University of Westminster

School of Computer Science and Engineering

MODULE: 5BUIS017W, "BIS Design and Architecture" (2022/2023)

Group Coursework

Justina Aikeremiokha -w18352261

Ali Jumah Ali Almajdoub - w1843542

Sohag Noman - w1856656

Jubair Ahmed - w1826461

Summary

Introduction

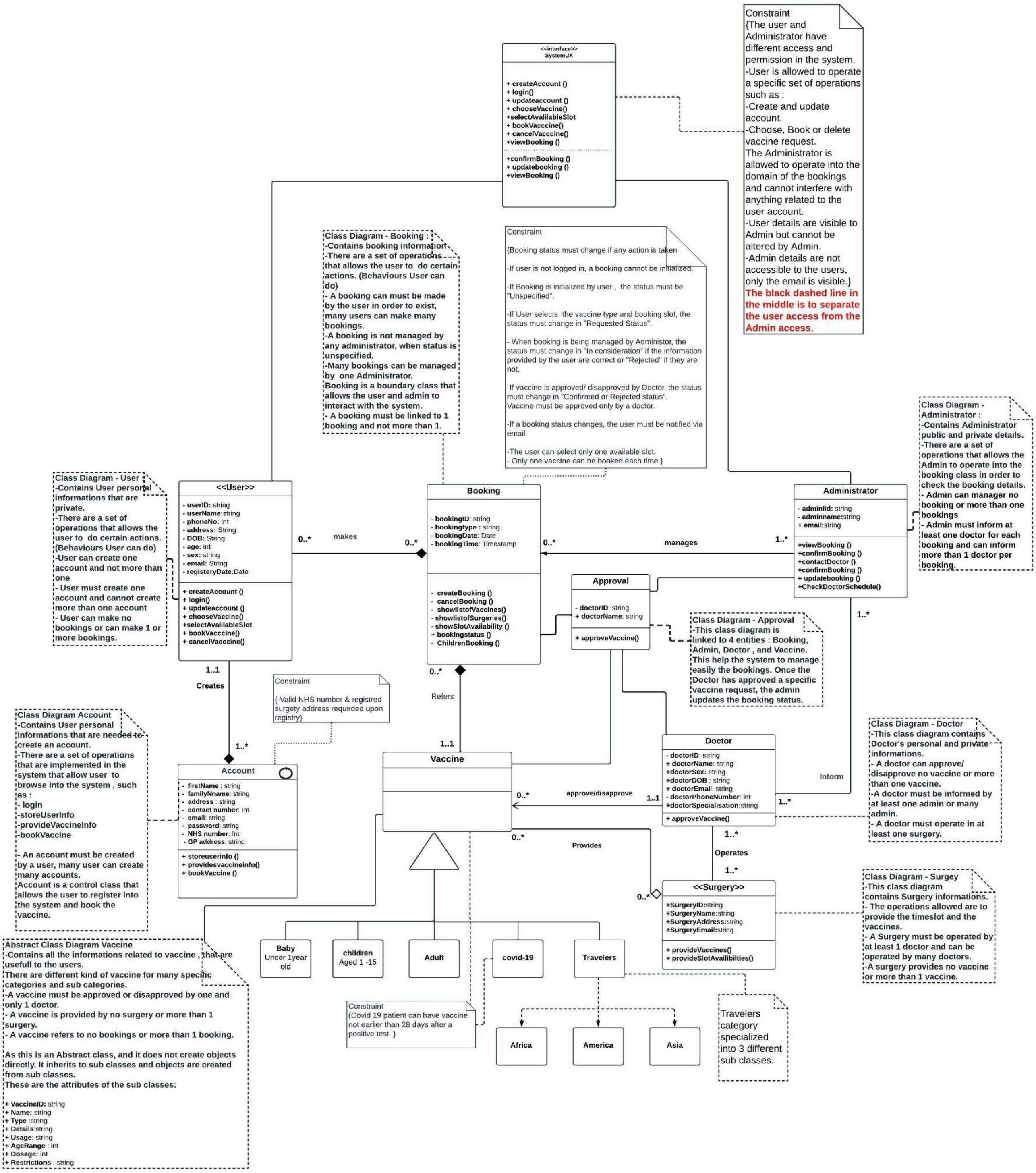
This coursework will demonstrate our understanding of the major concepts of systems architecture and design techniques required for the successful development of an online vaccine booking system. We will also demonstrate our understanding of the importance of implementing these techniques in the design phase and how it assists the design to accomplish its primary purpose in order to ensure that the system is comprehensive and adequate for its intended usage.

