

*Date: 29th March, 2023*

**SENG 401**  
**Group 26**  
**Project Final Report - uFlourish**  
**University of Calgary**



Source: <https://www.istockphoto.com/illustrations/software-architecture>

## **Table of Contents**

<b>Table of Contents</b>	<b>1</b>
<b>Group Members</b>	<b>2</b>
<b>Introduction</b>	<b>3</b>
<b>Functional Requirements</b>	<b>5</b>
<b>Non-Functional Requirements</b>	<b>7</b>
<b>Requirements Traceability Matrix (RTM)</b>	<b>8</b>
<b>Architecture Diagram</b>	<b>11</b>
<b>Sequence Diagrams</b>	<b>12</b>
<b>Class Diagram</b>	<b>15</b>
<b>Testing</b>	<b>16</b>
<b>Test Cases + Outcomes</b>	<b>17</b>
<b>Sources/Further Reading</b>	<b>1</b>
<b>Project Repository</b>	<b>19</b>

## **Group Members**

Kairos Koh Jia Jun (30201952)

Simrat S. Benipal (30129328)

Luis Miranda (30053613)

Rei Tsunemi (30121202)

Muhammad Ahsan Zia (30121304)

Chun-chun Huang (30112677)

Shabit Hassan (30062679)

## Introduction

The Wellness Center (WC) is a medical facility located within and operated by the University of Calgary, organized to serve the varied spectrum of healthcare needs expressed by the university community. As a holistic, centralized healthcare resource, the WC offers a wide range of services to students, faculty, and staff, including: (i) primary and secondary medical care; (ii) counseling and short-term psychotherapy; (iii) chiropractic services; and even (iv) massage therapy.

Special administrative considerations are entailed by the provision of healthcare services within a university setting, in contrast to community settings, as workflow must operate congruently with the organizational processes of the university itself. Permitted only to render services to university affiliates, the WC must authorize and identify all patients with the university's *Central Authentication System* (CAS). Moreover, the clinic must adhere to an additional set of COVID-19 screening protocols prescribed by the university, beyond Alberta Health guidelines, including a requirement for all patients to submit COVID-19 screening forms to the university database on days they visit.

Provision of a diversity of services beyond primary care also requires more comprehensive coordination, especially with regards to staffing, patient billing, and patient registration.

The WC continues to make gradual refinements to its administrative workflow in response to inefficiencies and an increasing university population, leveraging several technologies to enhance client interaction. The existing patient-facing technology combination includes:

- *Medeo Health*, a third-party patient portal for appointment booking.
- The University of Calgary's custom COVID-19 screening portal.
- CAS, used for authentication and authorization.
- A custom mental health screening portal, accessible on in-clinic tablets and used to perform preliminary DSM-5 assessments.
- E-mail, used to coordinate registration for counseling and psychotherapy.
- Phone lines, used to coordinate the vast majority of administrative servicing of patients.
- A custom appointment sign-in system, also accessible on in-clinic tablets, which is yet to become operational.

However, this combination appears to manifest a wide spectrum of issues with efficiency, data accessibility, and security.

The use of multiple, disjoint technologies not only reduces the efficiency of administrative workflow by requiring additional effort in the management of data consistency, but may greatly inconvenience patient experience. This is quite disadvantageous for scalability, with accommodation of each new use case requiring additional administrative and client-side management overhead, which is problematic given the various use cases still requiring automation (e.g., billing, paper registration forms).

Moreover, the technologies themselves are relatively inadequate. To illustrate, the patient portal lacks retention of appointment history, is indifferent to the registration status of patients under non-primary care providers, and delegates authorization to an external system unaffiliated with CAS. Concerningly, the WC's reliance on unsecured e-mail in coordinating access to counseling and psychotherapy is rather inappropriate, given the sensitivity of health information being communicated and retained. WC clientele remain primarily reliant on phone servicing for appointment booking and other automatable tasks, manifesting as lengthy queues.

Such issues may be effectively resolved by the implementation of a centralized platform. However, it is due to the relative uniqueness and diversity of use cases that the commercial availability of a platform perfectly capturing client interaction needs is improbable, hence the WC's resortment to a collection of disjoint systems. Nevertheless, it is likely that the WC would be open to a custom solution, given their experimentation with a custom mental health screening portal and appointment sign-in.

Herein, we describe the development of ***uFlourish***, a web-based patient portal that provides a centralized platform for client interaction. Inspired by the existing organizational pattern of disjoint services, along with significant foreseeable extensibility, *uFlourish* follows a microservices architectural pattern with intraservice cohesion enforced by the MVC pattern. The system was developed within the ASP.NET framework, featuring extensive support for Web API and MVC.

