Corona Fighter

Bachelor Project

Ziad Akram Bathish 273442

Angel Iliyanov Petrov 266489

Chunhui Liu 273452

Supervisor:

Joseph Chukwudi Okika

VIA University College

68284 characters

Software Technology Engineering

Semester 7

4-June-2021

Table of content

Abstract

The goal of this project is to help users solve related medical and life problems under the impact of COVID-19 with the help of the Corona Fighter website, as this is the most convenient option for users who have to stay at home due to the outbreak.

The main features that have been included in the system are the ability for users to choose a role among doctors, patients and volunteers, the ability for users to post or comment on others' posts in the system, the ability for users to create private groups and invite other users to join, and for doctors to give lectures. Volunteers are able to offer help to patients who are in need.

The choice of technologies that have been used for the development of the project involve two criterias: ease of development and testing. Technologies such as GitHub, Visual Studio, Microsoft SQL Management Studio were used in order to produce an end-product efficiently.

The web application is based on the ASP. NET MVC framework, code is written in C # and the database for the system is implemented using SQL.

The result of the project is to solve the life, physical and psychological problems of the people affected by the epidemic to the best possible extent that the system can offer to its users.

1. Introduction

As previously mentioned in the background description for this project, the outbreak of COVID-19, which began in January 2020, has affected the lives of everyone around the world. Since COVID-19 is a respiratory disease and can be transmitted through the air, people have to reduce their time for outdoor social activities, which has greatly affected many people's work, study and daily life.

The project aims to create a web application where patients can get help with coronavirus from doctors who have experience in dealing with the virus and can provide counseling and guidance to patients remotely and volunteers can use the information on the website to organize mutual events between hospital staff and volunteers to help patients in need.

As well as protecting themselves as healthy people, many are willing to contribute to the fight against the epidemic. Patients and governments also need the help of volunteers. "The Danish health agency SSI wants to recruit up to 15,000 people in Denmark to help better monitor the penetration of the coronavirus pandemic." [1] *(The Local news@thelocal.dk@thelocaldenmark, 2020)*.

In many cases, doctors could find it difficult to apply direct treatment to patients when face-to-face in a hospital or a clinic. Professional skills of doctors can provide guidance to patients online if they could get a good description of the patient's symptoms.

Some of the delimitations that this project will have are that the system will not automatically verify licenses of doctors who have just registered in the system and uploaded their licence file to the system and would therefore need to be verified manually by an admin.

The purpose of this project report is to familiarize the reader how each requirement is part of the Corona Fighter system by performing an extensive analysis & design explanation on the structure of the Corona Fighter system in the form of diagrams and

descriptions. Choices of technologies in the system will also be touched upon in this report. The reader will be familiarized with what actors there are in the system and what their roles are. Finally, an extensive testing procedure of the system will be available in this report that covers various types of testing methodologies.

2. Analysis

The analysis phase will be focused on the problem domain and what can be done to solve it. Usually, the working process here needs to be done closely with the client.
Analysis phase starts by defining the functional and non-functional requirements, then moves to work on use case diagrams and the use case description to see how many actors will be interacting with the system and what their functionalities are. Activity and System Sequence diagrams were made to have a better understanding of the interaction between the actor and the system.
Analysis is a part of the Elaboration phase where the maximum workload will be on the Analysis together with the design.
At the end of this phase, the Domain Model is created. It's purpose is to show the domain concepts and their relations.

2.1 Requirements

The requirements that are going to be used in this project are made up of functional and non-functional requirements.
Requirements need to be defined at the beginning of the project, the MoSCoW method helps to understand requirements priorities.
The requirements make use of the SMART principles which stand for Specific, Measurable, Attainable, Realisable, Time-bounded/Traceable. Using SMART would help when performing acceptance testing in the Test chapter.

2.1.1 Functional Requirements

MUST

1. The administrator should be able to authenticate the doctor's documentation in the administrator's verification panel of the system.
2. The user should be able to set what kind of role they have - Patient, Doctor or Volunteer in the system.
3. The user should be able to manage their account details, which include their username and password into the system.
4. The user should be able to manage their post by posting and deleting it.
5. The user should be able to write comments on posts.
6. The patient should be able to create their private group and manage it.
7. The user should be able to search and join a group.
8. The user should be able to leave a group by withdrawing from it.
9. The doctor should be able to manage their own corona-related lecture in the lectures section.
10. The user should be able to see all available corona-related lectures available to join in the lectures section.
11. The user should be able to search for posts by keywords in the search section.
12. The user should see statistics by date of the coronavirus outbreak in Denmark in the statistics section.

SHOULD

13. The administrator should be able to add video resources to the library to allow users to watch these resources.
14. The user should be able to have an online library of video resources to aid them mentally and physically in the mental wellbeing section of the system.
15. The user should be able to get notifications from their groups and from their own posted comments when someone interacts with the user's activities in the system.

COULD

16. The user should be able to save, view, and delete their already saved posts, which are in their bookmarked list.
17. The user should be able to chat with other users in the system.

2.1.2 Non-Functional Requirements

1. The web application should have a responsive design to view it on multiple devices' screens.
2. The system should have an uptime of 95 percent or more.
3. The User Interface must be in English language.
4. The system UI should follow Interaction Design principles.

2.2 Actor Descriptions

The users of the system include **administrators, doctors, patients and volunteers.**

- A user should be able to see posts in the view and leave comments on the already existing posts. Users are also able to create their own posts both for everyone to see in public or in private groups. Users can view groups and select ones to join or leave. The User can view a list of lectures which contains information about the lecture such as description and time. They are also able to select ones to join if needed. Users can see a list in the OfferHelp page and ask for help if necessary.
- A Patient should be able to create groups with information about location and describe which kind of help they need.
- A Doctor should be able to create corona-related lectures by providing a zoom link.
- A Volunteer should be able to offer help by providing information about location, time and contact on the OfferHelp page.
- An Admin should be able to add corona-related video resources in the video library. Admins can authenticate the doctor's documentation to decide whether to approve a doctor application or not.

2.3 Use Case Diagram

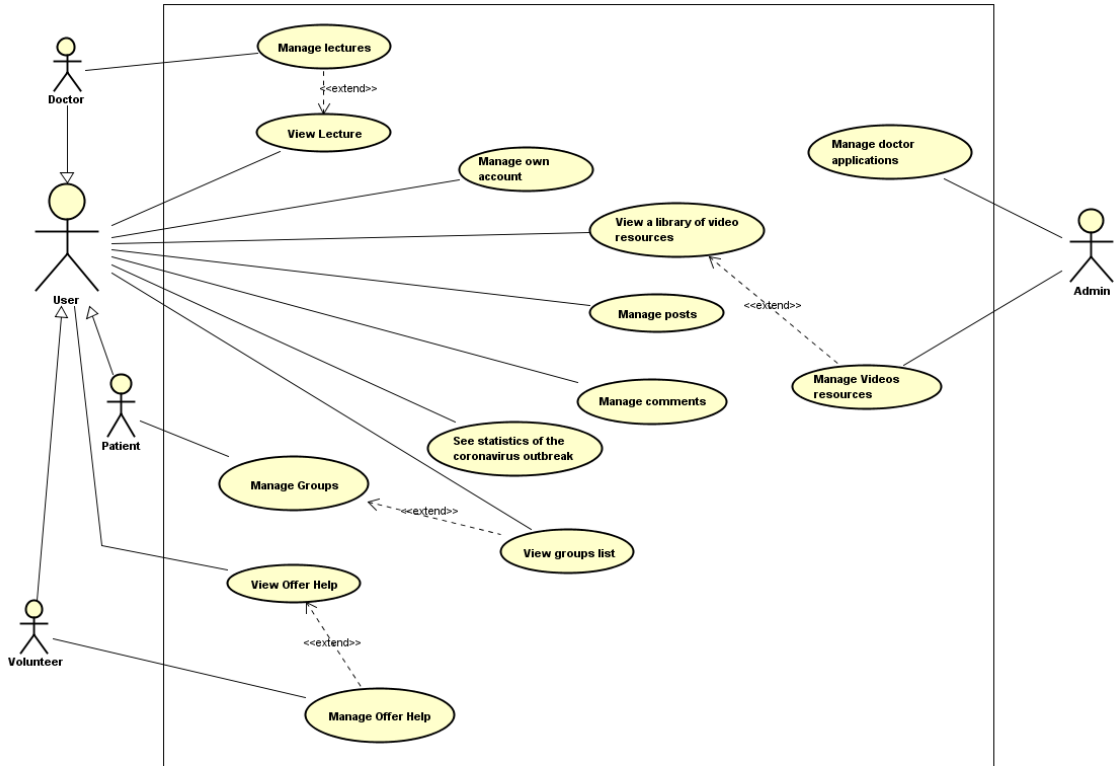


Figure 1 - Use Case Diagram

The Use Case diagram serves to show the various ways in which a user is able to interact with the system. It is a simplified representation of some of the relationships that are between use cases, actors and the system. An actor takes on responsibility for the Use Cases that they are associated with in the system. In the above Use Case Diagram, there are five actors - Admin, Doctor, Patient, User and Volunteer. There are 10 Use Cases in total. The Admin is responsible for Managing doctors data and Adding video resources in the system. The Doctor is able to manage lectures and do all User related functionalities. The Patient is able to manage groups and do all User related functionalities. Volunteer actor can offer help and do all User related functionalities. The User can manage their own account, manage posts, manage comments, see statistics of the coronavirus outbreak and view a library of video resources. The User is not able to do the other Use Cases that are part of Doctor, Patient and Volunteer since he is not a child of these actors.

2.4 Use Case Descriptions

Figure 2 shows a Use Case description for managing posts. The base sequence includes creating and deleting a post.

ITEM	VALUE
UseCase	Manage posts
Summary	User is able to create and delete their own post.
Actor	User
Precondition	[Create a post] User is logged in. [Delete a Post] User is logged in The post exists The post is created by the same user who wants to delete it [View a Post] User is logged in The post exists
Postcondition	Posts can be posted, viewed and removed
Base Sequence	Craete post (1) User enters relevant data about posts. (2) User clicks "Send" button. Delete post (1) User inputs post title in delete page. (2) User clicks "Delete" button. View post (1) User clicks "Post" page. (2) A list posts will show up.
Branch Sequence	
Exception Sequence	[Create post:(2) If post name already exist] System notifies that user needs to enter a new post name. [Delete post:(2) If post name not exist] System notifies that user needs to enter a new post name.
Sub UseCase	
Note	

Figure 2 - Use Case Description for Manage Posts

2.5 Activity Diagrams

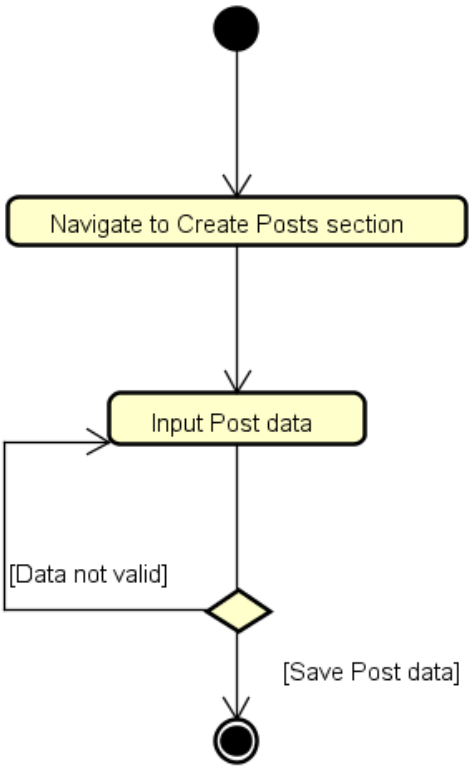


Figure 3 - Activity Diagram for Create posts

Activity Diagrams are used to describe dynamic aspects of the Corona Fighter system. An 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 [2] *(UML - Activity Diagrams - Tutorialspoint, 2021)*.

The above activity diagram (Figure 3) shows how a user would be able to create new posts. The activity starts with a user navigating to the Posts section of the web application. From then on, they are able to Create their own posts. When a user creates a post they would need input post data which will then be saved in the system. This action has a guard “Data not valid” meaning that if the input by the user is not valid then the user would need to input valid data and no changes would be made until doing so.

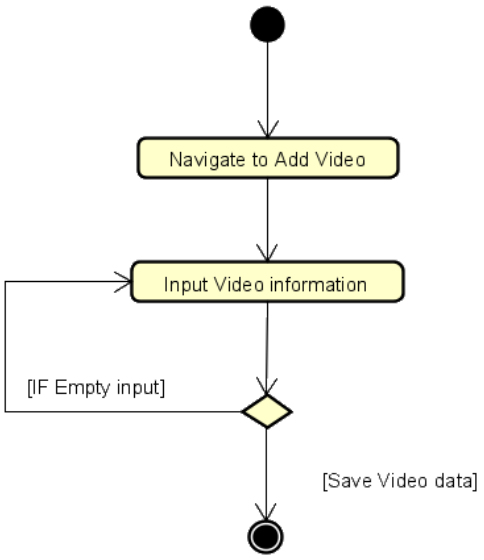


Figure 4 - Activity Diagram for Adding video resources

Figure 4 shows how an Admin is able to add a new video entry into the system. The activity starts with the Admin navigating to the Add Video section of the system. They then need to input information about the video such as video title, video URL and video thumbnail URL. If these fields are empty, the video will not be added to the list of videos that are in the system and the Admin would need to do this step again with valid input. Since there is no way to check if the input by the Admin contains actual valid video information, the system may or may not load the provided data depending on its validity. For example, if the Admin inputs a string of text “Top 10 ways to avoid Coronavirus” into the Video URL section, the system will not be able to redirect the Admin to the video if there is no valid link. This is also valid for video thumbnail URL inputs.

Finally, if all data is entered, the Admin can press the Submit button and the data about the video will be saved into the system.

2.6 System Sequence Diagram

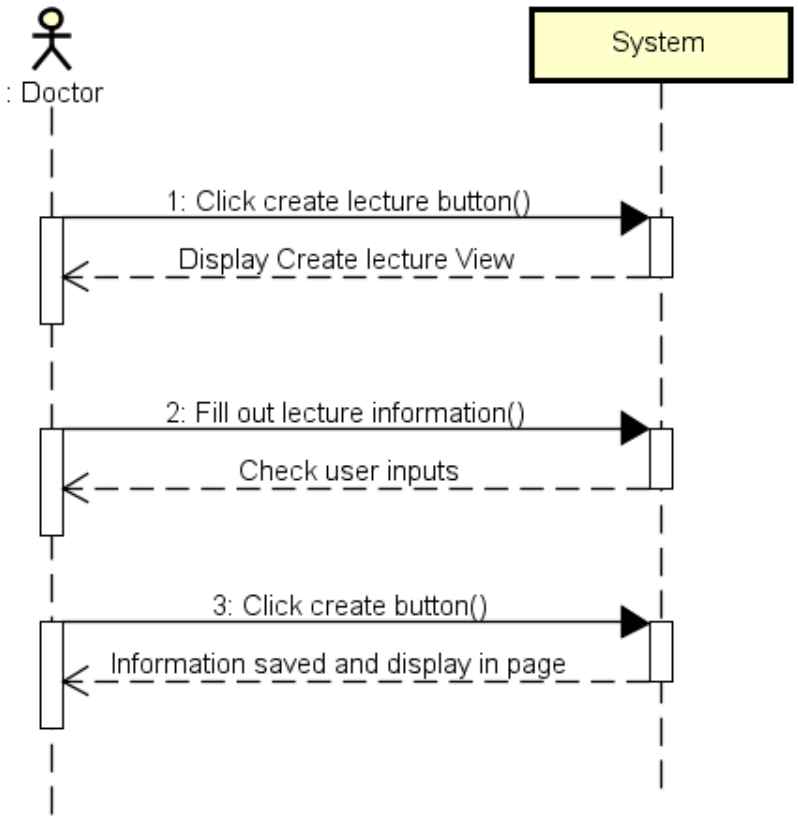


Figure 5 - System Sequence Diagram for Creating lecture

In software engineering, a system sequence diagram (SSD) is a sequence diagram that shows, for a particular scenario of a use case, the events that external actors generate, their order, and possible inter-system events. System sequence diagrams are visual summaries of the individual use cases [3] *(System sequence diagram - Wikipedia, 2021).*

The above system sequence diagram (Figure 5) shows how a doctor would be able to create lectures. The flow starts with a doctor clicking the create lecture button. Then, doctors are able to create lectures by filling out lecture information including lecture time, topic and also a zoom link. After that lecture information will be saved in the database and displayed in view.

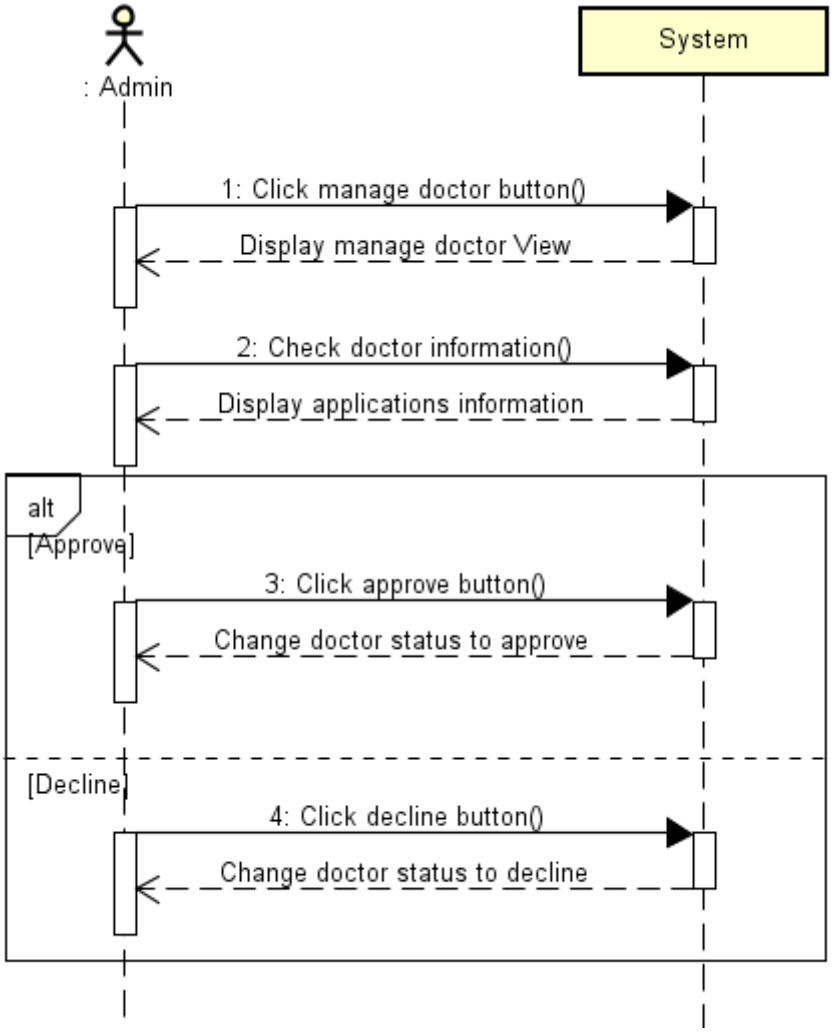


Figure 6 - System Sequence Diagram for Managing doctor

Figure 6 shows how an Admin is able to authenticate the doctor's documentation to decide whether to approve an application made by a doctor or to decline it. The flow starts with the admin going to the Manage Doctor application view. Then, an admin is able to check the doctor's documentation. After that, the Admin can go to the Approve application view and click the approve button if the Admin has established that the doctor’s documentation is authentic. Then the doctor will be able to sign up on the website. Otherwise, if the application has been declined, doctors will not be able to sign up with a doctor identity.