Our group was able to recognize and apply the basic principles of interface design and engage confidently in academic and professional discussions, to reflect on the team-work experience, and to justify the business requirement needed providing solutions and suggestions.

Class diagram

Task 1

Produce a detailed design class diagram (including all the boundary, control and entity classes as well as any other notation(s) required) with as much semantics necessary for the implementation. Hint: there are restrictions for specific vaccines.



Constraint
{The user and Administrator have different access and permission in the system.
-User is allowed to operate a specific set of operations such as :
-Create and update account.
-Choose, Book or delete vaccine request.
The Administrator is allowed to operate into the domain of the bookings and cannot interfere with anything related to the user account.
-User details are visible to Admin but cannot be altered by Admin.
-Admin details are not accessible to the users, only the email is visible.}
The black dashed line in the middle is to separate the user access from the Admin access.

Class Diagram - Booking :
-Contains booking information
-There are a set of operations that allows the user to do certain actions. (Behaviours User can do)
- A booking can must be made by the user in order to exist, many users can make many bookings.
-A booking is not managed by any administrator, when status is unspecified.
-Many bookings can be managed by one Administrator.
Booking is a boundary class that allows the user and admin to interact with the system.
- A booking must be linked to 1 booking and not more than 1.

Constraint

{Booking status must change if any action is taken

-If user is not logged in, a booking cannot be initialized.

-If Booking is initialized by user , the status must be "Unspecified".

-If User selects the vaccine type and booking slot, the status must change in "Requested Status".

- When booking is being managed by Administor, the status must change in "In consideration" if the information provided by the user are correct or "Rejected" if they are not.

-If vaccine is approved/ disapproved by Doctor, the status must change in "Confirmed or Rejected status".
Vaccine must be approved only by a doctor.

-If a booking status changes, the user must be notified via email.

-The user can select only one available slot.
- Only one vaccine can be booked each time.}

Class Diagram - Administrator :
-Contains Administrator public and private details.
-There are a set of operations that allows the Admin to operate into the booking class in order to check the booking details.
- Admin can manager no booking or more than one bookings
- Admin must inform at least one doctor for each booking and can inform more than 1 doctor per booking.

Class Diagram - User
-Contains User personal informations that are private.
-There are a set of operations that allows the user to do certain actions. (Behaviours User can do)
-User can create one account and not more than one
- User must create one account and cannot create more than one account
- User can make no bookings or can make 1 or more bookings.

Class Diagram - Approval
-This class diagram is linked to 4 entities : Booking, Admin, Doctor , and Vaccine. This help the system to manage easily the bookings. Once the Doctor has approved a specific vaccine request, the admin updates the booking status.

Constraint

{-Valid NHS number & registred surgety address requirded upon registry}

Class Diagram Account
-Contains User personal informations that are needed to create an account.
-There are a set of operations that are implemented in the system that allow user to browse into the system , such as :
- login
-storeUserInfo
-provideVaccineInfo
-bookVaccine

- An account must be created by a user, many user can create many accounts.
Account is a control class that allows the user to register into the system and book the vaccine.

Class Diagram - Doctor
-This class diagram contains Doctor's personal and private informations.
- A doctor can approve/ disapprove no vaccine or more than one vaccine.
-A doctor must be informed by at least one admin or many admin.
- A doctor must operate in at least one surgery.

Class Diagram - Surgey
-This class diagram contains Surgery informations.
- The operations allowed are to provide the timeslot and the vaccines.
- A Surgery must be operated by at least 1 doctor and can be operated by many doctors.
-A surgery provides no vaccine or more than 1 vaccine.

