COMPUTER SCIENCE

DAT 20I

2nd SEMESTER FINAL PROJECT

COVID-19 TEST CENTER

02/06/2021

By Krishna Prasad Khanal

Omar Said Farah

Table of Contents

"Nothing lasts forever, Technology to our rescue."

ABSTRACT

The success of any Institution such as Danish Health Authority depends on its ability to acquire accurate and timely data about its operations, to manage this data effectively, and to use it to analyze and guide its activities. The Covid-19 Test Center management system offer User, Secretary and Administrator with a unified view of data from multiple sources. The main objective of this project is to build a User (I.e., test patient user) database system that will store records of data. It is purposed to save time and life by enabling a single pane of glass (central location) presentation and management. The system is intended to accept process, generate ID number, user name, CPR number, date and time booked in relation to user information's accurately. The system is also intended to deliver better services to Administrator and Secretary, provide meaningful, consistent, timely data, information and finally promotes efficiency by enabling better tracking system.

INTRODUCTION

Covid-Response IT solution has been contacted and contracted by the Danish Health Authority.

We live in unprecedented times and cities are at the heart of the fight against the Covid-19 pandemic. Our challenges are greater today, but our combat weapons are also stronger. Data economics, collaboration between scientists and information in public opinion become extremely relevant in this struggle.

As an event never before experienced, the sharing of knowledge (solutions, insights, policies, etc.) is now more important than ever.

Therefore, the Danish Health Authority has decided it is time for an efficient IT administration system with better data storage, maintenance mechanism and most of all better visibility in order to make better informed decisions.

1. PROBLEM STATEMENT

The problem of having no tracking system in the face of the current virus outbreak, there is a high chance of the virus spreading fast and to a point of non-stop state.

It will be resulted in the inability of properly establishing some measurements and using proper tracking system of the pandemic.

Moreover, it is cumbersome, time consuming and disastrous result of high number of deaths.

With the increase in the volume of people infected growing exponentially, decentralized traditional system will not cut it anymore.

As a result of this, an advanced and centralized IT System has been in demand of time.

Hence the aforementioned problems prompted for the Design and Implementation of the Web based Application test center, to enable the Public Danish Health Institutions to work more efficiently.

2. PURPOSE OF THE PROJECT

The purpose of this project is to design and implement a system that will mitigate the aforementioned problems and avoid worst case scenario of such a case of large deaths.

It is also created for better visibility and explore the intricacies associated with Software Design.

3. SCOPE, LIMITATIONS

Covid-19 Software system will be a web-based application as per the agreement with our clients (Mr. Janus Pederson, Mr. Douglas Beaver, and Mr. Kristoffer Michael).
In making the online booking the User must be able to book a time slot and view its data (i.e., for his booked time), in regards to the test centre, Secretary must be able to book, view and update user details as well as being able to search like by CPR numbers, by name, by test status whether it is positive or negative. Administrator is also able to manage the whole system. Moreover, Administrator and Secretary from the Infection Detection Center and Vaccine Center, they can send message to the User. And then the User is able to see or receive the message in his dashboard. User, Secretary and Administrator must all login into the system before proceeding to their specific duties.

Although it is a web-based Management system, User is only able to book a timeslot as well as to view its bookings via searching with either their name or CPR.

Our system will not tie into any inventory details, payroll for the workers (secretary, administrator) and so on so forth. However, it will include time management, login capabilities and information regarding to Test center, Vaccine center and Infection detection.

4. Feasibility Study

In short, it's the evaluation and assessment of the proposed project. Answering questions like 'Do we have or can we create the technology to do this?', 'Do we have the people resources?', will we get our ROI?'…etc. There are mainly four area being Economic, Technical, Schedule & Organizational.

- Schedule Feasibility, may need strict scheduling which relates to the amount of time allocated to the development of the system or application (02/06/21 – 4 weeks). This may definitely benefit in conjunction with Gant Chart.
- Technical Feasibility, which mainly focus on Hardware and Software. As per our project, because it's 'Web-based Application' therefore we can safely say there's no hardware needed. However, in terms of Software, we are hosting online using Technology like – Spring & Spring Boot, MySQL, Java, HTML, Thymeleaf. And the use of GitHub for better collaboration and version control capabilities to mention some.
- Economic Feasibility – Assessing in accordance to the allocated budget.
- Organizational Feasibility – Staff training and the use of internet as our application is web-based.

FURPS

Users of the System

1. **User**
2. **Secretary**
3. **Admin**

Functional Requirements

1. **User** (only read & create operations)
    - Login and logout
    - Book a time slot (for both test center & vaccine center)
    - View his data (i.e., personal bookings)
2. **Secretary** (can read, create & update operations)
    - ➢ Login & Logout
    - ➢ Book a time slot for users
    - ➢ View & search (ex. by CPR numbers)
    - ➢ Update User details
3. **Admin** (create, read, update & delete – all crud operations
    - ✓ Login & Logout
    - ✓ Manage the system (has full control)

Non-Functional Requirements

- ❖ availability
- ❖ Browser support
- ❖ Performance
- ❖ Legal requirement
- ❖ Standard /or Regulation (i.e., GDPR)
- ❖ Documentation

SECTION 1: I.T ORGANIZATION

In this section, ITO, we will try to illustrate the Stakeholders Analysis & Interests, Risk Analysis & Mitigations, Swot Analysis & lastly Stakeholder's Grid.

Now, in Stakeholder Analysis & Interests, we can see that there are three important aspects: Who they (stakeholders – we have got around eight of them) are, what power (ex. High, medium or low) do they have & what are their interests.

Stakeholder Analysis and their interests			
Stakeholder Id	**Stakeholder's Name:**	**Key Interests and description**	**Power**
1	Administrator	Administrator is the key stakeholder of our system, he will frequently use the system. He is also interested in the reliability and accuracy of the data, easy to access and use its features.	He has a **high** power. we need to keep satisfying & watching him closely. If he wants, he could shut the project down as he has full power.

