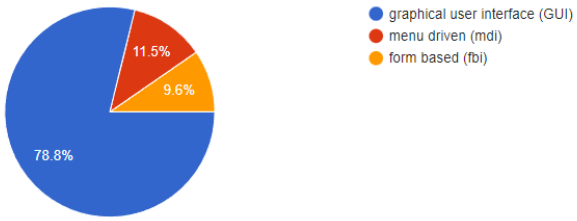
52 responses



Questions Responses 52 Settings

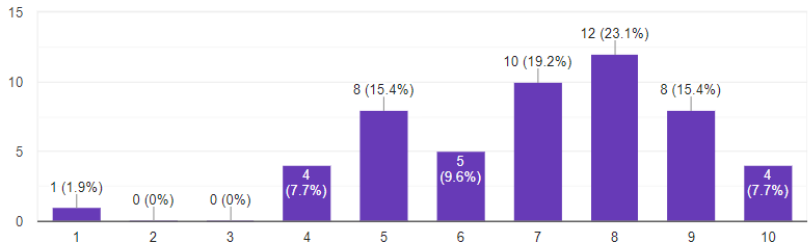
As a user, what type of interface would you prefer?

52 responses



How would you rate other apps based on their interface?

52 responses



Functional Requirements

1. **Patient ID:** Designating an ID to a particular patient after registration so that the system keeps all his/her results and systems interactions accounted for.

2. **Patient Report:** Reported results of either the tests or the vaccination confirmations. These are allocated to the patients after the service providers allow it.

3. **Covid Tracker:** The number of people which have been recorded being infected by either the app or adding other sources. This will help the system showcase real-time tracking results.

4. **Covid Status:** A particular patient's Covid status is shown. These can be shown as: Not Vaccinated, Possible Infection, Infected, Single Dose Vaccinated, Fully Vaccinated, Booster Applied.

5. **Response System:** How fast the system is responding to the patient activity on the app. It also may help keep a record of patient activity and health.

6. **Connecting to Doctor:** Some of the features may require direct or indirect help for the respective doctor to provide. Thus, an inline connection as well as availability notification is taken care by this requirement.

7. **Ambulance Options:** During emergency situations, ambulances are brought, patients can allocate what sort of resources they want in the ambulance according to what he/she requires.

8. **GUI:** Interface which will help to navigate through the app screens.

9. **Directed Help:** When additional help of the service providers are required directly in some sort. These only happen direct contact is required for some confirmation or action.

Non-Functional Requirements

1. Response Speed: Time between input given and output provided. Having a faster rate keeps the system running smooth for the user and really helpful during emergencies.

2. Data & Network Security: Data is a sensitive aspect both for the user and service provider. Securing the databases and networks using firewalls to prevent prevent unauthorized access.

3. Capacity: The capacity of a system refers to the amount of storage it offers. When using some applications, users can adjust and save settings based on their preferences.

4. Localization: A localized application has features that match the geographical location of its users, including aspects such as: languages, currencies, measurements, etc.

5. Portability: Portability means how effectively a system performs in one environment compared to another.

Traceability Table:

Features/Requirements	R1	R2	R3	R4	R5	R6	R7	R8	R9
F1-Registration	X								
F2-Login	X							X	
F3-Covid Check Questions			X	X	X				
F4-Covid Certificates		X	X	X					
F5-Covid Test					X				
F6-Test Appointment Schedule								X	
F7-Test Appointment Selection	X			X					
F8-Covid Vaccination					X				
F9-Vaccination Schedule								X	
F10-Vaccination Selection	X			X				X	
F11-AI Customer Chat						X			X
F12-Emergency Reach/Call					X		X		X
F13-Helpline Number					X	X			X
F14-Switch User/Logout	X								

Features

1. Registration

-R1- Every user has to enter their own details and also use their respective email addresses and phone numbers.

2. Login

-R1- The user has an inert ID in the system which activates when logging through his/her particular username and password.
-R8- The interface will direct if any problems are during log in. If problems arise it will direct in particular solution models.

3. Covid Check Questions

-R3- After the questionnaire ends if the patient is found to be infected they are added onto this directory.
-R4- When tracked as infected(possibly), status can be changed to 'Possibly Infected'.
-R5- The response gets recorded and notified to service providers for further actions.

4. Covid Certificates

-R2- Reporting is completed in this feature where when the patient has completed a number of vaccinations, their certificates are provided.
-R3- When doubly vaccinated tracker will recheck the patient for infection with Feature 3 and then clear her number off the tracker.
-R4- Similar as above, changing her marked status.

5. Covid Test

-R5- When the person is not infected or vaccinated the system will direct them to proceed towards a test.

6. Test Appointment Schedule

-R8- The interface will help in moving through particular locations and particular timings for these.

7. Test Appointment Selection

-R1- When confirming the selected date and location for the appointment, the inert ID is also allocated here.
-R4- After the test is completed, the service providers may or may not change the person's status.

8. Covid Vaccination

-R5- When the person is not vaccinated the system will direct them to proceed towards their vaccinations or the booster.

9. Vaccination Schedule

-R8- The interface will help in moving through particular locations and particular timings for these.

10. Vaccination Selection

-R1- When confirming the selected date and location for the vaccination center, the inert ID is also allocated here.
-R4- After the vaccination is completed, the service providers may or may not change the person's status.
-R8- If any assortments are to be viewed about the appointment, the interface will provide the details.

11. AI Customer Chat

-R6- If the customer requires particular details of the doctor's schedule or availability.
-R9- If the AI chatbot understands that the customer's problems can't be solved remotely they are directed to the call handlers.

12. Emergency Reach/Call

-R5- Quickly the decisions are to be made by the system during emergency and direct them properly.
-R7- Emergency Ambulance options can be shown here according to the patient's requirements.

-R9- If direction from the service providers are required.

13. Helpline Number

-R5- The system needs to understand what the customer what to do and direct their call to the respective problem handlers of various locations.
-R6- A doctor's direct consultation can be required.
-R9- Customer Service can provide various solutions ranging from immediate to complex ones.

14. Switch User/Logout

-R1- If switching the inert ID will change. If logging out the ID will deactivate until next activation.

INDEX

Project Introduction

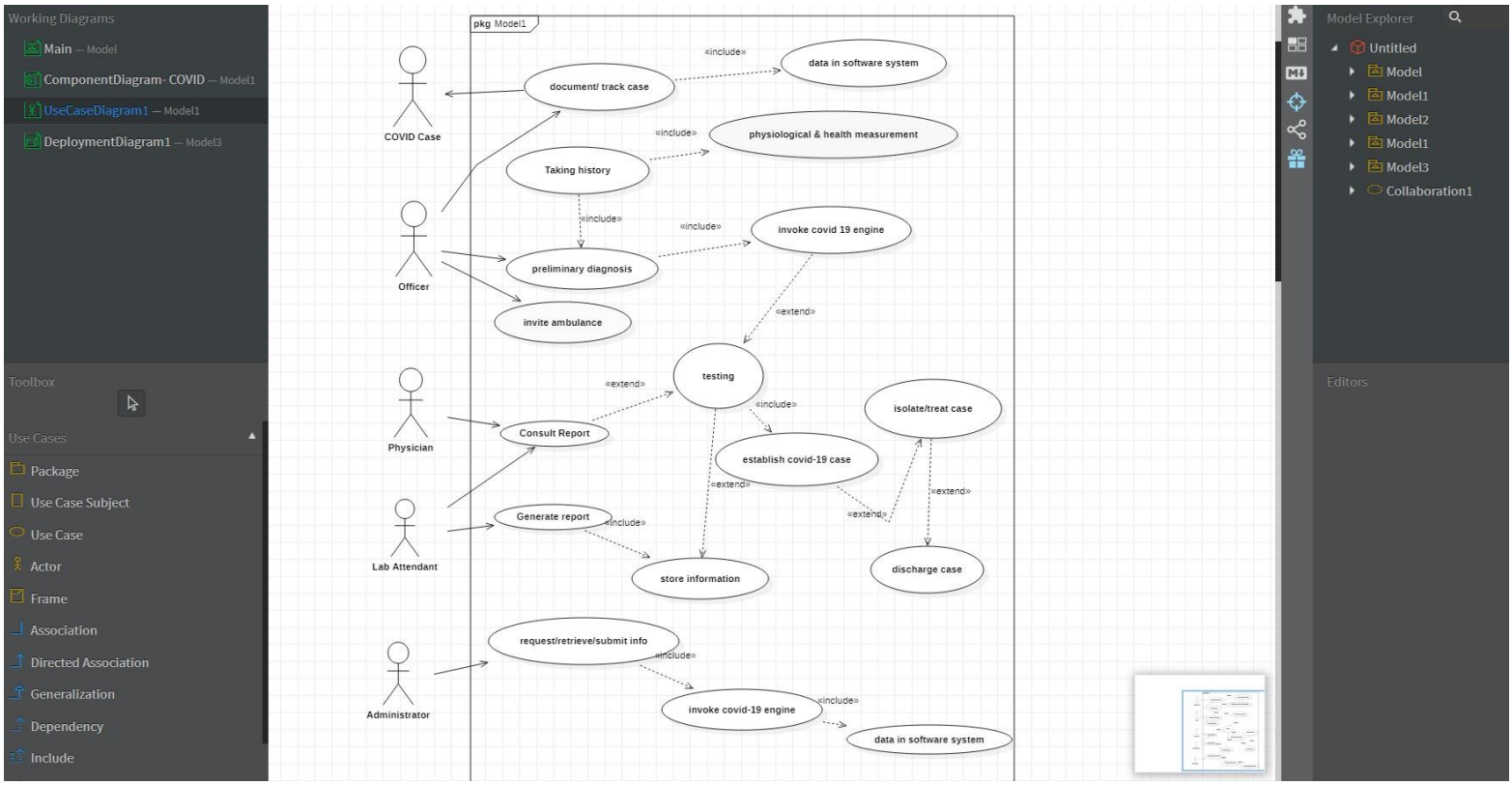
This project is a COVID Tracking and Treatment System that takes into account the world's current demands during the epidemic.
This project's application functions without the need of a complicated database management system or a complex graphical user interface. It employs a straightforward command line interface. This system's users are individuals who are suspected of being infected with the new coronavirus. Following this, the user has access to a variety of procedures that may be employed as needed. This is covered in full in the UML diagrams. Following down the potential user, tracking his past, testing him/her, noting the outcomes, and recording them in software are some of the activities.

Software used

The programme StarUML is used to build the project's UML (Unified Modeling Language) diagrams. StarUML is an open source software modelling tool that works with the UML framework to model systems and applications. A UML diagram's aim is to graphically portray a system together with its primary players, roles, actions, and classes in order to better comprehend and document information about the system.