2.7 Mockups

While the analysis phase is almost done, mockups were made to have a better understanding of how the system will look like. Simple paper drawings for few functionalities were made so the team knows how the system front-end will be.

For the Home page the system will show statistics and news of Covid-19, there will be Sign up and Login buttons in the top navigation bar, where the sign up button will return a page with sign up format and login will return another page with two fields Email and password to login.

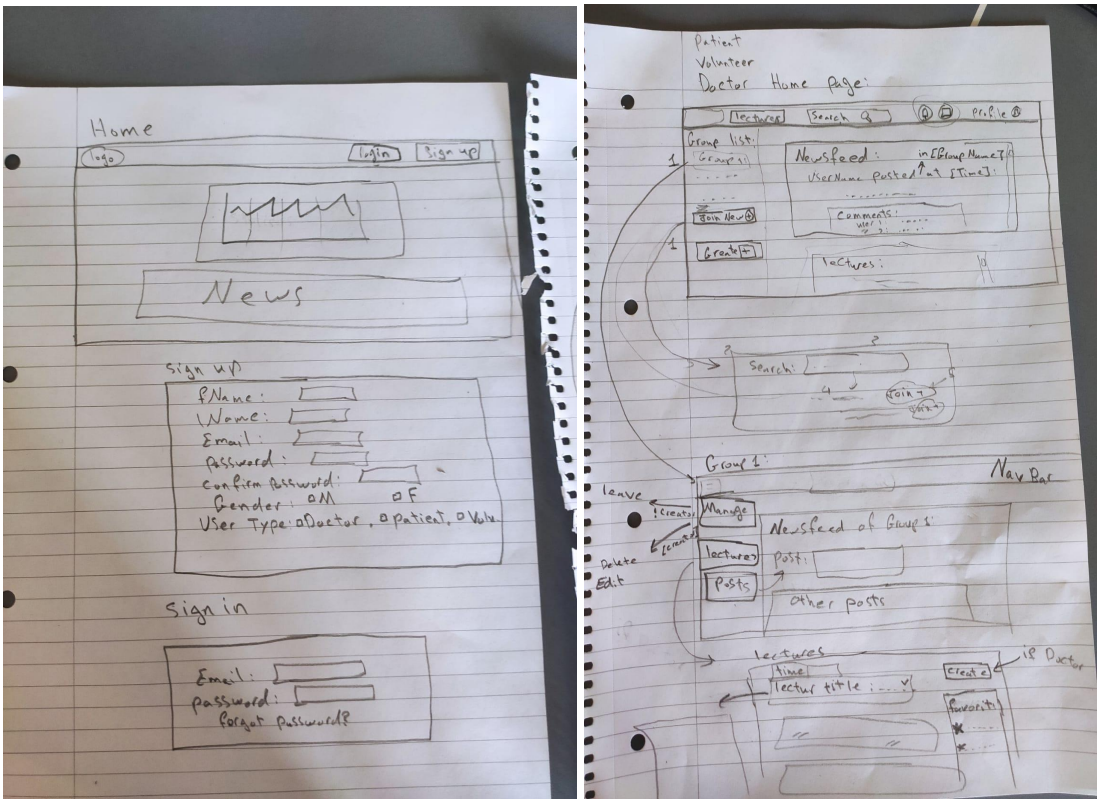


Figure 7 - Mockup sketch prototypes of system User Interface and transitions

2.8 Domain Model

All previous analyses have led to a conclusion of the following domain model, where it shows the main entities and interactions between them in the system. The system will have a user who can sign up and be authenticated in the system.

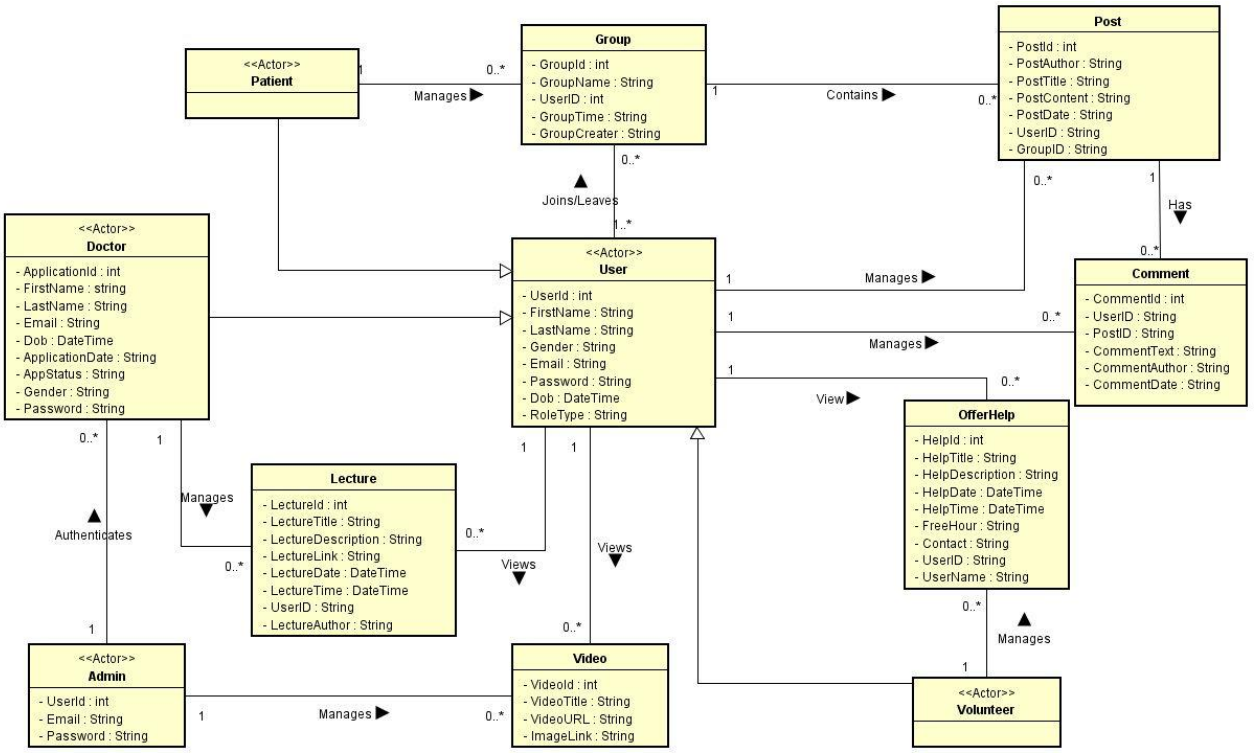


Figure 8 - Domain Model of system

The RoleType attribute for a User defines what the role of the user is in the system. The role can be Patient, Doctor, Volunteer but not Admin.

The following section describes two example entities in the system:

User: The user entity has the role to facilitate data about registered users.
A user has the following operations:

- Join/Leave Group
- Manage/View Post
- Manage/View Comment
- View OfferHelp
- Views Video
- Views Lecture

The Doctor entity can perform the Manage Lectures Use Case and inherits all other Use Cases from the User. The following is a list of all Use Cases that the Doctor can perform:

- Manage Lectures (Doctor can manage lectures by creating, editing or deleting them and is specific only for this entity)
- View Lecture
- Manage own account
- View a library of video resources
- Manage posts
- Manage comments
- See statistics of the coronavirus outbreak
- View groups list
- View Offer Help
- Manage Help Offer

3. Design

Where the analysis focused on what the system can do, the design is focusing on how the system will be built, therefore the work process here is relevant to developers only.
The design phase here is moving the light spot to the design of the solution, it will be the connection bridge between the documented requirements and the implementation, where it converts the analysis to diagrams that can be implemented.
The architecture diagram will be created in the beginning to know what kind of systems will be used. Then the ER diagram will be normalized and prepared to help with implementing the database and create needed tables with attributes. In the end the class diagram will be designed and ready to be implemented using the proper technologies that will be specified in the “Choice Of Technology” section.
The Design is part of the Elaboration phase, the workload will be moved from the analysis to the design after finishing the analysis.

3.1 System Architecture

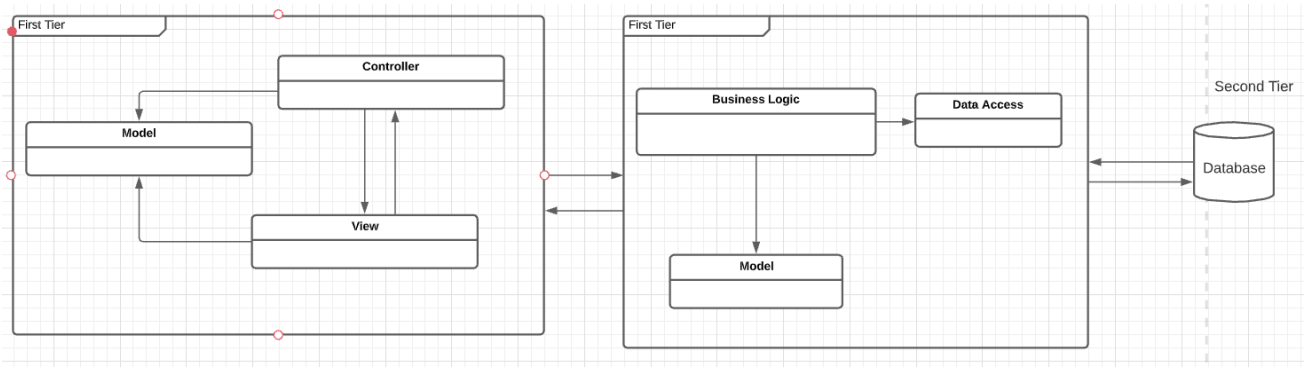


Figure 9 - Client/Server architecture diagram

The above diagram represents a 2-tier Client/Server architecture with 3 layers. The first tier consists of the first and second layer while the second tier consists of only the third layer. The first layer contains the Model-View-Controller. The second layer holds the Model, Business Logic and Data Access. The third layer holds the Database which acts as a remote server and is also part of the second tier.

3.2 ER Diagram:

The design of the database was done by making the conceptual ER diagram and applying the normalization process on it to organize the data and be able to implement it in Microsoft SQL Server Management.

The design of the conceptual ER diagram was done depending on the domain model, and after applying the first normal form (1NF), where each attribute can hold only atomic value.

The second normal form(2NF) was applied on Application table where User table and Application table were split into two tables so the attributes AppStatus and ApplicationDate are dependent on one primary key which is the applicationID.

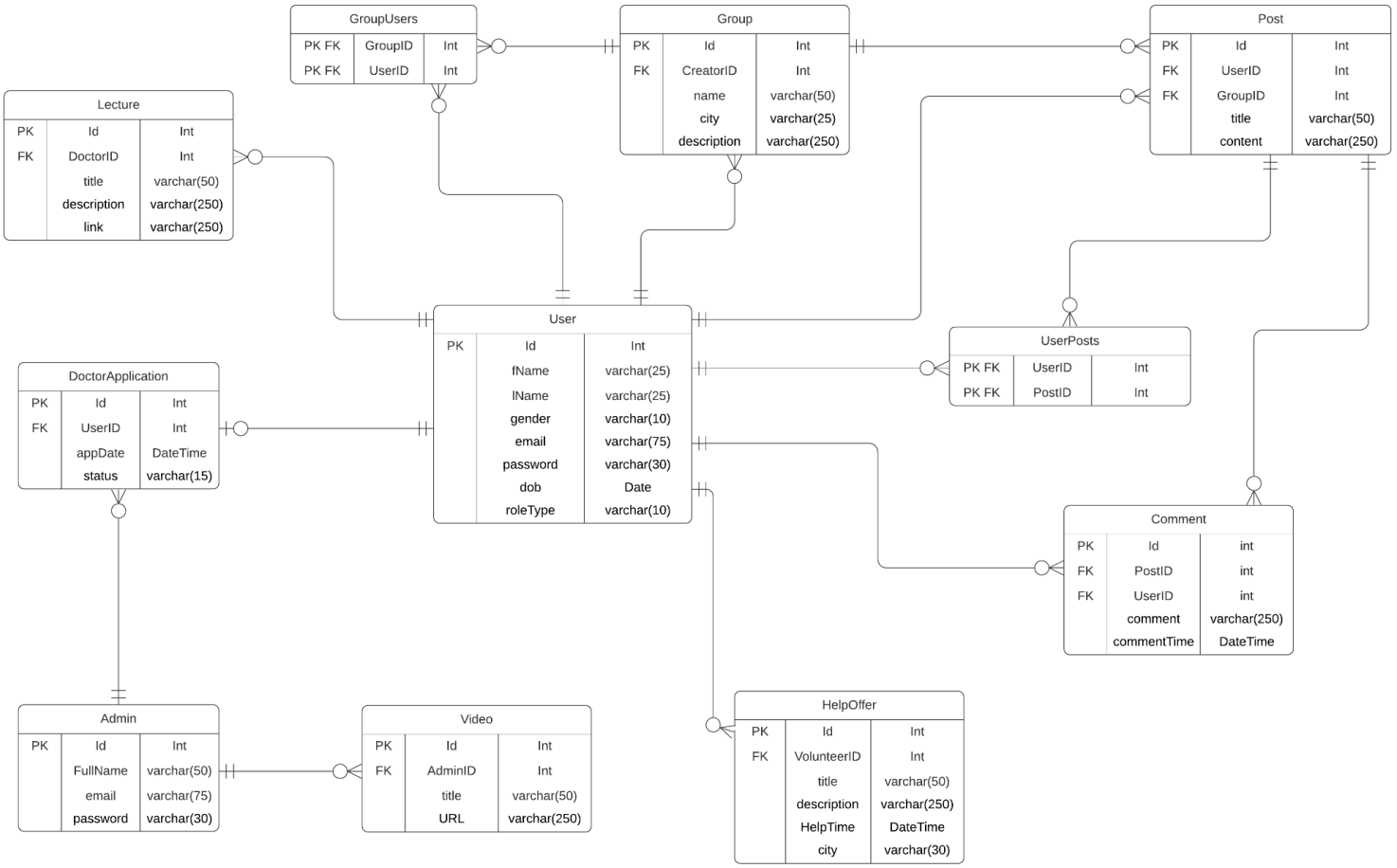


Figure 10 - ER Diagram

The normalization was done by removing transitive dependencies as well as referential dependencies, removing many to many relationships and removing any attributes that are not dependent on key attributes.

3.3 Class Diagram

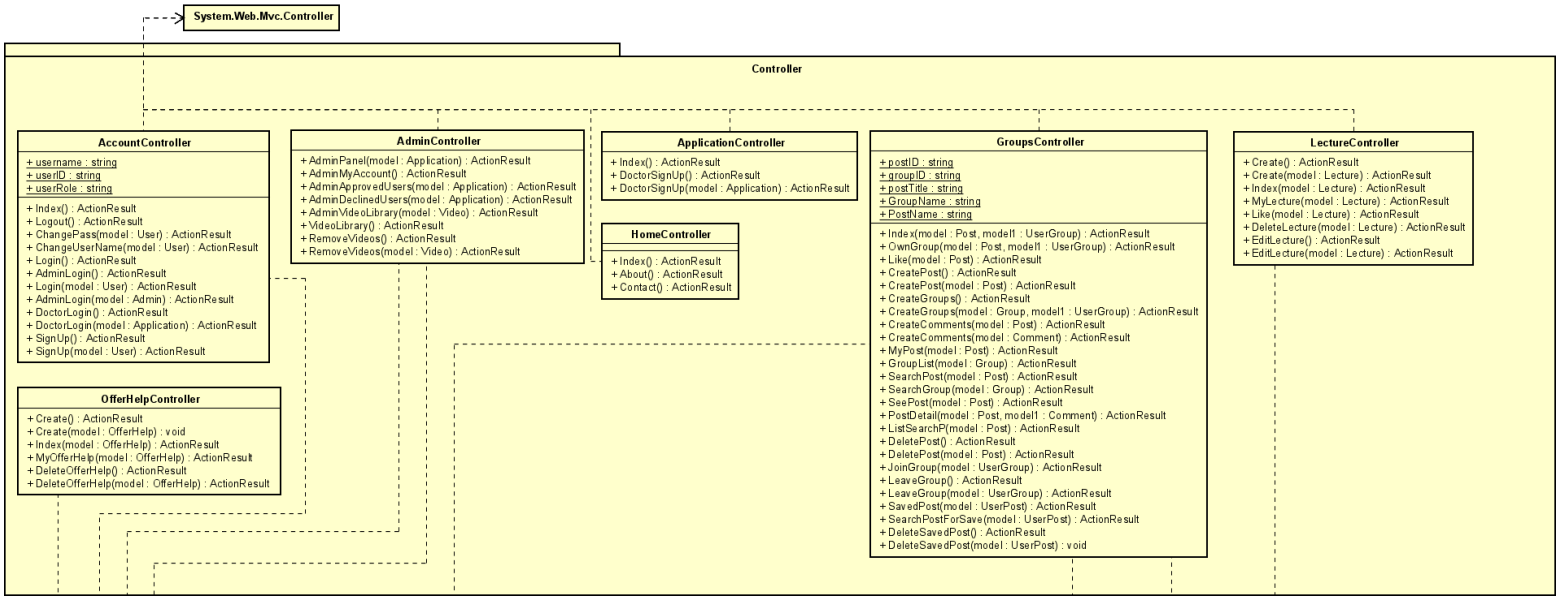


Figure 11 - Controller package

The system architecture is based on the Model-View-Controller pattern. The Controller section acts as an interface between the Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output [4] *(MVC Framework - Introduction - Tutorialspoint, 2021)*. There are a total of 7 controllers - Account, Admin, Application, Groups, Home, Lecture and OfferHelp controller.

Figure 12 - Business Logic package

The business logic is a part of the program that is responsible for defining real-world business rules that determine how data can be created, stored and changed [5] *(What really is the "business logic"?, 2021)*. CRUD is an acronym that stands for **Create**, **Read**, **Update** and **Delete**. Those are the four basic operations that you can perform on a database tuple.

The below examples explain some of the CRUD operations in the Business Logic that are performed onto the database:

Create: Method CreateGroup() in GroupProcessor is responsible for creating groups by constructing a model that contains group parameters such as GroupName, GroupCreator and others.

Read: Method LoadGroups() contains an sql query that selects all columns in the Group table and constructs a GroupModel list.

Update: Method LikeAddL in the LectureProcessor updates (saves) the number of likes for a given lecture when a user likes a specific lecture into the database.

Delete: Method DeleteLectures() in LectureProcessor takes as an argument the name of the lecture, searches for its name in the database and deletes it. Changes are saved afterwards.

Each method in the BusinessLogic calls the Data Access Object in order to communicate with the database.