2	Secretary	Secretary is a key stakeholder as well. He/she is interested to the reliability and accuracy of the data, easy of access & usage of the system.	He/she has a **medium** power. we keep close eye on them. He/she can influence the administrator.
3	User	Users are the main data source thus important stakeholder. They could have multiple interests such as system performance, handling of their data, easiness of use and responsiveness of the system.	Though user are key Stakeholders they have **medium** power but we still need to watch them closely too.
4	Health authority	Health authority is also especially important stakeholder. They are directly related to our system. Because our system is related to the health sector. We need also to follow their instructions.	They have **high** power. We need to fully satisfy them.
5	Product owner	In this case, product owner is also the key stakeholder. He/she is interested on the technology we use, design we implement, performance of the system and many other factors. In fact, we are fulfilling the requirements given by them.	He has **high** power as well.
6	Other developers	Other developers are also important stakeholders, because they will work in our system at some point for future maintenance source code.	They have **less** power.
7	Media	Media are also the key stakeholders because they will have interest on our public information and its usage for reference.	They have **medium** power.

8	Competitors	They have high interest. As they may intent to prevent us from achieving our goals.	They have **low** power.

Figure - Stakeholder Analysis & Interests

In Risk Analysis & Mitigations, it is basically a way, for us, to assess or identify the risks (depicted five of them) and how they may affect our project. And finally, after identifying them, to try to figure out how to minimize them.

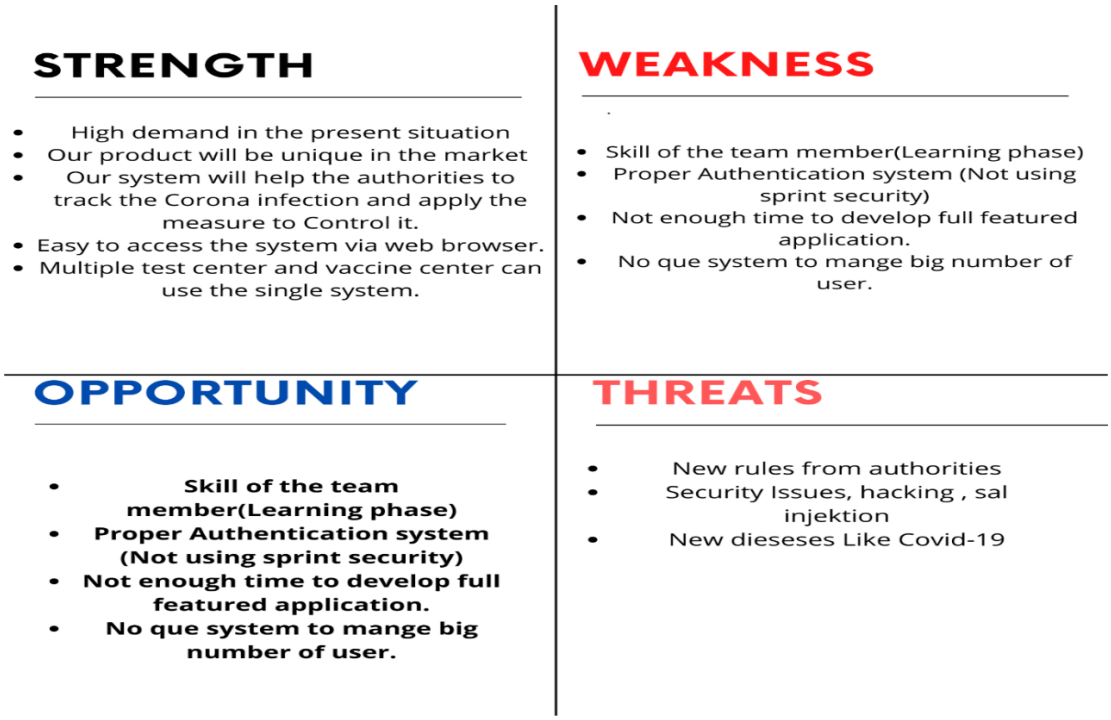
Risk Analysis and Mitigations						
Risk ID	Risk Categories	Risk Description and Impact to the project	Impact	Probability	Priority	Countermeasure Mitigation action
1	Organizational	Team member could be sick for example Covid-19, this could cause some delay and we may not be able to submit the project on time.	3	3	Medium	# We planned to finish as much work as early so that if someone get sick other can cover rest of the work. #We are working online in a collaborative fashion.
2	Technical	Failure of computer and loss of data or source code, this could cause the risk of re-writing everything & lack of time.	2	2	Medium	# We will save our work in cloud-based services (google docs) # GitHub
3	Product	Security (SQL injection and hack), As our system uses the database & user input there could be a possibility of SQL injection and missuses of user data.	3	3	High	# To limit the SQL injection, we will design the database field in such a way we would allow any concatenation. Input field will be properly validated.

4	Product data	User legal action against the use of data protection, User could complain against the use of data used by our system, Sometimes it could lead to legal dispute.	2	2	Medium	# We will take a user permission before they use our system.
5	Technical	As we use the file system for log It may throw the exceptions and error	1	1	Low	# As it is checked Exception, we will handle it using try & catch.

Figure - Risk Analysis & Mitigations

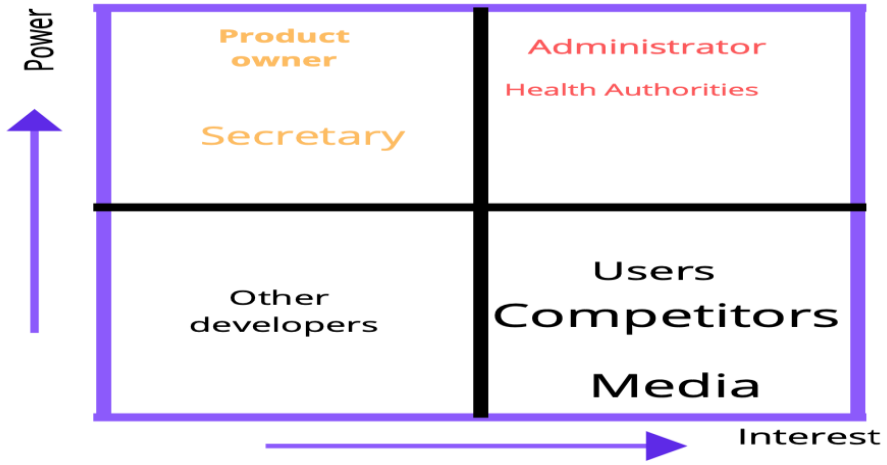
SWOT Analysis

It is a two-by-two grid used for planning process in order to help company overcome challenges by determining its Strength, Weakness, Opportunity & Threats.