Use-case Diagram

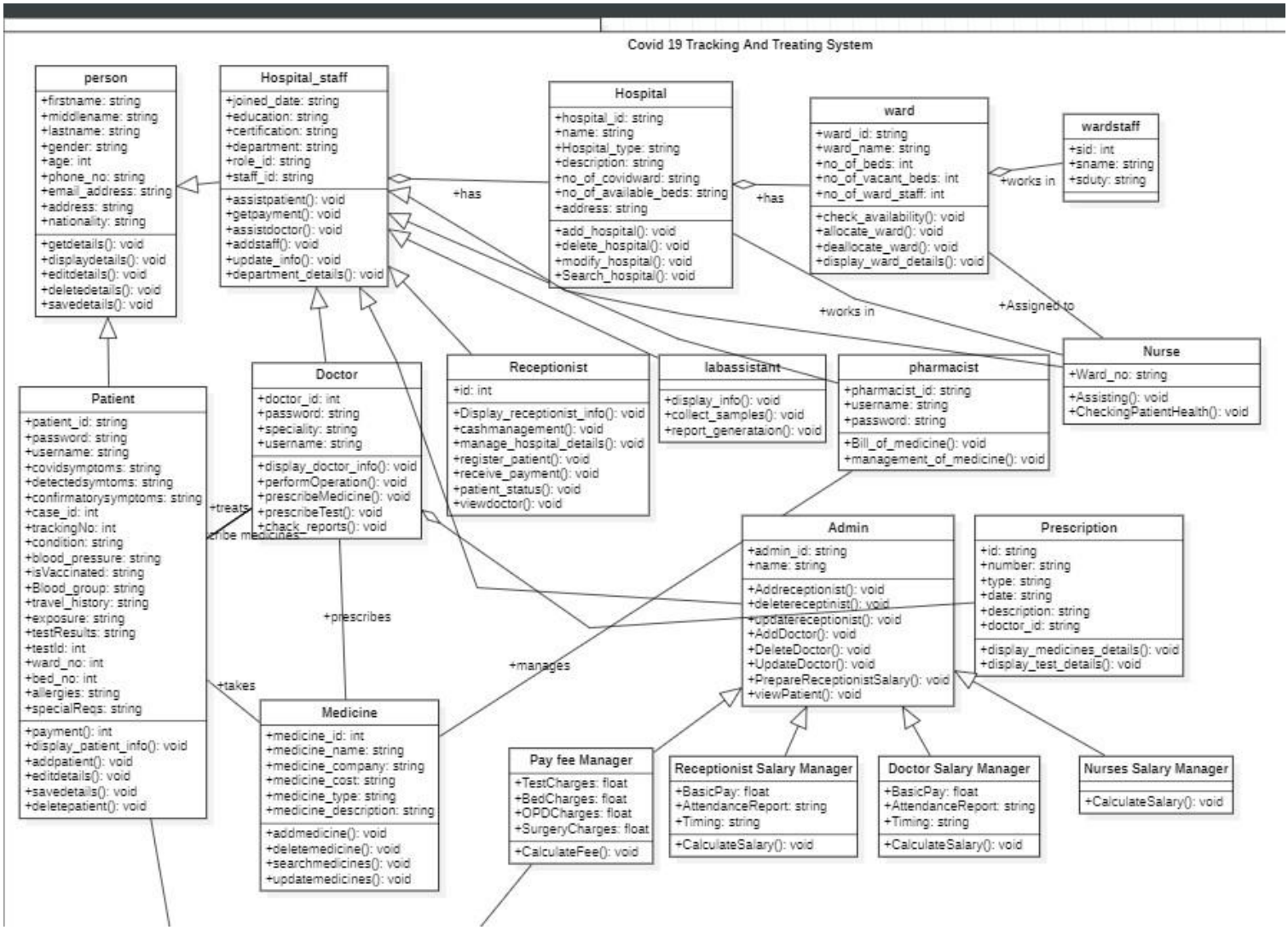
Actors, use cases, and their relationships are depicted in use case diagrams. The diagram is used to represent an application's system/subsystem. A use case diagram depicts a certain system capability. Here is a snapshot of the same, with notations on the left.



This Use Case graphic describes the operation's progression. The case is first documented/recorded, then the data is saved in the appropriate programme, the history is monitored, and physiological measures are done. The early phases of testing are then initiated, and as part of this, the ambulance is summoned and the results are recorded. The individual is next tested, and the results of his tests are reviewed. If the guy tests negative, he is promptly discharged. Otherwise, he is isolated, treated, and discharged when he tests negative. These dates are saved in the relevant database and hence appear in the country/state statistics.

Class Diagram

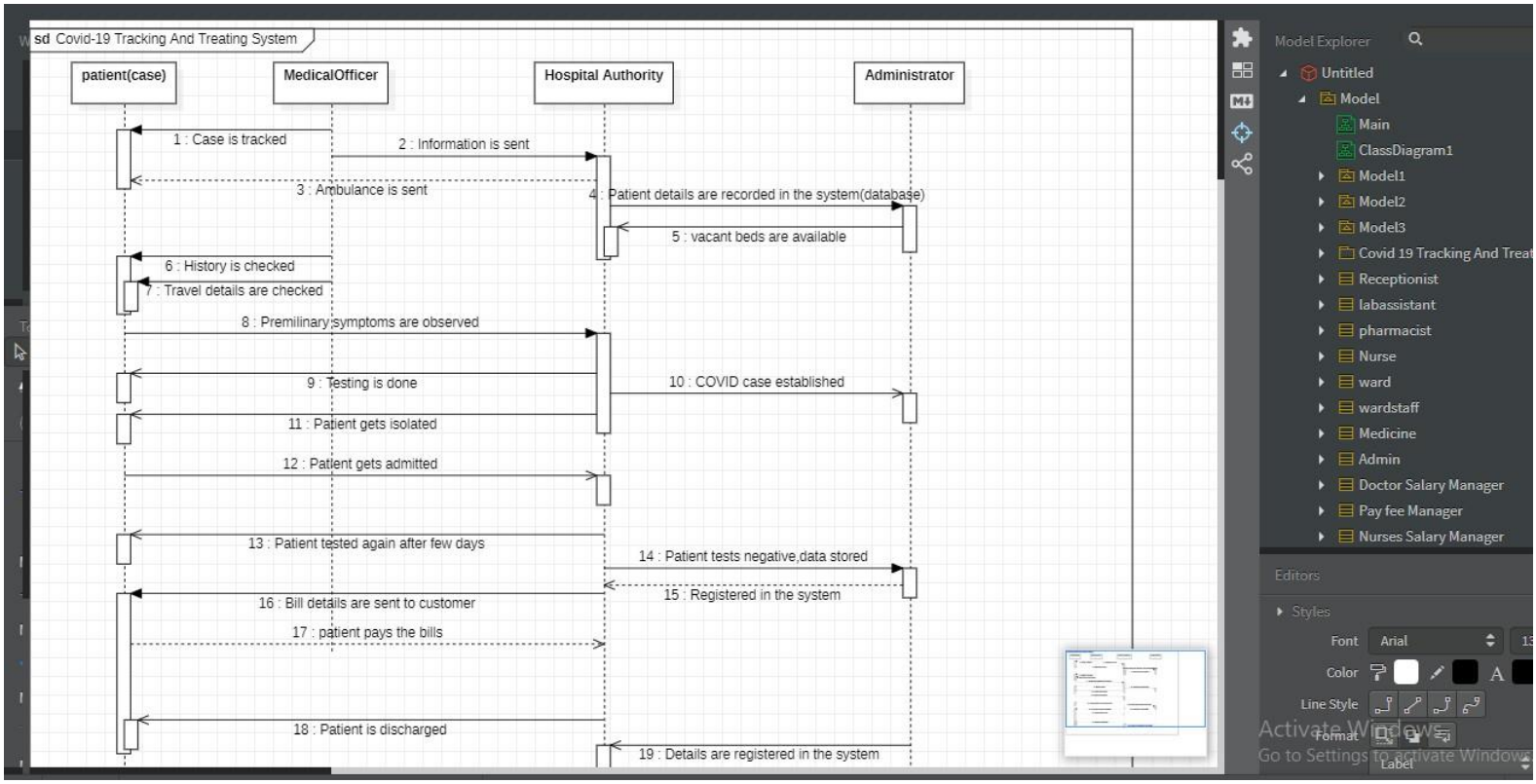
The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. Hereafter is the screenshot of the same.



The classes person,hospital_staff,hospital,ward,ward_staff,patient,doctor,receptionist,lab_assistant,pharmacist,nurse,admin,prescription,medicine,pay_fee_manager,receptionist_salary_manager,nurse_salary_manager,doctor_salary_manager.

