JACOBS UNIVERSITY

SOFTWARE ENGINEERING

ASSIGNMENT 1

GROUP MEMBERS

Suraj Giri
Aayush Subedi
Nimesh Acharya
Sandip Kumar Sah
Bidur Niraula

Table of Contents

1. Introduction

1.1 General Idea

The web service for Corona disease management is a service to help manage and restrict the spread of COVID-19. The application has a total of four types of users: Owner, Visitor, Hospital, and Place Owner (Location). The objective of this web service is to provide the facilities to track, record, and analyze COVID infections, and to help in contract tracing.

Visitors, in general, are those users who visit various places. They can scan the place specific unique QR code, which registers the users' data in a database. Place Owners are those users who can register for places which will be visited by the Visitors. To prevent frauds in test results, only Hospitals can publish the test results of the Visitors. Owner is the owner of the software, I.e., the client has the access to the database which contains all the data from Visitors, Place Owners, and Hospitals. Only the Owner also has the capacity to do different manipulations to the data.

To summarize, the web service has unique features that would help contain the spread of COVID-19. The features in detail will be discussed in the Design part of this document.

1.2 Goals and Objective

The main goal of this web application is to find out the analytics about the ongoing COVID-19 pandemic. The web application will be accessible to everyone: from local citizens to policy makers so that the number of infected people would be predicted, infected people could be reached, and isolated and probable infected people could be tested again for verification of the contraction. The whole goal of the application is to get the analytics that would be used by the policy makers to control the spread of COVID-19.

2. Requirements

2.1 User Requirements

Users can log in as an Owner, Visitor, Hospital, or Place Owner. Each user can log in as their respective roles with the combination of Email and Password but before that, they are supposed to sign up with a unique Username, unique Email, and a selection among the four mentioned user types. The Users also have the 'forgot password' option which uses their respective email to send the password recovery link. In addition to these functionalities, the users have the following roles:

2.1.1 Owner

The person or company who asked us to design the application is the owner. The owner gets the analytics from our application and processes it accordingly. The owner has all the rights of this application once it is handed in. The owners must enter the DOB, country, and phone number while signing up. The owner also must verify the hospital once any hospital is registered, after which they will get permission to manipulate data.

2.1.2 Visitor

When a person visits a registered place, S/he is asked to scan a QR code. Upon scanning, depending on if s/he is already registered or not, s/he is asked to fill out a form which is used to create a Visitor user for the visiting person.

2.1.3 Hospital

Hospitals are the users who are authorized to mark the visitor(citizen) as Positive or Negative test result. They can only manipulate the data once their account is verified by the owner. Based on the hospital's mark we need to find the people who were infected and help with contact tracing.

2.1.4 Place Owner

The person who is supposed to own or register a certain location is the place owner. This is a location where one will find a QR code and register that they were present at that same location for some time. Once the owner registers he she will automatically get a specific location based QR code.

2.2 System Requirements

Different users have different functionality and so different system requirements, these are described below:

Visitor:

- A normal user shall download the web app or shall load the website before scanning the QR code.
- When a normal user visits a registered location, s/he scans QR code with unique ID of location
- If the user is not a registered user, he/she shall register using the form that will be generated by the website / app. After the user confirms his sign-up data, the data is sent to the server to create a visitor account for him/her in the database.
- Upon the successful creation of the visitor account, the server sends 200 OK responses.
- If the user who scans the QR code is registered user or after an unregistered user registers to the app data (user ID, name, phone Number, etc.). Date, time, and users' data are sent to the server to store his visit to the place.

Hospital:

- To create a hospital user, the hospital must sign up on the website using the data (license number, username, email, location, etc.), after signing up for the hospital.
- The account of hospital is confirmed by owner and after getting approved, the hospital can carry out test and record it.
- The hospital gets a form to enter the test results of the patient(visitor). The form contains patient ID, name, date and result. By submitting form test record form, the hospital also marks a patient if s/he is infected.