Stakeholder Grid

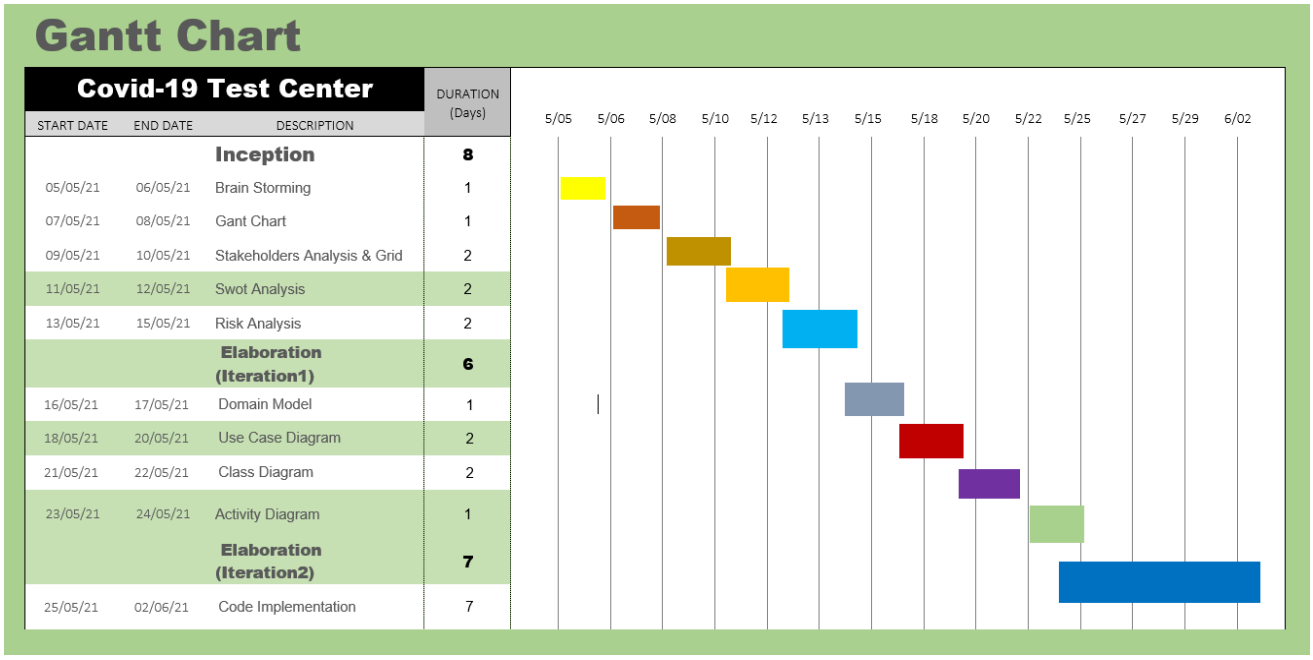
It is also known as Power-Interest grid that helps to determine stakeholders based on their power (or influence) and interests in the project at hand. However, it is primordial to do Stakeholders Analysis first before attempting it.



SECTION 2: SOFTWARE DEVELOPMENT (SWD)

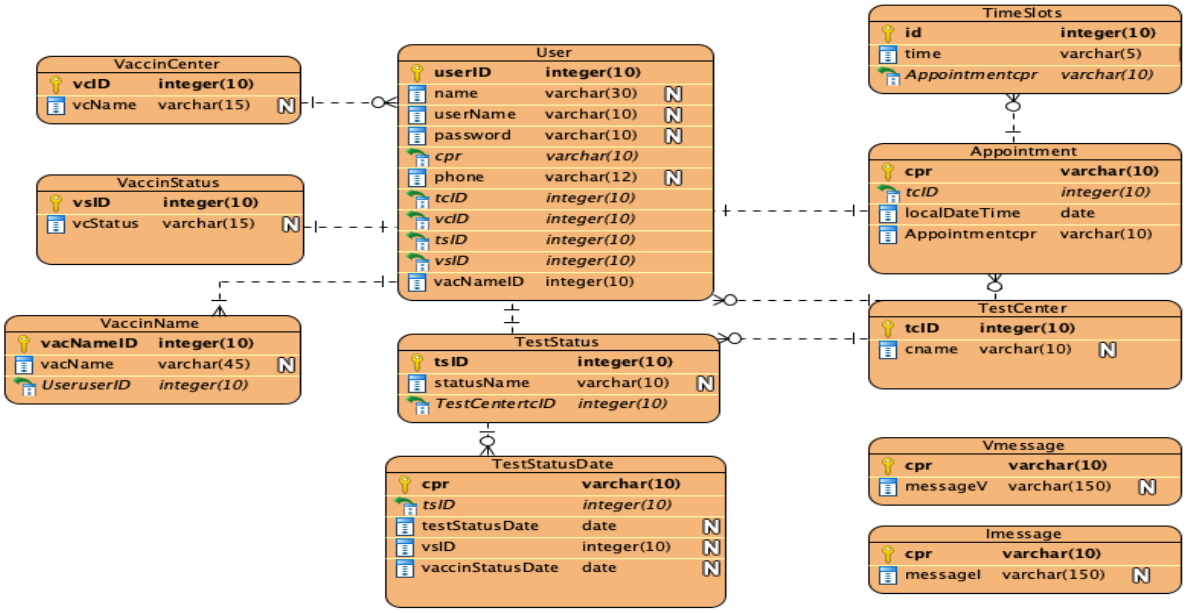
GANTT CHART

For planning the implementation of any project of any size, Gant Chart is the go-to and start-with tool. They are the most useful Project Management Technique there is. Therefore, here below is our Gantt Chart for the Covid-19 Test Center, it allows us to adjust our plan, monitor our project & plan and better explain our project. It goes from Brain Storming to Analysis and Design to Code implementation.