Abstract Class Diagram Vaccine
-Contains all the informations related to vaccine , that are usefull to the users.
There are different kind of vaccine for many specific categories and sub categories.
-A vaccine must be approved or disapproved by one and only 1 doctor.
- A vaccine is provided by no surgery or more than 1 surgery.
- A vaccine refers to no bookings or more than 1 booking.

As this is an Abstract class, and it does not create objects directly. It inherits to sub classes and objects are created from sub classes.
These are the attributes of the sub classes:

+ **VaccineID:** string
+ **Name:** string
+ **Type** :string
+ **Details**:string
+ **Usage**: string
+ **AgeRange** : int
+ **Dosage:** int
+ **Restrictions** : string

Constraint
{Covid 19 patient can have vaccine not earlier than 28 days after a positive test. }

Travelers category specialized into 3 different sub classes.

The following class diagram shows all the classes required for the online vaccine booking system creation. After the analysis, we were able to specify the role that each class plays in the system, and the responsibilities of each class in relation to the system. We followed the BCE pattern to make sure that the responsibilities of the classes are also reflected in the relations and interactions between the different categories of classes in order to ensure the robustness of the design. As a class diagram consists of Classifiers, Features and relationships we have tried to meet all the requirements in order to create a correct class diagram:

-We have booking as an active class which changes its state based on the action of the other related classes.

-We have concrete classes

-We have 'Vaccine' as an abstract class

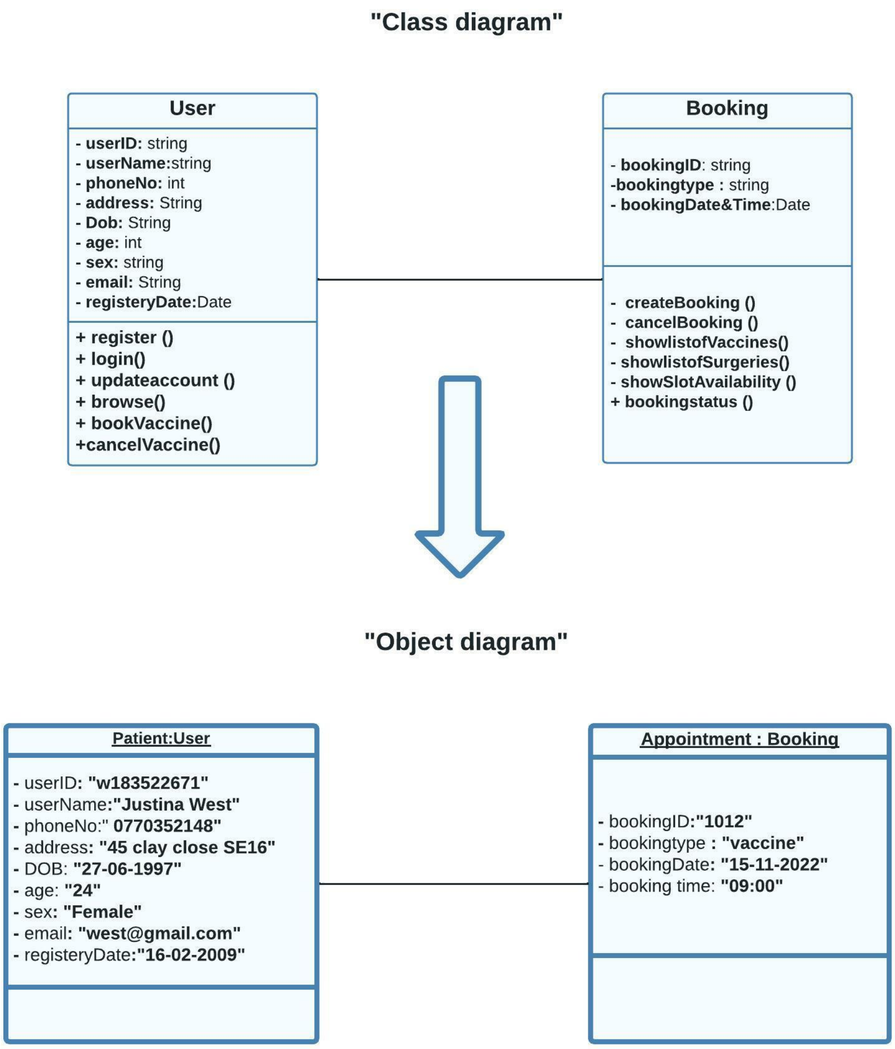
-We have SystemUX as Interface

We included different tags linked to the entities in order to add clarity and detailed information in the class diagram.