Sequence Diagram

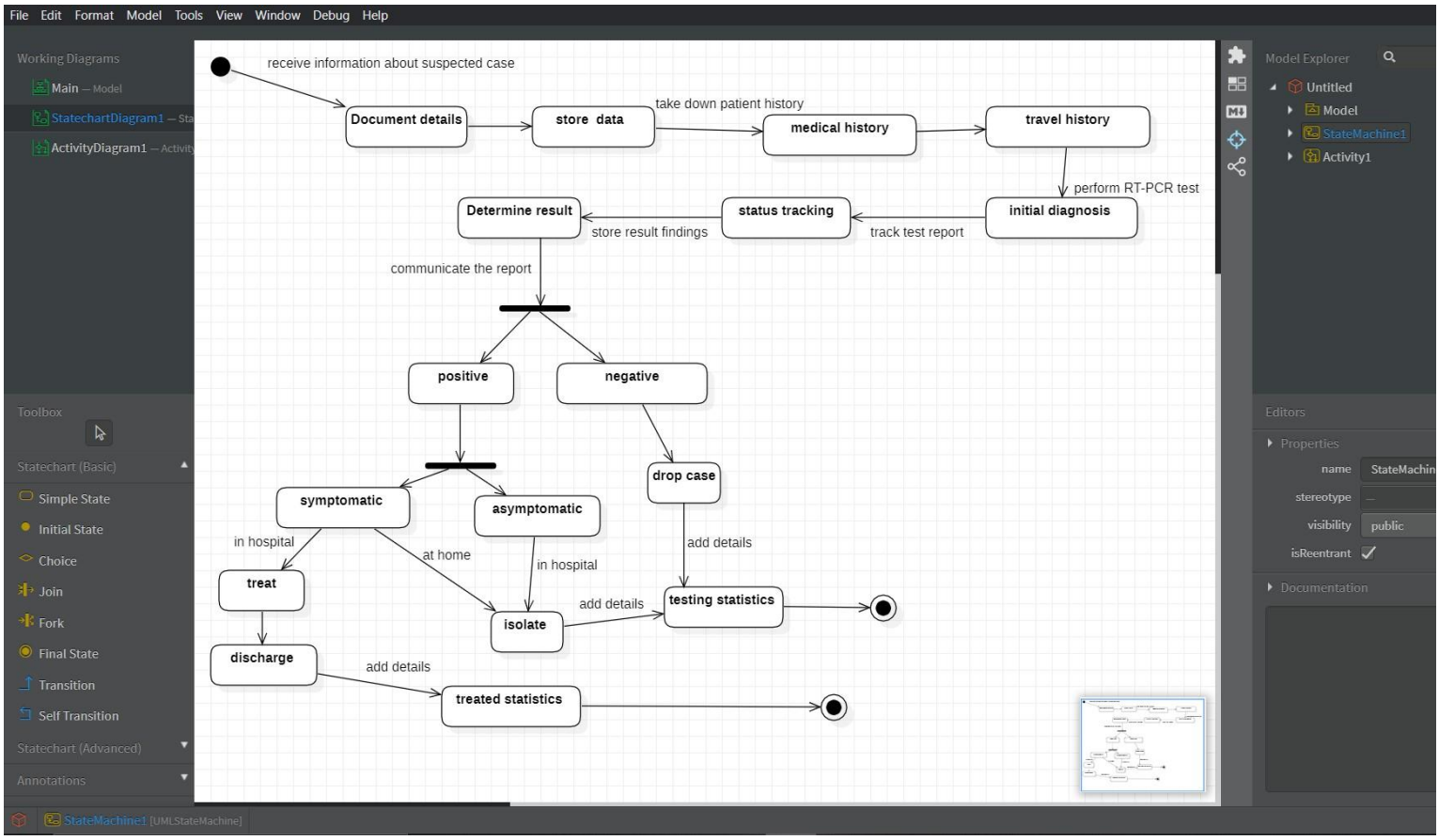
A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function.



The sequential approach to our model is explained in the above diagrams with numbers labelled from 1 to 15.The notations are given on the left. The sequence diagram is distributed being the 4 actors/ cases, namely- patient(cases), medical officer, the hospital authority and the admin behind the database.

State-Transition Diagram

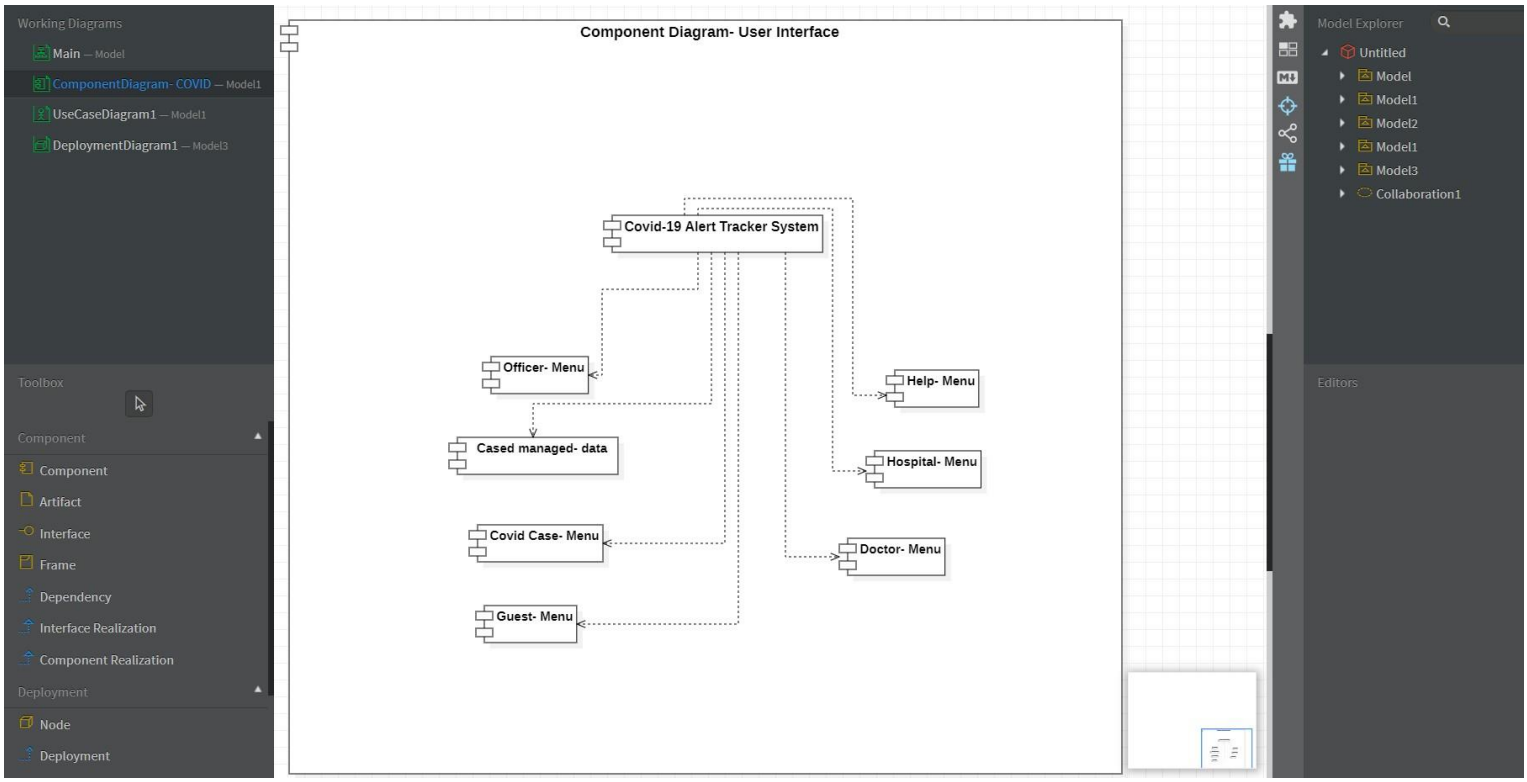
It describes all of the states that an object can have, the events under which an object changes state , the conditions that must be fulfilled before the transition will occur, and the activities undertaken during the life of an object. Here is a snapshot of the same, with notations on the left.



The above diagram shows a flowchart on how the data will flow,keeping in mind all the situations that might occur and choices a patient might take.for example-if he is positive for covid and wishes to isolate at home, instead of being admit in hospital and how details of patient will be taken and recorded.

Component Diagram

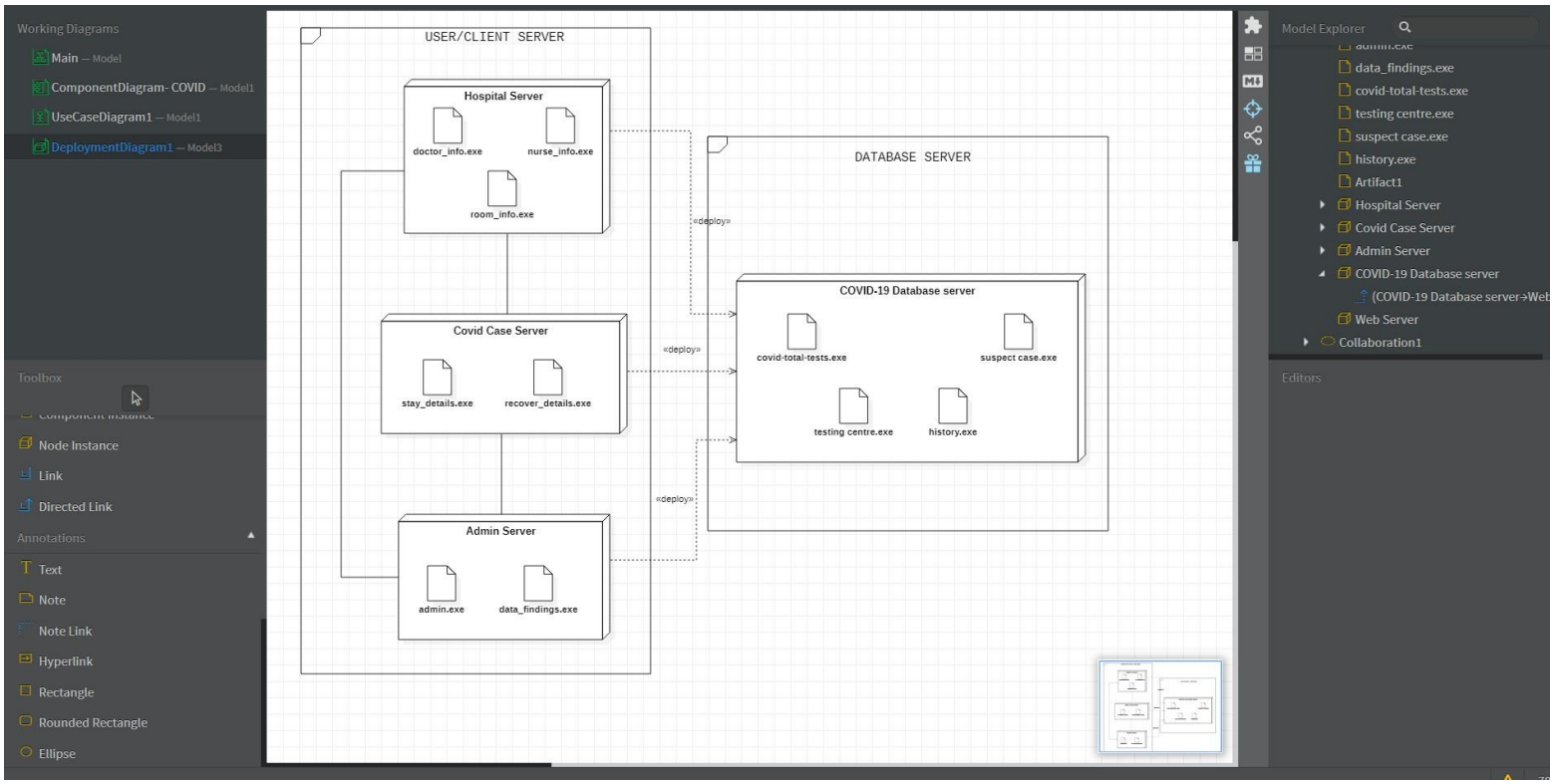
A component diagram, often known as a UML component diagram, illustrates how physical components in a system are organised and wired together. Component diagrams are frequently used to assist model implementation details and double-check that planned development covers all part of the system's essential functions.