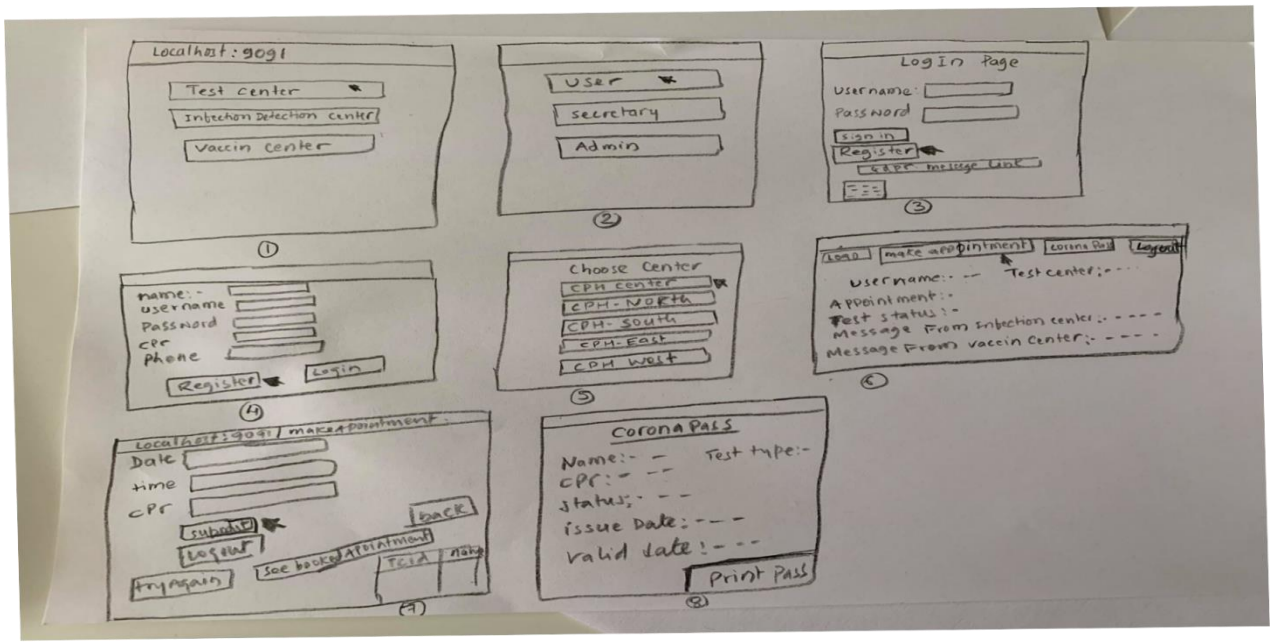
Entity Relationship Diagram (ERD)

Data in our project into Entities and defines relationship between Entities.



PAPER PROTOTYPE (user-centered design)

This is the process of creating a paper representation of digital product, in short, a working model of the product even before developing it. Great to get early feedback from users.



DOMAIN MODEL

Domain model, also called Conceptual model, is the basis for the design of the software or system.

After defining the requirements and writing some Use Cases, we start to transition from Analysis (understanding the problem we're trying to solve) to Design (how we're going to organize our solution). In our case, here is a Conceptual Model for the Covid-19 Test center System describing its different components & their interactions. The starting best practice for creating a Conceptual Model is to first outline all the nouns from the requirements perspective.

Visualizing the Domain model:

- Test center – A User (class) login and book a time slot to take a test. The Test center (class) records and render that time slot unavailable. User come in the day and get tested. The center send all tests to Detection infection center (class). User can view its data after result. The System classify the test results as Positive or Negative. User may receive a Corona-pass if negative.

Class when used for Domain modeling, it is a visualization of the real-world concept.

Once defined classes, the relationship between classes is called Association. At the end of an association is known as roles.