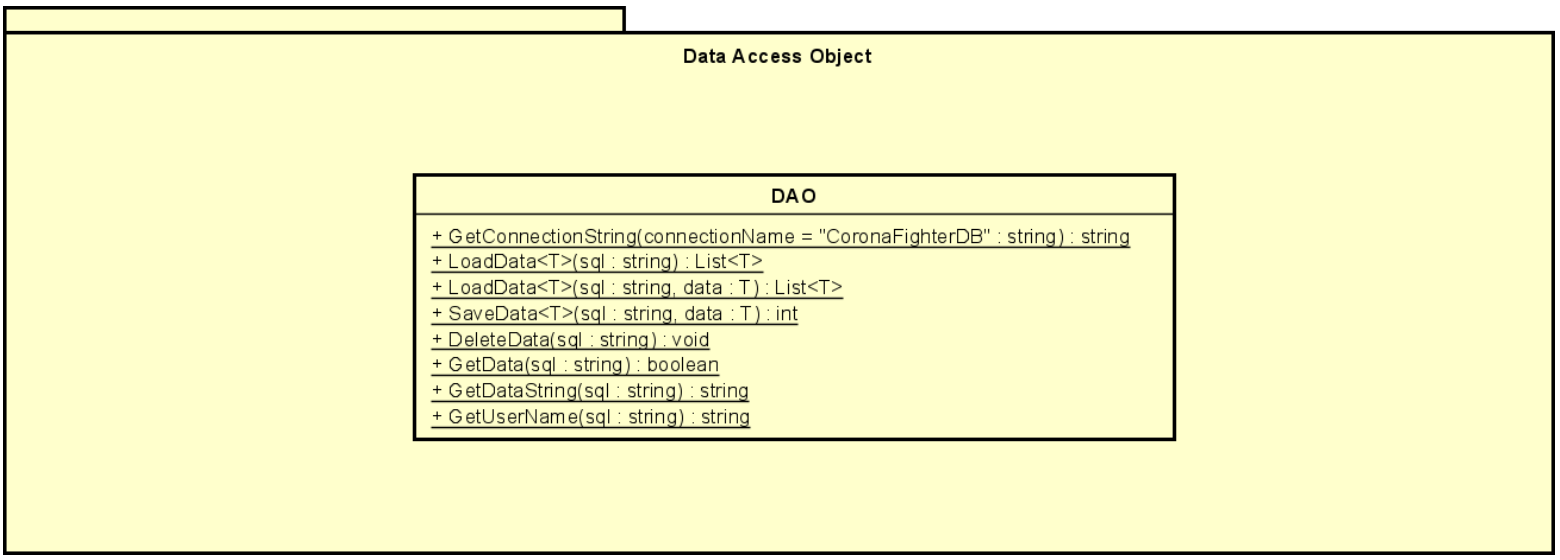


Figure 13 - Data Access Object package

The Data Access Object (DAO) class is used to bridge a connection between the first tier and second tier in order to communicate data between the different parts of the system. The class is used to isolate the application/business layer from the persistence layer (usually a relational database, but it could be any other persistence mechanism) using an abstract API [6] *(The DAO Pattern in Java | Baeldung, 2021).*

When a CRUD operation is made from within the business logic, the DAO is called which executes the relevant query onto the database and the end-result of the operation is returned from it. The DAO object does not contain connection parameters but instead looks them up in the Web.config file in the project by the name of the connection name which in this case is “CoronaFighterDB”.

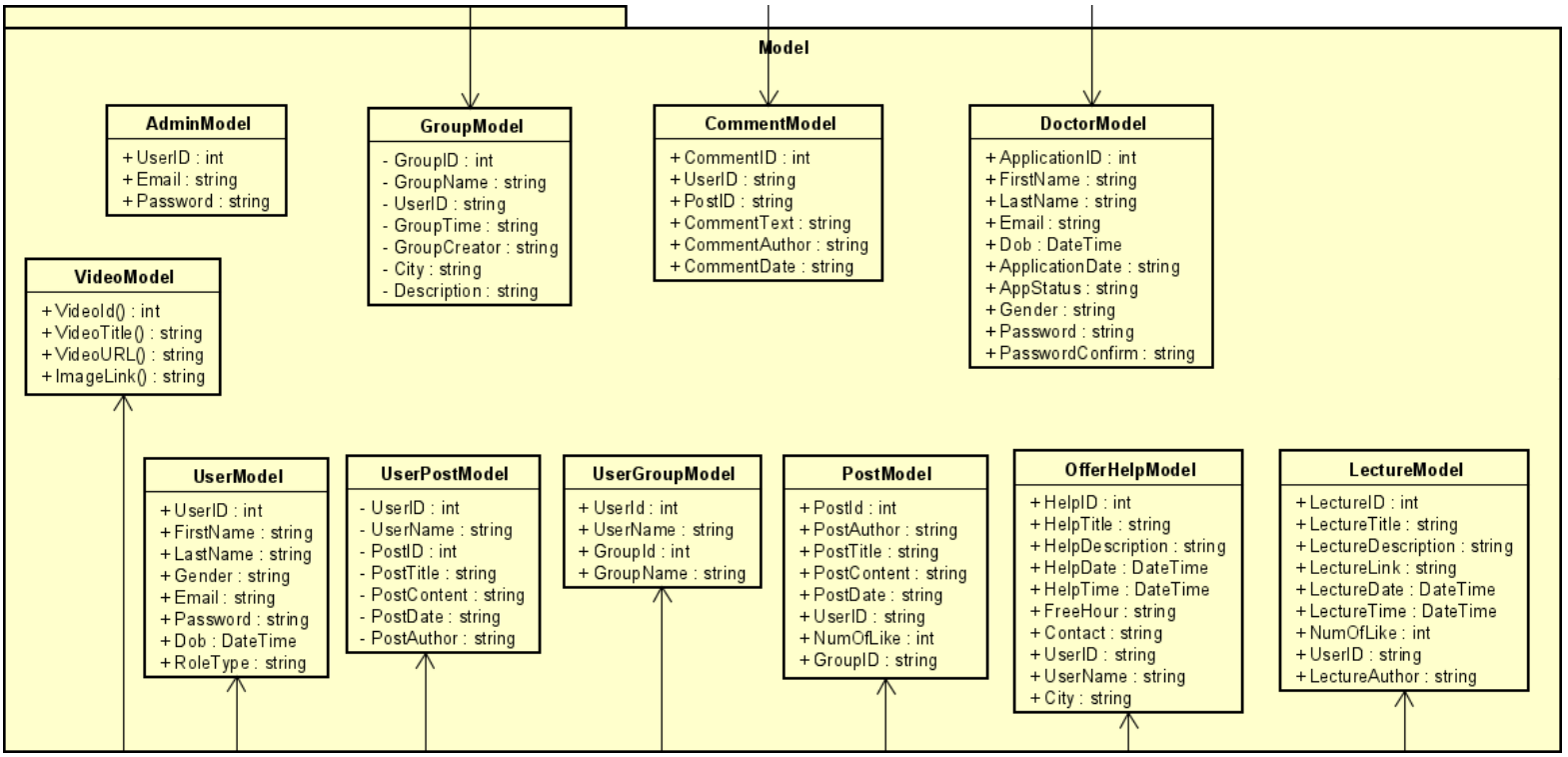


Figure 14 - Model package

The Model package contains Model classes that consist of only attributes. These classes do not contain any logic describing how to present the data to a user. Each model in the CoronaFighter system is composed of attributes that correspond to the specified model. For example, the UserGroupModel is used in UserProcessor class (BusinessLogic) in the LoadJoinedGroups method to form a list of UserGroupModels. Each UserGroupModel contains a UserID, UserName, GroupID and GroupName.

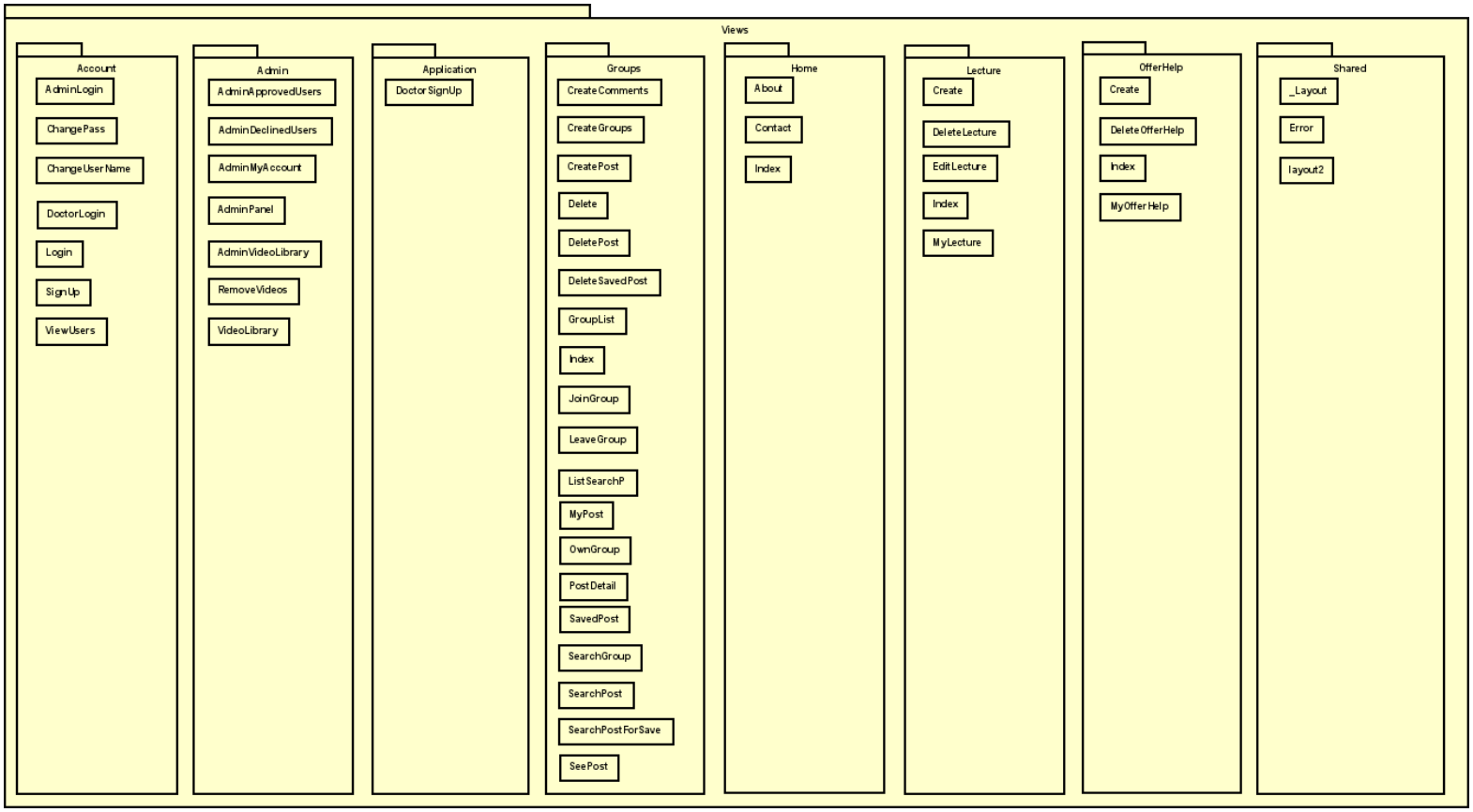


Figure 15 - Views Package

The View presents the model's data to the user and communicates with the model in case of updates triggered by the user. The view knows how to access the model's data, but it does not know what this data means or what the user can do to manipulate it [7] *(MVC Design Pattern - GeeksforGeeks, 2021)*. Each layout in the CoronaFighter project is categorized under a corresponding package, for example, the DoctorLogin is part of the Account package.

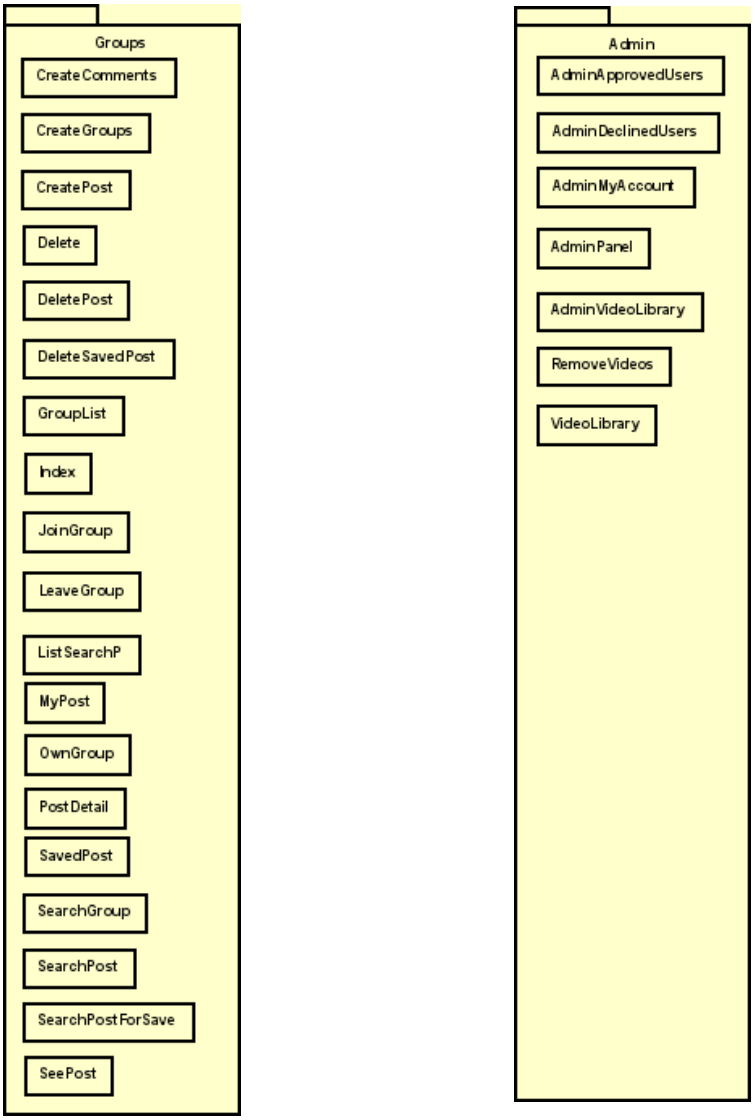


Figure 16 - Groups and Admin packages examples

An example of some of the packages in the View are the Groups and Admin packages. The Groups package contains a total of 19 CSHTML web pages necessary for displaying Group-related functionalities in the view. Admin contains a total of 7 pages.

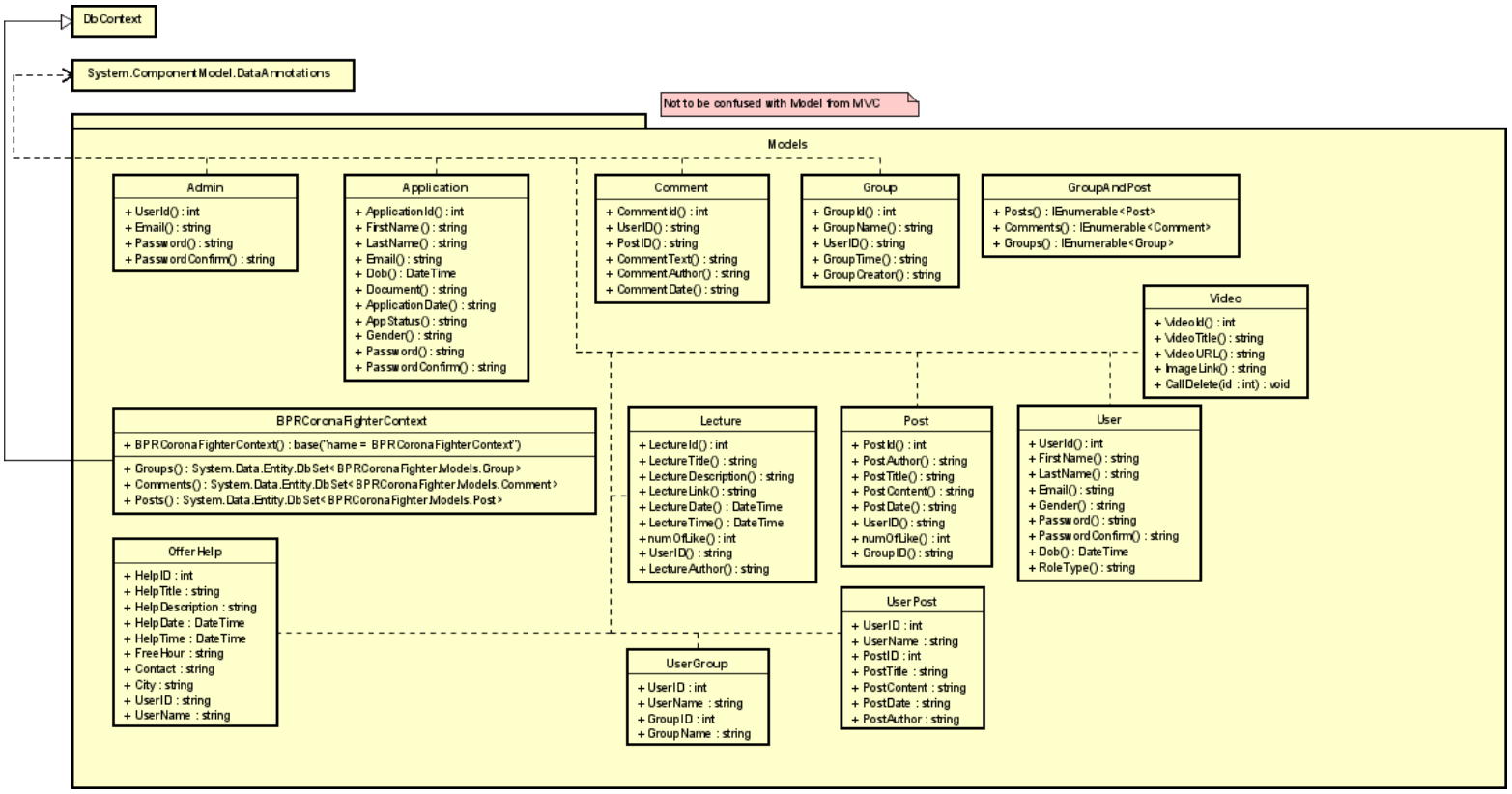


Figure 17 - Models package

Data Annotations are validations that are put into models to validate the input from a user. ASP.NET MVC provides a unique feature in which models can be validated by using the Data Annotation attribute [8] *(Model Validation Using Data Annotations In ASP.NET MVC, 2021)*.

To use data annotations the namespace System.ComponentModel.DataAnnotations must be imported into the model as can also be seen in Figure 17. There are multiple types of data annotations in ASP.NET MVC. An example of a data annotation is the *Required* annotation. This attribute specifies that the value is mandatory and cannot be skipped.

3.4 Sequence diagram

Sequence Diagrams were made to understand how the functions will go through the system and object interactions in the system.