Object diagram

Task 2

Identify an object of the class “User” and create its “object diagram”. Do the same for an object of the class “Booking”.



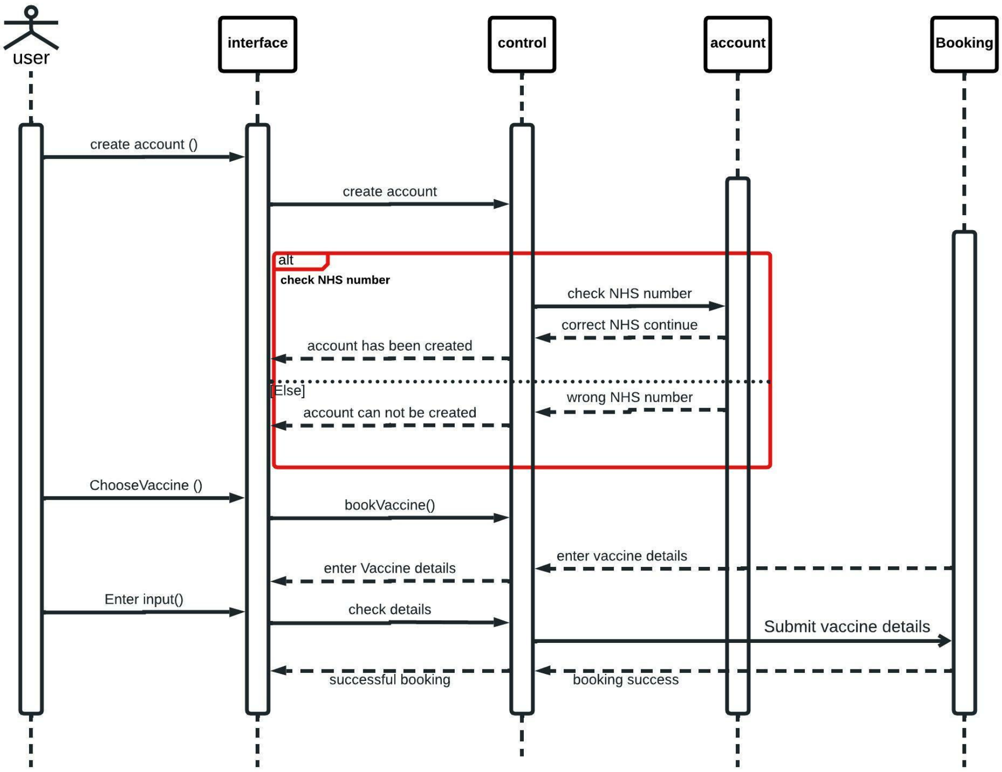
The diagram above illustrates the distinction between a "User" and "Booking" class diagram and object diagram. Structure diagrams, such as Object and Class diagrams, serve similar functions. In contrast, a class diagram displays an abstract model made up of classes and the connections between them. On the other hand, an object diagram captures the static perspective of interactions between objects at a specific moment and illustrates and portrays an instance at a specific time or moment that is formed throughout run time.

We emphasised the objects of the classes "User" and "Booking," which we recognised as "Patient" and "Appointment," respectively, and added all relevant characteristics and values as well.

Sequence diagram

Task 3

Draw a sequence diagram that shows how a user creates an account to book a vaccine. Hint: the user must provide a valid NHS number, otherwise an account cannot be created.



UML Sequence Diagrams are interaction diagrams that describe an event across all objects involved in a single interaction. It captures the responses of all objects involved in a single case.

The sequence diagram in the above diagram describes the sequence of events that occur when a user creates an account in order to book a vaccine. The interface sends the request to control after the user creates an account. The control asks for an NHS number. Following the user's entry of an NHS number. The NHS number is checked by the control. If the NHS number is correct, the control will allow you to proceed with the account creation. The control will not allow you to create an account if the NHS number is incorrect.