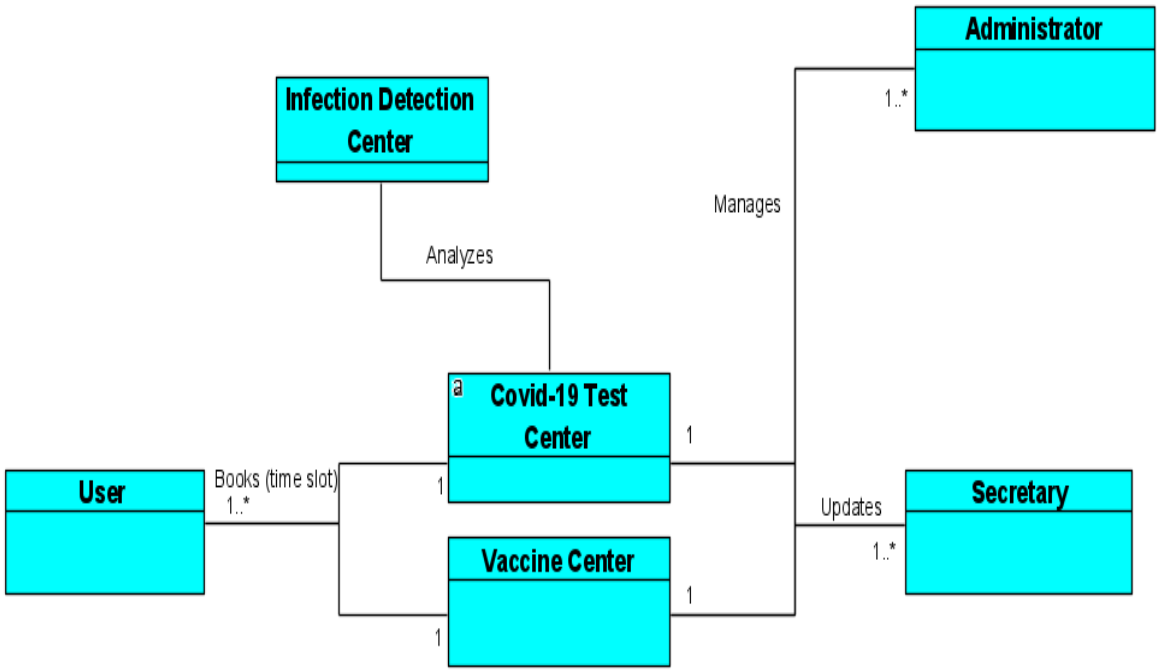


Figure – Domain Model

USE CASE DIAGRAM

The section below outlines the Use Case Diagram/ behavior Diagram for the Covid-19 Test Center System. A Use Case is a diagram of several Use Cases & multiple Actors at the same time, as we can see clearly below. We have got Use Case diagram & written Use Cases (further down below) which both show different perspectives and complement one another.
We started by listing Use Cases first before drawing anything in Visual Paradigm Application program. It helps us tremendously to ask ourselves, in the case of a particular Use Case, question like - is it a goal that our Actor want to accomplish? Use Cases describe a set of actions/ tasks that the system should or can perform in collaboration with one or more external users of the system (actors). For UC06 Register User, use case, the precondition and post-condition have been articulated in fully dressed manner along with Actors and the associations/ relationships (internal/ external) have been clearly stated. And two other Use Cases, namely UC01 Delete User & UC04 Update User, are written in a casual description.
In short, Use Case allow Actors to accomplish a Goal, thus Use Case should be a Goal itself not part of it.
Therefore, as we can see below, we can safely say we have eight goals that our Actors (i.e., User, Administrator & Secretary) should accomplish while interacting with our System.
We have just opted for two kind of Actors – three primary actors (I.e., User, Secretary & Administrator) and one secondary actor namely Danish Health Authority.

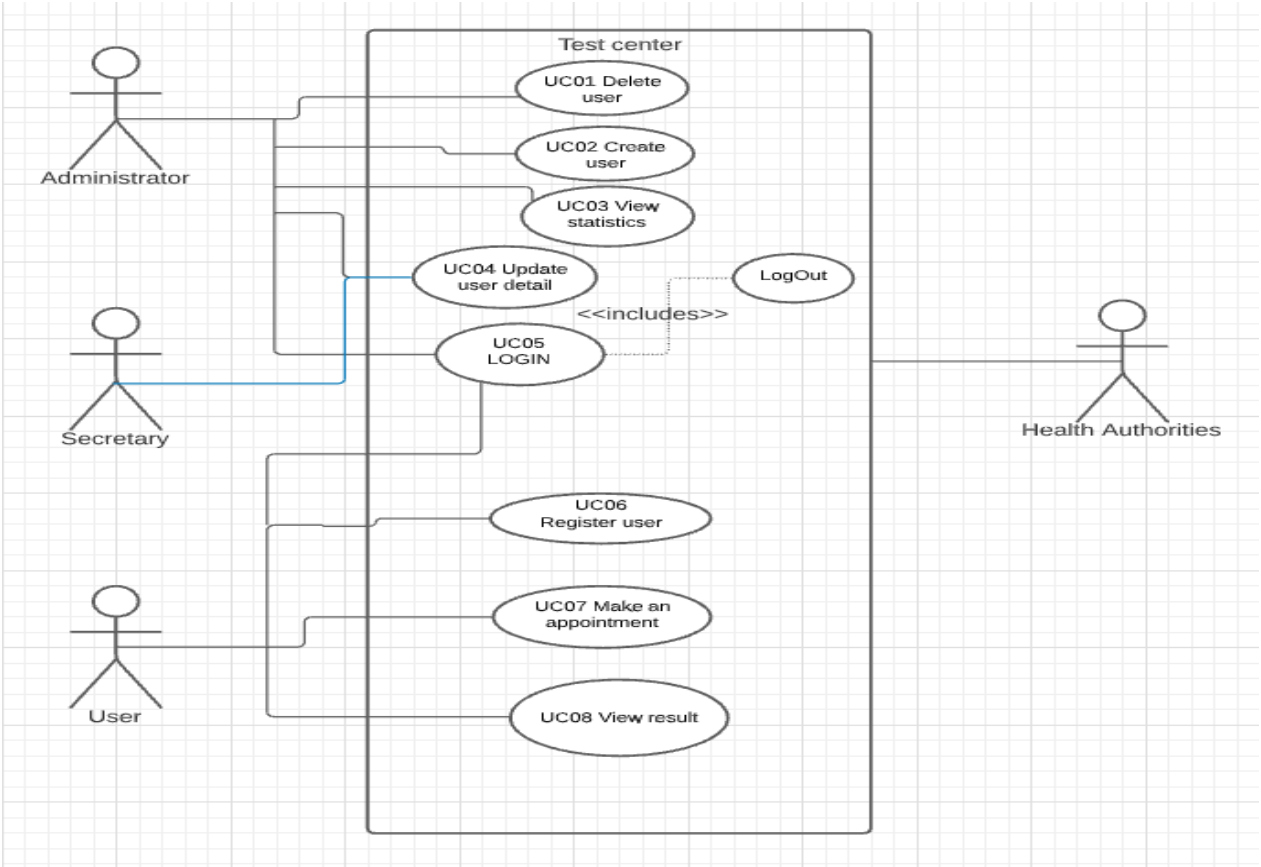


Figure – Main Use Case

Fully Dressed Use Case Description

UC06 - Register User

Primary Actor: - User

Stakeholder and Interest:

User wants to register himself by providing the data as system requirements.

Pre-condition:

The registration window must be already opened.

Post-condition:

In success, user will be registered himself in the system.

Main success Scenario:

1. User clicks in register button.
2. System displays the registration form.
3. User enters name, CPR number, contact number, and clicks on register me button.
4. System validates the data and saves in the database and displays the conformation of successful registration.

Alternative Scenario:

1. User clicks in register button.
2. System displays the registration form.
3. User enters name, CPR number, contact number and clicks on register me button.
4. System validates the data and shows the error message and redirects to the registration form again.

Special Requirement:

1. Easy to use User interface.
2. Responsive webpage to be able to use in Mobile devices.

Casual use case Description

UC04 - Update User

1. Secretary clicks on update user button.
2. System displays the update form with the placeholder value of recent information of user.
3. Secretary clicks on the input field to change the data for update and clicks on update button.
4. System validates the data and updates the data in database and displays the confirmation message.

Casual use case Description

UC01 - Delete User

1. Administrator clicks on delete user button.
2. System displays the list of all users with delete button.
3. Administrator finds the user by its name search and clicks in delete button.
4. System deletes the user in database and displays the confirmation message.