The system implements five major use cases:

- UC-1    **Managing account information:** *Enabling patients to manage their account information, including payment card and insurance information.*
- UC-2    **Refilling prescriptions:** *Enabling patients to request refills for existing prescriptions, without scheduling a full appointment.*
- UC-3    **Requesting mental health support:** *Enabling patients to request counseling and/or short-term psychotherapy.*
- UC-4    **Making payments:** *Enabling patients to view outstanding charges in their accounts.*
- UC-5    **COVID-19 screening:** *Enabling patients to submit COVID-19 screening forms directly from their account.*

# Functional Requirements

The functional requirements of the system are the features/functions of the software product that it is able to achieve. For our project, *uFlourish*, we have various functional requirements that we wanted the system to achieve. Below is a breakdown of the functional requirements for each use case.

## UC-1    **Account information:**

- The system shall allow new users to create a new account by providing an email and password.
- The system shall allow authenticated and authorized users to complete the user profile or edit the existing profile
- Once the sign-up user completes their profile, they are able to enter their payment and insurance information.
- The system shall not allow users to edit their email.

## UC-2    **Refilling prescriptions:**

- The system shall allow authenticated and authorized patients to request refills for prescriptions, specifying the medication name and date of request.
- The system shall make refill requests visible to administrative staff, which they shall be able to approve or decline.
- The system shall allow patients and administrative staff to view the status of the refill requests to which they are authorized.

## UC-3    **Requesting mental health support:**

- The system shall allow authenticated and authorized patients to schedule appointments, by selecting the date, viewing the list of available booking slots, and scheduling an appointment.
- The system shall allow the authenticated patients to view, edit, or cancel their existing appointments.
- The system shall allow the admin/medical staff to view all the appointments requested and make any changes if required.

UC-4    **Making payments:**

- Administrative users shall be able to create a new charge to a user's account, specifying their ID and details of the transaction.
- Administrative users shall be able to delete a charge to a user's account.
- Administrative users shall be able to modify a charge to a user's account.
- The system shall allow patients to view a list of their financial transactions.

UC-5    **COVID-19 screening:**

- The system shall provide the COVID-19 screening form to be completed each individual coming into the clinic for an appointment.
- The form shall require the individual to provide their name, UCID, and age, and select possible COVID-19 symptoms.
- Upon submission of the form, the system shall notify the user if they are safe to visit the clinic or not through a pop-up message.

## Non-Functional Requirements

Non-functional requirements of the system are some attributes or characteristics of the software that have no major effect on the user/business but define how the system should interact with the environment/users. These are mostly related to performance, scalability, maintainability etc.

1. **Performance:** The website should load within a certain timeframe such as three seconds or less. The use of MVC architecture and having separate databases for different services, our system is able to provide huge performance benefits. The system can take multiple appointments at the same time. Since taking appointments, refilling prescriptions, registering users are all done on separate databases, multiple users can access all these at the same time.
2. **Security:** The website should be secure and protected against threats such as hacking, malware, and data breaches. *uFlorish* uses an authentication system that limits the users on what they can do with the system. All the users have to be authenticated before using the system. If certain users are not allowed to use the system, they cannot log in and book appointments etc. A normal user/client of the system is not able to approve prescriptions on their own, only the medical staff or administrators are able to do so, providing extra validation.
3. **Usability and Consistency:** The system should provide an easy-to-use, simple interface to interact with the user. The interface is divided into different web pages, which makes it easy to differentiate between different services. Having different services on different web-pages makes the learning process of the system easy.
4. **Compatibility:** The web-based system should be able to be accessed from any Internet connected device, making it compatible with a wide range of devices from laptops, desktops to mobile phones and tablets etc.