The following Sequence Diagram is showing the detailed process of the "Sign up" function. The user starts by clicking on the "Sign Up" button, then the view asks the controller to return the "SignUp View", the user will fill up the format and will click on the "sign up" button, the View will send the request to the controller to handle the inputs. First step in the controller is to check from the Model if all inputs are valid, if yes, the controller starts handling the inputs by calling the "CreateUser" function in the Processor. Here the process will be moved from handling the front-end to the back-end. The processor now is responsible to assign these inputs to the Business Model and creates a query to instantiate a new user in the database, then it will call "SaveData(sql, Data)" function in the DAO class, which takes two parameters the assigned data and the sql query. The DAO class will connect to the database and execute the sql query with the assigned data and it will catch any errors that may happen in the connection or in the execution. If there are no errors the Controller will return the Home view with a welcoming message.

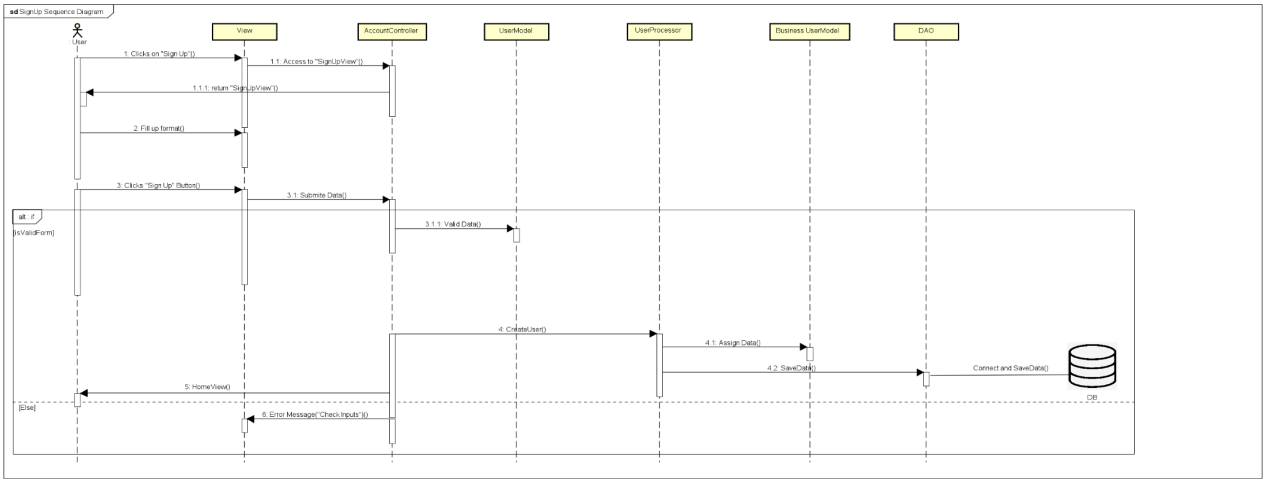


Figure 18 - Sign Up Sequence Diagram

Another example of the system where the user creates a group, the user needs to be logged in to see the “Create Group” button. When the user clicks on “Create Group" button the View will ask the controller to return “CreateGoupeView", the user here will be able to see 3 inputs: group name, the city name to specify the target city, and group description to give other users a better idea about the group. when the user fills up the format and clicks on the “Create” button that will fire an action in the controller which will check the inputs as well as if the group name already exists. If inputs are valid and the name does not exist, the controller will send the assigned data to the processor to handle this data and assign them to the model and then to the DAO class which will make the connection and execute the create group query to the Database.

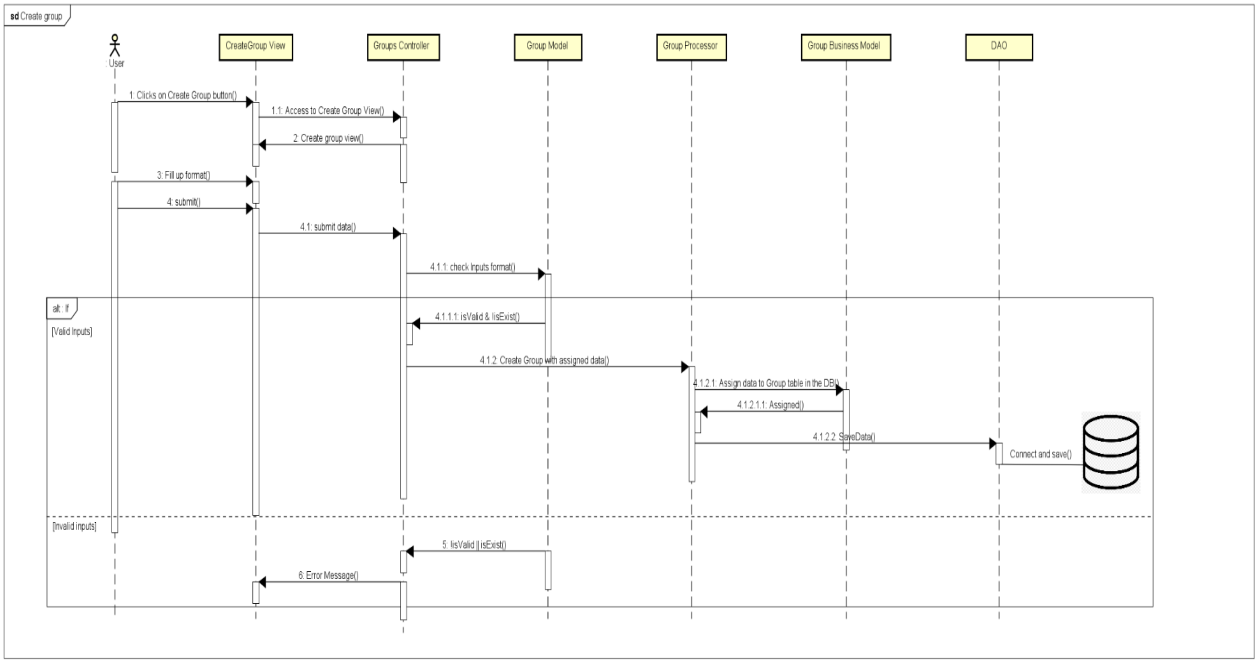


Figure 19 - Create Group Sequence Diagram

3.5 Design Patterns

MVC

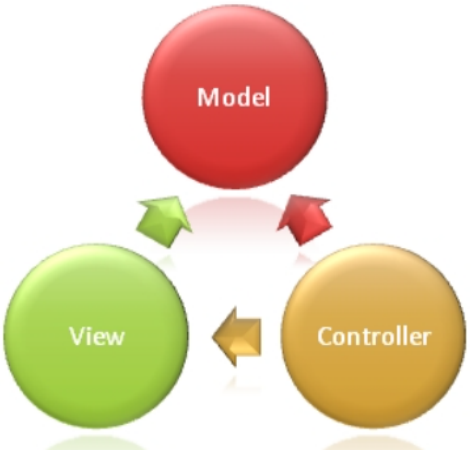


Figure 20 - MVC model [9] *(Overview of ASP.NET Core MVC, 2021)*

MVC stands for Model-View-Controller and is a software architectural design pattern. The goal of MVC is to separate functionality, logic and the interface in an application. This has the added benefit of organizing the system and allows multiple developers to work on the project with ease.

The Model is responsible for getting and manipulating data. The View takes care of the User Interface of the system.Finally, the Controller is responsible for taking user input.

The front-end of the Corona Fighter system will make use of the MVC pattern to have more separation of concerns according to the SOLID principles. The system will make use of multiple controllers, models and views to follow the single-responsibility principle where each of them will have only a specific functionality.

3.6 Choice of technology

The concept of technology may be defined as the expertise, equipment and procedures used by an organization to transform resource inputs into outputs [10] *(Choice of technology, 2021)*. It can be defined as the methods, skills and techniques that will be used to produce the end-product.

3.6.1 ASP.net MVC

ASP.NET gives a powerful, patterns-based way to build dynamic websites using the MVC pattern that enables a clean separation of concerns. [11] (ASP.NET MVC Pattern | .NET, 2021)

The Corona Fighter Web application makes use of ASP. NET framework, which is a unified Web development platform. This platform provides a safe, stable and scalable system.

The .NET Environment can make use of the .NET Framework and is compatible with C # and others. C# is a required programming language in this project due to the fact that the system uses the .NET Environment.

ASP.NET technology is designed to be completely object-oriented and primarily Web oriented.

3.6.2 GitHub Desktop for version control

GitHub Desktop is a tool that allows developers to interact with GitHub from their personal computer. It is an open-source tool which enables better productivity by encouraging the development team to interact in a simple and efficient way.

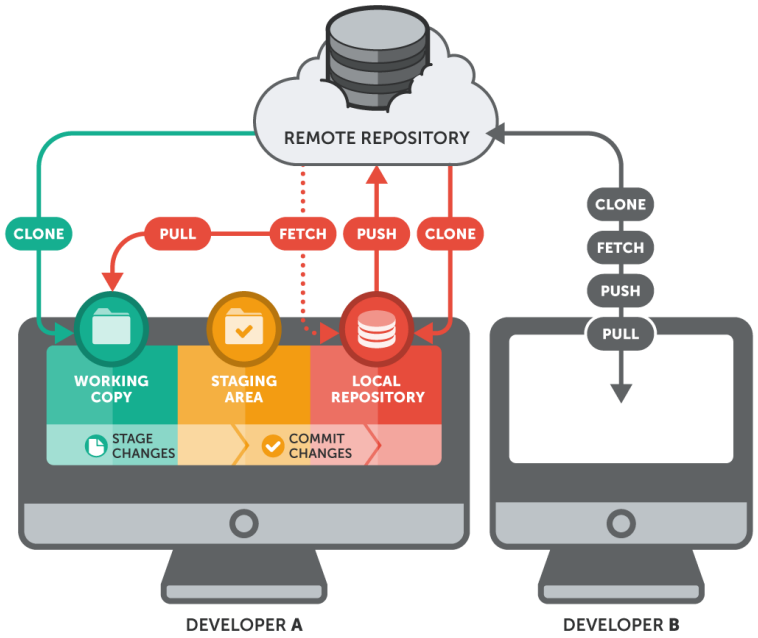


Figure 21 - Example of operations between two different local machines and remote repository [12] *(Tools to develop fast and with efficiency, 2021)*

Some of the features that GitHub Desktop provides are:

- Current changes viewing
- History of changes and rollback (reversion)
- Support for multiple work repositories
- Support for multiple work branches
- Pull/Push operations based on any new changes

The GitHub website version together with the desktop version will be used to have full control of the system implementation by the development team. This way, if a problem occurred with one of the machines of the system developer, a remote copy of the system can be restored if necessary or reverted if the remote repository has an issue which was pushed by accident.

GitHub Actions will be used for Continuous Integration (CI) where the pushed code will be tested before merging with the master remote branch.

3.6.3 Visual Studio for code management

The Visual Studio integrated development environment (IDE) was used during this project to develop, execute and debug project code. The built-in code editor provided ease of access to each section of the project structure directory. This IDE provides the possibility to have multiple projects into a single solution. The first project is the front-end - the ASP.NET Core Web App. This is where the project code is facilitated. The second project is the backend which contains the Business Logic, Data Access and Models. The third project contains each test that is performed on the other two projects. Each project is of different type as indicated by the icon next to their name as can be seen in Figure 22.

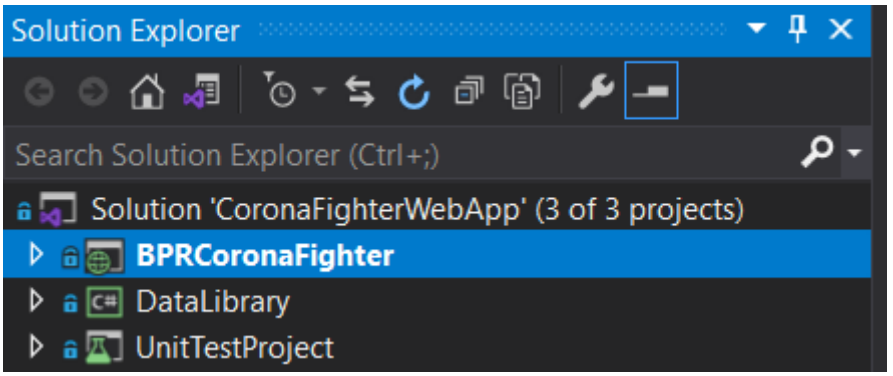


Figure 22 - Different project types within a single Visual Studio solution

3.6.4 Programming language for system code

C# is an Object-Oriented programming (OOP) language developed by Microsoft that runs on the .NET Framework. C# is used to develop web apps, desktop apps, mobile apps, games and much more [13] *(C# Tutorial (C Sharp), 2021)*. When developing a web application the best way to do this through Visual Studio is with the use of C# as is the case for the Corona Fighter web application.

HTML stands for HyperText Markup Language. It is the code which is used to structure a web page and its content. HTML consists of a series of elements which are used to enclose, or wrap, different parts of the content to make it appear a certain way, or act a certain way. [14] *(HTML basics - Learn web development | MDN, 2021)*

CSHTML (C-Sharp HTML) is the View Engine in the MVC Framework and uses Razor pages. Razor View engine is a markup syntax which helps to write HTML code in web pages using C#. Pages that are part of the view in the Corona Fighter project use this to display the View to the user and communicate with the model and thus other parts of the system when the user interacts with the view.

JavaScript is a scripting language that can further extend the functionality of HTML-based web pages. The Corona Fighter web application uses JavaScript in the form of dialog popups alerts for places in the Views where a user would need to be alerted about invalid input. JavaScript can be used as a separate script located elsewhere from the View or inside the View.

With these four programming languages, the system will be able to function properly and fulfil its purpose as a web application.

3.6.5 SQL: Structured Query Language

SQL was chosen as the language of accessing the database of the system. Because it allows to:

create tables and views easily in the database, also insert, retrieve, update, delete data in these tables.

To be able to implement the defined ER diagram using SQL, there will be a need of using the Microsoft SQL Management Studio that allows to create and execute queries to access and manipulate the database.

3.6.6 Microsoft SQL Management Studio (SSMS) for database management

SQL Server Management Studio (SSMS) is an integrated environment that allows access to the SQL Server and manage it.

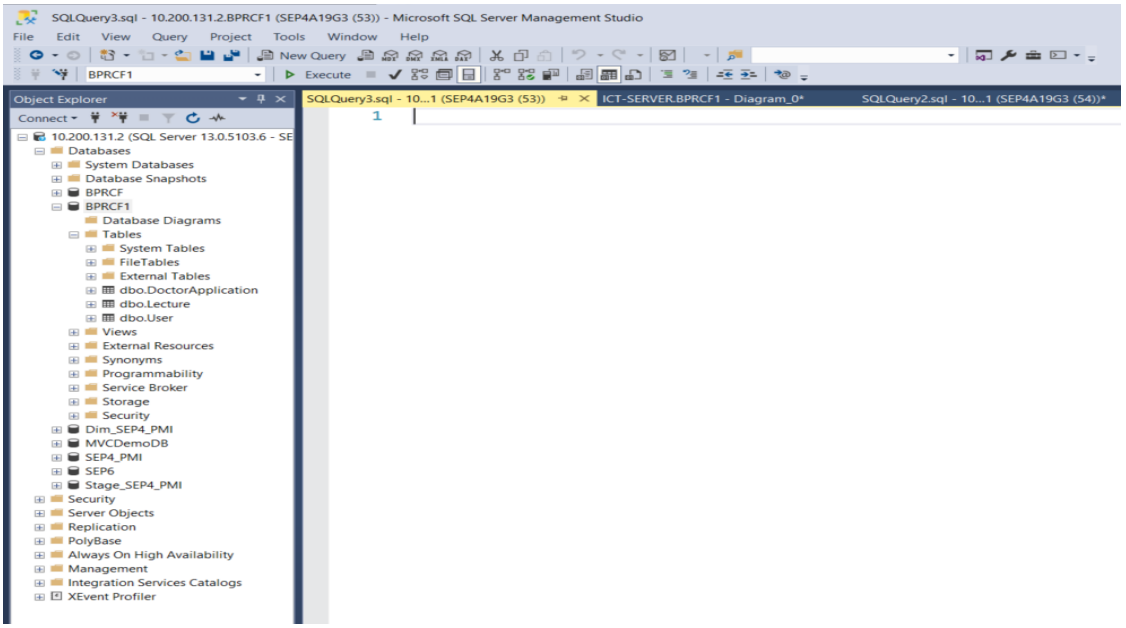


Figure 23 - SSMS

The SSMS was used in the project for the following reasons:

1. It provides many different tools to have a better control of the database.
2. Easy to use, add and modify databases, queries, tables, etc.
3. It allows to connect to remote servers, where the team will use VIA Server to have a remote database where all members can access and manipulate it when needed (there is also a need of using VIA VPN to have access).
4. It works very well with Visual Studio and Asp .Net, and it is possible to access the databases on the server using Visual Studio, so it will provide a better developing environment.

3.6.7 Entity Framework

Entity Framework is an open-source object-relational mapping framework, which will be used in the system while the system is a .Net application. The framework will help to work with the database immediately from the second layer classes.

The system will use Entity Framework Database-First approach where the database will be created first with all needed tables, then using Entity Framework the connection to the database will be created and the operations such as create, read, update and delete will be done using objects of the second layer classes.

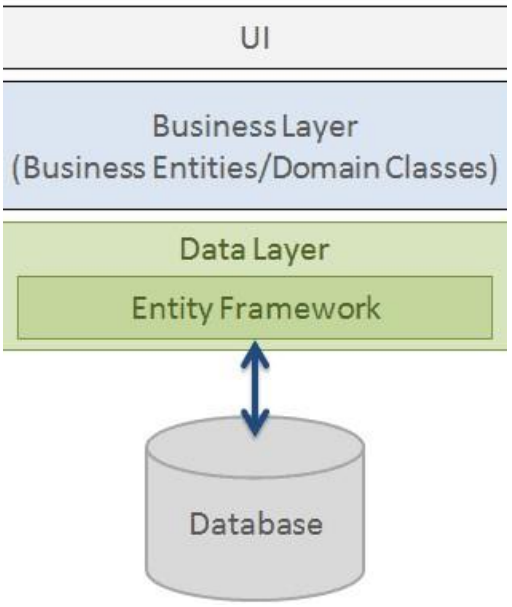


Figure 24 - Entity Framework [15] *(What is Entity Framework?, 2021)*

3.7 Data protection

3.7.1 What is personal data?

Personal data is any information that relates to an individual who can be directly or indirectly identified. Names and email addresses are personal data. Location information, ethnicity, gender, biometric data, religious beliefs, web cookies, and political opinions can also be personal data. [16] *(What is GDPR, the EU's new data protection law? - GDPR.eu, 2021)*

3.7.2 What is data processing?

Data processing is any action performed on data, whether automated or manual. This includes collecting, recording, organizing, structuring, storing, using, erasing and any type of manipulation of user data by the data processor.

3.7.3 What is a data subject?

The person whose data is processed. These are customers or site visitors.

3.7.4 What is a data controller?

A data controller is a person who decides why and how personal data will be processed.

3.7.5 What is a data processor?

A third party that processes personal data on behalf of a data controller.