ACTIVITY DIAGRAM

Activity Diagram below depicts the graphical representation workflow of the UC05 Login Use Case.
It portrays the control flow from a start point to a finish point illustrating the various decision paths while the activity is being executed.
In short, the Start black point is the initial state before an application is opened. As an activity represents execution of actions, the actions here are ‘Get User Credentials’ & ‘Validates’ them through a decision-making process depicted as diamond shape in the below figure. Hence the different stages are - entry point to validation stage through decision-making process where the login is either successful or rejected based on the validation.
The following diagram is drawn with two main activities: Get User credentials & Validates User credentials.
The above Activity Diagram depicts the User Login which can also be used for the Administrator & Secretary Login cases.

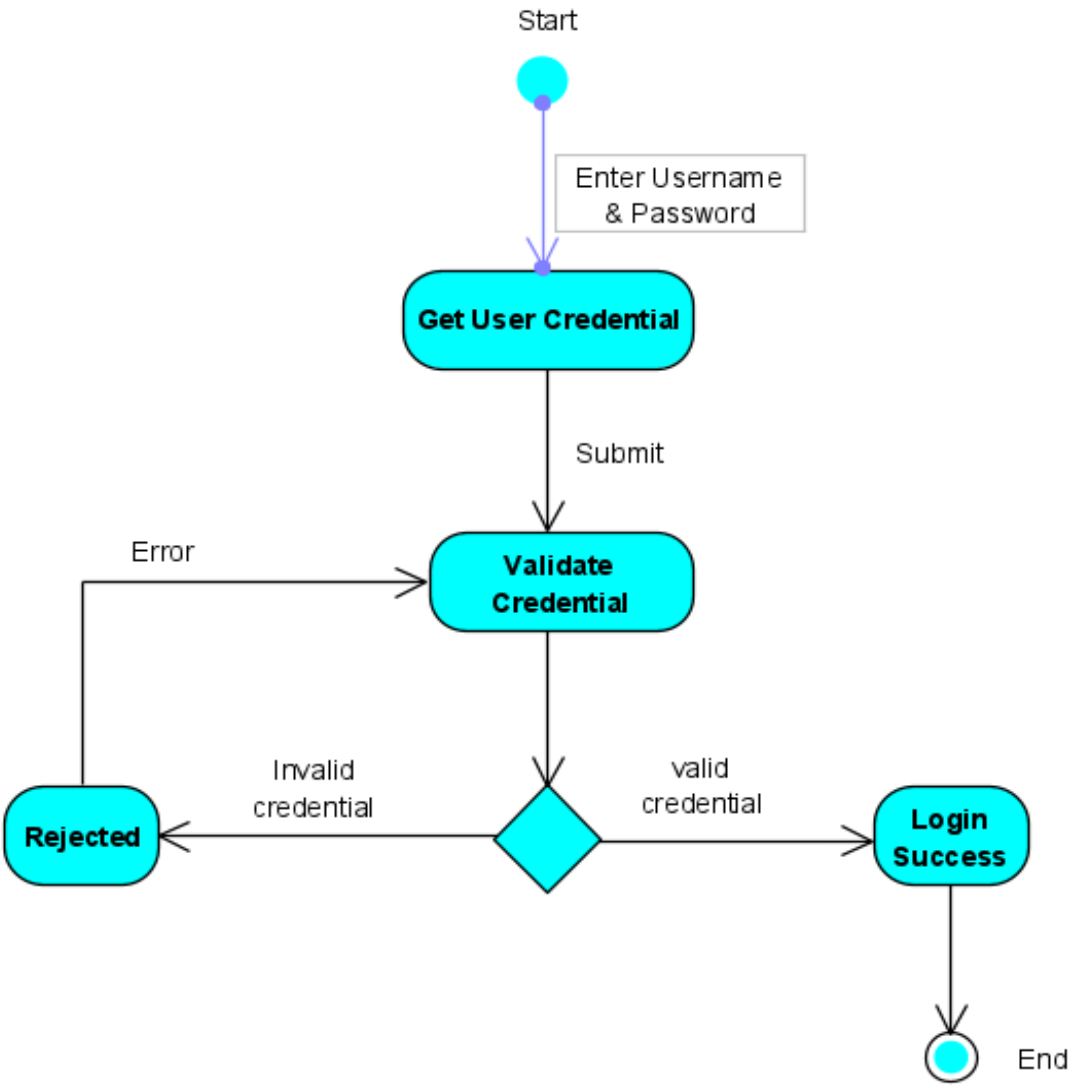
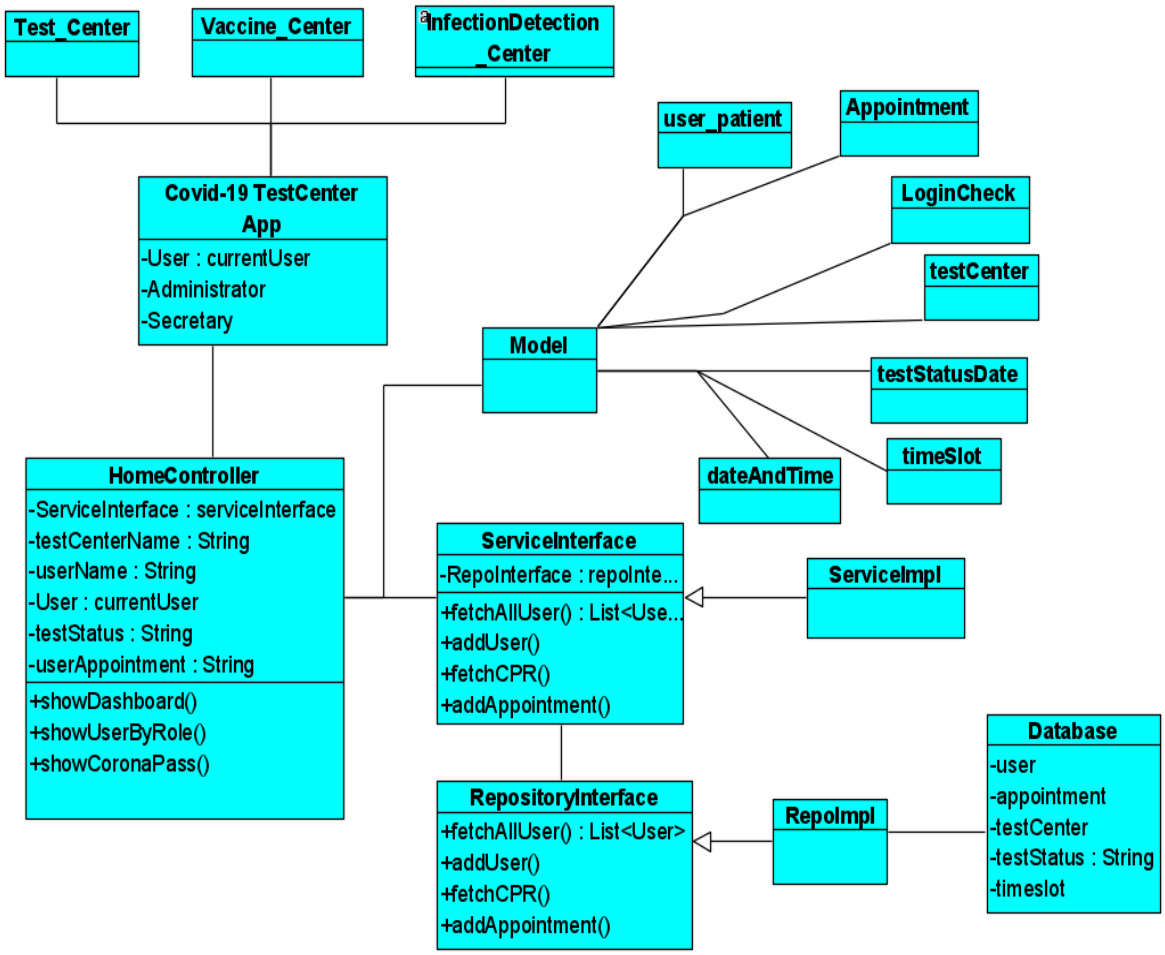


