Deliverable 4 Report

Team Squad 8

A. Deliverable 4 Requirements' Status:

Requirements:

Section No.	Requirement Name	Status	Description
3.2.1	Doctor Availability Tracker	Finished	Proper form for doctors to add, delete, or edit their availability details; Should be able to view all their recorded slots.
3.2.2	Calendar Style widget	In Progress	Provides an interactive calendar widget for the patients to choose their appointment dates and times.
3.2.3	Appointment Booking system	In Progress	A formatted booking system with available time slots and doctor's information in a calendar style widget for the user(patient) to select from
3.2.4	Patient upcoming appointments list	Finished	Updates a list to store the logged in patient's upcoming appointments with date, time, logged in patient, and assigned doctor's name for every appointment. If there aren't any upcoming appointments, the message "No Upcoming Appointments" is displayed
3.2.5	Patient previous appointments list	Finished	A list that shows and saves the history of the logged in patient's previous appointments with date, time, logged in patient's name, and the assigned doctor's name for every appointment.
3.2.6	Doctor day-to-day appointments list	In Progress	A list that shows day-to-day appointments for each doctor in the dashboard.
3.2.7	Enabling notification capabilities backend	Finished	Enabled an in-app notification system for the patients to set preferred notification method for reminders.
3.2.8	Appointment deletion capabilities	In progress	Implemented a provision for patient/doctor where under extreme circumstances they will have an option to cancel/delete their appointments.
3.2.9	Appointment Doctor Confirmation capability	Finished	Enabled feature so that a notification is sent to the doctor to confirm the appointment after patient books one.
3.2.11	Basic search to get relevant doctors by name	In Progress	A search interface system with dropdown menus, text inputs, and checkboxes to search doctor's name.
3.2.12	Two-Factor Authentication (2FA)	Finished	Enabled 2FA to enhance data privacy and security of all users.

3.2.13	Clear dashboard for doctors	In Progress	Designed a doctor dashboard where doctors will be able to update their information and available timings.
3.2.14	Contact Us backend implementation	In Progress	Successfully saving logged in user's information in the "Contact Us" page of the website
3.2.15	Patient Appointment Deletion Capabilities	In Progress	A logged in user identifying as a patient can successfully delete already made appointments
3.2.16	DB Table Schema	Finished	A table for "Appointments" successfully added to the database to extend the web application's features

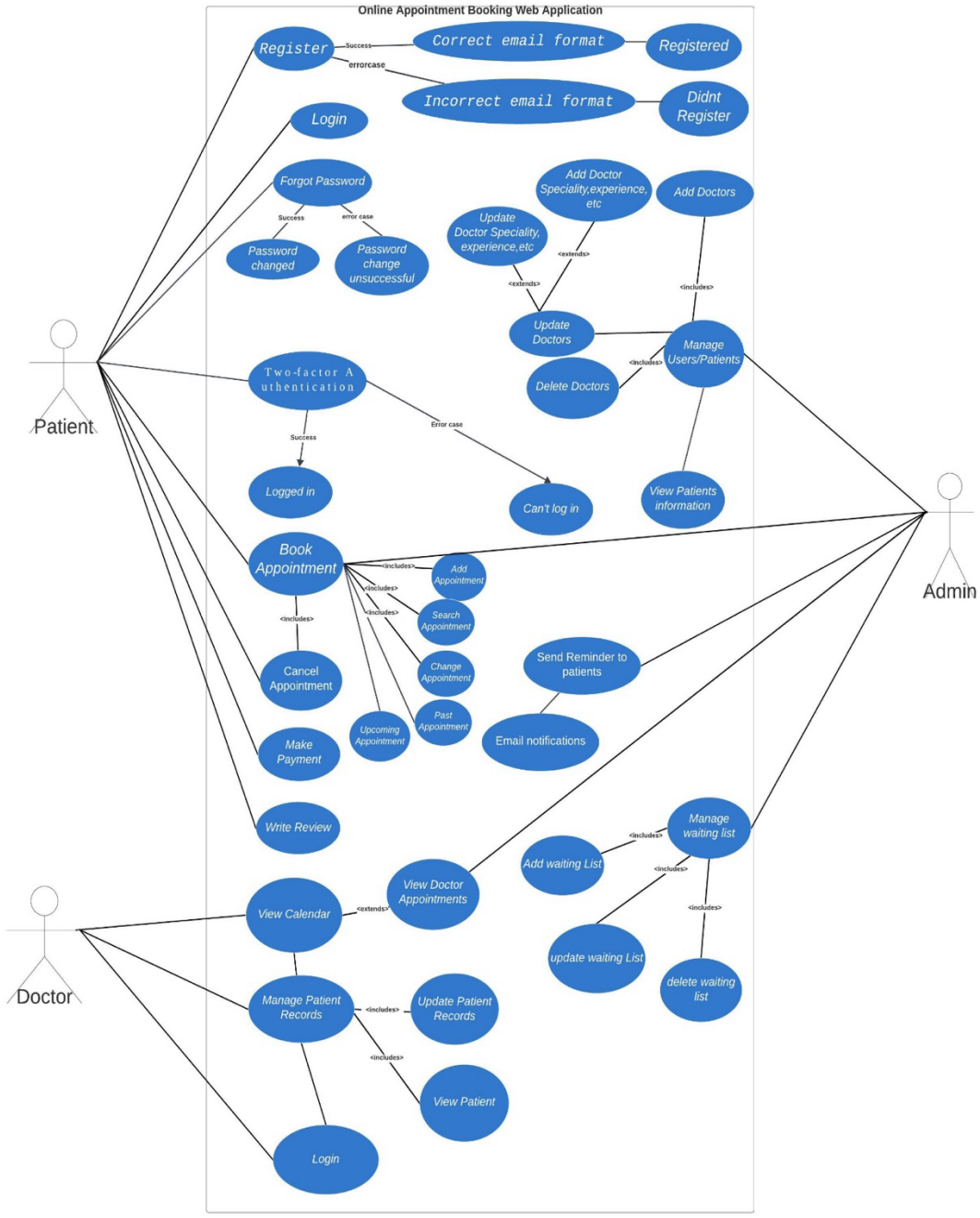
We have made significant progress for this deliverable both in the front end and back equally. The above table shows all the features that were planned to be implemented during this phase. The most prominent changes that were made to our that include finishing up some left-over features from the previous phase that include adding the patient dashboard, profile page, update details form as well as new features from this current phase involving a two-factor authentication setup, patient past and previous appointments storing/display, doctor availability tracker, and an email integration system. There are also other features that are being implemented and are under progress. MedVoyage has completed a lot of functionalities according to the plan but there are also a few features that are currently still in progress and will be pushed to the next phase.