Place Owner:

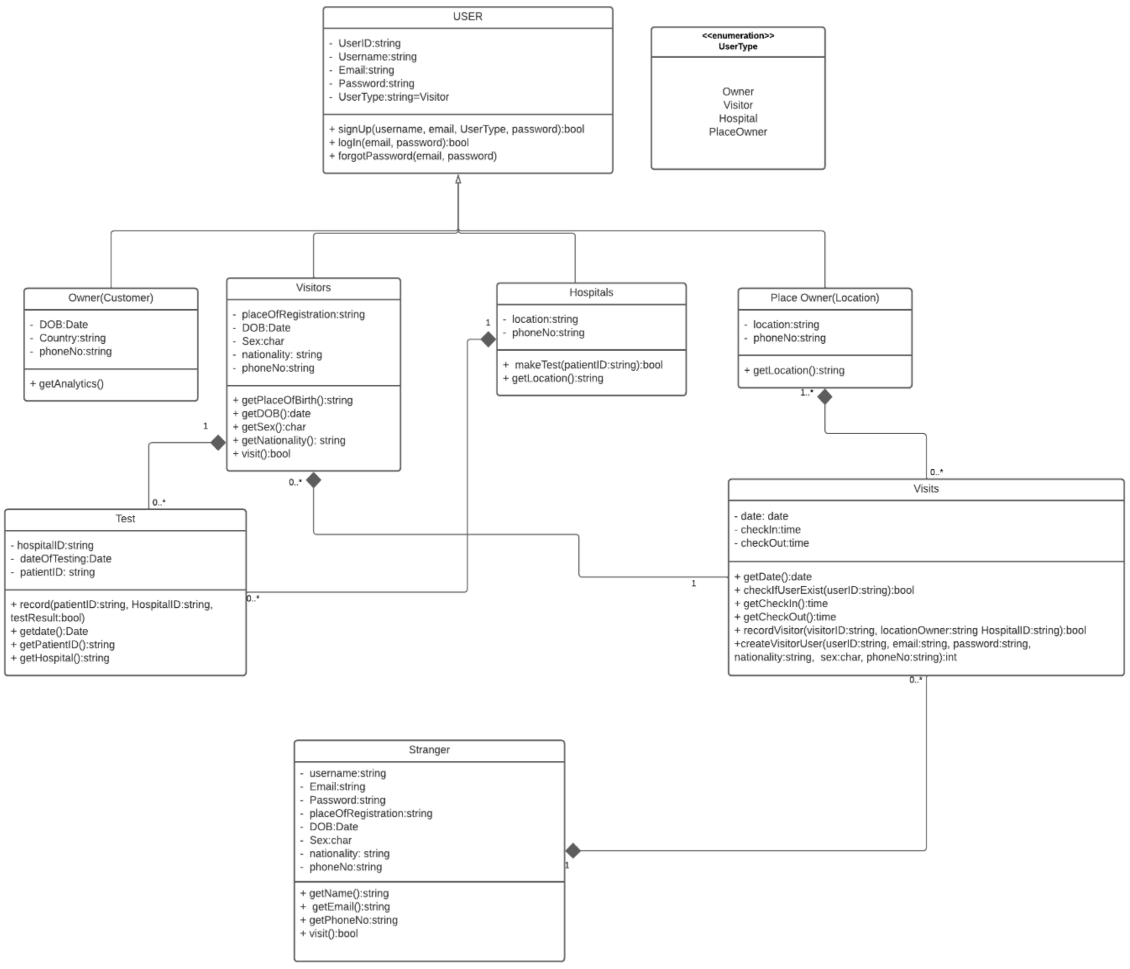
- To create a place owner user, the place owner shall sign up on the website using data like place, Username, DOB, userID, password.
- After signing up, the place owner will get a unique QR code which contains location owner ID in encoded form.
- This QR code shall be used by visitors to sign up.