The structural components of our system are depicted in the component diagram above. The basic components are kept in packages, which will assist us in redirecting to the construction of this tracking and treatment system. The primary components are the Officer, Cases handled, the number of positive cases, the visitor on the menu, and the Hospital personnel, which includes the admin, physicians, and nurses.

Deployment Diagram

A deployment diagram is a sort of UML diagram that depicts a system's execution architecture, containing nodes such as hardware or software execution environments and the middleware that connects them. Deployment diagrams are commonly used to depict a system's physical hardware and software. The physical structure of our simulated system is given below.



We have three primary nodes: the Hospital Server, the COVID Case Server, and the Admin Server, all of which are connected to a single parent node, the COVID-19 database server. These contain artifacts/extension files with information about that specific user/data.

Explanation of these files:

doctor info.exe- keeps a record of all the physicians and authorities involved.

nurse info.exe- keeps track of all nurses who are supporting their corresponding physicians.

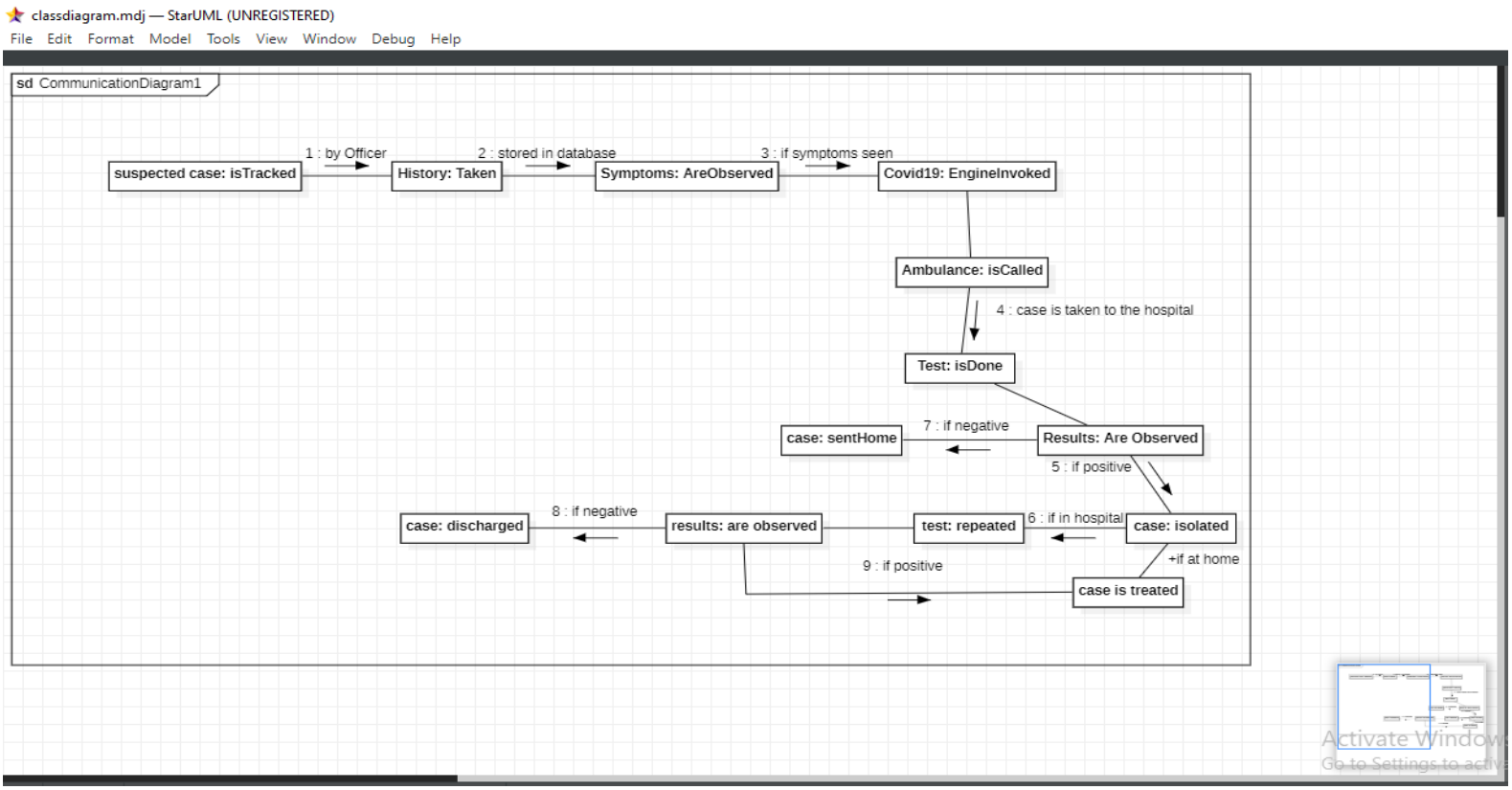
room info- keeps track of vacated/in-use rooms, as well as their statuses (deluxe, basic), and pricing.

Admin.exe- the login gateway for the administrator, who is responsible for managing all database entries and records for statistical and analytic reasons.

data findings.exe- the findings observed in several patients in order to reach a pharmaceutical conclusion
covid total test.exe- Records the total number of tests performed at the hospital on a daily/weekly/monthly basis.
testing center.exe- stores information regarding the hospital's affiliated labs, government go-through centres, and so on.
Suspectcase.exe- holds information about those who have been tracked and are suspected to be potential cases of the novel coronavirus

Collaboration Diagram

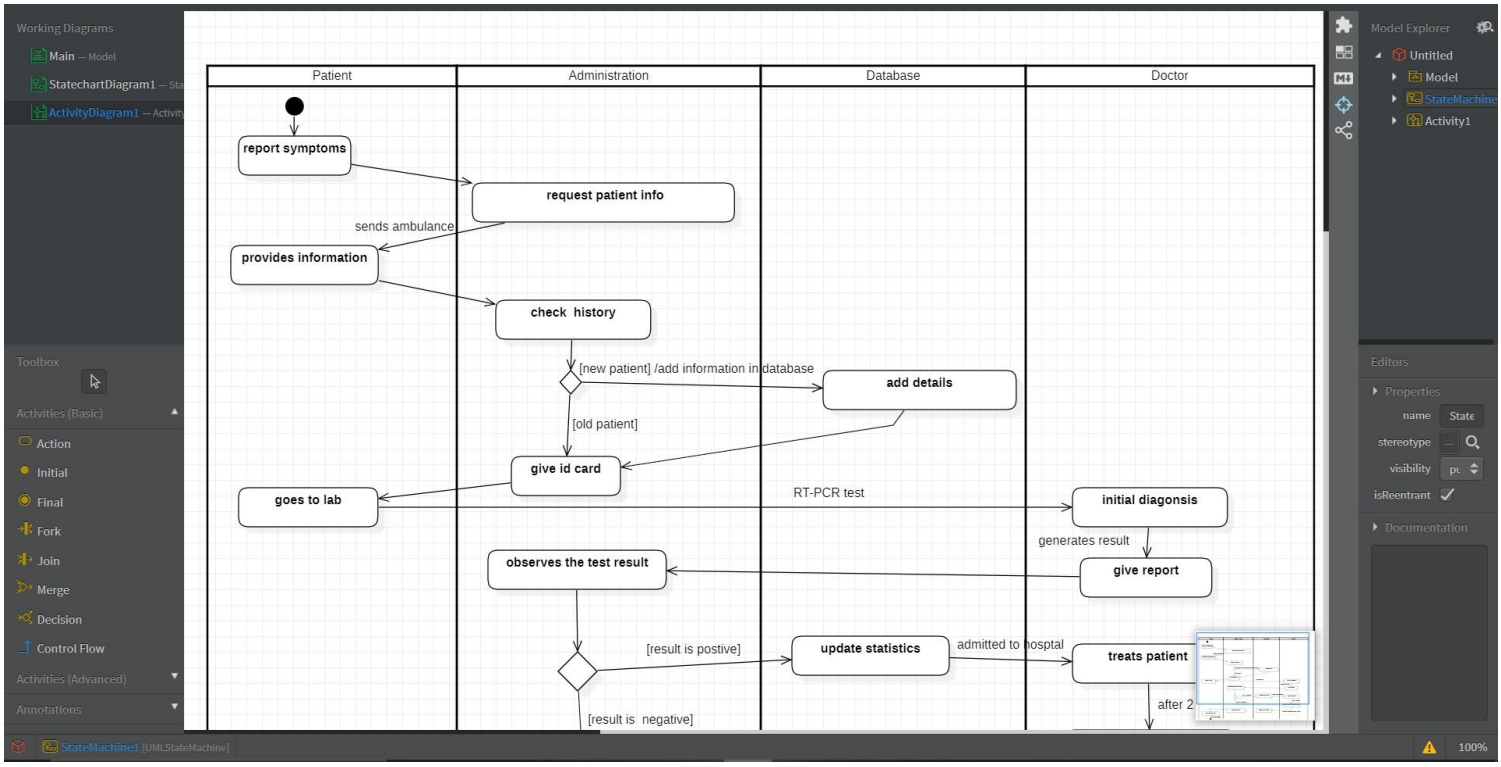
A collaboration diagram is a type of visual presentation that shows how various software objects interact with each other within an overall IT architecture and how users can benefit from this collaboration. A collaboration diagram often comes in the form of a visual chart that resembles a flow chart. Hereafter we have shown our collaboration diagram.