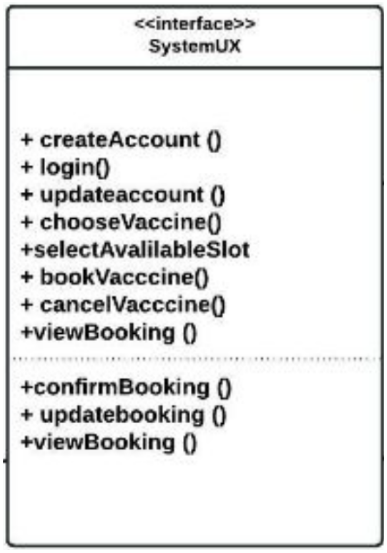
You can reserve the vaccine once you have created an account. after making a booking request. Through the interface, the control requests that the user submit certain information. The control will then use the account to issue requests for booking. User information will be reviewed following user information verification. The vaccine will be successfully booked if the user's data are accurate

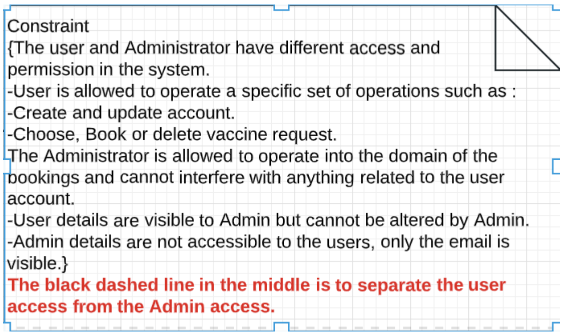
Interfaces

Task 4

Apparently, users and administrators have different needs and they must have access to different parts of the system. As a result, different design decisions are required. As far as design decisions are concerned, abstract interfaces become very useful. Explain the use of interfaces by classes to simplify implementation. Referring to your class diagram produced for task 1 above, provide and discuss ONE example where you would use interfaces.

In UML modelling, interfaces are model elements that define sets of operations that other model elements, such as classes, or components must implement.
We can use interfaces in class diagrams to specify a contract between the interface and the classifier that realises the interface. Each interface specifies a well-defined set of operations that have public visibility.
By implementing an interface, classes are guaranteed to support a required behaviour, which allows the system to treat non-related elements in the same way – that is, through the common interface.





Constraint
{The user and Administrator have different access and permission in the system.
-User is allowed to operate a specific set of operations such as :
-Create and update account.
-Choose, Book or delete vaccine request.
The Administrator is allowed to operate into the domain of the bookings and cannot interfere with anything related to the user account.
-User details are visible to Admin but cannot be altered by Admin.
-Admin details are not accessible to the users, only the email is visible.}
The black dashed line in the middle is to separate the user access from the Admin access.