Figure – an activity diagram for User Login (UC05)

CLASS DIAGRAM

We are at the last stage of transitioning from Design to Implementation. Hence, it is much easier to translate Class Diagrams into code.



SECTION 3: SOFTWARE CONSTRUCTION (SWC)

The aim in this section is to showcase the Programming process, the way to the Implementation. As per the agreement and requirement with our clients, the application system is web-based.
Moreover, the technologies used are: Spring Boot, IntelliJ Idea IDE, MySQL database. And we decided Apache Maven as our building tool of choice.
For the web presentation, we are following the MVC design patterns for better layered approach. As it hid our business logic by only rendering it to view what is necessary through the use of the single point of contact with the Controller.

Design to Implementation
During coding and implementation, we will create the actual product i.e., write/code the set of programs to render business functionality as per Software Design Pattern. We are specifically using Spring Web & Thymeleaf dependencies with Apache Maven. As our backend server, we are using MySQL database for optimal productivity, scalability & performance.

The output of the coding is the source code for the Software.
In the following section, we will try to showcase some snippets of our code for presenting the bigger picture & show the transitioning from Design to Implementation.
As we already mentioned previously, our code is adhering to the MVC Design Pattern we will thereby start with the Controller (I.e., home Controller), followed by the different components such as Service, Repository, Data model & lastly but not least the template views.
The Controller (see below) contains our business logic, it is in fact responsible to handle requests, add data model and return the views to be rendered as response. The latter with the help of Thymeleaf template engine.

```
@Autowired
ServiceInterface serviceInterface;

@GetMapping("/")
public String showDashboard() throws IOException {

    List<Appointment> allAppointments = serviceInterface.fetchAllAppointments();
    Date myDate = new Date();
    SimpleDateFormat formatter = new SimpleDateFormat( pattern: "dd/MM/yyyy");
    String strDate = formatter.format(myDate);
    String month = strDate.substring(3, 5);
    String day = strDate.substring(0, 2);

    saveLogs( myLogs: myDate.toString() + "\n" + getIpAddress());

    for (int i = 0; i < allAppointments.size(); i++) {
        String day1 = allAppointments.get(i).getLocalDateTime().toString().substring(8, 10);
        String month1 = allAppointments.get(i).getLocalDateTime().toString().substring(5, 7);


        if ((Integer.parseInt(day1) < Integer.parseInt(day)) && (Integer.parseInt(month) <= Integer.parseInt(month1))) {
            long cpr = Long.parseLong(allAppointments.get(i).getCpr());
            serviceInterface.deleteAppointment(cpr);


        }
    }

    return "home/dashboard";
}
```

In the above snapshot, we have first injected the ‘service interface’ by using @Autowired annotation and then created the show dashboard endpoint to handle the home request. Using the fetchAllAppointment () method from the interface, we have fetched them and stored in a list of type appointment named ‘all Appointment’. In short, in this snippet, we have used List collection, handled the ‘IO Exception’ by throwing it and lastly standard for loop.

```java
@GetMapping("/index")
public String showIndex(Model model) {
    List<Appointment> allAppointments = serviceInterface.fetchAllAppointments();
    List<User> allUsers = serviceInterface.fetchAllUser();
    String appointmentDetails = "";
    int testStatusID = 0;
    int vaccinNameID = 0;
    String recentCpr = "";
    String testCenterMessage = "";
    String vaccinCenterMessage = "";

    for (int i = 0; i < allAppointments.size(); i++) {
        if (allAppointments.get(i).getCpr().equals(currentUser.getCpr())) {
            appointmentDetails = "You have an appointment on  " + allAppointments.get(i).getLocalDateTime().toString();
            recentCpr = currentUser.getCpr();
        }
    }
    for (User allUser : allUsers) {
        if (allUser.getCpr().equals(recentCpr)) {
            testStatusID = allUser.getTsID();
            vaccinNameID = allUser.getVacNameID();
        }

    }
    switch (testStatusID) {
        case 0:
            testCenterMessage = "You are not yet tested or your test result is not ready";
            break;
        case 1:
            testCenterMessage = "Your test status is positive.";
            break;
        default:
            testCenterMessage = "Your test status is negative ";
            break;

    }
    vaccinCenterMessage = switch (vaccinNameID) {
        case 0 -> "You are not called for vaccination";
        case 1 -> "you are vaccinated 1st time";
        default -> "Your vaccination is completed";
    };
    model.addAttribute( s: "currentUser", currentUser.getUserName());
    model.addAttribute( s: "testCenterMessage", testCenterMessage);
    model.addAttribute( s: "vaccinCenterMessage", vaccinCenterMessage);
    model.addAttribute( s: "userAppointment", userAppointment);
    model.addAttribute( s: "TestCenterName", TestCenterName);
    return "home/index";
}
```