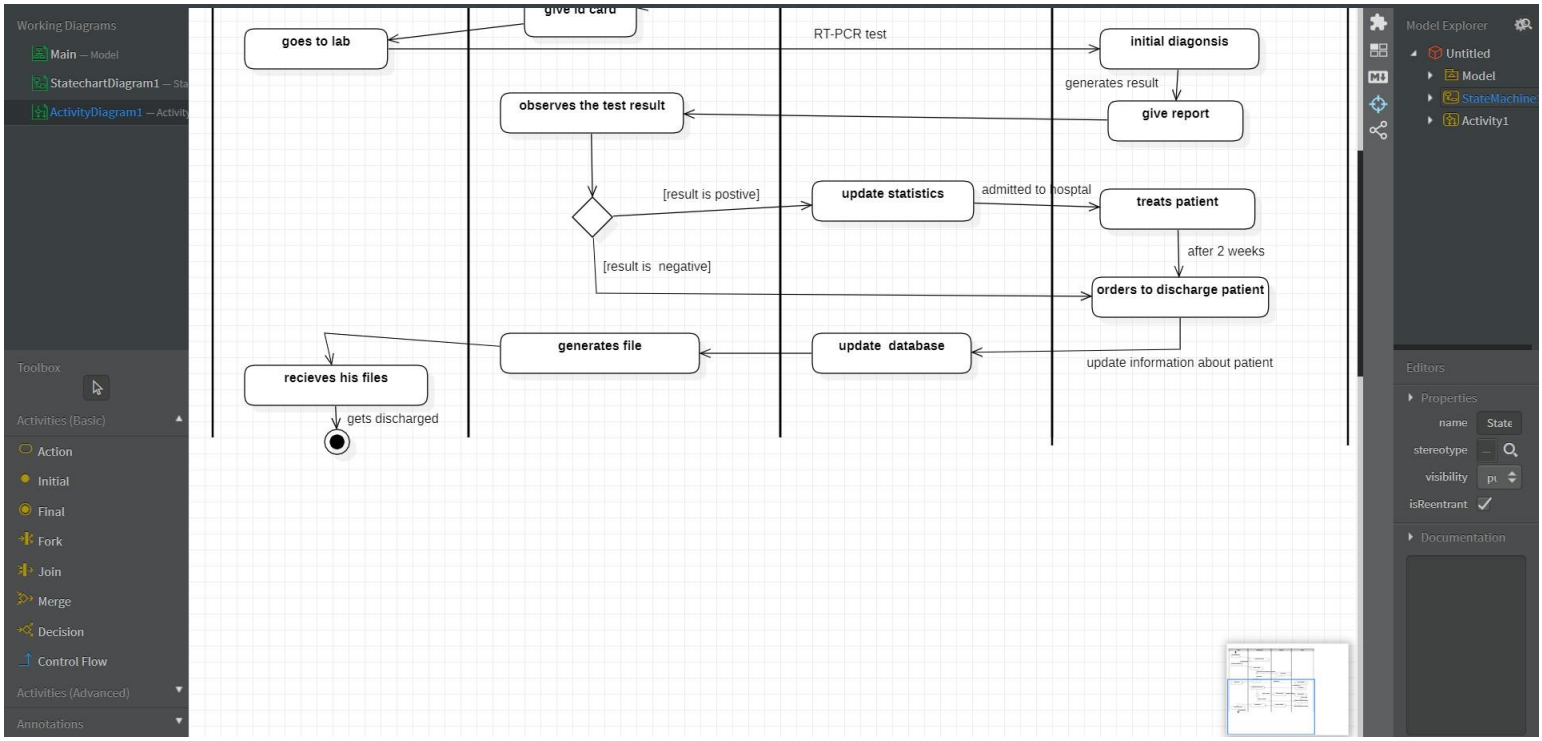


In the above flow diagram, it is seen that the collaboration diagram helps us understand object architecture within a system rather than the flow of a message in a sequence diagram. An object is an entity that has various attributes associated with it. It is a concept of object oriented programming. There are multiple objects present inside an object-oriented system where each object can be associated with any other object inside the system. When we look up to the lifelines, they are all followed in order as seen in a flowchart. The messages help us understand how does the next step proceed. This goes from tracking the case, their history, jotting the symptoms, calling the ambulance and eventually testing and treating the case.

Activity Diagram

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. Hereafter we have shown the activity diagram. The notations are on the left.





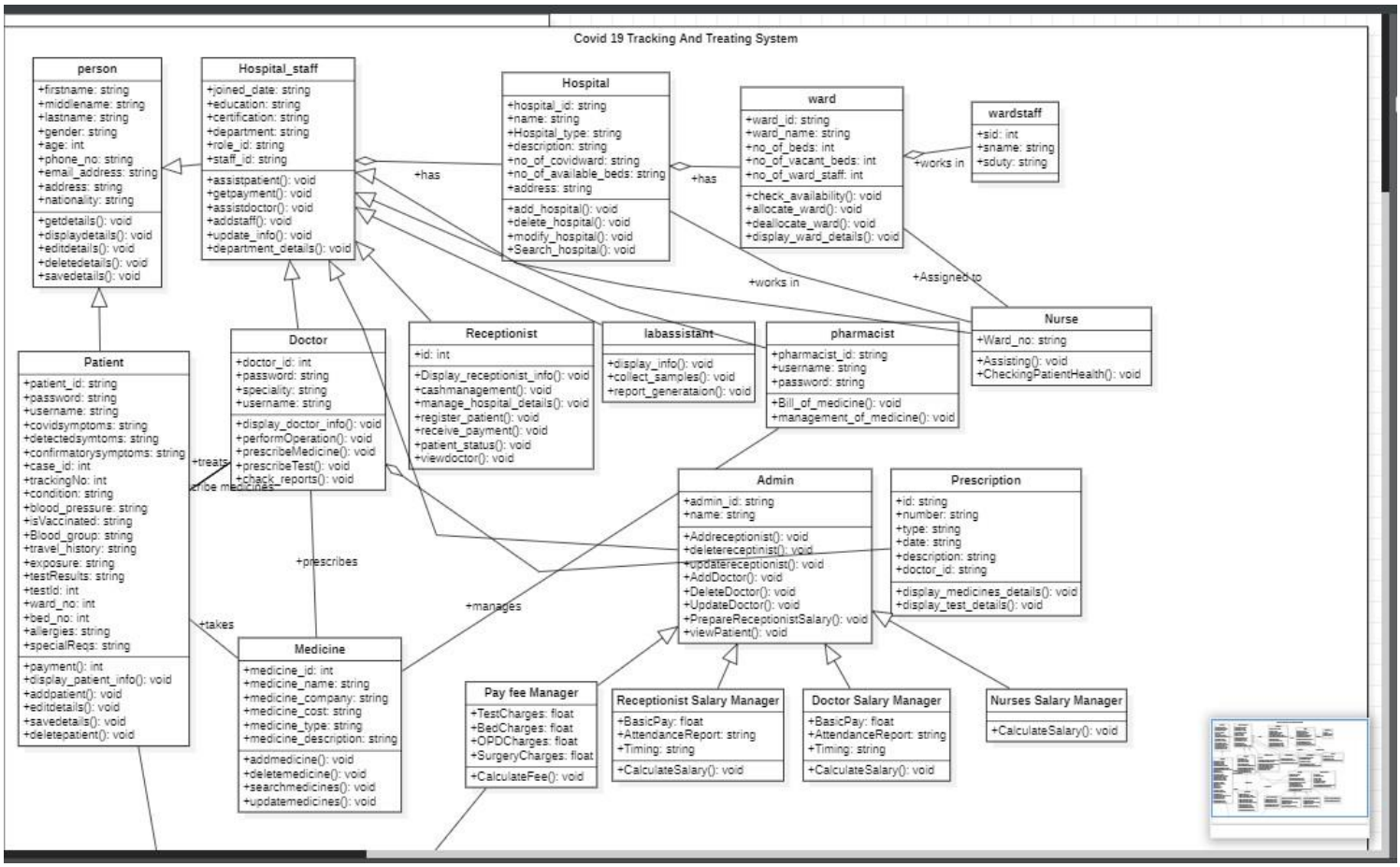
The above activity diagram has been drawn to show a flow of the activities/the processes involved in the stages of testing. This can be looked upon as a swimlane diagram that goes from tracking to tracing to treating and finally discharging and storing details and how the patient,doctor and administrator of the hospital is involved in the same.

Code-generation

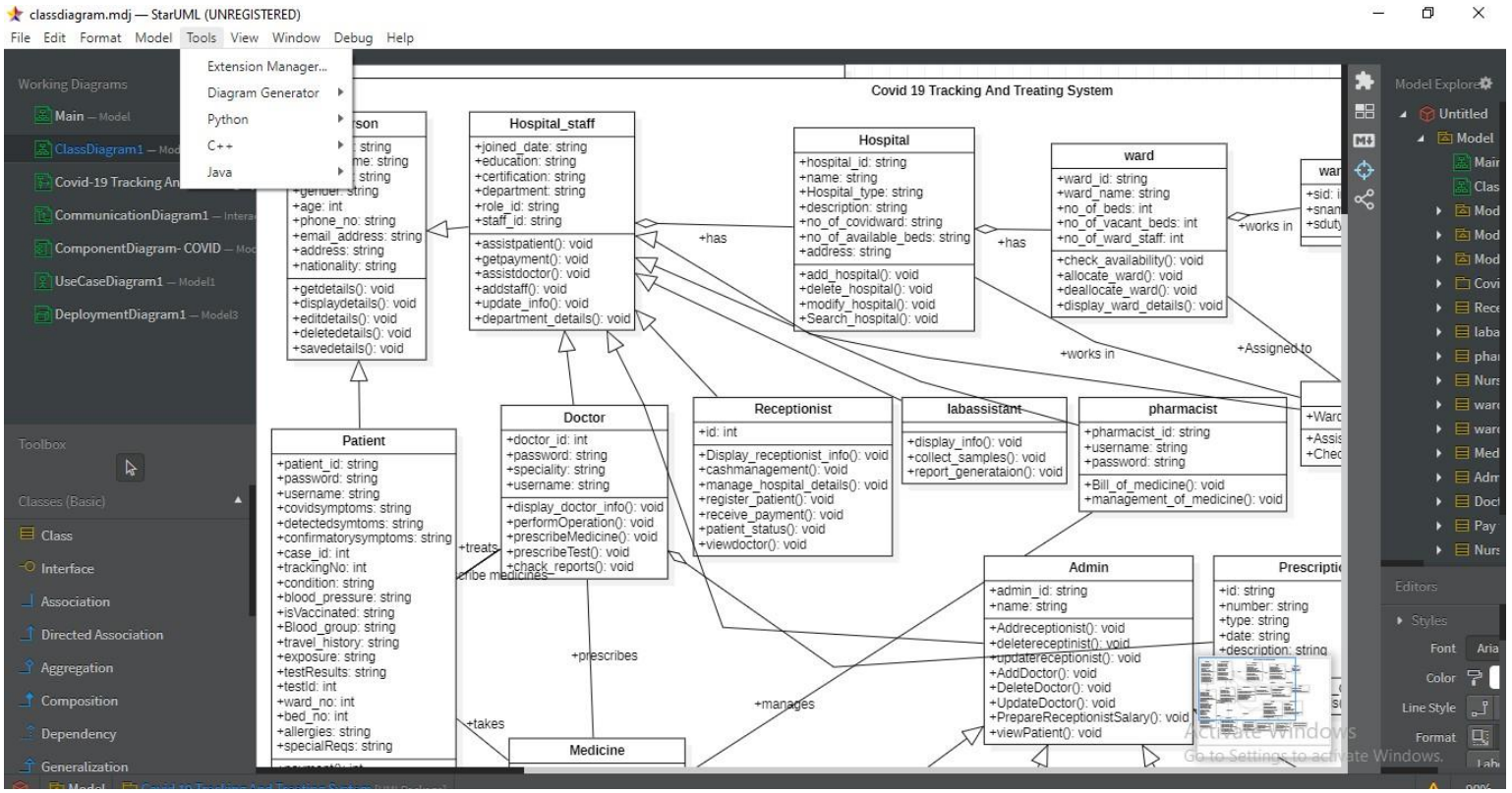
Steps:

As stated earlier, the codes have been generated in JAVA language. The procedure to generate these codes is as follows.