## Requirements Traceability Matrix (RTM)

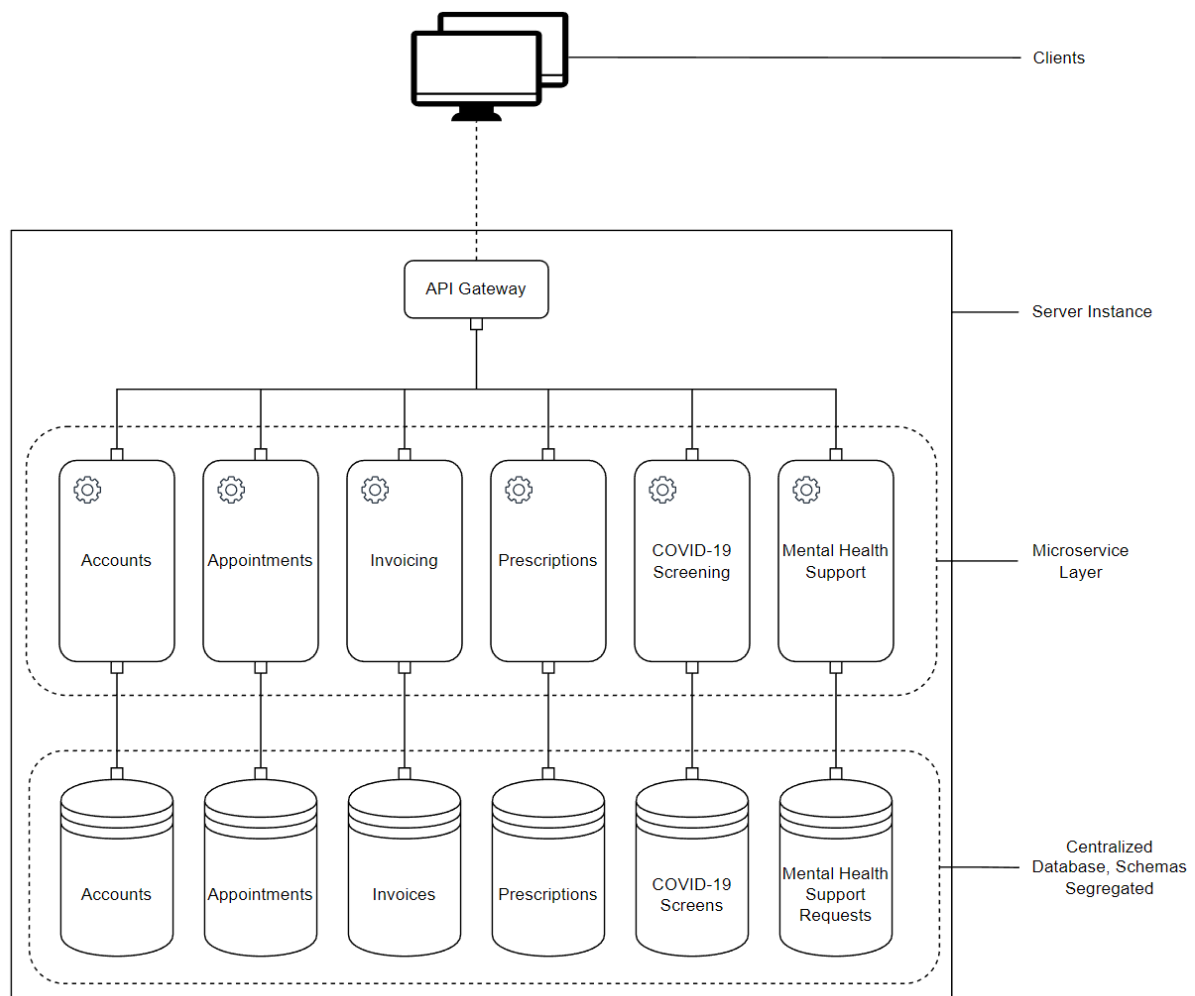
Note: Priority observes a descending scale from 1 (highest) to 5 (lowest)

Req-No	Description	Rational	Functional Specification	Priority	Requirement Met	Comment
FR-1.1	Account Creation	Users are required to be able to access the system.	The system shall allow new users to create a new account by providing an email and password.	1	Yes	N/A
FR-1.2	Changing General Information	Users may be required to update account information as real-world details change.	The system shall allow authenticated and authorized users to complete the user profile or edit the existing profile	2	Yes	N/A
FR-1.3	Changing Payment Information	Users may be required to update payment information as real-world details change.	Once the sign-up user completes their profile, they are able to enter their payment and insurance information.	2	Yes	N/A
FR-1.4	Changing User Email	Users should not be able to edit their sign-in email as it may be used to distinctly identify them in the system.	The system shall not allow users to edit their email.	1	Yes	N/A
FR-2.1	Requesting Refills	Booking appointments strictly to request refills may be wasteful of healthcare resources, engendering a need for a remote solution.	The system shall allow authenticated and authorized patients to request refills for prescriptions, specifying the medication name and date of request.	3	Yes	N/A
FR-2.2	Refill Requests Approval/Denial	Administrative users should be able to use their medicolegal judgment to satisfy or decline healthcare requests.	The system shall make refill requests visible to administrative staff, which they shall be able to approve or decline.	3	Yes	N/A
FR-2.3	Refill Request Status Visibility	Administrative users should be able to use their medicolegal judgment to satisfy or decline healthcare	The system shall allow patients and administrative staff to view the status of the refill requests to which they are authorized.	3	Yes	N/A