Owner:

- To create a place owner user, owner, shall sign up on the website using data: place, Username, DOB, userID, password.
- After login, the owner shall be able to see analysis
- Owner shall be able to see list of infected visitors
- The owner shall be able to see the list of visitors who visited a place

3. System Design

UML Diagram



UML Class Diagram

There are 4 types of users: Visitors, Owner of the Application, Place Owners, and Hospitals, which are child classes and inherit the attributes and methods of the parent class User. Different users have different attributes and methods which are described below.

1. **User Class (Abstract class):**

   The attributes of this class are:

   - UserID:string: to store the unique userID of every user
   - Username:string : to store the username of every user
   - Email:string : to store the email of every user
   - Password:string: to store the password of every user
   - UserType: : to store and distinguish between the types of user.

   The methods provided by this class are:

   - Signup: This method is used by child classes to create users of distinct types and is called by child classes.
   - Login: This method is called by child class to login as users logs into the app.
   - Forgot Password: This method is also called by the child classes to recover the passwords (if they forget them).

2. **Owner(Clientof the application):**

   The attributes of this class are:

   - DOB:Date : to store the date of birth of the client
   - Country:string : to store the country of the client
   - phone:string : to store the phone number of the client

   The methods provided by this class are:

   - getAnalytics(): This method is used by the client or the owner of the application to get all the analytics from the application.

3. **Visitors:**

The attributes of this class are:

- placeofRegistration:string: to store the location of the place where a visitor signs up for the first time.
- DOB: Date: to store the date of birth of visitor account
- Sex:char: to store the gender of the visitor
- nationality:string: to store the nationality of the visitor
- phone: to store the phone number of the visitor

The methods provided by this class are:

- getPlaceofBirth(), getDOB(), getSex(), getNationality(): These are the basic getter methods to get the place of birth, date of birth, gender, and nationality simultaneousluy.
- visit(): This method is used to confirm if the visitor has visited the place or not.

4. **Hospitals:**

The attributes of this class are:

- location:string: to store the location of the hospital
- phone:string: to store the phone number of the hospital
- licenseNo:int : to store the licence number of the hospital for verification purposes

The methods provided by this class are:

- makeTest(patientID:string):bool: This method is used to confirm whether the visitor has already done a test or not.
- getLocation():string : This method is used to get the location of the hospital

5. **Place Owner(Location):**

The attributes of this class are:

- location:string: to store the location of the owner of the place.
- phoneNo:string : to store the phone number of the owner of the place.

The methods provided by this class are:

- getLocation(): This method is used to get the location of the place where a visitor visits.

Other than the classes mentioned above, there are classes: Tests and Visits which provide methods that are used for performing of tasks and proper functioning of the system.

6. **Test:** Tests are instantiated by hospitals to create objects to store data about individual patients' test.

   The attributes of test class are:

   - hospital_id: to store the hospital_id
   - dateOfTesting: to store the date on which the test was performed.
   - patient_ID: to store the id pf the patient(visitor in this case)

   The methods provided by this class are:

   - getDate(), getPatient(), getHospital(): These are basic get methods to get the date, patient information and hospital information simultaneously.
   - record(): This method is used to store the attributes of participated object to the database which are to be accessed by product owner.

7. **Visit:**

   This class is instantiated to store the data about persons who visit to a place.

   The attributes of this class are:

   - date:date : to store the date when the place was visited
   - checkin:time : to store the check in time (when the place was visited or entered)
   - checkout:time: to store the checkout time(when the place was left)

   The methods provided by this class are:

   - getData(), getCheckin(), getCheckout(): These are general getter methods to access attributes to objects.
   - checkUserExist(): This method is used to chceck if the user with same credentials like same email or phone number exists or not.
   - createVisitoruser(): This method is used when a person visits the place for the first time and likes to sign up as a visitor.
   - infectedContactList(): This method is used to get data about analytics of a visitor who visited the same place as an infected person on the same day --for contact tracing.
   - storeinfectedContactList(): This method is used to store the data about contact tracinfg into the databasae.