1. Go to your respective class/ component diagram- whichever you wish to generate the code for.



2. Click on tools

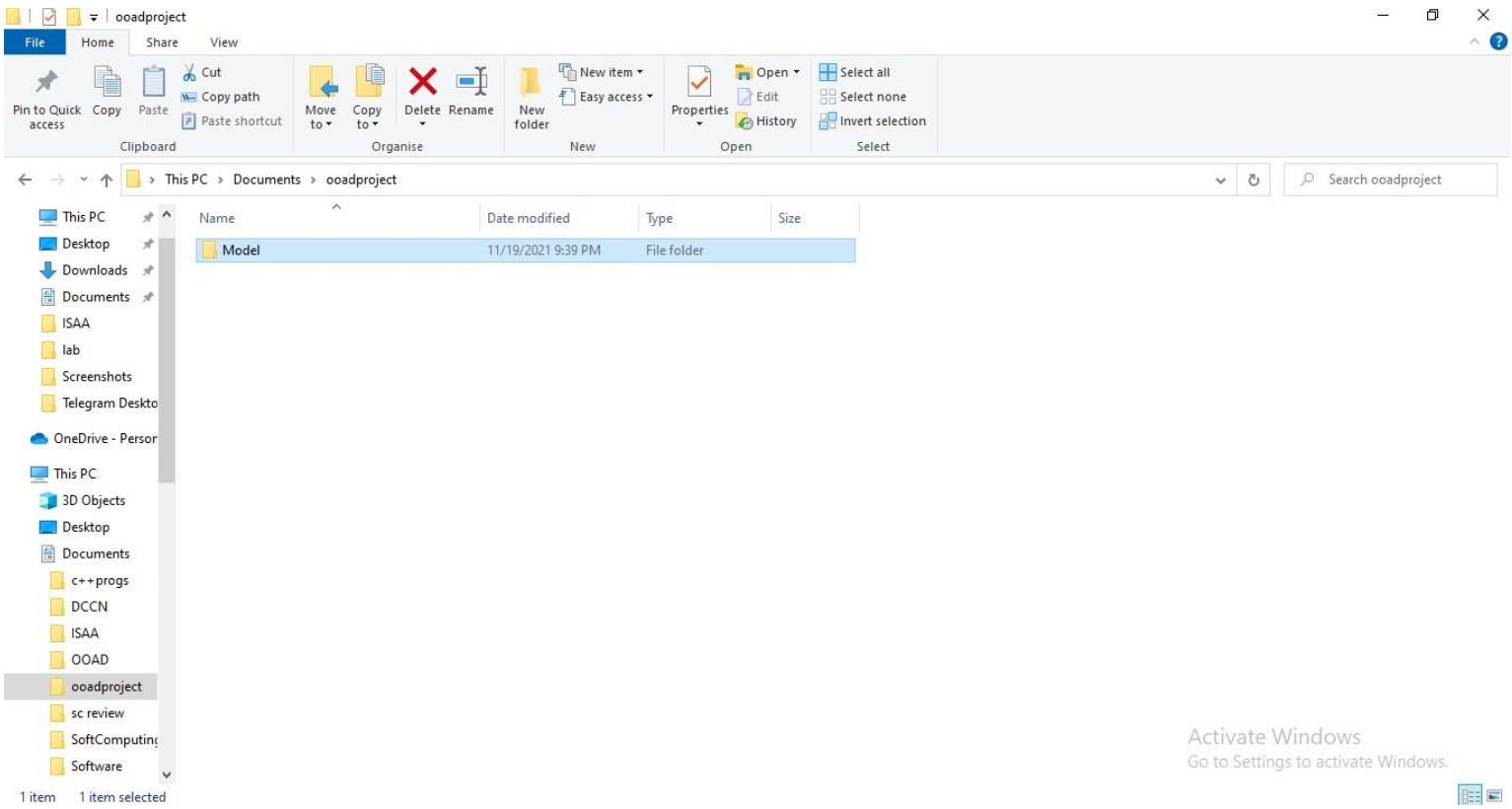


3. Go to extension manager

4. A registry opens up-type in the language that you wish to install.

5. Install the URL extension for that particular language.

6. A pop-up opens up, asking you to create a folder wherein you wish to save your commands.I created a folder, namely ooad_project and selected insert.



7. The respective codes are automatically saved using the model folder and are viewable in whichever app you wish them to see.

The codes for our COVID 19 Tracking and Treating System's Class Diagram is shown hereby:

For patient-

```
package Covid 19 Tracking And Treating System;

import java.util.*;

/**
 *
 */
public class Patient extends person {

    /**
     * Default constructor
     */
    public Patient() {
    }
```

```
/**
 *
 */
public string patient_id;

/**
 *
 */
public string password;

/**
 *
 */
public string username;

/**
 *
 */
public string covidsymptoms;

/**
 *
 */
public string detectedsymtoms;

/**
 *
 */
public string confirmatorysymptoms;

/**
 *
 */
public int case_id;

/**
 *
 */
public int trackingNo;
```

```
/**
 *
 */
public string condition;

/**
 *
 */
public string blood_pressure;

/**
 *
 */
public string isVaccinated;

/**
 *
 */
public string Blood_group;

/**
 *
 */
public string travel_history;

/**
 *
 */
public string exposure;

/**
 *
 */
public string testResults;

/**
 *
 */
public int testId;
```

```
/**
 *
 */
public int ward_no;

/**
 *
 */
public int bed_no;

/**
 *
 */
public string allergies;

/**
 *
 */
public string specialReqs;

/**
 *
 */
public Doctor prescribe medicines;

/**
 *
 */
public Doctor treats;


/**
 *
 */
public Medicine takes;


/**
 * @return
 */
```

```
public int payment() {
    // TODO implement here
    return 0;
}

/**
 * @return
 */
public void display_patient_info() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
public void addpatient() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
public void editdetails() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
public void savedetails() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
```

```
    public void deletepatient() {
        // TODO implement here
        return null;
    }

}
```

For hospital-

```
package Covid 19 Tracking And Treating System;

import java.util.*;

/**
 *
 */
public class Hospital {

    /**
     * Default constructor
     */
    public Hospital() {
    }

    /**
     *
     */
    public string hospital_id;

    /**
     *
     */
    public string name;

    /**
     *
     */
    public string Hospital_type;

    /**
     *
     */
```

```
public string description;

/**
 *
 */
public string no_of_covidward;

/**
 *
 */
public string no_of_available_beds;

/**
 *
 */
public string address;




/**
 * @return
 */
public void add_hospital() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
public void delete_hospital() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
public void modify_hospital() {
```

```
        // TODO implement here
        return null;
    }

    /**
     * @return
     */
    public void Search_hospital() {
        // TODO implement here
        return null;
    }
```

For Doctor-

```
package Covid 19 Tracking And Treating System;

import java.util.*;

/**
 *
 */
public class Doctor extends Hospital_staff {

    /**
     * Default constructor
     */
    public Doctor() {
    }

    /**
     *
     */
    public int doctor_id;

    /**
     *
     */
    public string password;

    /**
     *
```

```
 */
public string speciality;

/**
 *
 */
public string username;




/**
 * @return
 */
public void display_doctor_info() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
public void performOperation() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
public void prescribeMedicine() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
```

```
    public void prescribeTest() {
        // TODO implement here
        return null;
    }

    /**
     * @return
     */
    public void chack_reports() {
        // TODO implement here
        return null;
    }

}
```

For Hospital_staff-

```
package Covid 19 Tracking And Treating System;

import java.util.*;

/**
 *
 */
public class Hospital_staff extends person {

    /**
     * Default constructor
     */
    public Hospital_staff() {
    }

    /**
     *
     */
    public string joined_date;

    /**
     *
     */
    public string education;
```

```
/**
 *
 */
public string certification;

/**
 *
 */
public string department;

/**
 *
 */
public string role_id;

/**
 *
 */
public string staff_id;


/**
 * @return
 */
public void assistpatient() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
public void getpayment() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
```

```
    public void assistdoctor() {
        // TODO implement here
        return null;
    }

    /**
     * @return
     */
    public void addstaff() {
        // TODO implement here
        return null;
    }

    /**
     * @return
     */
    public void update_info() {
        // TODO implement here
        return null;
    }

    /**
     * @return
     */
    public void department_details() {
        // TODO implement here
        return null;
    }

}
```

For prescription-

```
import java.util.*;

/**
 *
 */
public class Prescription {
```

```
/**
 * Default constructor
 */
public Prescription() {
}

/**
 *
 */
public string id;

/**
 *
 */
public string number;

/**
 *
 */
public string type;

/**
 *
 */
public string date;

/**
 *
 */
public string description;

/**
 *
 */
public string doctor_id;


/**
 * @return
 */
```

```
    public void display_medicines_details() {
        // TODO implement here
        return null;
    }

    /**
     * @return
     */
    public void display_test_details() {
        // TODO implement here
        return null;
    }

}
```

For receptionist_salary_manager

```
package Covid 19 Tracking And Treating System;

import java.util.*;

/**
 *
 */
public class Receptionist Salary Manager extends Admin {

    /**
     * Default constructor
     */
    public Receptionist Salary Manager() {
    }

    /**
     *
     */
    public float BasicPay;

    /**
     *
     */
    public string AttendanceReport;
```

```
    /**
     *
     */
    public string Timing;

    /**
     * @return
     */
    public void CalculateSalary() {
        // TODO implement here
        return null;
    }

}
```

For Admin-

```
import java.util.*;

/**
 *
 */
public class Admin extends Hospital_staff {

    /**
     * Default constructor
     */
    public Admin() {
    }

    /**
     *
     */
    public string admin_id;

    /**
     *
     */
    public string name;

    /**
     * @return
```

```
 */
public void Addreceptionist() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
public void deletereceptinist() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
public void updatereceptionist() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
public void AddDoctor() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
public void DeleteDoctor() {
    // TODO implement here
    return null;
}

/**
 * @return
```

```
     */
    public void UpdateDoctor() {
        // TODO implement here
        return null;
    }

    /**
     * @return
     */
    public void PrepareReceptionistSalary() {
        // TODO implement here
        return null;
    }

    /**
     * @return
     */
    public void viewPatient() {
        // TODO implement here
        return null;
    }

}
```