		requests.				
FR-3.1	Appointment Reservation	Patients should be able to circumvent physical queues to reserve appointments at their leisure time.	The system shall allow authenticated and authorized patients to schedule appointments, by selecting the date, viewing the list of available booking slots, and scheduling an appointment.	2	Yes	N/A
FR-3.2	Patient Appointment Management	Patients should be able to circumvent queues to manage appointments at their leisure time.	The system shall allow the authenticated patients to view, edit, or cancel their existing appointments.	2	Yes	N/A
FR-3.3	Administrative Appointment Management	Administrative staff should have access such that they can view and edit any changes to the appointments if needed.	The system shall allow the admin/medical staff to view all the appointments requested and make any changes if required.	2	Yes	N/A
FR-4.1	Issuing Charges	Administrative users should be the only users possessing the ability to declare charges to a patient's account.	Administrative users shall be able to create a new charge to a user's account, specifying their ID and details of the transaction.	4	Yes	N/A
FR-4.2	Admin access: Delete charges	Administrative users should be the only users possessing the ability to remove charges from a patient's account.	Administrative users shall be able to delete a charge to a user's account.	4	Yes	N/A
FR-4.3	Admin access: Modify charges	Administrative users should have the access to modify any charges to the users, if the need arises.	Administrative users shall be able to modify a charge to a user's account.	4	Yes	N/A
FR-4.4	Financial Transactions	Patients should have the ability to view a history of all financial transactions.	The system shall allow patients to view a list of their financial transactions.	4	Yes	N/A
FR-5.1	COVID-19 form	Users are required to enter COVID information before entering the clinic.	The system shall provide the COVID-19 screening form to be completed each individual coming into the clinic for an	5	Yes	N/A

			appointment.			
FR-5.2	COVID-19 form information entry	Users have to provide their name, UCID and any COVID related symptoms to verify if they are safe to enter the clinic, and retain a record of this on the university's database.	The form shall require the individual to provide their name, UCID, and age, and select possible COVID-19 symptoms.	5	Yes	N/A
FR-5.3	COVID-19 form results	Users should receive immediate feedback regarding whether they are safe to enter the clinic or not.	Upon submission of the form, the system shall notify the user if they are safe to visit the clinic or not through a pop-up message.	5	Yes	N/A
NFR-1	Performance	Users should be able to access a functioning website that has great performance with page-loading times	The website should load within a certain timeframe such as three seconds or less.	3	Yes	N/A
NFR-2	Security	Users should be secure against any possible malware tools, especially for payment processing	The website should be secure and protected against threats such as hacking, malware, and data breaches.	1	Yes	N/A
NFR-3	Usability and Consistency	The interfaces should be easy to use and learn.	The system should provide an easy-to-use, simple interface to interact with the user.	1	Yes	N/A
NFR-4	Compatibility	The interface should be available to view from different types of internet connected devices	The web-based system should be able to be accessed from any Internet connected device, making it compatible with a wide range of devices from laptops, desktops to mobile phones and tablets etc.	2	Yes	N/A