Deliverable 4 Report

Team Squad 8

B. UML design for Deliverable 4:

a. Use Case diagram:

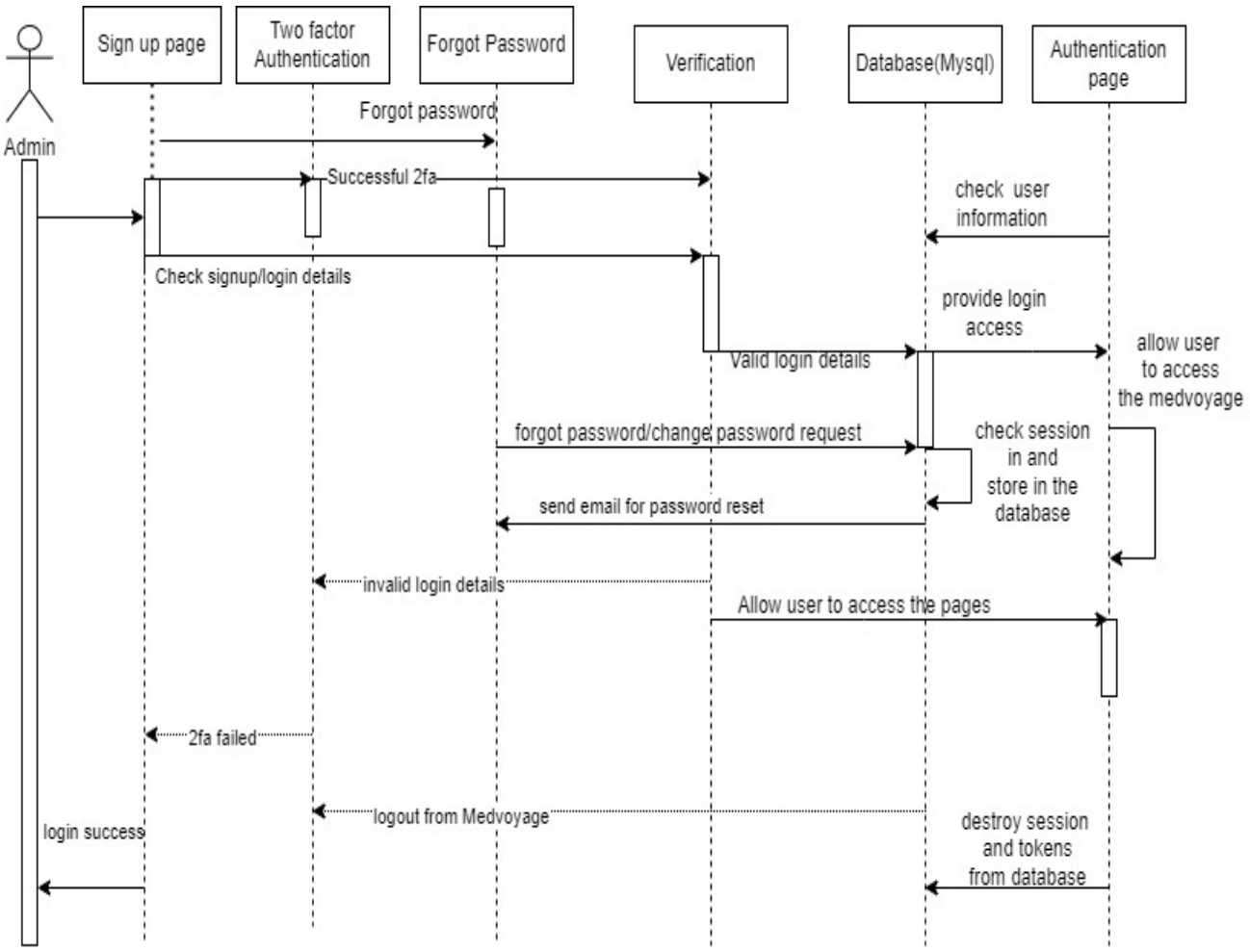


a. Use Case Diagram representing the system's functionalities and the external entities interacting with them