In the above index endpoint named ‘show Index’ taking model as parameter, it is clear we are using both the standard for loop to iterate through all appointment & enhanced for loop for users. As we can see, we are also using two kind of ‘switch’ – standard & switch expression (for ‘test Status ID’ & ‘vaccine Center Message’ respectively).

```java
@GetMapping("/login")
public String showlogin() {
    return "home/login";
}

@PostMapping("/user")
public ModelAndView login(@ModelAttribute LoginCheck loginCheck, Model model) {

    if ((loginCheck.getUserName().equals("admin")) && (loginCheck.getPassword().equals("admin123"))) {
        ModelAndView modelAndView = new ModelAndView( viewName: "administrator/administratorDash");

        return modelAndView;

    } else if ((loginCheck.getUserName().equals("secretary")) && (loginCheck.getPassword().equals("secretary123"))) {
        ModelAndView modelAndView = new ModelAndView( viewName: "secretary/secretaryDash");

        return modelAndView;
    } else if ((loginCheck.getUserName().equals("secretaryI")) && (loginCheck.getPassword().equals("si123"))) {
        ModelAndView modelAndView = new ModelAndView( viewName: "infectionCenter/secretaryIdash");

        return modelAndView;
    } else if (logIncheck1(loginCheck.getUserName(), (loginCheck.getPassword()))) {
        System.out.println(logIncheck1(loginCheck.getUserName(), (loginCheck.getPassword())));
        ModelAndView modelAndView = new ModelAndView( viewName: "home/chooseTestCenter");
        UserName = loginCheck.getUserName();
        String vennueName = TestCenterName;


        model.addAttribute( s: "UserName", UserName);
        model.addAttribute( s: "vennueName", vennueName);
        model.addAttribute( s: "currentAppointment", currentAppointment);
        return modelAndView;
    } else if ((loginCheck.getUserName().equals("adminI")) && (loginCheck.getPassword().equals("ai123"))) {
        ModelAndView modelAndView = new ModelAndView( viewName: "infectionCenter/secretaryIdash");
        return modelAndView;
    } else if ((loginCheck.getUserName().equals("secretaryV")) && (loginCheck.getPassword().equals("sv123"))) {
        ModelAndView modelAndView = new ModelAndView( viewName: "vaccinCenter/secretaryVdash");
        return modelAndView;
    } else if ((loginCheck.getUserName().equals("adminV")) && (loginCheck.getPassword().equals("av123"))) {
        ModelAndView modelAndView = new ModelAndView( viewName: "vaccinCenter/secretaryVdash");
        return modelAndView;
    } else {
        ModelAndView modelAndView1 = new ModelAndView( viewName: "home/dashboard");

        return modelAndView1;
    }
}
```

As we can see in the above user endpoint, we have hard coded the login for the Administrator & Secretary. For the User, we are fetching from the database. In here, we are using ‘If ...Else’ condition for checking purposes and lastly adding to the model Attribute.

CONCLUSION

Information is an indispensable tool many organizations, or institution, use to advance decision making. Large number of User's data are generated either manually or electronically on daily basis. As of any epidemy, there is some kind of substantial demand for timely data to reflect its impact. Thus, because as we are dealing here with the whole population, the need for the hour is mostly focused on centralized accurate data.

Covid-19 (2019 novel Coronavirus) is an epidemic which is still current as of today. Tracking plus accurate information gives better visibility and lead to better informed decision for the Danish Health Authority. Hence, give them the time to propose some solution that can fort probably flatten the curve of the infection or the impact.
In order to manage data overtime, there is the need to use past records without any missing data. The database system which captures and maintains longitudinal data would provide an accurate and reliable data about current and past data.

By following Spring pattern (i.e., MVC, dependency injection & so on so forth), we have achieved as benefits Loose Coupling (i.e., single responsibility) and Separation of Concern for our code.

The system is free of errors and very efficient and less time consuming due to the care taken to develop it. All the phases of software development cycle are employed (see Gantt Chart) and it is worthwhile to state that the system is very robust.

Recommendation

Since Population or User database system is very broad, the scope of this project covers only a small aspect of User information system since the stipulated timing within which the project is expected to be executed is too short. And we only used information that were deemed necessary.
This report could be useful to any person, or institution for that matter, who wants to do a project on similar topic.

Opportunity & Lesson Learned

During this project, we were able to understand better what goes in a centrally managed system of a Test, Vaccine or Infection Detection center. Sure, we know only the Test Center was mandatory but we wanted to stretch and test our knowledge and do all of them.
This was effectively done through reading of literature of our Curriculum and research.
Most of Java learned concepts from Interface to Inheritance, Exception to File handling, Java Collection (i.e., List, ArrayList) to Generic… are used except Set, While & Do While Loop.
The whole process of developing the system was an opportunistic challenge. Seeing the system into a tangible system was a rewarding exercise.

BIBLIOGRAPHY
(Whole group)

Books:

- Craig Larman: Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development - Third edition (2004)
- Murach's MySQL (3ed)
- Building Java Programs
- The Organization
- Google machine search
- Video presentation link: https://youtu.be/q7T8nbJAhMM

- Our project GitHub link: https://github.com/KrishnaKEA/CovidTestCenter

In order to experiment with our web Application, we are providing here some user details, please see below:

Test center login information:
For *user*: testUser User name: tu123 Password: tu123
For *secretary*: User name: secretary Password: secretary123
For *administrator*: User name: admin Password: admin123

For Infection detection center:
Secretary:
Username: secretaryI Password: si123
Administrator:
Username: adminI Password: ai123

For vaccine center:
Secretary:
Username: secretaryV Password: sv123
Administrator:
Username: adminV Password: av123