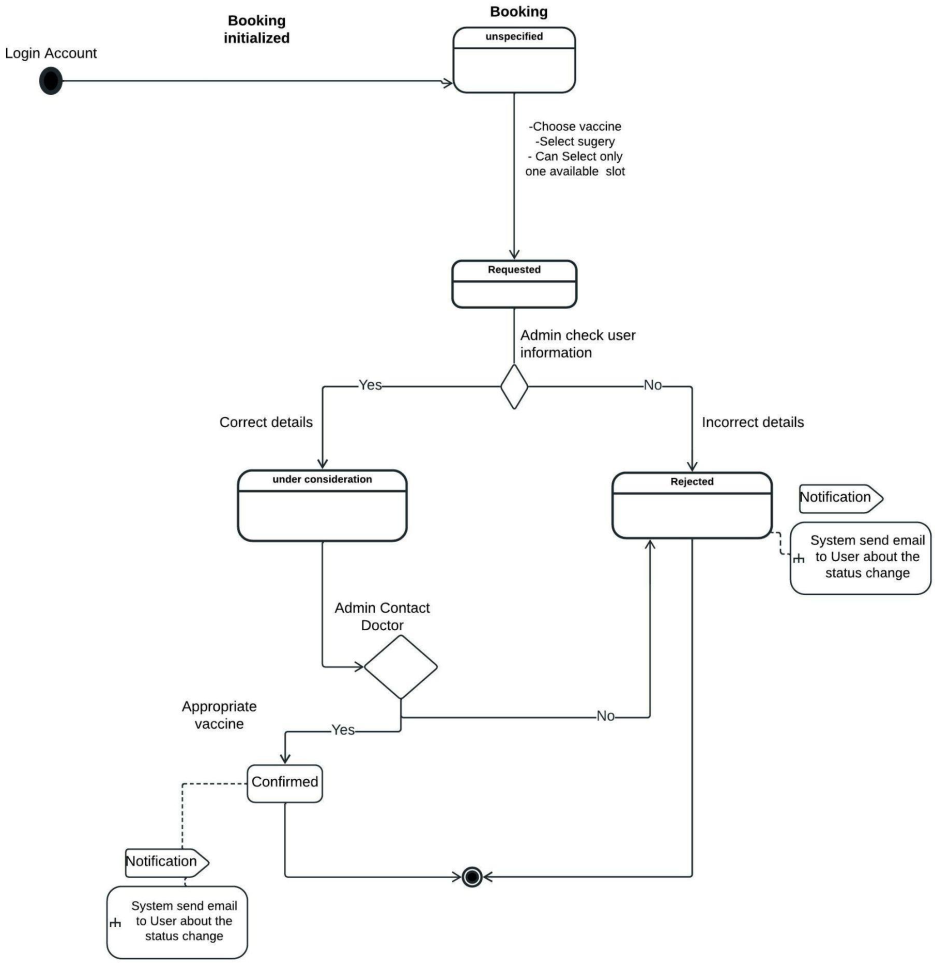
Interface 'SystemUX' **Constraints**

In our case we have created an interface called 'SystemUX' that allows the user and Administrator to interact with the system. This interface contains a set of operations that are also present in the 'User' & 'Administrator' classes. The dashed line is a separator that indicates the 2 entities don't have the same privileges. (See picture at left hand side)
Administrators and User have constraints specifying different access and privileges into the system.(Picture right hand side)

State machine diagram

Task 5

During the above scenario, the status of the booking changes from "unspecified" to "confirmed" or "rejected". Draw a state diagram for an object of the class "Booking".



The behavioural diagram shown above details all the state, transitions and events that take place when a user tries to create a 'Booking' .

The booking state is 'unspecified' at first. However, when a User logs into their account to make a booking, the status of the booking changes to 'requested' when an event occurs, such as when they choose a vaccine, surgery or available slot.

If the details are correct, the booking is placed under consideration; if they are incorrect, the booking is rejected.

Following that, the administrator will contact the doctor, who will determine whether the requested vaccine is appropriate for the User. When the doctor approves the vaccine, the booking status changes to 'confirmed,' and the system notifies the user.

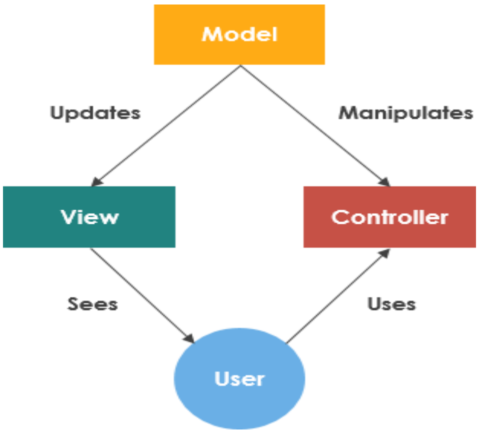
Model View Controller (MVC)

Task 6

Briefly explain (maximum 300 words) what MVC design pattern is, in software architecture context. You are expected to explain the responsibilities of each component of the pattern and how they collaborate with each other. You should use an example from the 'vaccine booking' scenario above, to clarify your points.

MVC (or Model-view-controller) is a popular software framework for successfully and efficiently relating the user interface to underlying data models. MVC is an architectural pattern, which implies it governs the whole application design.

MVC Framework has been widely used by many software developers and other software frameworks and libraries. Traditionally used for desktop graphical user interfaces (GUIs), this pattern has become popular for designing web applications.