Deliverable 4 Report

Team Squad 8

b. Sequence Diagram

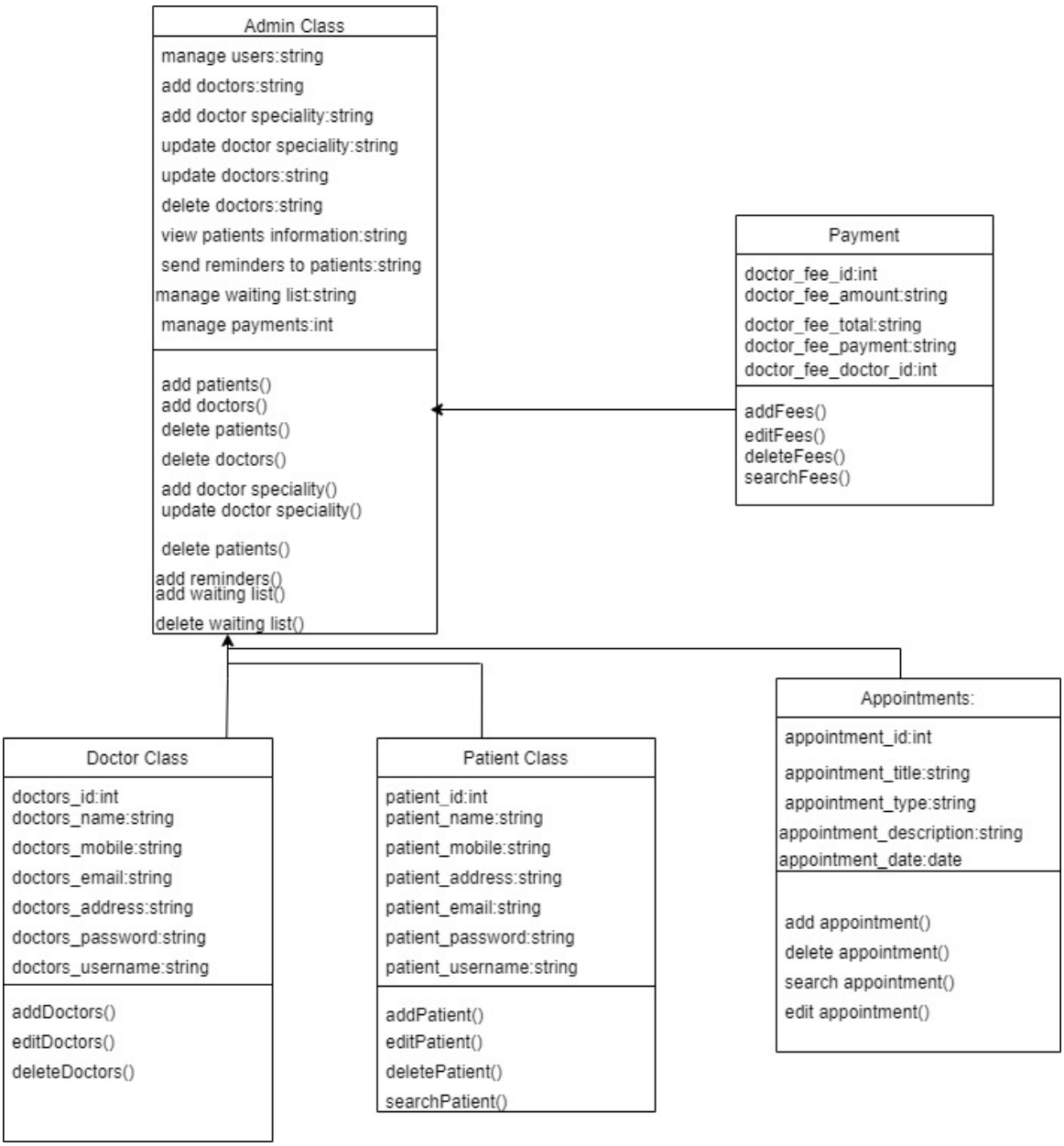


b. Sequence Diagram showing the time-ordered interactions between objects using messages.