## Architecture Diagram

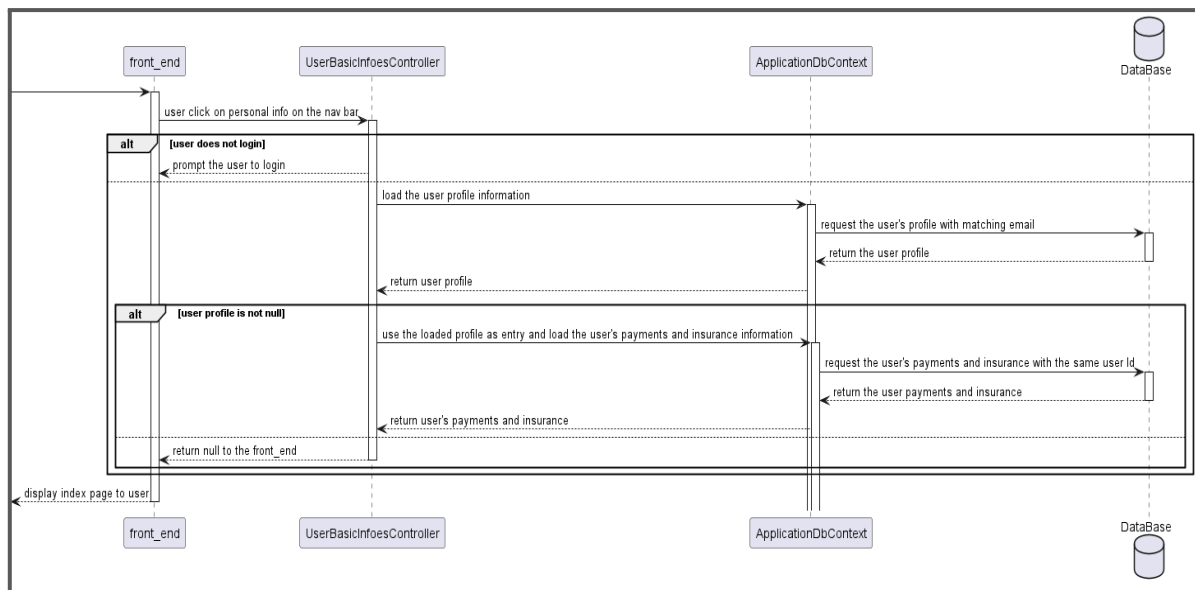


*uFlorish* uses a customized architecture style which is a combination of Microservice and MVC architectural styles. The system uses Microservice architectural style to have multiple independent services each having a separate database on its own. The use of microservice is required to break a big application into smaller sub-applications which can be developed simultaneously and by multiple team members. This makes the system easy to scale and maintain, as each of the services can be scaled horizontally or vertically easily.

MVC architecture style is used mostly among web based applications due to the advantage of separating user interface, business logic and database into separate layers, making it easy to maintain. In our system each of the use cases are divided into microservices and then implemented using the MVC architecture. This made the system to be deployed easily and maintenance of the code was easily handled by the team members.

# Sequence Diagrams

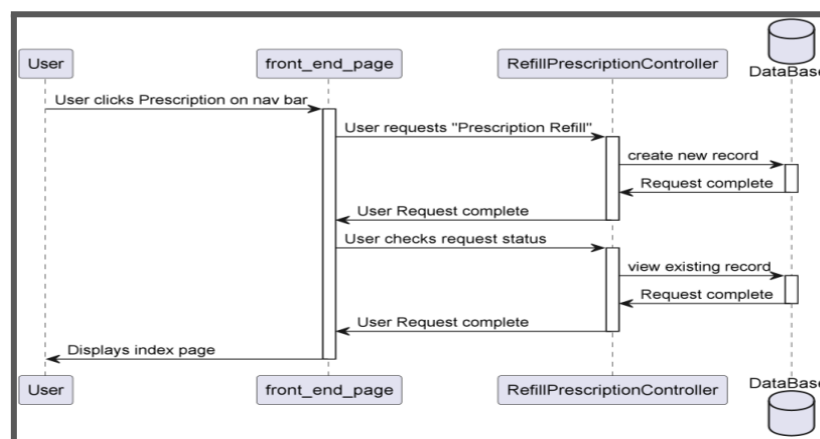
## Personal information page



### Explanation:

Users will be prompted to login if they have not yet signed in. Once they sign in and try to access the personal information, UserBasicInfo controller will request the user profile with the signed-in email through our central repository which is ApplicationDbContext object. If the user already creates the profile, the UserBasicInfo model will be returned to the UserBasicInfoesController, then the controller will try to load the payment and insurance linked to the user profile. Otherwise, null will just be returned to the frontend without trying to load the payment and insurance information, and users will be prompted to complete their profile when they try to access personal information.

## Refill Prescriptions

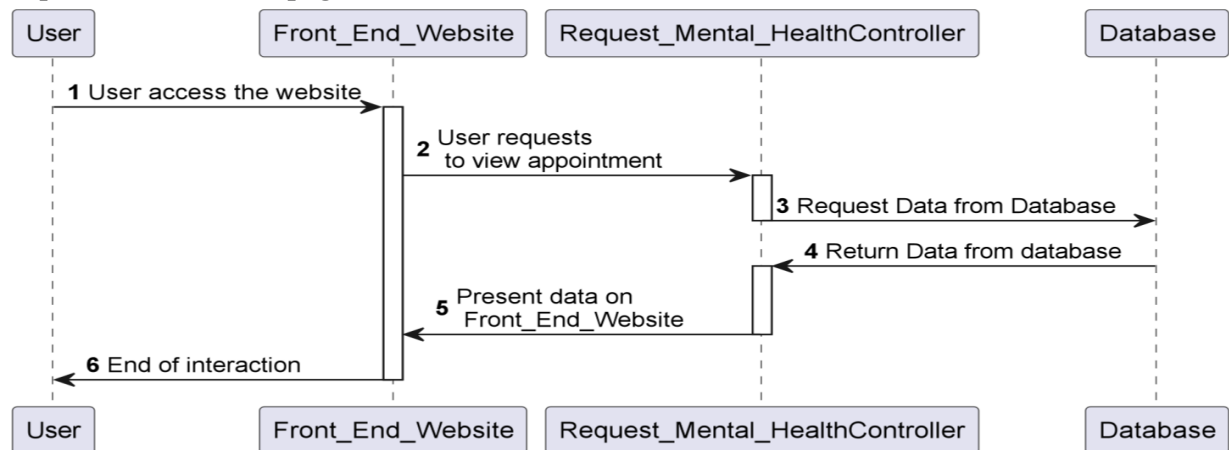


### Explanation:

The user can access the "Prescriptions" front end page by clicking on "Prescriptions" in the nav bar. Once at the front end page, the user can then request a "Prescription Refill" which then transfers over the control to "RefillPrescriptionContoller" which then checks the database to get the list of prescriptions and then the control is transferred back. "RefillPrescriptonController" then transfers