3.7.6 What is GDPR?

GDPR is an acronym for General Data Protection Regulation and was implemented by the European Union parliament on the 14th of April, 2016 and took effect from the 25th of May, 2018. It is a set of laws that regulate how personal data can be collected, stored and used. The organization that collects personal data is called the data controller. The data controller has an obligation to protect the data it collects by keeping it private and only using and sharing it in an authorized manner. Data controllers often share data with data processors. These are organizations who do some form of analysis and research on the data provided to them by the data controllers but do not collect it themselves. An example of this is a payment processing company which acts as a data processor. Under GDPR, users' personal data can only be accessed if there is a legitimate reason to do so.

The most common way is for a user to give consent of use – in other words, the customer has explicitly given access to a business which can then access the customer's data. Individuals have a number of rights over their data. This includes having the right to have all of the individual's data erased from the service that they would like to remove their data from and the right to have their data shared with another company. GDPR applies to every company that collects personal data from EU data subjects, regardless of where the company is established in the world.

3.7.7 How can the development team of the Corona Fighter web application use GDPR in their system?

Under the GDPR, it is the legal responsibility of website owners and operators to make sure that personal data is collected and processed lawfully. One of the most common ways for personal data to be collected and shared online is through website cookies [17] *(GDPR and cookie consent | Compliant cookie use, 2021)*. Since we use a web application in our project, the front-end of our application could display visual data about GDPR consent information and prompt the end-user.

The European Data Protection Board's (EDPB) guidelines from May 2020 clarifies what constitutes valid consent on websites in compliance with the GDPR. EDPB guidelines state that a website's cookie banner is not allowed to have pre-ticked checkboxes and continued scrolling or browsing by users cannot be considered as valid consent for processing of personal data.

Users must freely give a clear and affirmative action to indicate their consent in order for your website to activate cookies and process personal data.

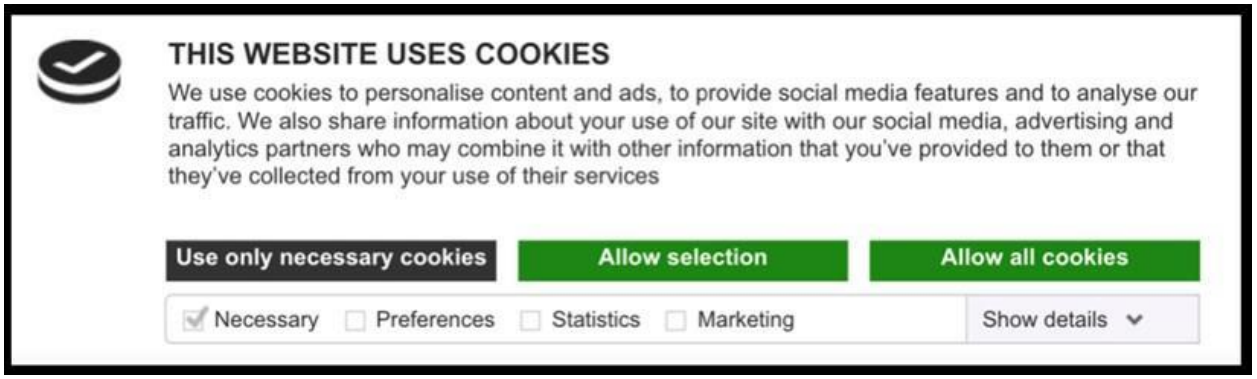


Figure 25 - Cookies prompt usage on a website [17] *(GDPR and cookie consent | Compliant cookie use, 2021)*

In Figure 25, a User prompt pop-up is displayed upon visiting our website. The users are able to control their own data privacy on our website therefore making it GDPR compliant.

4. Implementation

The Corona Fighter website uses the ASP.net MVC pattern in the first layer of the system architecture in order to structure the program User Interface.

When the project utilizes the MVC pattern, requests are forwarded to the controllers that contain the backend method logic within each function. The controllers themselves utilize their equivalent class that matches their name in the Business Logic - for example, LectureController would call upon LectureProcessor. A Controller may use more than one Processor to accomplish its targeted end-functionality. When these classes within the Business Logic are called upon, they have the responsibility to instantiate a class from the Model package that a function in the Business Logic would need to make use of. To get data from the remote database, Processor classes call upon the Data Access Object - DAO with a parameterized query such as SELECT or INSERT in order to fetch or save data into the remote database.

Finally, the Controller chooses the appropriate View to display, and afterwards, it provides it with the Model as a return statement. The View renders the final page, based on the data from the Model.

4.1 Group function

The following sections explain how to implement the group functionality, including the complete path to view the group list, create a group, and join or leave a group.

4.1.1 View groups list

The following snippet of code is responsible for handling joining of groups. A user must be logged in first so that they are able to see the following functions.

```
public ActionResult GroupList(Group model)
{
    var data = LoadGroups();
    List<Group> groups = new List<Group>();
    foreach (var item in data)
    {
        groups.Add(new Group
        {
            GroupName = item.GroupName,
            GroupTime = item.GroupTime,
            GroupCreater = item.GroupCreater,
        });
        groups.Reverse();
    }
    return View(groups);
}
```

Figure 26 - GroupList method in GroupsController class

A user would be able to see a list of groups and see information about them. The user can then select which group or groups they wish to join from the list. The foreach loop creates a Group object with the GroupName, GroupTime and GroupCreator and adds this Group object to a list of Groups. Since the list elements are added from old to new, we would like to order the groups by time from newest created to oldest. This is accomplished with the help of the Reverse() method which re-orders the list in the correct way.

```
public static List<GroupModel> LoadGroups()
{
    string sql = @"select  groupName, groupTime, groupCreater,userID
                        from dbo.[Group];";
    return DAO.LoadData<GroupModel>(sql);
}
```

Figure 27 - LoadGroups method in GroupProcessor class

In the above example code, the system calls upon the LoadGroups() method which is responsible for getting a list of groups from the database by executing an sql query on it.

```
public static List<T> LoadData<T>(string sql)
{

    using (IDbConnection cnn = new SqlConnection(GetConnectionString()))
    {
        return cnn.Query<T>(sql).ToList();
    }
}
```

Figure 28 - LoadData method in DAO class

The LoadData() method in DAO class is responsible for opening the connection to the remote database and returning a query result as a list.

```
@foreach (var item in Model.Posts)
{
    <div class="card gedf-card">
        <div class="card-header">
            <div class="d-flex justify-content-between align-items-center">
                <div class="d-flex justify-content-between align-items-center pt-5">
                    <div class="mr-2">
                    </div>
                    <div class="ml-2">
                        <span class="text-muted h7" style="margin-left: 0px; font-size: 12px;">
                            Author:
                            @Html.DisplayFor(modelItem => item.PostAuthor)
                        </span>
                        <br />
                            <span class="text-muted h7" style="margin-left: 0px; font-size: 12px;"> Title:
                                @Html.DisplayFor(modelItem => item.PostTitle)
                        </span>
                        <br />
                            <span class="text-muted h7" style="margin-left: 0px; font-size: 12px;">
                                @Html.DisplayFor(modelItem => item.PostDate)
                        </span>
                    </div>
                    </div>
                </div>
            </div>
        </div>
        <div class="card-body">
            <p class="card-text text-muted">
                @Html.DisplayFor(modelItem => item.PostContent)
            </p>
        </div>
        <div class="card-footer">
            <button type="button" class="fa fa-comment mr-2" onclick="location.href='@Url.Action("CreateComments", "Groups")'" />
        </div>
    <br />
}
```

Figure 29 - Index view

A foreach loop was used to go through the data that was fetched from Model.Posts and stored into the item variable. The Html.DisplayFor() method is used to display PostAuthor, PostTitle, PostDate and PostContent in the OwnGroup.cshtml view.

The Url.Action calls the GroupsController using the keyword “Groups”, therefore, the Url.Action method recognizes that this is an invocation to GroupsController and calls the CreateComments() method.

4.1.2 Join a group

```
public ActionResult JoinGroup(UserGroup model)
{
    ModelState.Remove("UserId");
    ModelState.Remove("UserName");
    ModelState.Remove("GroupId");
    if (ModelState.IsValid)
    {
        model.UserId = int.Parse(AccountController.userID);
        model.UserName = AccountController.username;
        if (GetGroupID(model.GroupName) == "Nothing")
        {
            return Content("<script language='javascript' type='text/javascript'>"+
                "alert('Nothing Found! ');history.go(-1);location.reload();</script>");
        }else{
            model.GroupId = int.Parse(GetGroupID(model.GroupName));
            int recordsCreated = JoinGroups(model.UserId, model.UserName, model.GroupId, model.GroupName);
            postTitle = model.GroupName;
            return RedirectToAction("GroupList", "Groups");
        }
    }
    return View();
}
```

Figure 30 - JoinGroup method in GroupsController class

Users are able to write a groupName that is in the group list, then join it. The ModelState.IsValid method is used to check whether the user input is valid or not - if the user does not provide any input and presses the search button, then, they will not be able to see any groups. If the user inputs valid search data but there are no found groups, the system will present them with a message that nothing was found from the performed search query. The system will then return the user to the initial search page. Finally, if the search result is valid, the system will call the JoinGroups() function.

```
public static int JoinGroups(int userID, string username, int groupID, string groupname)
{
    UserGroupModel data = new UserGroupModel
    {
        UserId = userID,
        UserName = username,
        GroupId = groupID,
        GroupName = groupname,
    };
    string sql = @"insert into dbo.[UserGroup] (userID, userName, groupID, groupName)
                        values(@UserId, @UserName, @GroupId, @GroupName);";
    return DAO.SaveData(sql, data);
}
```

Figure 31 - JoinGroups method in GroupProcessor class

The JoinGroups() method takes a userID, username, groupID and groupNameas input from the user and uses this input to form a query.

```
public static int SaveData<T>(string sql, T data)
{
    try
    {
        using (IDbConnection cnn = new SqlConnection(GetConnectionString()))
        {
            return cnn.Execute(sql, data);
        }
    }
    catch (Exception)
    {

        return 0;
    }

}
```

Figure 32 - SaveData method in DAO class

The DAO method SaveData() is called to save user data into the remote database. If there is an exception such as a problem with connecting, the return value would be zero, otherwise, the return statement would be how many rows were altered in the database by the performed query.

4.1.3 Go to your groups

```
public ActionResult SearchGroup(Group model)
{
    ModelState.Remove("City");
    ModelState.Remove("Description");
    if (ModelState.IsValid)
    {
        groupID = GetGroupID(model.GroupName);
        if (groupID == "Nothing")
        {
            return Content("<script language='javascript' type='text/javascript'>"+
                "alert('Nothing Found! ');history.go(-1);location.reload();</script>");
        }
        else
        {
            int recordsCreated = SearchGroups(int.Parse(groupID));
            GroupName = model.GroupName;
            ViewBag.GroupName = GroupName;
            return RedirectToAction("OwnGroup", "Groups");
        }
    }
    return View();
}
```

Figure 33 - SearchGroup method in GroupsController class

Users are able to go to their joined groups. The GetGroupID() method is used to transform the user input variable groupName into groupID due to the fact that the function SearchGroup() uses an ID to perform a search. The SearchGroups() method is used to check if data exists in the database. If yes, the search result will be displayed in the view of the user and thus, the user will be able to see private posts in the OwnGroup page which will refresh automatically.

```
public static int SearchGroups(int groupID)
{
    GroupModel data = new GroupModel
    {
        GroupId = groupID,
    };
    string sql = @"select groupName, groupTime, groupCreater,userID
                from dbo.[Group] where groupID ='" + @groupID + "'";
    return DAO.SaveData(sql, data);
}
```

Figure 34 - SearchGroups method in GroupProcessor class

The SearchGroups() method takes a groupID input from the user and uses this input to form a query.

4.1.4 See joined groups

```
var data1 = LoadJoinedGroups(AccountController.username);
List<UserGroup> listOfGroups = new List<UserGroup>();
for (int i = 0; i < data1.Count; i++)
{
    listOfGroups.Add(new UserGroup
    {
        GroupName = data1[i].GroupName,
        UserName = data1[i].UserName,
    });
    listOfGroups.Reverse();
}
```

Figure 35 - Index method in GroupsController class

In the main post page, the user will be able to see groupName that they have already joined. Firstly, the system will retrieve a user's username from the AccountController which keeps track of the current logged in user. The UserGroup model is used as a parameter to create the listOfGroups. With this information, LoadJoinedGroups is called to get all groups that the current user is part of. The for loop searches data1 and if any

groups are found, they are added to the variable listOfGroups. Finally, the list is reversed so that the newly joined groups are on top in the View.

```
public static List<UserGroupModel> LoadJoinedGroups(string userName)
{
    string sql = @"select groupName from dbo.[UserGroup] where userName= '" + @userName + "'";
    return DAO.LoadData<UserGroupModel>(sql);
}
```

Figure 36 - LoadJoinedGroups method in GroupProcessor class

The LoadJoinedGroups() method forms an SQL query that selects the groups that are associated with a user's name meaning that the user is part of those groups. The list is then loaded into the index.cshtml. As a prerequisite, the user has to be logged in first as they are not able to navigate to this page if they are not logged in to the system.

4.1.5 See group members name

```
var data1 = LoadJoinedGroupsMembers(groupID);
List<UserGroup> listOfGroupsMembers = new List<UserGroup>();
for (int i = 0; i < data1.Count; i++)
{
    listOfGroupsMembers.Add(new UserGroup
    {
        GroupName = data1[i].GroupName,
        UserName = data1[i].UserName,
    });
    listOfGroupsMembers.Reverse();
}
```

Figure 37 - OwnGroup method in GroupsController class

With the method LoadJoinedGroupsMembers(), the user is able to see a list of members that are already part of the group that the user is viewing. A for loop is executed on the data1 variable which contains all members for a specific group. In the for loop, a new UserGroup model object is constructed and each user is assigned the group name and their user name.

```
public static List<UserGroupModel> LoadJoinedGroupsMembers(string groupID)
{
    string sql = @"select userName from dbo.[UserGroup] where groupID= '" + @groupID + "'";
    return DAO.LoadData<UserGroupModel>(sql);
}
```

Figure 38 - LoadJoinedGroupsMembers method in GroupProcessor class

The LoadJoinedGroupsMembers() method forms an SQL query that selects the user names that are associated with a group ID meaning that the user is part of those groups. The list is then loaded into the OwnGroup.cshtml.

4.1.6 Create a group

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult CreateGroups(Group model,UserGroup model1)
{
    ModelState.Remove("UserId");
    ModelState.Remove("UserName");
    ModelState.Remove("GroupId");
    if (ModelState.IsValid)
    {
        bool isdup = CheckDupG(model.GroupName);
        if (isdup)
        {
            return Content("<script language='javascript' type='text/javascript'>alert('Group already exist!');"+
                "history.go(-1);location.reload();</script>");
        }
        else
        {
            model1.UserId = int.Parse(AccountController.userID);
            model1.UserName = AccountController.username;
            model.UserID = AccountController.userID;
            model.GroupCreater = AccountController.username;
            int recordsCreated = CreateGroup(model.GroupName,DateTime.Now.ToString(), model.GroupCreater, model.UserID,model.City,model.Description);
            groupID = GetGroupID(model.GroupName);
            GroupName = model.GroupName;
            int recordsCreated1 = JoinGroups(model1.UserId, model1.UserName, int.Parse(groupID), model.GroupName);
            return RedirectToAction("OwnGroup", "Groups");
        }
    }
    return View();
```

Figure 39 - CreateGroups method in GroupsController class

The Patient user can create groups with the help of the CreateGroups() method. The method takes the Group model and the UserGroup model to form a new group. The UserID, UserName and GroupID parameters are removed by using the ModelState.Remove() method so that the system does not check their validation in the Group model. Then, an if statement is used to check ModelState.IsValid. If the statement is true, the system will then check if the group already exists in the database else it will create a new group with the initial input from the Patient. When a group has been successfully created, the Patient will be redirected to the page of their newly created group.

```
public class Group
{
    [HiddenInput(DisplayValue = false)]
    public int GroupId { get; set; }
    [Required]
    [Display(Name = "Group Name")]
    public string GroupName { get; set; }
    public string UserID { get; set; }
    public string GroupTime { get; set; }
    public string GroupCreater { get; set; }
    [Required]
    [Display(Name = "City")]
    public string City { get; set; }
    [Required]
    [Display(Name = "Description")]
    public string Description { get; set; }
}
```

Figure 40 - Group model

The above class is responsible for taking input from the user and parsing it to the model. The [Required] data annotation requires that the user inputs a GroupName, City and Description.

```
public static int CreateGroup(string groupName, string groupTime, string groupCreater, string userID,string city,string description)
{
    GroupModel data = new GroupModel
    {
        GroupName = groupName,
        GroupTime = groupTime,
        GroupCreater = groupCreater,
        UserID = userID,
        City = city,
        Description=description,
    };
    string sql = @"insert into dbo.[Group] (groupName, groupTime, groupCreater,userID,city,description)
                    values(@GroupName, @GroupTime,@GroupCreater,@UserID,@City,@Description);";
    return DAO.SaveData(sql, data);
}
```

Figure 41 - CreateGroup method in GroupProcessor class

The CreateGroup() method takes 6 parameters that contain information about a group - GroupName, GroupTime, GroupCreator, UserID, City, Description. Afterwards, a new GruopModel is instantiated and the main method parameters are assigned to the model parameters. When the data is put into the model, we then insert the model data into our

database with the help of an SQL INSERT query which we use when performing DAO.SaveData() with the SQL statement as a parameter and the model.

4.1.7 Leave a group

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult LeaveGroup(UserGroup model)
{
    ModelState.Remove("UserId");
    ModelState.Remove("UserName");
    ModelState.Remove("GroupId");
    if (ModelState.IsValid)
    {
        int recordsCreated = LeaveGroupss(model.GroupName);
        if (recordsCreated == 0)
        {
            return Content("<script language='javascript' type='text/javascript'>"+
                "alert('Nothing Found! ');history.go(-1);location.reload();</script>");
        }else{
            return RedirectToAction("GroupList", "Groups");
        }
    }
    return View();
}
```

Figure 42 - LeaveGroup method in GroupsController class

The user can leave groups with the help of the LeaveGroups() method. The method takes the UserGroup model to remove a patient from a group. The UserID, UserName and GroupID parameters are removed by using the ModelState.Remove() method so that the system does not check their validation in the Group model. Then, an if statement is used to check ModelState.IsValid. If the statement is true, then an integer variable named recordsCreated is assigned the outcome of the LeaveGroupss() method which returns the found number of groups which can either be 0 or 1. If it is 0, then, a popup dialog will appear on the user's view which informs them that nothing was found. If a group is found, the system will remove the user from the group and redirect them to the GroupsList page.

```
public static int LeaveGroupss(string groupName)
{
    UserGroupModel data = new UserGroupModel
    {
        GroupName = groupName,
    };
    string sql = @"delete from dbo.[UserGroup] where groupName ='" + @groupName + "'";

    return DAO.SaveData(sql, data);
}
```

Figure 43 - LeaveGroup method in GroupProcessor class

The LeaveGroupss() method is used to remove a user from a group that they have joined previously. The return statement of the method can be either 0 or 1 as an outcome of the performed SQL DELETE query whereby 0 means nothing found and 1 means the group was found in the system and the user was successfully removed from it.