Deliverable 4 Report

Team Squad 8

c. Class Diagram:



c. Class Diagram showing showcasing classes, their attributes, operations, and relationship between the various classes

Deliverable 4 Report

Team Squad 8

5. Different use cases for MedVoyage application :

Use case 1:

Use Case Number	UseCase-03
Use Case Name	Registration confirmation for Patients and Doctors
Overview	The Users (Patients/Doctors) shall start entering the registration information on the MedVoyage web application
Pre-condition(s)	The Users have arrived at the MedVoyage web application through the URL ● User database is properly configured. A separate table for patients and a separate table for doctors ● Web server is open, and the host is waiting for registrations of patients/doctors. ● The new users have access to the MedVoyage web application via url
Scenario Flow Main (success) flow:	1.User (patient/doctor) selects the signup option instead of login. 2.The MedVoyage system prompts the user(patient/doctor) for their personal information, including: Username, Password, and Email. 3.The user(patient/doctor) enters the requested information. 4.MedVoyage system verifies if the requested information is within the pre-specified constraints. a.If constraints are violated, Display error message b.Return to Step 2 5.User(patient/doctor) inputs the information. 6.MedVoyage system verifies the information. a.If information is invalid, display an error message specifying instructions. b.Return to Step 6 7.If the email is not entered in the correct format like if the @ symbol is missing or if there is .co or .com is missing then the sign up will not be possible. 8.MedVoyage system displays the confirmation of signup
Alternate Flows	1)The user confirms the MedVoyage system suggestion. 2)The MedVoyage system returns to step (7).

	3)Two factor Authentication is applied and if it is successful then the user could login.
Post Condition	New user(patient/doctor) information has been successfully added to the MYSQL database.

Use case 2 :

Use Case Number	UseCase-04
Use Case Name	Forgot Password
Overview	The users(doctors/patients) shall retreive their password with forgot password option.
Pre-condition(s)	The user(doctors/patient) has already completed registration/sign up page, they have a username, but they forgot password.
Scenario Flow Main (success) flow:	Main (success) flow: 1.The user(doctor/patient) connects to the MedVoyage system. 2.The user(doctor/patient) selects the option to login 3.MedVoyage System prompts the user(doctor/patient) to enter credentials. 4.The user(doctor/patient) enters credentials. 5.MedVoyage System validates credential information is correct. 6.The user forgot the password. 7.The user clicks the forgot password button 8.The username and email will be asked 9.Email notification will be sent to change the password 10.The user will be able to change the password
Alternate Flows	At any step, the user(doctor/patient) can instead click the cancel button. 1)Return to homepage(About Us page)
Post Condition	MedVoyage System redirects users(doctors/patients) to the Login page.

Deliverable 4 Report

Team Squad 8

C. Test Cases (unit tests) for Deliverable 4:

Test Function Name	Test Description	HTTP URL	HTTP Method	Expected HTTP Response Code	Expected Outcome
test_client_dashboard_content	Tests if the fields in the navbar are rendered correctly.	/client_dashboard/	GET	200	Contains navbar fields
	Tests if the footer is displayed correctly in the dashboard.				Contains footer
	Tests whether the font and content are correct				Font and content are displayed correctly.
test_client_Profile_content	Tests if the navbar on the homepage contains all the fields, the content and edit button is displayed along with it.	/client_profile/	GET	200	Contains navbar fields
test_client_profile_rendered_correctly	Tests if the form elements are rendered correctly	/client_profile/	GET	200	Contains "© MedVoyage"
test_client_profile_labels	Tests for the labels are correctly rendered.	/client_profile/	GET	200	Contains 'MedVoyage' and the specific journey-to-health text
test_update_user_info_page_is_rendered	Tests if the navbar on the homepage contains all the fields, the content and update button is displayed along with it.	/client_profile/	GET	200	Contains: "MedVoyage|About Us"
test_readonly_fields	Tests if the readonly fields are displayed as read-only fields correctly	/client_profile/			

Deliverable 4 Report

Team Squad 8

test_update_user_info_form_submission	Prepares data for form submission, checks if the form submission is successful and redirects to the expected page and checks if the user's information is updated in the database	/client_update/	POST	302	Page is successful with correct redirection to update form, updates the given user's information to the dashboard.
test__appt_nav	Checks to see if the page request with the patient appointments button is successful in the navigation and loads to view the patient appointments, once signup and log in have occurred	Various URLs ('/signup/', '/login/', '/home/')	POST GET	302 200	Page is successful with the patient appointments button loading and redirecting successfully once the specific user has logged in
test_appt_page	Checks to see if the patient appointment page with the patient's past and upcoming appointments successfully loads and redirects when the patient has signup and logged in	Various URLs ('/signup/', '/login/'. '/client_appointments/')	POST GET	302 200	The patient appointment page with the patient's past and upcoming appointments successfully loads and redirects when the patient has logged in