control back to the front\_end\_page. Once back at the front\_end\_page. The user can then again request the status for their prescription. This again hands the control over to the “RefillPrescriptonController” which again checks the database. After the database is checked, the control is handed back to the “RefillPrescriptonController” which can then hand the control back to the front\_end\_page

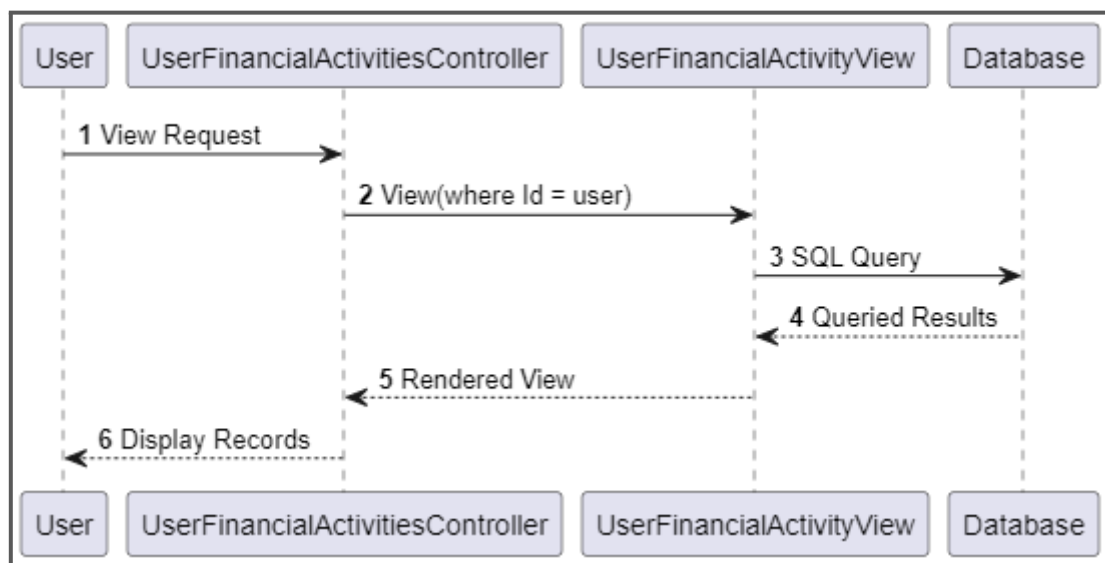
### ***Request Mental Health page***



### **Explanation:**

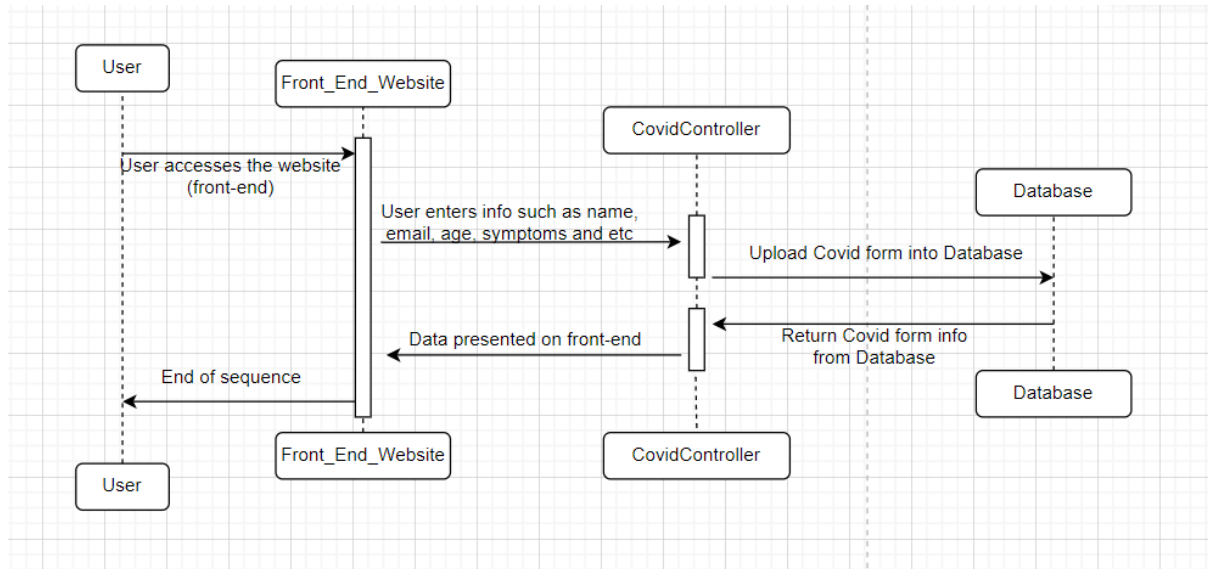
Once the user accesses the web application’s front end web page to view the request health support main page, the control is given to the “Request\_Mental\_HealthController”, which will check the database to get the list of stored appointments and transfer the data back to the controller. The controller transfers the information to the ‘view’ which is the front end website for the user. The other operations such as adding/creating a new appointment, editing or deleting the existing appointments have similar control structure as shown in the above sequence diagram.