4.2 Doctor verify

```
public ActionResult AdminApprovedUsers(Application model)
{
    ModelState.Remove("FirstName");
    ModelState.Remove("LastName");
    ModelState.Remove("Dob");
    ModelState.Remove("Gender");
    ModelState.Remove("Password");
    if (ModelState.IsValid)
    {
        int recordsCreated = ApproveApp(model.Email);
        if (recordsCreated == 0)
        {
            return Content("<script language='javascript' type='text/javascript'>"+
                "alert('Nothing Found! ');history.go(-1);location.reload();</script>");
        }
        else
        {
            return RedirectToAction("AdminPanel", "Admin");
        }
    }
    return View();
}
```

Figure 44 - AdminApprovedUsers method in AdminController class

The Admin can approve users with the help of the AdminApprovedUsers() method. The method takes an Application model to load approved users in the view. The FirstName,

LastName, Dob, Gender and Password parameters are removed by using the ModelState.Remove() method so that the system does not check their validation in the Application model. Then, an if statement is used to check ModelState.IsValid. If the statement is true, then an integer variable named recordsCreated is assigned the outcome of the ApproveApp() method which takes an Email parameter of a Doctor from the model which the Admin provided. If a Doctor is found, the system will update the status of the Doctor's application to Approved status.

```
public ActionResult AdminDeclinedUsers(Application model)
{
    ModelState.Remove("FirstName");
    ModelState.Remove("LastName");
    ModelState.Remove("Dob");
    ModelState.Remove("Gender");
    ModelState.Remove("Password");
    if (ModelState.IsValid)
    {
        int recordsCreated = DeclineApp(model.Email);
        if (recordsCreated == 0)
        {
            return Content("<script language='javascript' type='text/javascript'>" +
                "alert('Nothing Found! ');history.go(-1);location.reload();</script>");
        }
        else
        {
            return RedirectToAction("AdminPanel", "Admin");
        }
    }
    return View();
}
```

Figure 45 - AdminDeclinedUsers method in AdminController class

The Admin can decline users with the help of the AdminDeclinedUsers() method. The method is almost the same as the AdminApprovedUser() method whereby the only noticeable difference is that the Doctor's application is set to Declined status in the database.

```
public static int ApproveApp(string email)
{
    UserModel data = new UserModel
    {
        Email = email,
    };
    string sql = @"UPDATE  dbo.[Doctor] set appStatus='Approve' where email= '" + @email + "'";
    return DAO.SaveData(sql, data);
}
public static int DeclineApp(string email)
{
    UserModel data = new UserModel
    {
        Email = email,
    };
    string sql = @"UPDATE  dbo.[Doctor] set appStatus='Decline' where email= '" + @email + "'";
    return DAO.SaveData(sql, data);
}
```

Figure 46 - ApproveApp and DeclineApp method in UserProcessor class

The ApproveApp() and DeclineApp() methods serve to update the status of a Doctor's application in the system by performing an SQL UPDATE query which sets the appStatus to either Approve or Decline in the database.

4.3 Multi-model in one view

```
public class GroupAndPost
{
    public IEnumerable<Post> Posts { get; set; }
    public IEnumerable<Comment> Comments { get; set; }
    public IEnumerable<Group> Groups { get; set; }
    public IEnumerable<UserGroup> UserGroups { get; set; }
}
```

Figure 47 - GroupAndPost model

GroupAndPost class is a single "Big" class that has multiple models. It contains multiple models as individual methods. In the above example, the View model has four methods. This GroupAndPost class is passed to the view as a model.

```
public ActionResult OwnGroup(Post model, UserGroup model1)
{
    ViewBag.UserName = AccountController.username;
    ViewBag.UserRole = AccountController.userRole;
    ViewBag.GroupName = GroupName;
    //string groupN = ViewBag.GroupName;
    //model.GroupID= groupID ;
    var data = LoadOwnPosts(groupID);
    List<Post> listOfOwnPosts = new List<Post>();
    for (int i = 0; i < data.Count; i++)
    {
        listOfOwnPosts.Add(new Post()
        {
            PostId = data[i].PostId,
            PostTitle = data[i].PostTitle,
            PostContent = data[i].PostContent,
            PostDate = data[i].PostDate,
            PostAuthor = data[i].PostAuthor,
            UserID = data[i].UserID,
            numOfLike = data[i].numOfLike
        });
        listOfOwnPosts.Reverse();
    }
```

Figure 48 - OwnGroup method in GroupsController class part one

```
var data1 = LoadJoinedGroupsMembers(groupID);
List<UserGroup> listOfGroupsMembers = new List<UserGroup>();
for (int i = 0; i < data1.Count; i++)
{
    listOfGroupsMembers.Add(new UserGroup
    {
        GroupName = data1[i].GroupName,
        UserName = data1[i].UserName,
    });
    listOfGroupsMembers.Reverse();
}
GroupAndPost cp = new GroupAndPost();
//cp.Comments = listOfComments;
cp.UserGroups = listOfGroupsMembers;
cp.Posts = listOfOwnPosts;
return View(cp);
```

Figure 49 - OwnGroup method in GroupsController class part two

The two for loops in the OwnGroup() method differ from the one used in other previously described methods. Using a foreach loop causes issues with the IEnumerable interface and therefore, a more simplistic approach was used to construct new Post and UserGroup models. Following that, at the end of the method the GroupAndPost object is instantiated through the name "cp" and the methods UserGroups and Posts are assigned the listOfGroupMembers data and listOfOwnPosts data that were fetched from the previously performed for loops in data and data1 variable. Finally, the data is returned to the View from the Model in the return statement.

```
@using BPRCoronaFighter.Models
@model GroupAndPost
```

Figure 50 - OwnGroup view

```
@foreach (var item in Model.UserGroups)
{
    <ul>
        @Html.DisplayFor(modelItem => item.UserName)
    </ul>
}
```

Figure 51 - OwnGroup view

In Figure 55, the using BPRCoronaFighter.Models means that the model GroupAndPost is being used in this view. In the foreach loop, the Model.UserGroups references the GroupAndPost model which contains multiple children models. One of these child models is UserGroups. A list is used to display all of the usernames that are in the database in the OwnGroup.cshtml page. These usernames are the users, including the currently logged in patient, which have joined the group.

4.4 Database SQL Implementation

While SQL will be the language of implementing the database, it is important to create the database, tables and related attributes according to the ER diagram before going forward.

To access the remote VIA SQL server, it is required to be authenticated on the server.

The development team has access to server 10.200.131.2, which was provided by VIA Database teacher.

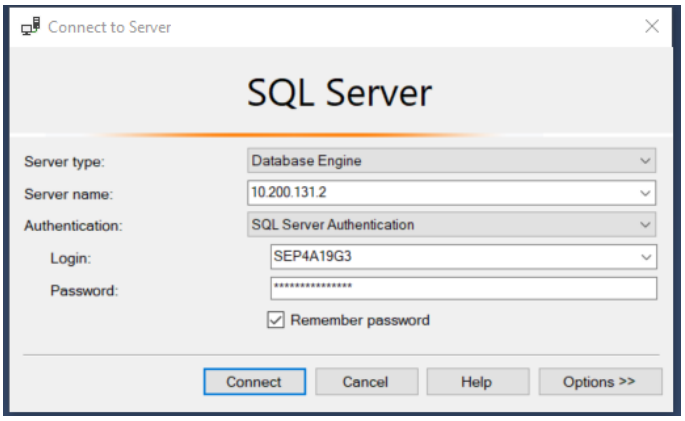


Figure 52 - SSMS Server Connection

After connecting to VIA SQL server, the database is created and the tables will be created by implementing the attached queries below:

4.4.1 Create User table

The table user has 8 attributes, one of them is the primary key which is the UserID that can represent the user in other tables as a foriegn key.

When a user will be created it is required to give their name, gender, email, password, date of birth and what role type they have in the system.

The UserID is defined as IDENTITY(1,1) which means that it will be auto incremented by 1 when a new user signs up.

```
1  CREATE TABLE [User](
2  UserID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
3  FirstName varchar(25) NOT NULL,
4  LastName varchar(25) NOT NULL,
5  Gender varchar(15) NOT NULL,
6  Email varchar(75) NOT NULL,
7  [Password] varchar(30) NOT NULL,
8  DOB date NOT NULL,
9  RoleTYPE varchar(10) NOT NULL
10 )
11
```

Figure 53 - SQL Create User Table

4.4.2 Create Group table

While the group can be created only by a user, therefore it is required to have a CreatorID which refers to the user who created the group.

It is necessary to give a name for the group and a city name to help other users know which city this group belongs to.

```
30 CREATE TABLE [Group](
31 GroupID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
32 CreatorID int FOREIGN KEY REFERENCES [User](UserID),
33 GroupName varchar(50) NOT NULL,
34 [Description] varchar(250),
35 City varchar(50) NOT NULL
36 )
37
```

Figure 54 - SQL Create Group Table

4.4.3 Group Users table

The GroupUsers table is needed to know group members, this table is important to avoid many-to-many relationship between User and Group table.

Here GroupID and UserID are foreign keys where they are primary keys at the same time, where they represent a group and a user but at the same time this representation is unique.

```
39 CREATE TABLE GroupUsers (
40 GroupID int FOREIGN KEY REFERENCES [Group](GroupID),
41 MemberID int FOREIGN KEY REFERENCES [User](UserID),
42 PRIMARY KEY (GroupID, MemberID)
43 )
44
```

Figure 55 - SQL Create GroupUsers Table

5. Test

After the Corona Fighter system is implemented, the system is tested to verify that the requirements are met and the source code is completed. The following section analyzes the testing process. The section will look at the various types of tests that were performed on the system which include unit, integration and acceptance tests. Each chapter will familiarize the reader with the purpose of the test used.

5.1 Unit testing

Unit tests serve to test small pieces of code such as methods which are part of a system. A unit test can perform an isolated test on a piece of code without the need of external dependencies such as files or a database which would normally be necessary when running system code. In C#, the **[TestClass]** attribute denotes that a class contains unit tests. The **[TestMethod]** attribute indicates that a method is a test method. A test consists of three parts:

- Arrange - Used to set up a test before execution the main code;
- Act - Execute main code of test;
- Assert - Check outcome of main code in test against a predefined outcome.

The unit test project used was the Unit Test Project that uses C# and is based on .NET Framework in Visual Studio.

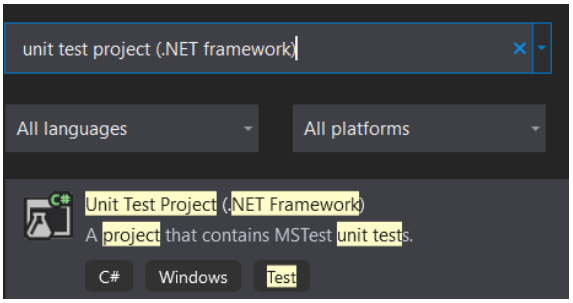


Figure 56 - Unit Test Project in Visual Studio

```
[TestMethod]
public void GroupIndexTest()
{
    // Arrange
    GroupsController controller = new GroupsController();
    var model = new Post();
    var model1 = new UserGroup();
    // Act
    ViewResult result = controller.Index(model, model1) as ViewResult;
    // Assert
    Assert.IsNotNull(result);
}
```

Figure 57 - GroupIndexTest method in TestGroupController class

The GroupIndexTest() starts with the Arrange part of the test by creating a new instance of the GroupsController. The model variable is assigned a new Post() instance and model1 variable is assigned a new UserGroup() instance. Following that, the Act part of the test has a ViewResult which calls the Index method that is part of the GroupsController with two parameters - model and model1. These two models are sections of the Groups page. In the Assert final phase of the test, the ViewResult is checked by performing Assert.IsNotNull(result). If the result View is Null, then, the test will fail, if not, the test will pass.

```
[TestMethod]
public void OwnGroupTest()
{
    // Arrange
    GroupsController controller = new GroupsController();
    var model = new Post();
    var model1 = new UserGroup()
    {
        GroupId = 15,
    };
    // Act
    ViewResult result = controller.OwnGroup(model, model1) as ViewResult;
    // Assert
    Assert.IsNotNull(result);
}
```

Figure 58 - OwnGroupTest method in TestGroupController class

The OwnGroupTest() method starts with the Arrange part. As in the GroupsIndexTest, the same variables are instantiated whereby, the only difference is that the UserGroup instance has its GroupID variable instantiated with a value of 15. The Act phase performs the same function as GroupsIndexTest and the assert function checks if the result variable that contains the predefined data is null or not.

```
[TestMethod]
public void CreatePostTest()
{
    // Arrange
    var model = new Post()
    {
        PostTitle = "test",
        PostContent = "test",
        PostDate = DateTime.Now.ToString(),
        PostAuthor = "test",
        UserID = "15",
        GroupID= "5"
    };
    var controller = new GroupsController();
    // Act
    var results = controller.CreatePost(model) as ViewResult;
    var results2 = CreatePosts(model.PostTitle, model.PostContent,
        DateTime.Now.ToString(), model.PostAuthor, model.UserID, model.GroupID);
    // Assert
    Assert.AreEqual(1, results2);
}
```

Figure 59 - CreatePostTest method in TestGroupController class

The CreatePostTest() method differs from the OwnGroupTest and GroupIndexTest() methods that were previously described. This method starts with creating a new instance of a Post model and providing sample parameter values for all parameters which are part of the Post class. Afterwards, a variable named controller is instantiated with a new instance of a GroupsController(). In the Act phase, the model data is fed into the CreatePost() method of the new controller instance and the outcome is saved into the results variable. The results2 variable saves the outcome of the CreatePosts() method. This method is fed real data from model1 such as “test” for model1.PostTitle. Finally, the Assert part tests if the outcome of results2 is 1 which means that the data has already been saved successfully. If results2 returns a different value, then, there is an error with saving the data.

5.2 Integration tests

Integration testing is to test modules/components at integration time to verify that they work as expected [18] *(What Is Integration Testing, 2021)*. Integration testing begins after all the system modules have been assembled and implemented. Every test case belongs to a requirement. They have prerequisites, steps, expected result and pass or not.

In the following section, three sets of test cases that serve as functionalities of the system will be presented:

3.Requirement:
The user should be able to manage their account details, which include their username and password into the system.
Prerequisites:
User is logged in.
Steps:
1.Click change password or change user name button.
2.Enter new information.
3.Click save button.
Expected Result:
User information has been updated.
Pass or Not: Passed

Figure 60 - Test case Requirement 3

- This test case shows that when a user needs to manage an account such as change password or change user name that the system must do.

9.Requirement:
The doctor should be able to manage their own corona-related lecture in the lectures section.
Prerequisites:
User is logged in.
Create lecture Steps:
1.Click the create lecture button.
2.Enter information.
3.Click create button.
Edit lecture Steps:
1.Click the edit lecture button.
2.Enter new information.
3.Click edit button.
Delete lecture Steps:
1.Click the delete lecture button.
2.Enter leature name.
3.Click delete button.
Expected Result:
For creating lecture:Data saved and displayed on the page.
For editing lecture:Data updated and displayed on the page.
For delete lecture:Date deleted.
Pass or Not: Passed

Figure 61 - Test case Requirement 9

- This test case shows that when a doctor needs to manage a lecture includes creating, editing and deleting that the system must do.

13.Requirement:
The administrator should be able to add video resources to the library to allow users to watch these resources.
Prerequisites:
Admin is logged in.
Steps:
1.Click the create video button.
2.Enter information.
3.Click create button.
Expected Result:
Data created and videos displayed on the page.
Pass or Not: Passed

Figure 62 - Test case Requirement 13

- This test case shows that when an admin needs to add video resources that the system must do.

5.3 Acceptance testing

Acceptance testing will be performed to ensure that all requirements have been successfully completed.

In this process, the user tests the functionality, describes the expected results and actual performance of the system, and assesses whether the requirements have been successfully implemented.

1. **The administrator should be able to authenticate the doctor's documentation in the administrator's verification panel of the system.**
   Expected outcome:
   Admin can check the doctor licence and decide to approve or not.
   Does outcome matches expected:YES

2. **The user should be able to set what kind of role they have - Patient, Doctor or Volunteer in the system.**
   Expected outcome:
   Users can set the user role when logging in.
   Does outcome matches expected:YES

3. **The user should be able to manage their account details, which include their username and password into the system.**
   Expected outcome:
   Users can change password and username.
   Does outcome matches expected:YES

4. **The user should be able to manage their post by posting and deleting it.**
   Expected outcome:
   Users can write posts or delete posts written by themselves.
   Does outcome matches expected:YES

Figure 63 - Test case Requirement 1-4

6. Results and Discussion

The report served the purpose of showcasing the various features of the Corona Fighter web application and its development. The finished system was tested using various test types and almost all requirements were fulfilled in the final product which satisfies the development team's expectations and ambitions for this project. The development team put all its experience gained from the previous semesters to their best into making this project. The system supports multiple types of roles. Therefore, a Doctor, Patient, Volunteer and Admin can use the system and these users can have a unique User Interface presented to them. The Analysis chapter contains a mockup model which will be of great help throughout the development of the system. The development team was able to visualize and agree on core functionalities of the system in the mockup model that were assembled in the final product.

The design of the system was made using the Client/Server model using three layers - the first containing the MVC which contains the front-end of the system. The second layer contains the Business Logic and the third layer contains the database - both of which are back-end. The tools used to aid the development of the system were also documented in the Choices of Technologies chapter. GDPR was also briefly discussed in the design chapter.

Testing was performed to ensure the end-product fulfills its intended purpose. Integration testing was applied immediately after the components of the system were assembled for the given Use Case. The process of integration testing involves following the steps for a given Use Case that is associated with the assembled part of the system. Acceptance testing was also performed to ensure that all requirements were implemented by accepting or rejecting a scenario for a function's behaviour based on its outcome compared to the expected scenario.

To conclude, the final product of the system contains all necessary features that were expected by the development team with the exception of having chat functionality between users and receiving notifications. Apart from that, the system is fully usable and ready to be put into practice.

7. Conclusions

The Corona Fighter application managed to successfully achieve all functional and non-functional requirements except for chatting and getting notification requirements. Each actor that is a user can fulfill their assigned use cases in the system. The Doctor, Patient and Volunteer are able to do their specific Use Cases and the ones that they inherit from the User actor in the system. The Admin of the system is able to fulfill his Use Cases - Manage Videos and Manage Doctor Application.

An Activity Diagram was shown in the Analysis to present the flow of steps necessary for Adding Posts and Managing Video Resources. The next diagram in the Analysis chapter is the System Sequence Diagram which shows a particular scenario and how the system responds to it. The Domain Model shows each entity such as User or Group and how they interact in the system.

The Design section of the report starts with presenting the Architecture Diagram and ER Diagram of the system. The Architecture Diagram served to visualize the Client/Server interaction in our system. The ER Diagram shows how the database was constructed and what kind of tables and attributes it contains. The next diagram in this chapter is the class diagram which has the aim to show the construction of the client-side of the system and showcase the different packages. Following that, a sequence diagram was made to showcase the Sign Up and Create Group functionalities in the system. The Design section also covers the use of MVC and Choice of Technologies. The Choice of Technologies chapter aims to describe the tools that were used to build the system, the organization of the system, the programming languages used, the SQL language and tools that were used as well as how the Entity Framework is used to communicate between the data layer and database. GDPR is also part of this chapter and describes what the GDPR act is and how it can potentially be used in the Corona Fighter web application.