Deliverable 4 Report

Team Squad 8

test_no_appt_listing_when_no_appt_made	Checks to see if the message "No Upcoming Appointments" displays under "Upcoming Appointments" if there are no scheduled appointments in the portal by the logged in patient	Various URLs ('/signup/', '/login/', '/client_appointments/')	POST GET	302 200	Page successfully displays "No Upcoming Appointments" when there are no scheduled future appointments in the portal by the logged in user
test_appt_no_nav_logged_out	Checks to see if the patient appointment button doesn't pop up in the navigation bar when no sign up and log in have occurred in terms of successful loading	'/home/'	GET	200	As expected, the page with the patient appointments button in the navigation bar does not pop up as the user has not logged in or signed up as expected
test_page_inaccessible_when_not_logged_in	Tests to check if the patient appointments page doesn't display when no user has logged in	'/client_appointements/'	GET	200	As expected, the patient appointment page is not displayed if the user has not logged in
test_add_slot	Tests to check if doctor availability timings are recorded	/add_slot/	POST	200	Slot should be added to the database
test_delete_slot	Tests to see if an already recorded time slot is deleted	/delete_slot/	DELETE	200	Slot should be removed from the database

test_edit_slot	Tests to see if an already recorded slot is updated with changed values	/edit_slot /	PUT	200	Slot details should be updated in the database

These test cases are the test cases that were written to check the functionalities of newly added features and we have also upgraded the other test cases that were previously mentioned in deliverable 3.

D. A user manual that tells us how to install/use your program. This is meant for the end-user of the software.

User Manual (more of the user manual is under "How to use the basic functionalities of the MedVoyage Web-Application" section)

The initial setup to launch our website on the browser is like the deliverable 3.

Installation of the VS Code:

Visual Studio Code (VS Code) could be used for code editing. To install it, visit the official VS Code website at https://code.visualstudio.com/ and download the version appropriate for your operating system (Windows, macOS, or Linux).

Follow the installation instructions for your specific operating system:

- On Windows, run the installer executable and follow the on-screen instructions.
- On macOS, drag the VS Code application to the Applications folder.
- On Linux, use the provided package manager or download the .deb or .rpm package, then follow the installation instructions.

After installation, open Visual Studio Code from your applications or program menu. You can configure settings globally (user settings) or on a per-project basis (workspace settings). VS Code has built-in Git support. If you're working with a Git repository, you'll see version control icons in the left sidebar. You can stage, commit, and push changes directly from VS Code.

Docker setup:

Docker provides a consistent and isolated environment for applications, making it easier to deploy and manage software across different computing environments. To install and set up Docker, you'll need to follow specific steps based on your operating system. Docker is available for Windows, macOS, and various Linux distributions.

To download the docker desktop,

Visit the Docker Desktop for Windows page: https://www.docker.com/products/docker-desktop.

Visit the Docker Desktop for Mac page: https://www.docker.com/products/docker-desktop.

Deliverable 4 Report

Team Squad 8

Download the Docker Desktop installer for Windows/Mac Installer. Run the installer and follow the installation wizard's instructions. During the installation, Docker may prompt you to enable Windows Subsystem for Linux (WSL) 2. Allow this, as WSL 2 is required for Docker to run, or you can install Windows subsystem for Linux using the terminal directly by running the wsl --install command or you could get it directly from the Microsoft store for windows OS. Once the installation is complete, Docker Desktop should be available in your windows applications. Launch it. To verify whether your Docker is configured correctly try running the docker --version command from your terminal.

Cloning the repo into the VS Code:

If you haven't already, you'll need to install Git on your computer. You can download Git from the official website: https://git-scm.com/downloads.

Launch Visual Studio Code on your computer. You should be able to see clone repository in a new window. Click on clone repository and choose clone git repo, it will prompt you to add the link or repository name. Copy and paste the repo link there https://github.com/kalyancheerla/MedVoyage.

Choose the directory in which you want the repo to get cloned and open in a new window. This will create a new directory with all the workspace files of your repo. You can clone the repo using the terminal also. In VS Code, you can open the integrated terminal by clicking on "View" in the top menu and selecting "Terminal." Use the terminal to navigate to the directory where you want to clone the Git repository. You can use the ***cd*** command to change directories. Use the git clone command with the repository name as follows:

```
git clone https://github.com/kalyancheerla/MedVoyage.git
```

After the repository is cloned, you can open it in Visual Studio Code. You can now work on the files in the cloned repository, commit changes, and interact with Git directly from the integrated terminal or by using the source control features provided by VS Code.