### ***My Financials***



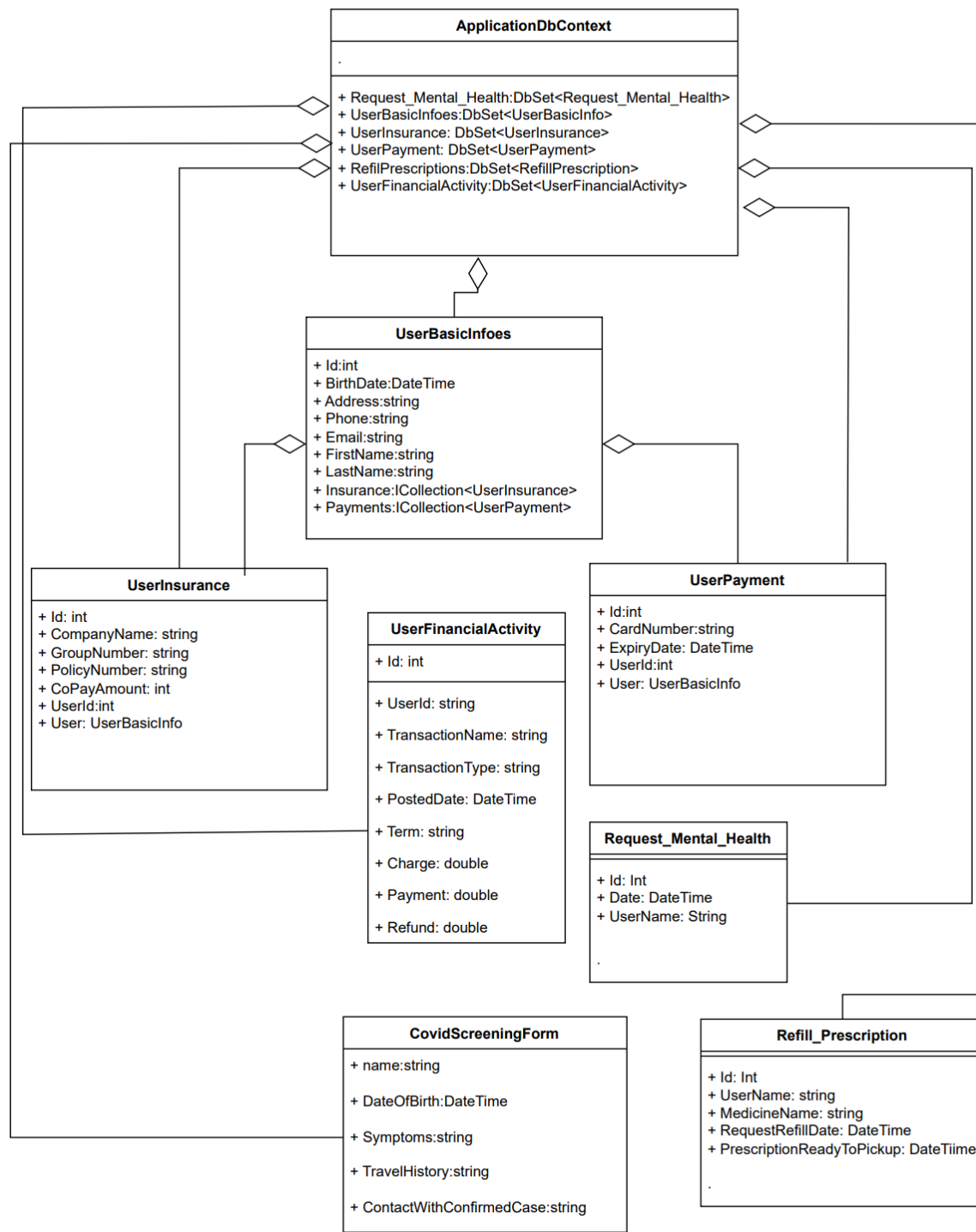
**Explanation:**

As the user enters the “My Financials” page on the front-end, a view request is sent to the back-end system where the controller handles it. The controller then directs a view call to the view page where the database is then queried using SQL. The results are then propagated back to view where a rendered version is then displayed for the front-end to view.

***Covid Screening Form*****Explanation:**

As the user accesses the website’s front-end and then the Covid form, they are prompted to enter their personal information and further details required in the covid screening procedure. The completed form is then uploaded to the database and can also be pulled out and assessed.

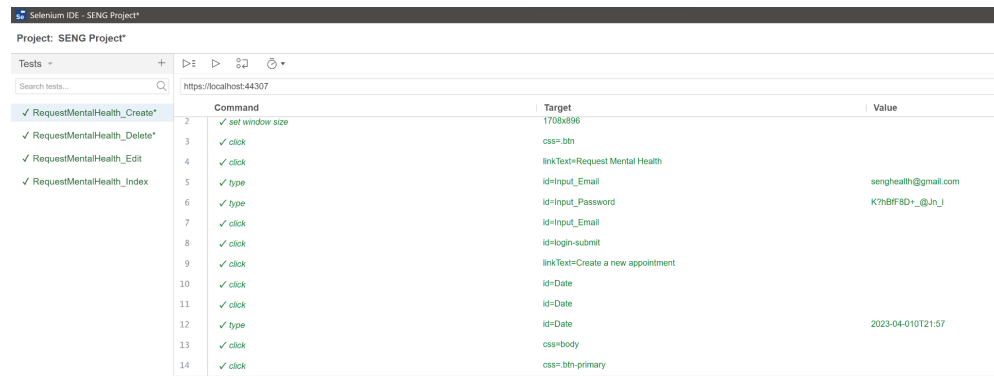
# Class Diagram



The class diagram above is only showing all the models within our system, not our entire project. The central repository which is called ApplicationDbContext contains every model table as Dbset objects. When the controllers are trying to access the database, they will all need to have a reference object of the ApplicationDbContext and pick the needed data from the Dbset within the ApplicationDbContext object. This class diagram is a great way to track what tables we have in our database.



# Testing



Project: SENG Project*			
Tests +			
Search tests... https://localhost:44307			
	Command	Target	Value
✓ RequestMentalHealth_Create*	2 ✓ set window size	1708x896	
✓ RequestMentalHealth_Delete*	3 ✓ click	css= btn	
✓ RequestMentalHealth_Edit	4 ✓ click	linkText=Request Mental Health	