The implementation has the purpose to show the SQL code implementation of the database and code snippets of the web application.

The testing section shows how unit tests, integration tests and acceptance tests were performed on the system.

Overall, the outcome of the system shows that most of the functionalities that were planned initially were completed. The system can handle multiple types of actors and each actor can only perform the functionalities that they have been given as can be seen in the Use Case Diagram which implies all system users have their own unique role in the system. The project idea was selected based on the current world situation. Working on such a system that gave the development team a huge gain of experience.

Finally, the project development team is satisfied with the project outcome. The system was successfully developed and almost all goals made by the development team were satisfied.

8. Project future

The project can be deployed at the current time, but there are still a few more improvements to attract users to the Corona Fighter Webapplication.

The project was done without any collaboration with a company. The development team does not find any problem with that and the project can still be deployed and help people.

The Covid-19 vaccines help with decreasing the number of Corona cases, but that does not mean the project has no future.

The Covid-19 variants are still coming out in different countries and in many cases the vaccines could not be efficient on these variants, therefore the project can still help people during the pandemic. And it is necessary to re-mention that the project can help in many cases of new pandemics, not only in Covid-19 pandemic and as such, it can be repurposed.

The project can be improved by adding more functionalities which could not be implemented due to the lack of time such as, chatting system , integrated voice and video calls system, coupons and gifts that can be given to doctors and volunteers to motivate them to join the system, and the development team is thinking to update the system and make a phone app version of it. Also an improvement can be done which is to change the system architecture to a Three-Tiers system, where the second layer will be a web-API, which will allow to have many options for the front-end and run the system on different operating systems and different devices.

The General Data Protection Regulation (GDPR) framework can be also a good choice to improve the system privacy and protect users data in the system.

When these improvements are done, the system will be ready to be deployed using Google Cloud or Azure, and the team wishes that the system will be a point of interest for the health ministry in Denmark, so it will give the system better authentication and better support.

9. Sources of information

1. Thelocal.dk. 2021. Danish health authorities seek volunteers to track coronavirus. [online]
Available at:
<https://www.thelocal.dk/20200319/danish-health-authorities-seek-volunteers-to-track-coronavirus> [Accessed 3 June 2021].

2. Tutorialspoint.com. 2021. UML - Activity Diagrams - Tutorialspoint. [online]
Available at: <https://www.tutorialspoint.com/uml/uml_activity_diagram.htm>
[Accessed 18 May 2021].

3. En.wikipedia.org. 2021. System sequence diagram - Wikipedia. [online]
Available at: <https://en.wikipedia.org/wiki/System_sequence_diagram>
[Accessed 3 June 2021].

4. Tutorialspoint.com. 2021. MVC Framework - Introduction - Tutorialspoint. [online] Available at:
<https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm> [Accessed 20 May 2021].

5. Software Engineering Stack Exchange. 2021. What really is the "business logic"?. [online] Available at:
<https://softwareengineering.stackexchange.com/questions/234251/what-really-is-the-business-logic> [Accessed 21 May 2021].

6. Baeldung. 2021. The DAO Pattern in Java | Baeldung. [online] Available at:
<https://www.baeldung.com/java-dao-pattern> [Accessed 21 May 2021].

7. GeeksforGeeks. 2021. MVC Design Pattern - GeeksforGeeks. [online] Available at: <https://www.geeksforgeeks.org/mvc-design-pattern/> [Accessed 21 May 2021].

8. C-sharpcorner.com. 2021. *Model Validation Using Data Annotations In ASP.NET MVC*. [online] Available at: <https://www.c-sharpcorner.com/article/model-validation-using-data-annotations-in-asp-net-mvc/> [Accessed 22 May 2021].

9. Docs.microsoft.com. 2021. Overview of ASP.NET Core MVC. [online] Available at: <https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-5.0> [Accessed 29 May 2021].

10. Slideshare.net. 2021. Choice of technology. [online] Available at: <https://www.slideshare.net/santhoshscaria/choice-of-technology> [Accessed 27 May 2021].

11. Microsoft. 2021. *ASP.NET MVC Pattern | .NET*. [online] Available at: <https://dotnet.microsoft.com/apps/aspnet/mvc> [Accessed 27 May 2021].

12. Speaker Deck. 2021. Tools to develop fast and with efficiency. [online] Available at: <https://speakerdeck.com/thedaviddias/tools-to-develop-fast-and-with-efficiency?slide=34> [Accessed 3 June 2021].

13. W3schools.com. 2021. C# Tutorial (C Sharp). [online] Available at: <https://www.w3schools.com/cs/> [Accessed 26 May 2021].

14. Developer.mozilla.org. 2021. *HTML basics - Learn web development | MDN*. [online] Available at: <https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics> [Accessed 27 May 2021].

15. Entityframeworktutorial.net. 2021. What is Entity Framework?. [online] Available at: <https://www.entityframeworktutorial.net/what-is-entityframework.aspx> [Accessed 2 June 2021].

16. GDPR.eu. 2021. *What is GDPR, the EU's new data protection law? - GDPR.eu*. [online]
Available at: <https://gdpr.eu/what-is-gdpr/> [Accessed 13 May 2021].

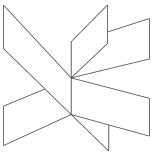
17. Cookiebot.com. 2021. GDPR and cookie consent | Compliant cookie use. [online]
Available at: <https://www.cookiebot.com/en/gdpr-cookies/> [Accessed 13 May 2021].

18. Software Testing Help. 2021. What Is Integration Testing. [online] Available at: <https://www.softwaretestinghelp.com/what-is-integration-testing> [Accessed 1 June 2021].

10. Appendices

- Appendix A: Analysis Diagrams
- Appendix B: Design Diagrams
- Appendix C: User Stories
- Appendix D: Integration Testing
- Appendix E: Acceptance Testing
- Appendix F: Project Description
- Appendix G: System Source Code
- Appendix H: User Guide

List of Figures



Corona Fighter

Process Project

Students:

Angel Iliyanov Petrov – 266489

Ziad Akram Bathish – 273442

Chunhui Liu – 273452

Supervisors:

Joseph Chukwudi Okika - JOOK@via.dk

21941 characters

Software Engineering

7th semester

4th of June 2021

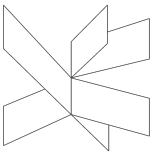
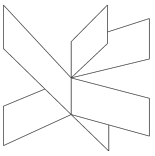


Table of content



1 Introduction

The project planning phase started during the sixth semester at VIA University College. The starting date of the bachelor project was on the 2nd of September 2020 with the start of BPR1. In the inception phase, we had time to come up with a project idea that appealed to us and our supervisor by writing a project description. Our initial goal was to develop a social media website. Later on, with the help of our supervisor, we decided that we should create something more practical and trending for current market needs that would aid the end-user and bring motivation in their daily life and wellbeing and as such, we decided to make a web application for allowing users to be more aware of the coronavirus.

BPR2 and the actual development of the system started on the 18th of February 2021 at which point we started work on system development and writing documentation. This process report has the aim of introducing its readers to the process that we undertook to complete our project. Firstly, we will talk about who the development team is and what their culture and personalities are. Following that, we will talk about the project initiation, the project description, the project execution and finally, we will touch upon supervision during the project.

Since the group consists of three members, it is very important that each group member is given responsibility for a portion of developing the system and can manage on their own unless they need help from the other group members. In contrast to previous semesters where we could have up to four group members, this semester gives us more responsibility to handle on our own.

Being the last semester for us at VIA University College, we are to apply any previously acquired knowledge in this project so that our final outcome meets the standard that we expect to accomplish.

In order to have a good workflow, the group must follow a development framework for documenting the workflow. In this project, we have utilized a Kanban board to keep track of our tasks.

2 Group Description

During BPR1, the group consisted of two members - Liu and Ziad. Shortly after, Angel joined the team and thus, the group was made up of three members. Since we had a good outcome for our sixth semester, we decided to stick together for our final bachelor project. These three group members are the development team behind the Corona Fighter web application. Each group member is of different nationality which produced a very vibrant and diverse group and gave us different perspectives to work with for applying our problem-solving skills to the project.

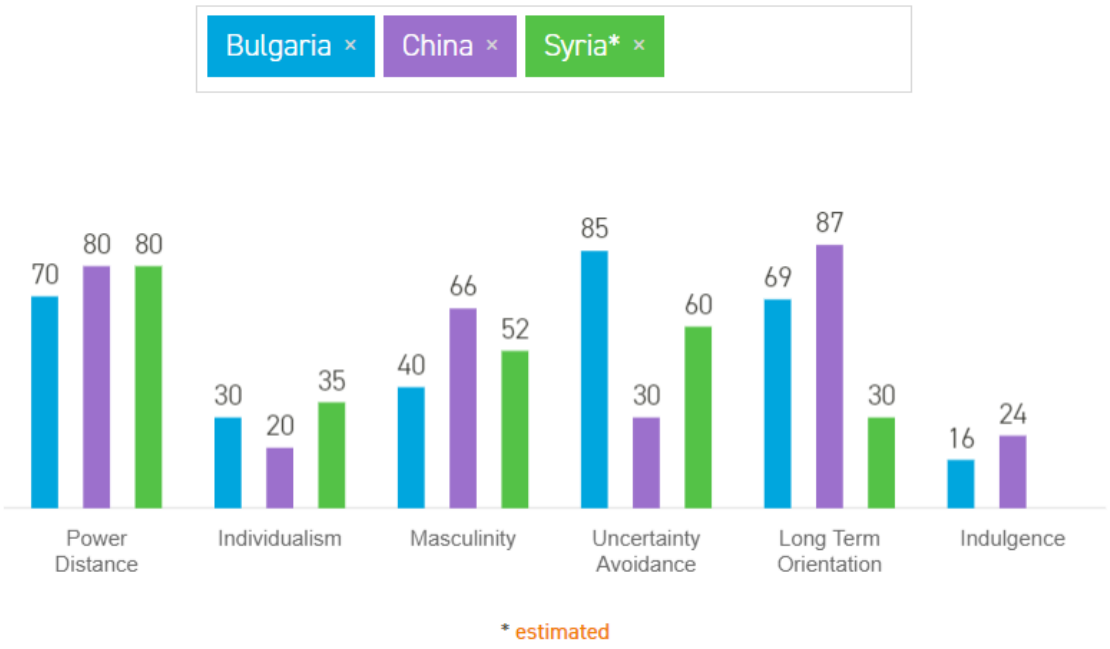
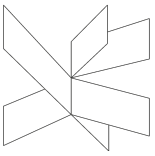


Figure 1 - Hofstede Insights six culture dimensions

As we can see in the figure above, according to the Hofstede Insights analysis [1] *(Compare countries - Hofstede Insights, 2021)*, the group members share similarities in some dimensions whilst they differ in others - Angel being from Bulgaria, Liu from China and Ziad from Syria. In contrast to Danish culture where there is a strong emphasis on individualism, the mentality of our group can be considered as "eastern" [2] *(Kotelnikov, 2021)* and as such all group members aim towards working in a collectivist manner and benefiting each other. This was seen in our group in terms of regular meetings at a group member's home, pair-programming with two or all of the group members, collective problem-solving on a specific problem and even mutual cooperation in other courses that we had together during the last semester.



East	INDIVIDUALISM / COLLECTIVISM	West
A human being is an integral part of the universe and the society. People are fundamentally connected. Duty towards all others is a very important matter. **COLLECTIVISM is stronger.**		A human being has an individualistic nature and is an independent part of the universe and the society. **INDIVIDUALISM is stronger.**

Figure 2 - Collectivism and individualism [2] *(Kotelnikov, 2021)*

2.1 Team roles in group

Each group member undertook the Belbin Team Role Inventory Test [3] *(Belbin Team Role Inventory Test, 2021)* which gave each of our group's members an overview of their strengths.

Chunhui Liu:

As a team member, my role tends to be a coordinator, plant and complete finisher according to the Balbing roles.

This means that I am a creative character, I like to provide ideas for the team, and I have the ability to rally the team together. I am good at communicating with team members and understanding each person's strengths and weaknesses, so as to put forward specific suggestions for each person's work. Which will help us achieve our goals more efficiently.

The test results showed that our team lacked the ability to deal with the outside world and have a strong driving force. As a result, it may be difficult to obtain external information and communicate effectively with supervisors. The lack of Shaper may lead to a decrease in the members' work enthusiasm, and they will be easily discouraged when they encounter setbacks. We will try our best to stay motivated and always encourage each other while doing our job to ensure that this is an active and competitive team.

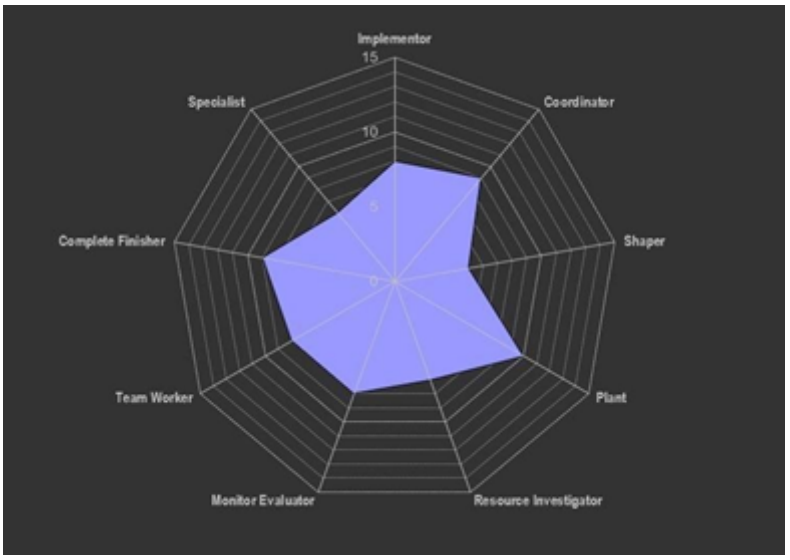


Figure 3 - Personal Role Inventory chart for group member Chunhui Liu

Ziad Akram Bathish:

According to the Belbin test, I attend to be a team worker and a specialist.

This means that I like to work and go in-depth with areas that are important to complete our project.

Where my strengths are is to cooperate and listen carefully to team members without any conflicts, and usually I provide good ideas that help to get better results. On the other hand, there are few weaknesses where I often avoid leading the team because I don't feel that I am good with making critical right decisions.

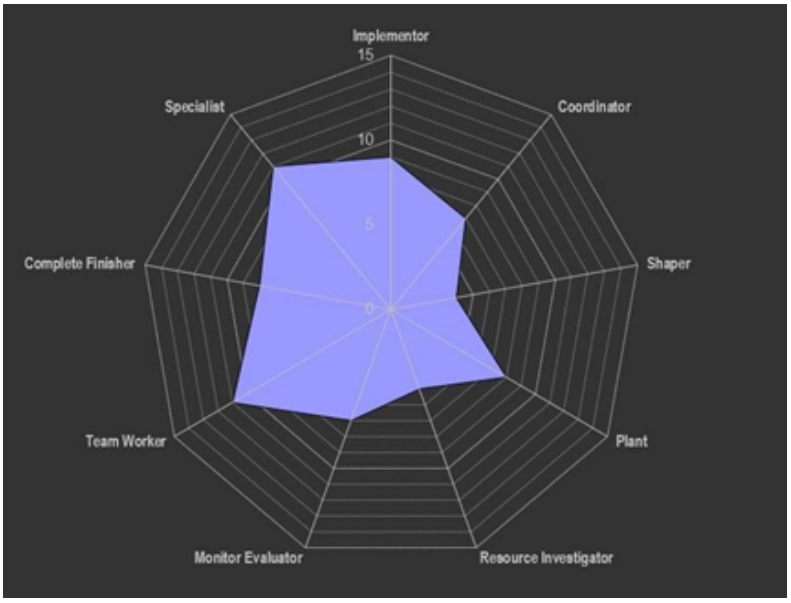


Figure 4 - Personal Role Inventory chart for group member Ziad Akram Bathish

Angel Iliyanov Petrov:

According to the Belbin Team roles test which I undertook, my personal character specializes in being a Coordinator, Plant and Specialist. This would imply that I excel in being mature, confident, and able to identify talent within my team as well as putting a clear objective for me and my team to follow towards reaching our success goal. Thanks to my Plant personality trait, I am able to offer high creativity and good problem-solving skills to my team whilst also contributing in-depth knowledge of a key area to the team thanks to my Specialist personality trait.

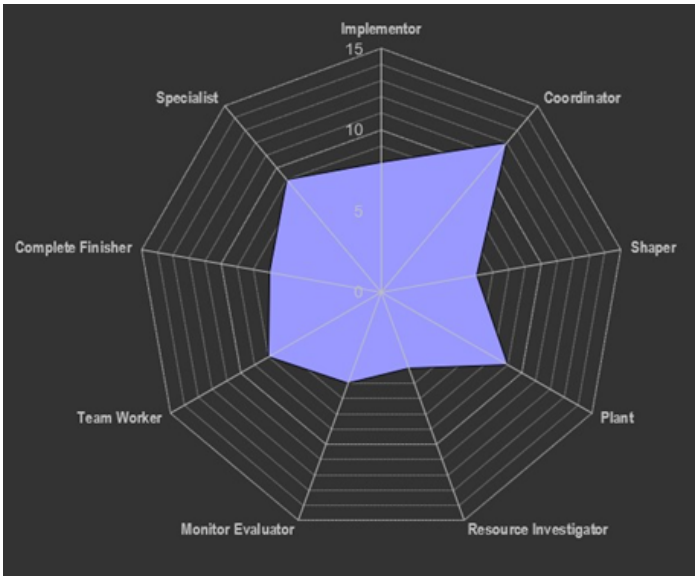
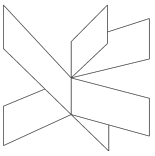


Figure 5 - Personal Role Inventory chart for group member Angel Iliyanov Petrov

2.2 Group member introduction

Angel Iliyanov Petrov

I am a 23-year-old student who is in his seventh semester at VIA University College. My experience with web development came from the fact that my internship was about deep-web development – extracting and filtering data from web sources. During my internship which was held in the United Kingdom, I had the chance to work in a global company with both front-end and back-end to accomplish my given tasks and had the opportunity to work with very talented, experienced, and bright people in my international development team. As such, my gained experience led me to pursue creating a project that was in the scope of web development, however, my interests in the software engineering sphere are not only limited to web development. During the development of the Corona Fighter application, I worked with UI for the different web pages and back-end logic implementation such as writing the logic in the various controllers of the application. I focused mainly on the implementation of administrator functionalities by creating a special admin panel which contained various administrator-related functionalities such as adding videos to the user video library. My decisive thinking led me to achieving great results in the project and fulfilling my expectations to a satisfactory acceptance.



Chunhui Liu

I am a 23 years old student from China who is interested in app design and development.

During the bachelor project,I worked mostly on webpage design and implementation.I did not have relevant work experience before, so I did a lot of preparation work before the project started, including the conception and preliminary planning of the project framework.Due to my interest in the web development and accumulated programming ability, I soon had a preliminary idea of the project and started to implement it.