Also, there is one more class: Stranger, which is used to store the information about a person who visits any place for the first time. This class is used to get the information about a person while s/he signs up for the first time.

8. **Stranger:**

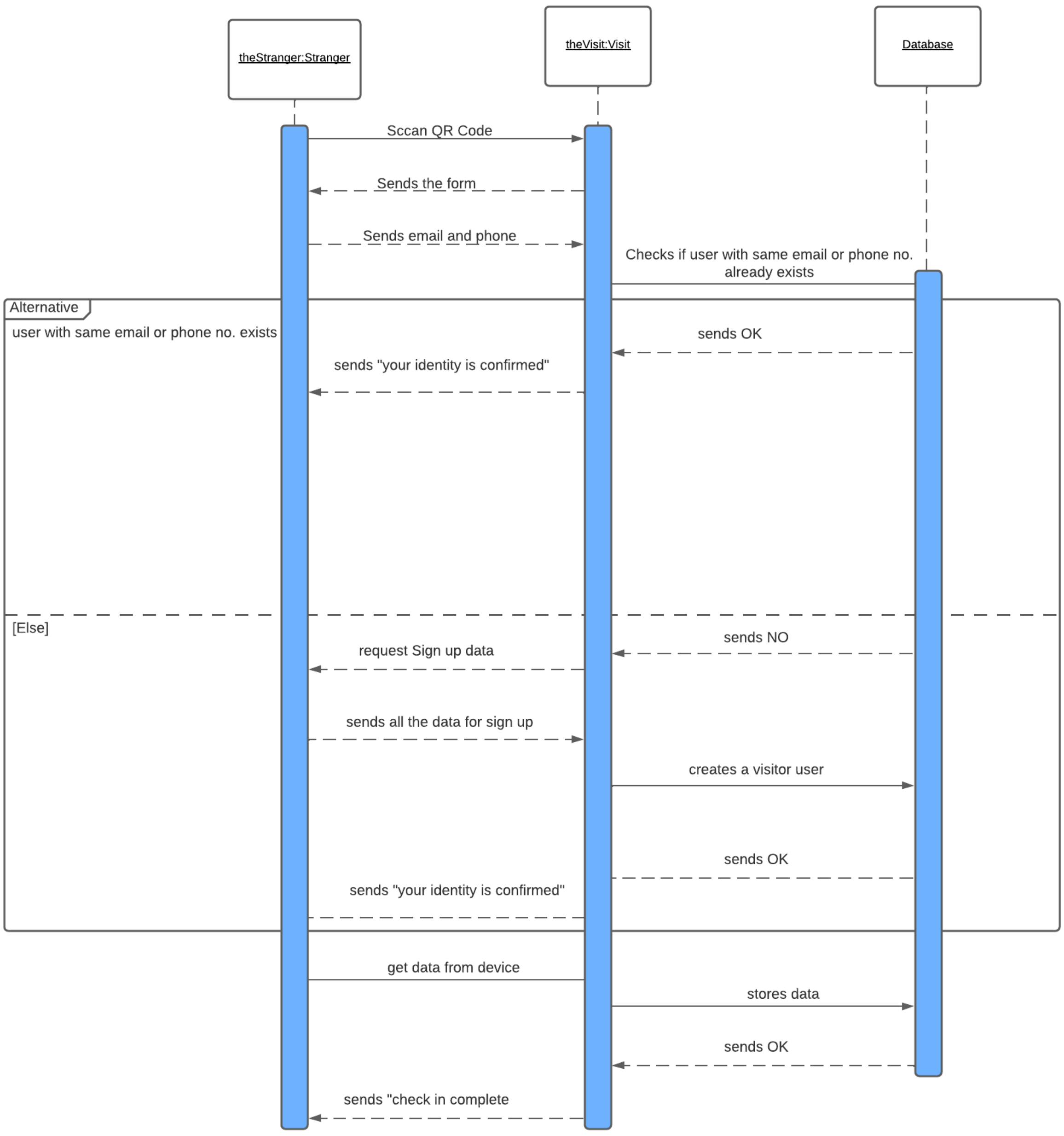
   The attributes of this class are:

- username:string : to store the username selected by the stranger while signing up
- Email:string: to store the email address selected by the strange while signing up
- Password:string :to store the password selected while signing up
- placeofRegistration:string: to store the location of a place where s/he signs up
- DOB:Date: to store the date on which s/he signs up
- Sex:char : to store the gender
- nationality:string: to store the nationality
- phoneNo:string : to store the phone number

The methods provided by this class are:

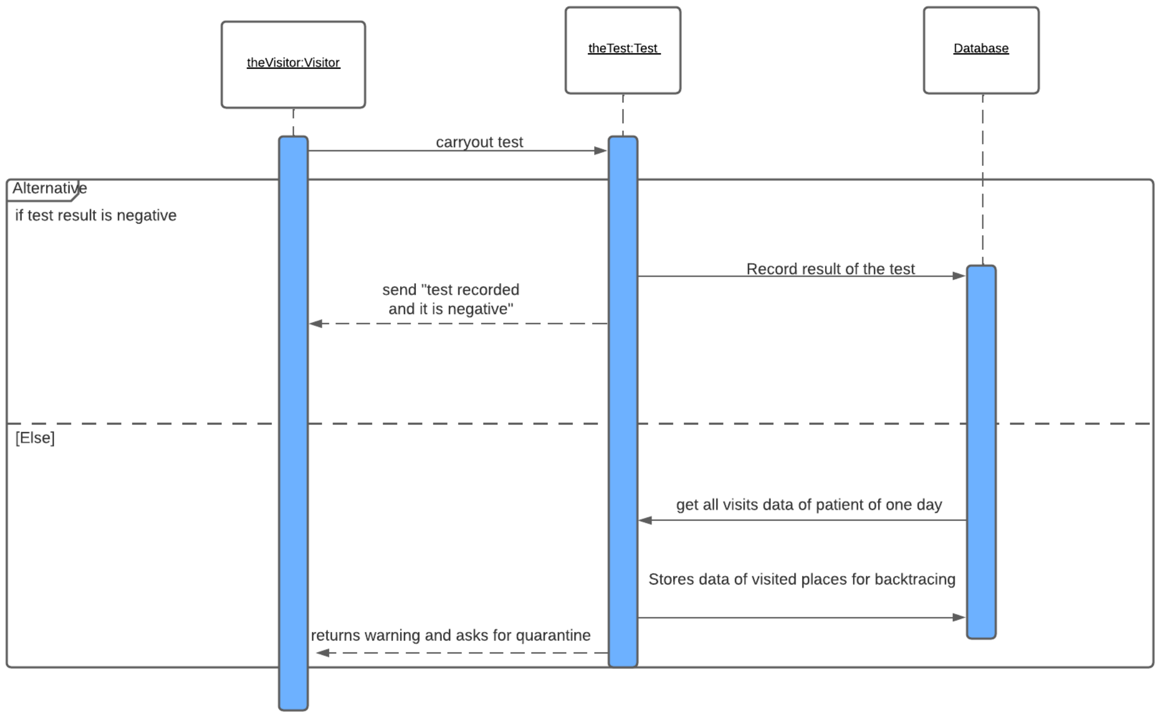
- getName(), getEmail(), getPhoneNo(): These are the basic getter methods for getting the name, email and phone number consecutively.
- visit(): This method is used to instantiate Visit class to store detail about user visit detail