In the Model – View – Controller (MVC) framework each component has distinctive responsibilities:

- Model: where the data of the application managed
- View: present the model to the user
- Controller: receives input from the user and passes it to the model in an understandable way.

The user sends data to the Controller; the controller communicates with the Model; the Model sends data to View; via View the user is presented with data. Views and controllers together comprise the user interface. A change-propagation mechanism ensures consistency between the user interface and the model.

If we had a web application for the online booking system and wanted to implement the MVC framework, this would be the following :

Controller: An example from the vaccine booking class diagram for controllers would be the system. This is because the system processes the booking once the user requests for an appointment. From the class diagram another controller would be creating an account. This is because before booking an appointment the user has to create an account.

Model: The booking class will allow the user and administrator to communicate if there are any changes on the booking status. If the user wants to cancel the appointment or reschedule the appointment they would need to log in to their account and go on to the booking section where it allows them to book an appointment or cancel the appointment. Now the model part of creating an account is the user has to input their personal information and NHS number so the doctor can access the details quickly.

View: The admin team will use this to view the booking request. Another thing would be the type of vaccine they would be provided as there are a variety of vaccines available based on the age group. The view part of creating an account is the functionality that allows the user to put the user input will and submit the request.

Graphical User Interface design principles

Task 7

All the above diagrams are related to the design of the online system. This design must be "translated" to graphical design. Write a report of no more than 300 words to explain any three (3) of the main design principles and how each one could be applied in the design of the graphical user interface. The report must be referenced.

User interface design is critical in defining how people interact with software products. This is because user interface design is concerned with how various visual components, such as colours, typography, and pictures, interact to produce a unified user experience. Every user interface design project's main objective is to develop a product that is both aesthetically pleasing and easy to interact with.

3 main design principles that could be applied to the online booking system :

-User control and freedom: Users frequently behave unknowingly. They require a clearly designated "emergency exit" in order to stop the unpleasant activity without having to go through a lengthy procedure. It generates a sense of freedom and confidence when users may easily back out of a procedure or undo an activity. Exits help users to keep control of the system and avoid being stranded or angry.

-Create an easy-to-navigate interface : Navigation should always be clear and self-evident. Users should be able to enjoy exploring the interface. Good UI puts users in their comfort zone by providing some context of where they are, where they've been, and where they can go next:

- Provide visual cues. Visual cues serve as reminders for users. Allow users to navigate easily through the interface by providing points of reference as they move through a product interface.
- Predictability. Users should be provided with cues that help them predict the result of an action.

-User control and freedom: Another of the main design principles because users often make mistakes or change their minds. User control allows them to exit a flow or undo their last action and go back to the system's previous state. This rule means that the user should always be able to quickly backtrack whatever they are doing.

Reflective group statement

Task 8

Write a statement (maximum 300 words) to reflect on your team work experience. This must be a group statement. All members are expected to contribute to each task and to the statement.

As a group, we agreed to share tasks. This allowed us to focus on other areas of the coursework while also reducing our workload.

We worked strategically and proactively as a team, arranging up to five meetings at Cavendish campus which were productively employed according to the meeting's agenda.

The team leader Ali Almajdoub , organised and kept the team informed about forthcoming meetings. The team's progress was aided by the contributions of Justina Aikeremiokha Ali Almajdoub , Sohag Noman , and Jubair Ahmed attendance.

Justina Aikeremiokha , Ali Almajdoub , Jubair Ahmed and Sohag Noman solid contributions on creating all the class diagram, object diagram, state machine diagram and sequence diagram was very helpful in moving the team forward in the task, as was Justina Aikeremiokha contribution of holding everyone accountable and making sure they knew what was expected of them. Before the project submission we practised our tutorial experiences in understanding and identifying useful semantics that have added meanings to our diagrams.

Jubair Ahmed , Justina Aikeremiokha , Ali Almajdoub , and Sohag Noman contributions and research ability in gathering information for the report were quite beneficial in completing all the questions about the Interface, MVC , and Graphical User Interface explanation.

Besides the normal meeting sessions, we had also asked for feedback to our module leader in order to deliver a successful and meaningful work.