In this Corona Fighter project, my main work is to complete the design and implementation of group and post functions, including UI design in the front section, logic development in the back end and realization of database-related functions.

In this project, I learned a lot about the communication methods of team members and improved my ability to be a good team member.

Ziad Akram Bathish

I came from Syria to Denmark 5 years ago, where I left my country because of the war and I came here to have a better future and live in a peaceful environment with my family. I am 26 years old now. I lost almost 4 years because of the war, but I came here and tried my best to continue with studying. I have always been interested in softwares and technologies but did not have any experience before starting at VIA.

Web development is one of the most interesting topics for me, where we started learning the basics in the third semester and in my internship I got a better experience with it where I worked on developing a survey system using the ASP.net core. Therefore it was interesting to work on the Corona Fighter Webapp using ASP .Net MVC, where it was a bit difficult in the beginning to prepare the system environment and analyse how the functions will be handled to send and receive data from the database and show it to the user.

3 Project Initiation

After forming the group, the project is initiated by brainstorming about what problems that people are facing and how the team can create a project to solve a common problem. The team came out with different ideas but in the end all members agreed on finding a solution to help corona patients where we make it easier for them to be in touch with doctors and volunteers.

After getting the acceptance from the teacher about the idea, the team started making research about the Covide-19 pandemic and looking for statistics about the countries that got affected by this pandemic.

While the group is formed and the idea is accepted by the supervisor and all members, the team is ready and motivated to start working and achieve best results in the end.

The Project Initiation was counted as part of our first Sprint where the tasks of forming the group and finding the idea were completed at the end of the sprint.

Sprint 1:

08/Feb/2021 - 14/Feb/2021

Task	Estimate	Status	Who
Group formation	3	Done	Liu, Ziad, Angel
Project Idea	3	Done	Ziad, Angel, Liu
Project Description	5	Assigned	Ziad, Angel, Liu

Sprint Description:
The working on the project started here where the group was formed and members started brainstorming to find a problem and what project can be made to solve the problem.
After finding the project idea the group got acceptance from the supervisor on the idea and the work started on writing the project description.

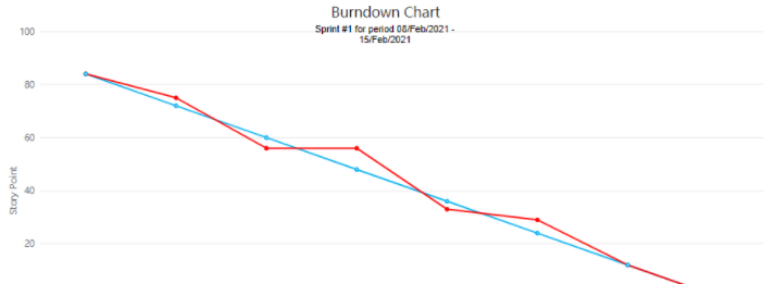


Figure 6 - First Sprint Backlog / Inception 2

4 Project Description

Before starting with project execution, it was important to work on project description and the most important part here was to define our problem statement where it was needed for the team to know exactly what the problem is and how to solve it.

The team started doing research to support the project idea. All members were able to understand the problem domain and can explain the background, define the purpose and make a plan to what risks, methodology and time schedule will be used.

The choice of methodology was good enough for our three members team while we used Kanban together with Unified Process that helped us to come with better outcomes and having better time management.

The team chose to use Sprint from Scrum and use it in the project to keep track of the working process, where the team started documenting all meetings and specifying what has been accomplished and what must be done later.

The project description was part of the first Sprint where all team members worked together to write a good background description with all necessary information sources, also tried our best to explain the problem statement to be easily understandable, and worked on all different chapters with almost daily communication to have the best results in the second Sprint where the team could finish the project description.

Sprint 2:

15/Feb/2021 - 21/Feb/2021

Task	Estimate	Status	Who
Project Description	5	Done	Angel, Liu, Ziad
Requirements	4	Assigned	Angel, Liu, Ziad
Analysis Planning	3	Assigned	Angel, Liu, Ziad

Sprint Description:
The sprint here started by continuing working on the project description where the team worked together to complete it.
At the end of the sprint the project Description was done and new tasks assigned for the team where the team now needs to work on requirements and Analysis planning to be able to move to the next sprint which will be about the Elaboration phase.

Figure 7 - Second Sprint Backlog / Inception 2

5 Project Execution

Project Execution started by gathering data where the team made a survey that has been sent to many people to check who had and who is having the Coronavirus currently, what they think about the web application and their suggestions about having a better system that can really help patients.

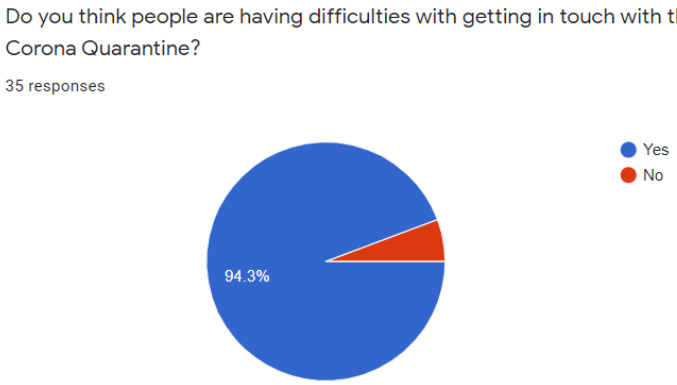


Figure 8 - Results from conducted Corona Fighter Web App survey

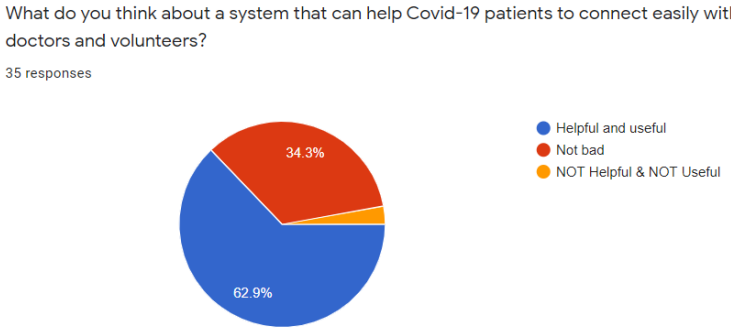


Figure 9 - Results from conducted Corona Fighter Web App survey

In the survey, doctors and volunteers were mentioned and there are a few questions about what they think and what can motivate them to join the system.

The attached image shows answers by doctors about what can motivate them:

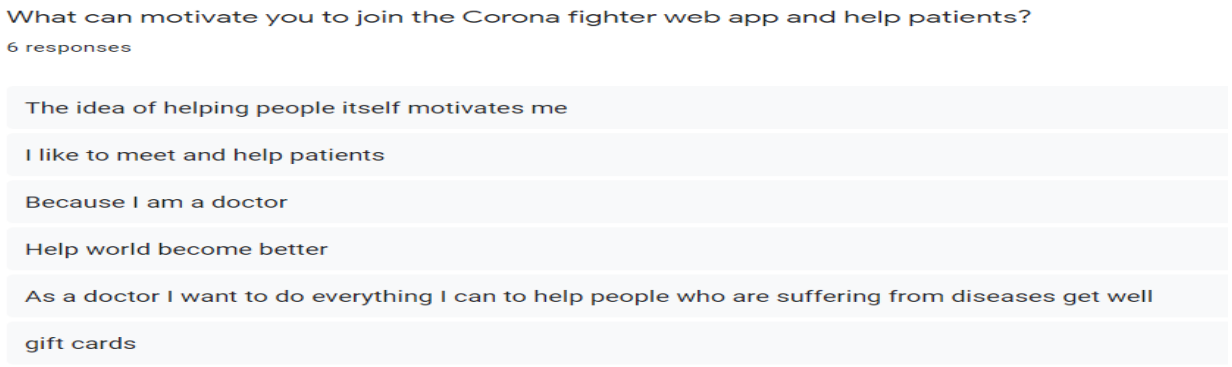


Figure 10 - Results from conducted Corona Fighter Web App survey

The processes here were assigned in the first and the second sprints where the data collection was done in the second sprint.

Depending on the collected data and project description, the team started writing the requirements based on the problem domain in the project description and following SMART principles to have Specific, Measured, Attainable, Relevant and time-based requirements.

When the functional and Non-functional requirements were ready the team could move to the Elaboration phase where we started with the analysis and moved forward to the design.

The process here was done in five sprints, at the end, the team started with the longest phase which was the Construction.

We started with implementing what we have in the design according to the Product Backlog.

Kanban backlog

Task NO.	Priority	Description
1	High	As an admin, I want to be able to authenticate doctors' documentation, so that doctors are verified in the system.
2	High	As a user, I want to be able to set what kind of user I am (Patient, Doctor, Volunteer), while signing up so that I am assigned a specific role in the system.
3	High	As a user, I want to manage my account details which include password and username, so that I am able to update my profile data.
4	High	As a user, I want to manage my posts by posting and deleting them, so that I am able to share or remove them.
5	High	As a user, I want to be able to write comments on posts, so that I am able to interact with others.
6	High	As a patient, I want to create a private group and manage my group/s, so that I have more interaction with my doctors and volunteers who can provide help.
7	High	As a user, I want to join a group by searching for the group, so that I can provide help and communicate with others.
8	High	As a user, I want to be able to leave a group by withdrawing from it, so that I am not associated with other users in the group.
9	High	As a doctor, I want to be able to manage my corona-related lecture by posting, editing and deleting it, so that I can let other users be aware of my lecture updates.
		As a volunteer, I want to be able to add and delete my offer help post, so that I

Figure 11 - Product Backlog

The Agile Coach was not the same person all the time, the team decided to switch between the roles every two sprints so all of us acquire experience and understand what is going on.

The tasks were moved from the backlog to the “To DO” list by the Agile Coach, and the other two members were picking up one task at the time.

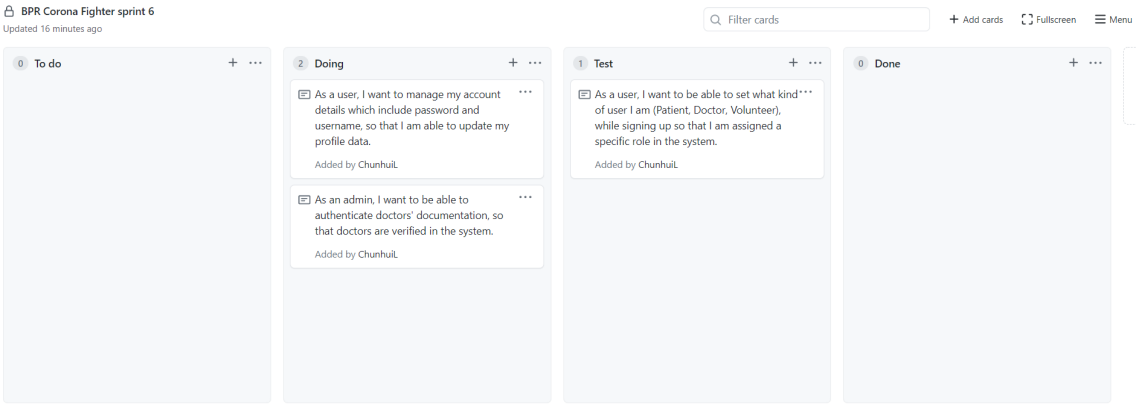


Figure 12 - Kanban board tasks in GitHub

The hard part here was the iterations where many times the team needed to go one or two steps back to the analysis and design to make small modifications on diagrams.

The Workflow was done here using Kanban board on Github where it was a really nice and flexible approach that helped us work together during the construction phase.
At the end of the project the team was almost done with all assigned tasks where most of them were in the “Done” List and “To Do” List was empty.

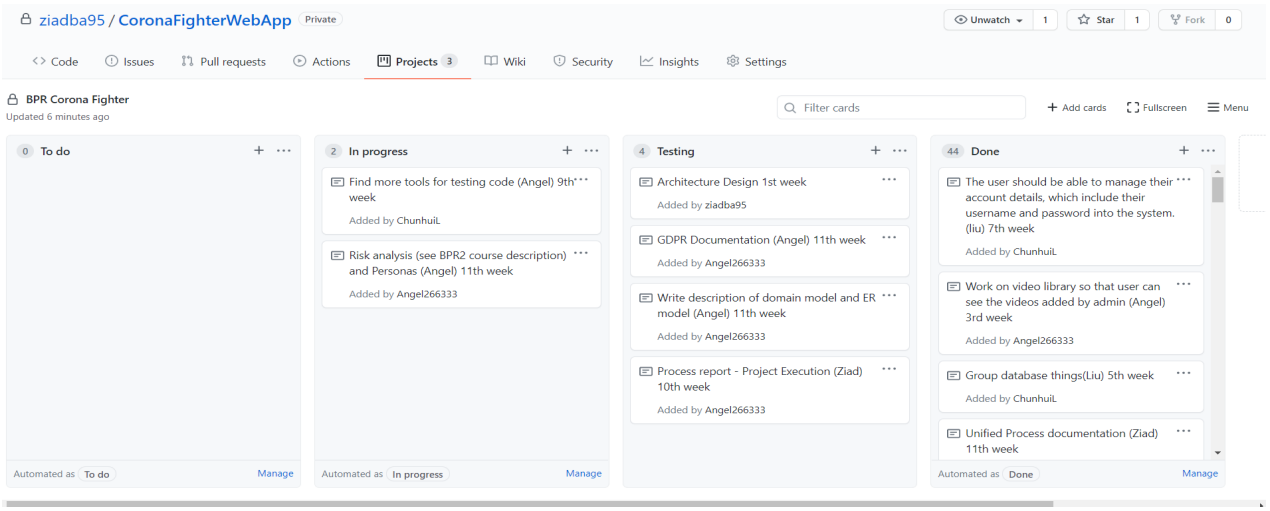
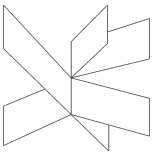


Figure 13 - Kanban board tasks in GitHub

The Transition phase of the project was about documentation, where the team started to give their full focus on it and tried to make a good report while it was also an iterative process where sometimes we discovered that there was something not so obvious as before so had to go back back and make the needed changes and go forward again. That happens in the Use Case Diagram where the team found out that the diagram still lacks view details so we tried to make it more specific and more understandable.
The process here was really good and the choice of the methods and technology helped us to have better control of the working process.
The Kanban approach together with the Unified process could be applied successfully while it was the first time we used these two approaches together, also the use of Sprints

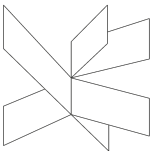


was a bit difficult but it came up with better results in having a detailed and trackable working process.

6 Personal Reflections

- **Angel**

  My personal opinion about how this project and semester is good with some exceptions regarding the coronavirus pandemic. We had classes and did group work online on Zoom most of the time due to the fact that VIA was not open because of the coronavirus for the most part of the semester which did have a marginal impact on our performance. The project concept and execution however seem fine to me in regards to what the market might be looking for currently and in the future. My group members were a pleasure to work with and I had absolutely no issues in terms of communication or any sort of cultural barrier that might have stopped me in doing so.

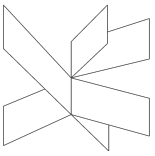


- **Liu**

  A year ago, when I first learned about this project, I was very nervous not only because of the difficulty of the project, but also because I did not know whether my team members had the confidence to complete the project together.
  But by now I can say that those fears were unfounded,my team members are very active and self-motivated. Even under the influence of the epidemic, it is difficult for us to organize offline meetings, but everyone still works hard and makes full use of Zoom for video conferencing.
  On the whole, the cooperation was pleasant. Although we came from different countries and different cultures, we still respected each other and completed the project efficiently.

- **Ziad:**

  For me, I knew Liu and worked together since the second semester, and Angel joined us last semester and worked together on Sep6. I was really comfortable and happy to work with them, where they were listening, helping and giving really useful ideas and solutions when we had problems.
  In this semester I wished that we had more time to work on the project and give it all our focus, but that was difficult while we have other courses and assignments that must be finished at the same time. But in general, it went well when all members did their best to work on it and help each other.

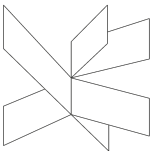


7 Supervision

The bachelor project took place during the 6th and 7th semesters. During the 6th semester, there were no scheduled meetings with our supervisor since the workload was not as intense as it would have been in the final semester. The only feedback that was given to us during the 6th semester was on the project description by the teacher of the BPR1 course.

During the 7th semester, meetings were held occasionally with the supervisor to discuss potential improvements and guide the project development team into the right direction. Meetings took place on Zoom and lasted around 30 minutes each. The supervisor provided help to us by overseeing our progress and giving feedback on the report and diagrams. The supervisor left useful comments so that we can follow his guidance to improve the progress and quality of the project and its documentation.

In contrast to previous semester projects where we had to follow a set of requirements given to us by the teachers this final bachelor project gave us total freedom to expand our horizons on what we wanted to do for the project which meant that we had to be more individualistic in our choices regarding the system which is a great way to prepare us for the real world where we are the ones in charge of a project. The supervisor was there only to give us guidelines on how we could better improve our project but not to tell us how to do the project which is a very good way of learning by us during this last semester. If we had any questions, apart from the Zoom meetings, we could always get in touch with our supervisor via email and he would reply to us momentarily.



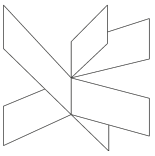
8 Conclusions

Overall, the bachelor project required a lot of team effort to complete in a relatively short amount of time this semester whilst keeping in mind that each group member had to focus on other courses as well. Even though we could not meet most of the time due to the pandemic and most of our meetings were online, we were able to finish the project to a very satisfactory level and deliver an almost ideal system.

Using Kanban helped with task distribution amongst group members. It also aided in managing our time together with Sprints for completing each of our assigned tasks. Kanban was new to all of us and we decided to try it this semester to gain more experience with different methods of working on a project.

Teamwork and communication were an invaluable part of our group which significantly improved our efficiency. Help between group members was also not uncommon to see. Whenever someone needed help another group member would try to solve the problem that they are facing together. Positivity in the team led us to having a fun and delightful way of working on the project.

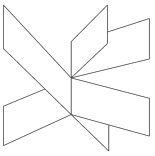
In conclusion, working on this project was an exciting way to learn many new skills and technologies and thus made us better talented at developing something that will train us for our future careers in the software industry.



9. Appendices

- Appendix A: Kanban Backlog
- Appendix B: Group Description and Contract
- Appendix C: Sprint Backlogs
- Appendix D: Survey Statistics

List of Figures



10. Sources of Information

1. Hofstede Insights. 2021. *Compare countries - Hofstede Insights*. [online] Available at: <https://www.hofstede-insights.com/product/compare-countries/> [Accessed 3 May 2021].

2. Kotelnikov, V., 2021. *EAST vs. WEST Differences: Eastern and Western values, attitudes towards life, culture insights, comparisons, cultural intelligence, Vadim Kotelnikov*. [online] 1000ventures.com. Available at: <http://www.1000ventures.com/business_guide/crosscuttings/cultures_east-west-phylosophy.html> [Accessed 3 May 2021].

3. Academia.edu. 2021. Belbin Team Role Inventory Test. [online] Available at: <https://www.academia.edu/29292303/belbin_test> [Accessed 1 June 2021].