Python interpreter setup:

Go to the repository cloned directory in your VS code and open a new terminal to install the Django and MySQL dependencies. In the new terminal use the following command:

```
pip install -r src/requirements.txt
```

This will open the document requirements.txt read the Django, MySQL and recursively install these dependencies.

Setup of MySQL Database in Docker:

We have already documented the tool script to run and setup the MySQL database in docker in the tools/setup_mysql_docker.sh document in our files. This script can be executed directly in Linux based environments, for windows we can execute the following commands:

```
docker run -p 3306:3306 --name mysqlinstance -e MYSQL_ROOT_PASSWORD='pa$$w0rd' -d mysql:latest
```

When you run this command, Docker will download the MySQL image if it's not already available locally, create a new container from that image, and start the MySQL database server within the container. The **-p**

option maps the MySQL port so that you can connect to it from the host or another machine. The MySQL root password is set to 'pa$$w0rd' for authentication.

The resulting container, named "MySQL instance," will be running in the background with the MySQL server accessible on port 3306, and you can interact with it as needed. This is a common way to run a MySQL database server in a Docker container for development or testing purposes.

```
docker exec -it mysqlinstance mysql -uroot -p'pa$$w0rd'
```

When one runs this command, one will be granted access to the MySQL server running inside the "mysqlinstance" container as the root user with the provided password. This allows one to interact with and manage the MySQL database server using the MySQL command-line client. One can execute SQL queries, manage databases, and perform various administrative tasks within the container's MySQL environment. MySQL command line opens, and one can create database *MedVoyagedb* and then exit from the MySQL using the exit command.

```
CREATE DATABASE MedVoyagedb;

exit
```

Creating tables and schema of models in MySQL DB:

Navigate to the directory where one has previously installed the Django and MySQL dependencies in the terminal and make migration files in that directory.

```
python src/manage.py makemigrations
```

It helps one create migration files for any changes one has made to their database schema, making it easy to apply these changes to the database when you run the "migrate" command. This is part of Django's database schema management and version control system.

```
python src/manage.py migrate
```

It helps you keep the database schema in sync with your project's models and ensures that the database is properly configured for your application. This command is typically run after creating or modifying migration files with the "makemigrations" command.

Running the Django webserver:

```
python src/manage.py runserver
```

In the same directory, after creating the migration files we can run the above command to run the webserver and this can be viewed on localhost:8000

Running unit testcases

```
python src/manage.py test src/main/tests/
```

This is a way to execute test cases for the "main" app within your Django project. This is a crucial step in ensuring the reliability and correctness of your Django application by verifying that various components

and features work as expected. Currently, we integrated GitHub Actions to run unit testcases on commits made on the main branch and pull requests that are created.

E. Clear instructions on how to compile/run both your program and your test case (the program must compile/run)

Prerequisites:

1. Install VS Code

2. Install Docker Desktop

Setting Up:

3. Clone the MedVoyage repository in VS Code.

4. Set up the Python interpreter.

5. Install necessary project dependencies using: `pip install -r src/requirements.txt`.

   - This installs both `django` and `mysqlclient` required for the MedVoyage project.

6. We could also set up the same using python virtualenv:

   - Create a virtual environment.

   - Source into that virtual environment.

   - Install the dependencies as mentioned above.

Database Setup:

7. Set up MySQL DB in Docker. Refer to `tools/setup_mysql_docker.sh` for guidance.

8. Run the Docker command: `docker run -p 3306:3306 --name mysqlinstance -e MYSQL_ROOT_PASSWORD='pa$$w0rd' -d mysql:latest`.

9. Access the MySQL instance with: `docker exec -it mysqlinstance mysql -uroot -p'pa$$w0rd'`.

10. When the MySQL command line opens, create the database using: `CREATE DATABASE MedVoyagedb;`. After creation, exit.

Django Setup:

11. Execute `python src/manage.py makemigrations` to create tables and schema based on models in the MySQL database.

12. Apply these migrations using: `python src/manage.py migrate`.

Running the Program:

13. To launch the Django web server, use the command: `python src/manage.py runserver`. This starts the server, and we can view the app at `localhost:8000/`.

Deliverable 4 Report

Team Squad 8

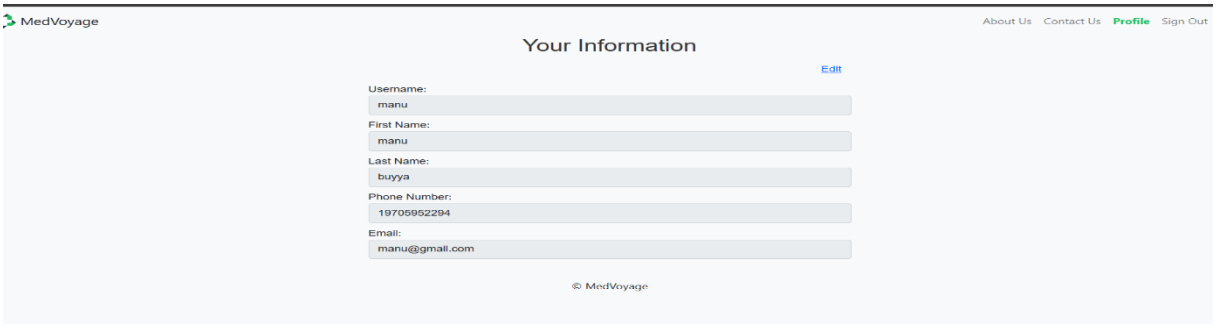
Running Test Cases:

14. For executing unit test cases, run: `python src/manage.py test src/main/tests/`.

How to use the basic functionalities of the phase two released features below:

Doctor Dashboard page: Users need to sign up with their details and credentials using the signup form that was developed in phase 1. After signing up as a doctor and filling in all your details and once you login using the same login credentials as doctor, the user will be directed to a doctor's dashboard as shown below. From here, the doctor will be able to see their upcoming appointments, previous appointments, that will be added later. They can sign-out using the sign out button on the top right corner of the page or will be directed to the profile page when clicked on the profile button on the nav bar of the web page.

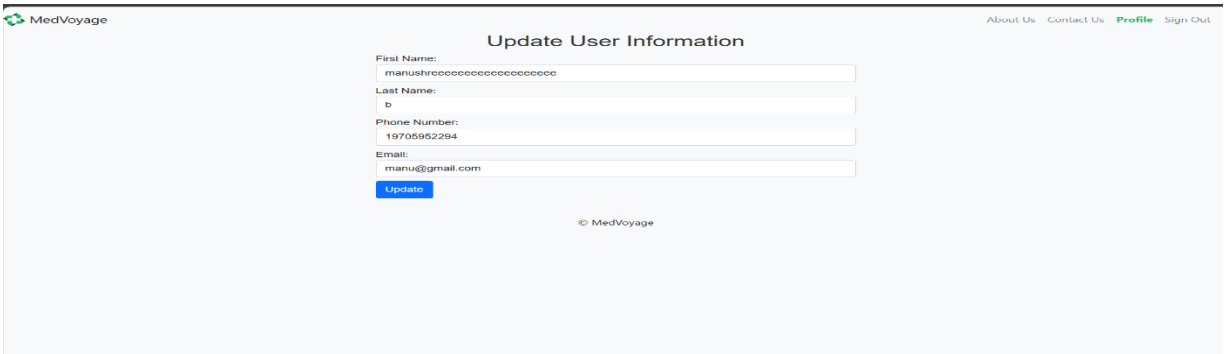
Doctor Profile page: When the doctor logs in with proper credentials and chooses to view the profile page using the button on the nav bar from their dashboard, they will be able to see their details which is only a read only form. There is an edit button enabled in the form which when prompted by the user will direct the user to another update from where the doctor will be able to view the fields and edit them accordingly and update the details accordingly.



Doctor Update Form: When the doctor clicks on the edit button in the doctor [profile page, he will be directed to the update form where the doctor will be able to edit the fields in this page and they will be updated accordingly on the profile page as well after clicking on the update button.They will be directed automatically to the profile page that shows profile with updated changes.

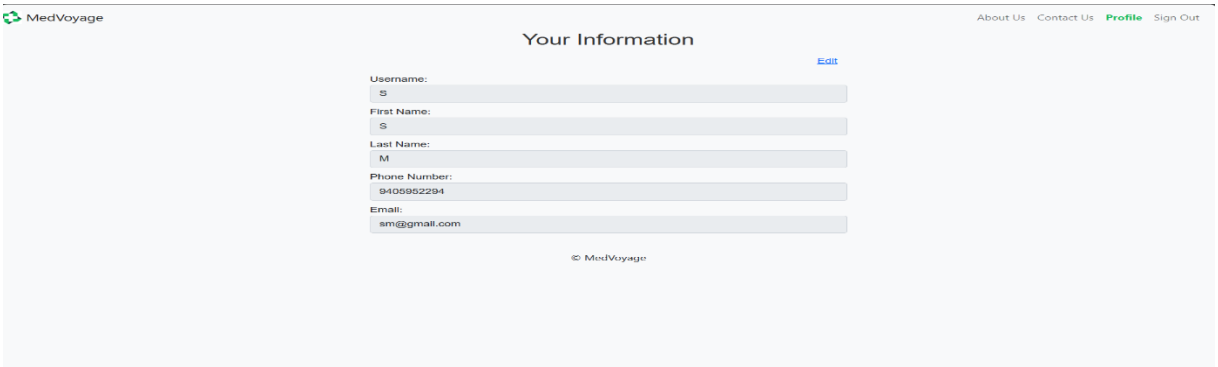
Deliverable 4 Report

Team Squad 8



Client Dashboard page: Users need to sign up with their details and credentials using the signup form that was developed in phase 1. After signing up as a patient and filling in all your details and once you login using the same login credentials as doctor, the user will be directed to a client's dashboard as shown below. From here, the client will be able to see their upcoming appointments, previous appointments, that will be added later. They can sign-out using the sign out button on the top right corner of the page or will be directed to the profile page when clicked on the profile button on the nav bar of the web page.