✓ RequestMentalHealth_Index	5 ✓ type	id=input_Email	serghealth@gmail.com
	6 ✓ type	id=input_Password	K7hBFF8D+_@Jn_J
	7 ✓ click	id=input_Email	
	8 ✓ click	id=login-submit	
	9 ✓ click	linkText=Create a new appointment	
	10 ✓ click	id=Date	
	11 ✓ click	id=Date	
	12 ✓ type	id=Date	2023-04-01T21:57
	13 ✓ click	css=body	
	14 ✓ click	css= btn-primary	

*The above image briefly shows how we are implementing our test cases, using Selenium (Smoke testing)*

*Smoke testing: Often done using selenium and is a form of unit testing where one can automate a “click-through” around the entire web-application*

## Test Cases + Outcomes

Test Cases for all requirements in RTM	Outcome
Successfully register and authenticate the user	Passes
Schedule (includes create, delete, edit and view) appointments for Mental Health Support	Passes
Have working alerts/notifications based on upcoming appointments	Passes
Working prescription of medicines	Passes
Successfully manage financials/payments	Passes
Working system management	Passes
Has compatibility with multiple systems	Passes
Performance is optimal	Passes
Security is up-to-par	Passes
Has good usability and consistency	Passes
Has good compatibility	Passes

## Sources/Further Reading

- [\*18,800+ Software Architecture Illustrations, Royalty-Free Vector Graphics & Clip Art - iStock\*](#)
- [\*PlantUML\*](#)
- [\*Selenium\*](#)
- [\*What is Smoke Testing? | Definition from TechTarget\*](#)

## Project Repository

The project can be found at the following Git-Hub address:

<https://github.com/kairoskoh/UFlourish-SENG401.git>