Sequential Diagram



Sequence Diagram 1

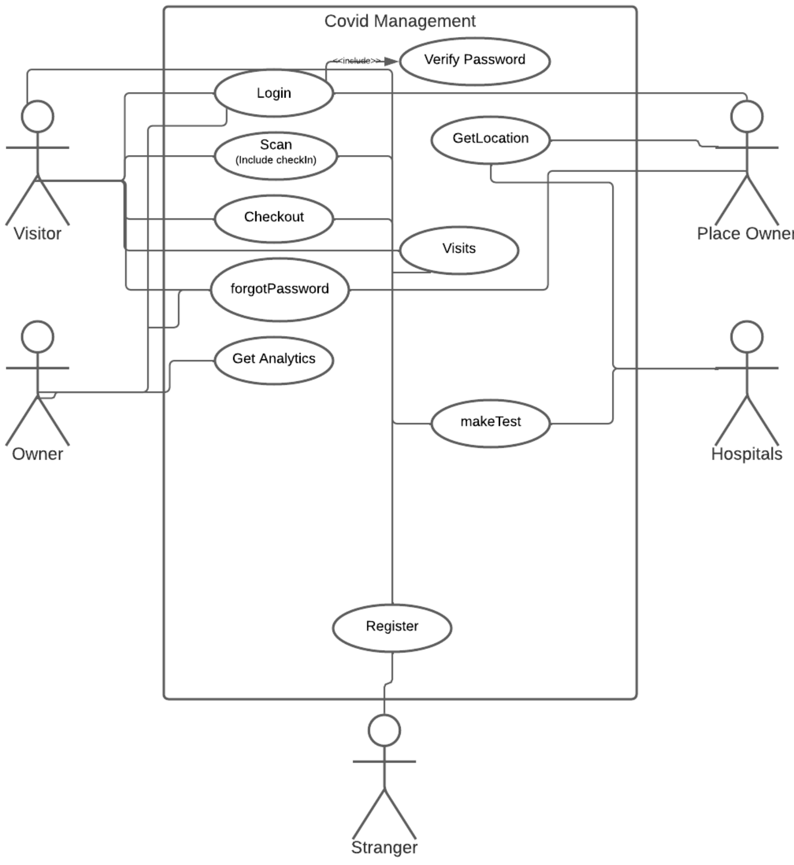
When a person visits a registered location, s/he scans the QR code upon scanning, the app sends the email and phone number stored in the app to the server. The server checks if the user with the same email or phone number exists if the user with the same email/phone number exists it sends ok. If a visitor account does not exist for the person currently visiting, the server asks user to sign up as a visitor and the user must sign up. After this visitor account exists for sure in database. Now the app sends date and time using system time to server, which server use to store detail as visit Object


theVisitor:Visitor
theTest:Test
Database
carryout test
Alternative
if test result is negative
Record result of the test
send "test recorded and it is negative"
[Else]
get all visits data of patient of one day
Stores data of visited places for backtracing
returns warning and asks for quarantine

Sequence Diagram 2

Only a user with a visitor account can take the test. Upon taking the test a test object is created the test object provides record method which is used to record all its attributes to database, if the test is negative the server sends back the test result to the user, if the test result is positive the test object calls infectedContactList() method which uses Warshall algorithm to recursively search for the other person who have visited the same place that other person has visited on the same day. These data are stored by calling storeInfectedContactList() method.

Use Case Diagram



Use Case Diagram 1

4. Conclusion

The objective of this web service is to provide a clean, secure, and easy-to-use service to track, analyze, register, and hence restrict the spread of COVID infections. The application takes notice of all the types of users and tries to encompass their requirements. The service also provides a link among several types of users and eases their coordination.

The aim of this document is also to provide a detailed explanation of the different services, their relationships among different users, and their functionalities to programmers, and other technical personnel. The instruction in this document facilitates the programmer to turn the conceptual design of a service into a well-working application.

5. References

https://lucid.app/

http://www.peter-baumann.org/

https://en.wikipedia.org/wiki/Class_diagram

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/

https://www.tutorialspoint.com/uml/uml_interaction_diagram.htm