For doctor_salary_manager

```
import java.util.*;

/**
 *
 */
public class Doctor Salary Manager extends Admin {

    /**
     * Default constructor
     */
    public Doctor Salary Manager() {
    }

    /**
     *
```

```
     */
    public float BasicPay;

    /**
     *
     */
    public string AttendanceReport;

    /**
     *
     */
    public string Timing;

    /**
     * @return
     */
    public void CalculateSalary() {
        // TODO implement here
        return null;
    }

}
```

For lab_assistant

```
import java.util.*;

/**
 *
 */
public class labassistant extends Hospital_staff {

    /**
     * Default constructor
     */
    public labassistant() {
    }

    /**
     * @return
     */
    public void display_info() {
```

```
        // TODO implement here
        return null;
    }

    /**
     * @return
     */
    public void collect_samples() {
        // TODO implement here
        return null;
    }

    /**
     * @return
     */
    public void report_generataion() {
        // TODO implement here
        return null;
    }

}
```

For medicine-

```
import java.util.*;

/**
 *
 */
public class Medicine {

    /**
     * Default constructor
     */
    public Medicine() {
    }

    /**
     *
     */
    public int medicine_id;
```

```
/**
 *
 */
public string medicine_name;

/**
 *
 */
public string medicine_company;

/**
 *
 */
public string medicine_cost;

/**
 *
 */
public string medicine_type;

/**
 *
 */
public string medicine_description;



/**
 * @return
 */
public void addmedicine() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
public void deletemedicine() {
```

```
        // TODO implement here
        return null;
    }

    /**
     * @return
     */
    public void searchmedicines() {
        // TODO implement here
        return null;
    }

    /**
     * @return
     */
    public void updatemedicines() {
        // TODO implement here
        return null;
    }

}
```

For nurse-

```
import java.util.*;

/**
 *
 */
public class Nurse extends Hospital_staff {

    /**
     * Default constructor
     */
    public Nurse() {
    }

    /**
     *
     */
    public string Ward_no;
```

```
    /**
     * @return
     */
    public void Assisting() {
        // TODO implement here
        return null;
    }

    /**
     * @return
     */
    public void CheckingPatientHealth() {
        // TODO implement here
        return null;
    }

}
```

Nurse_salary_manager

```
import java.util.*;

/**
 *
 */
public class Nurses Salary Manager extends Admin {

    /**
     * Default constructor
     */
    public Nurses Salary Manager() {
    }

    /**
     * @return
     */
    public void CalculateSalary() {
        // TODO implement here
```

```
        return null;
    }

}
```

Pay_fee_manager-

```
import java.util.*;

/**
 *
 */
public class Pay fee Manager extends Admin {

    /**
     * Default constructor
     */
    public Pay fee Manager() {
    }

    /**
     *
     */
    public float TestCharges;

    /**
     *
     */
    public float BedCharges;

    /**
     *
     */
    public float OPDCharges;

    /**
     *
     */
    public float SurgeryCharges;


    /**
```

```
     * @return
     */
    public void CalculateFee() {
        // TODO implement here
        return null;
    }

}
```

Pharmacist-

```
import java.util.*;

/**
 *
 */
public class pharmacist extends Hospital_staff {

    /**
     * Default constructor
     */
    public pharmacist() {
    }

    /**
     *
     */
    public string pharmacist_id;

    /**
     *
     */
    public string username;

    /**
     *
     */
    public string password;


    /**
```

```
     * @return
     */
    public void Bill_of_medicine() {
        // TODO implement here
        return null;
    }

    /**
     * @return
     */
    public void management_of_medicine() {
        // TODO implement here
        return null;
    }

}
```

Receptionist-

```
import java.util.*;

/**
 *
 */
public class Receptionist extends Hospital_staff {

    /**
     * Default constructor
     */
    public Receptionist() {
    }

    /**
     *
     */
    public int id;

    /**
     * @return
     */
    public void Display_receptionist_info() {
```

```
    // TODO implement here
    return null;
}

/**
 * @return
 */
public void cashmanagement() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
public void manage_hospital_details() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
public void register_patient() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
public void receive_payment() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
public void patient_status() {
```

```
        // TODO implement here
        return null;
    }

    /**
     * @return
     */
    public void viewdoctor() {
        // TODO implement here
        return null;
    }

}
```

Ward-

```
import java.util.*;

/**
 *
 */
public class ward {

    /**
     * Default constructor
     */
    public ward() {
    }

    /**
     *
     */
    public string ward_id;

    /**
     *
     */
    public string ward_name;
```

```
/**
 *
 */
public int no_of_beds;

/**
 *
 */
public int no_of_vacant_beds;

/**
 *
 */
public int no_of_ward_staff;



/**
 * @return
 */
public void check_availability() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
public void allocate_ward() {
    // TODO implement here
    return null;
}

/**
 * @return
 */
public void deallocate_ward() {
    // TODO implement here
    return null;
```

```
    }

    /**
     * @return
     */
    public void display_ward_details() {
        // TODO implement here
        return null;
    }

}
```

Ward_staff-

```
import java.util.*;

/**
 *
 */
public class wardstaff {

    /**
     * Default constructor
     */
    public wardstaff() {
    }

    /**
     *
     */
    public int sid;

    /**
     *
     */
    public string sname;

    /**
     *
     */
    public string sduty;
```

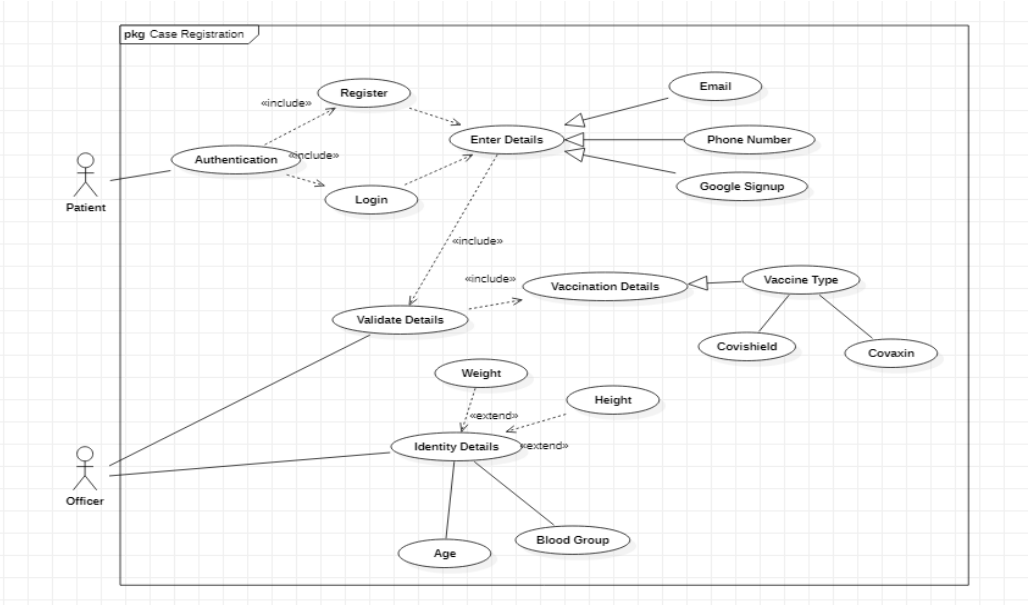
}

CONCLUSION

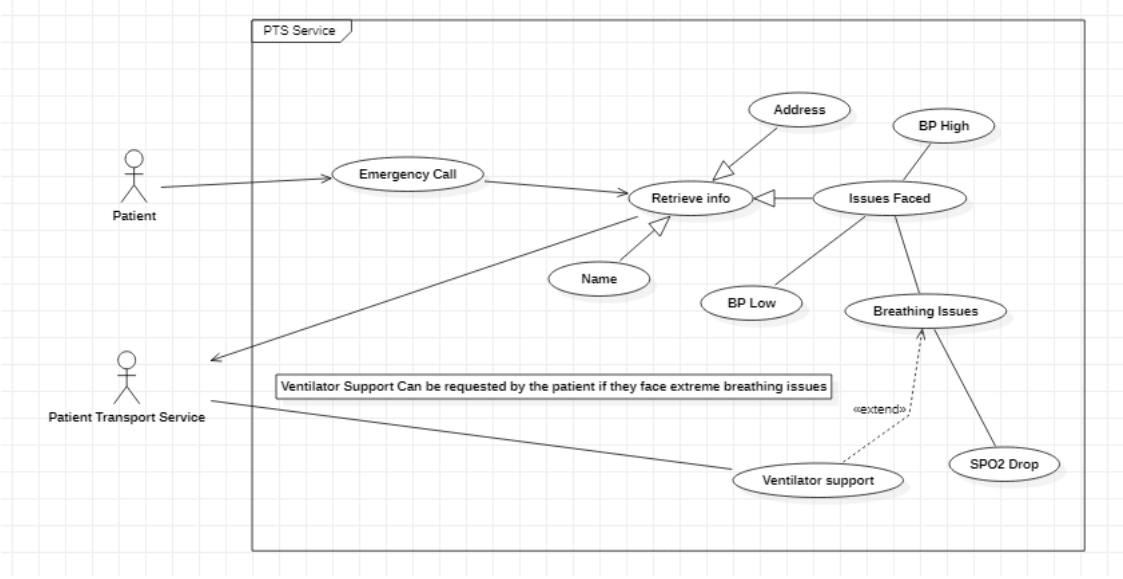
Thus, as seen, the codes for the project COVID-19 Tracking and Treating System have beensuccessfully executed and generated. This project gave an overview on how an efficient management of the COVID-19 patients can be ensured by the use of Object-Oriented Analysis and Design. From the start to the end, the main motive and intention of the system is to make it more visual, something that can explain what each component requires in a lesser amount of
time. By creating a template/outlay of the design mechanism using StarUML and notepad, I havebeen able to express my presentation and modelling. The functional aspects can be incorporated at a later date. We have thereby seen the benefits that OOAD brings to a project and to a designerlike us who have been evaluating/researching specific topics.

Use-Case Specification:

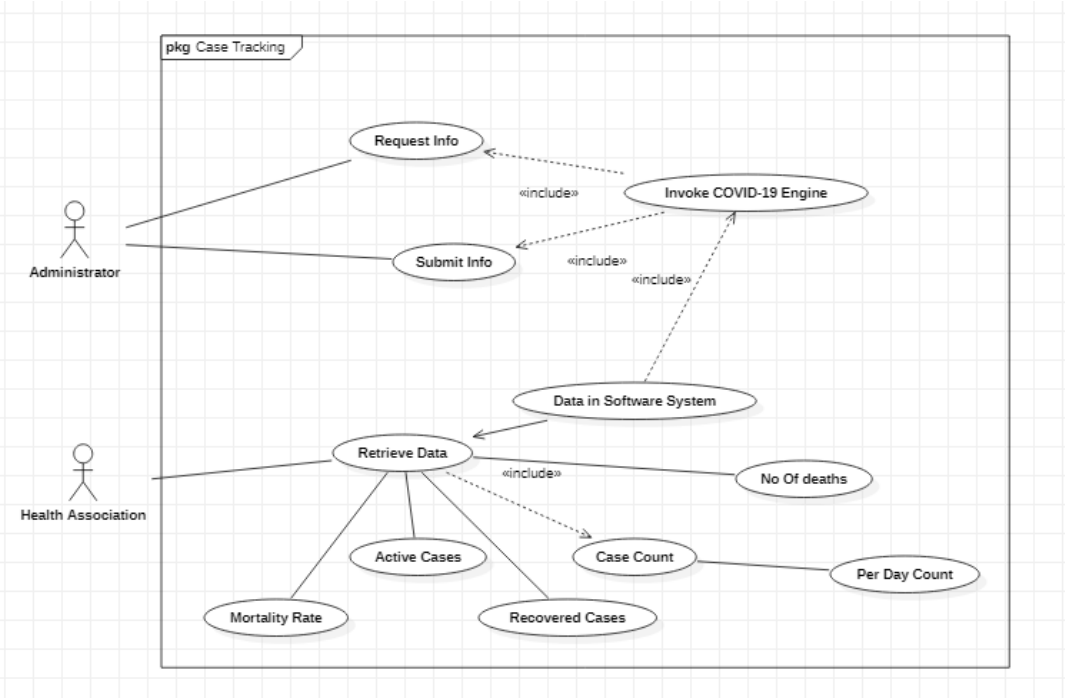
1. Case Registration



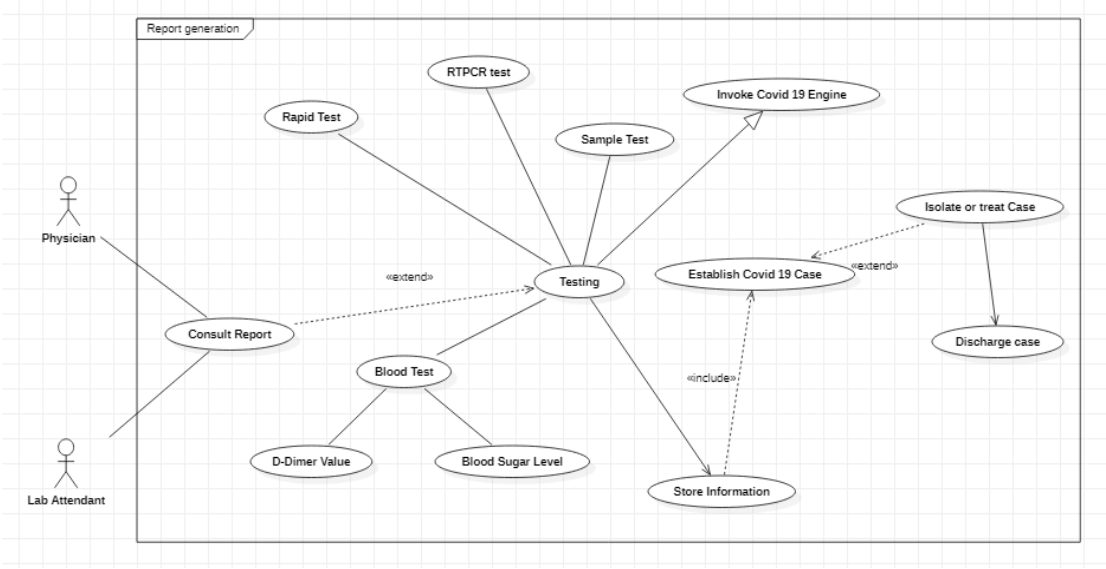
2. PTS Service



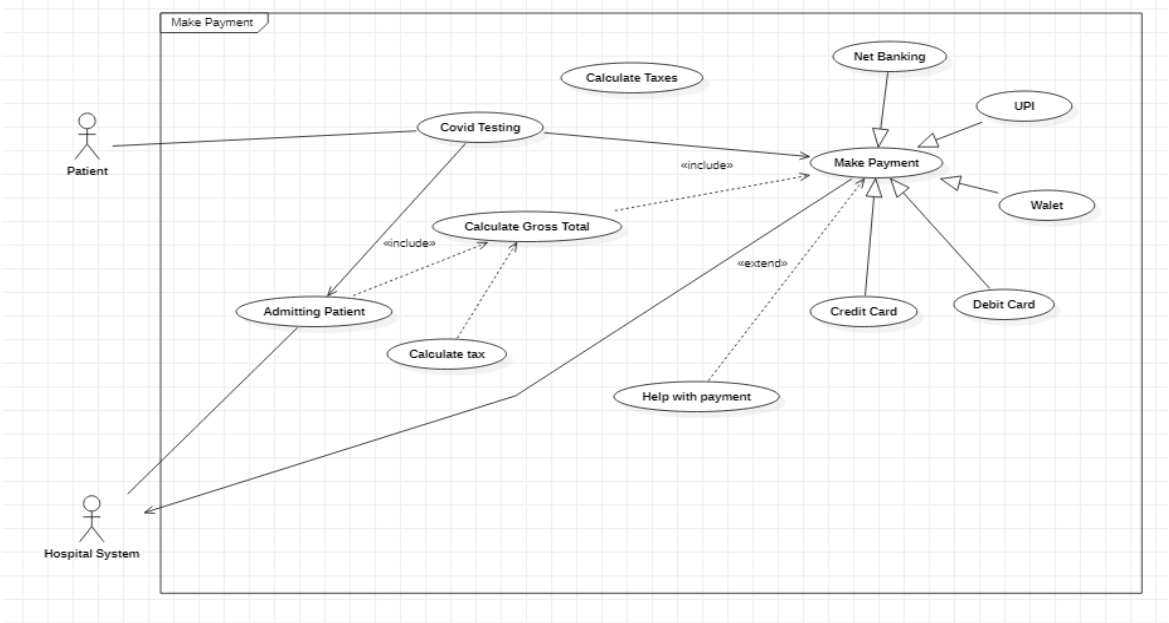
3. Case Tracking



4. Report Generation

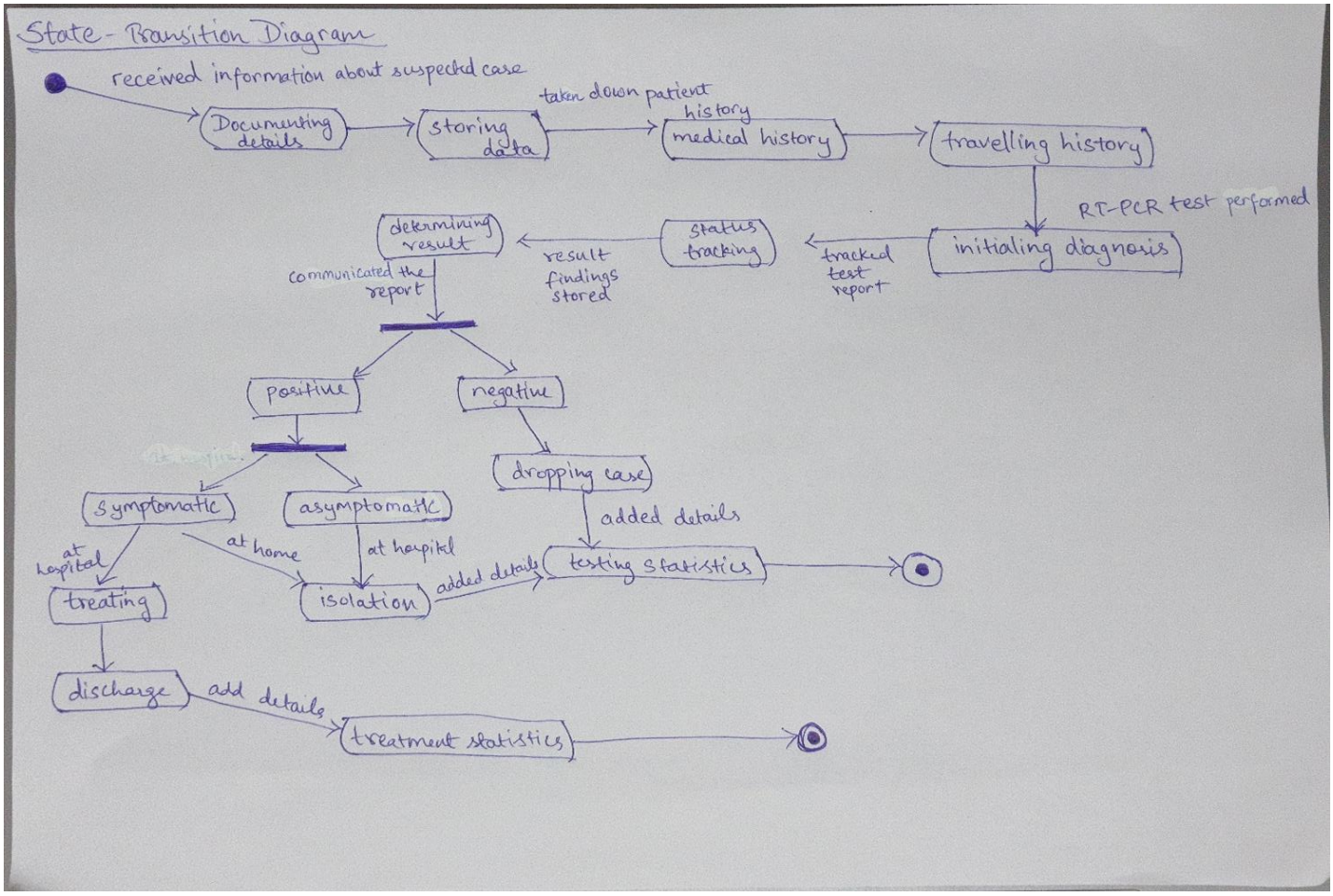


5. Make Payment



Corrections after Review-3

Change in the state transition diagram



Change in the component diagram