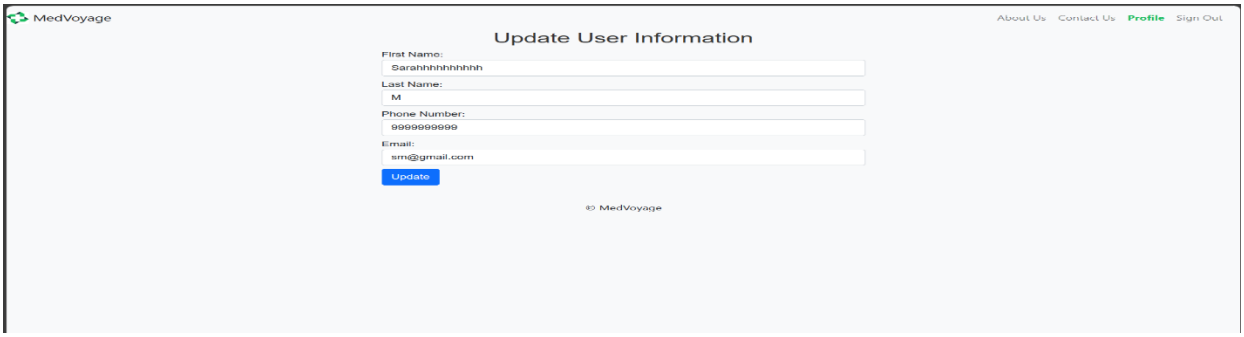
Client Profile page: When the client logs in with proper credentials and chooses to view the profile page using the button on the nav bar from their dashboard, they will be able to see their details which is only a read only form. There is an edit button enabled in the form which when prompted by the user will direct the user to another update from where the patient will be able to view the fields and edit them accordingly and update the details accordingly.



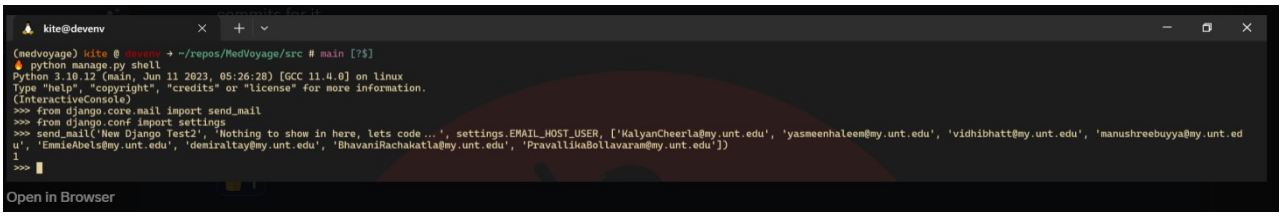
Deliverable 4 Report

Team Squad 8

Client Update form: When the patient clicks on the edit button in the client profile page, he will be directed to the update form where the patient will be able to edit the fields in this page and they will be updated accordingly on the profile page as well after clicking on the update button. They will be directed automatically to the profile page that shows profile with updated changes.



Backend Email Integration: The image below illustrates the backend of the email notification system, a newly integrated feature that MedVoyage has developed. This allows a notification system as displayed in the image below with the specific email addresses. This allows the users to confirm their signing up via a notification in their respective email provided while signing up. Once again, this was integrated into the MedVoyage web application to foster usability and responsiveness.

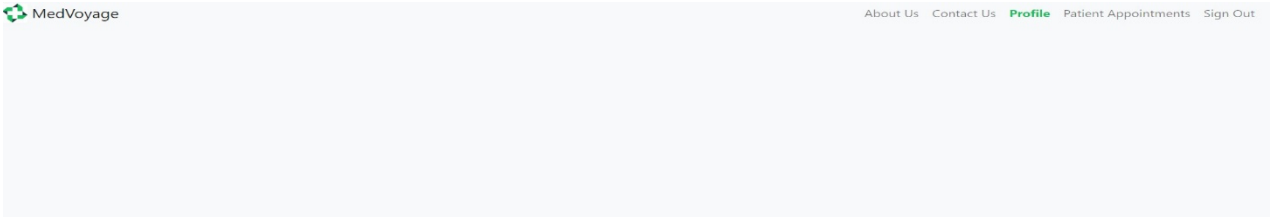


Backend/Frontend Patient Past and Upcoming Appointments Display/Store integration: This system permits storing of any past and upcoming appointments the patient has made with information including their name, date, time of the appointment, as well as the doctor they booked an appointment with. The first figure demonstrates how the “Patient Appointments” button is unavailable when the user has not logged in, to ensure security and privacy of the patient.

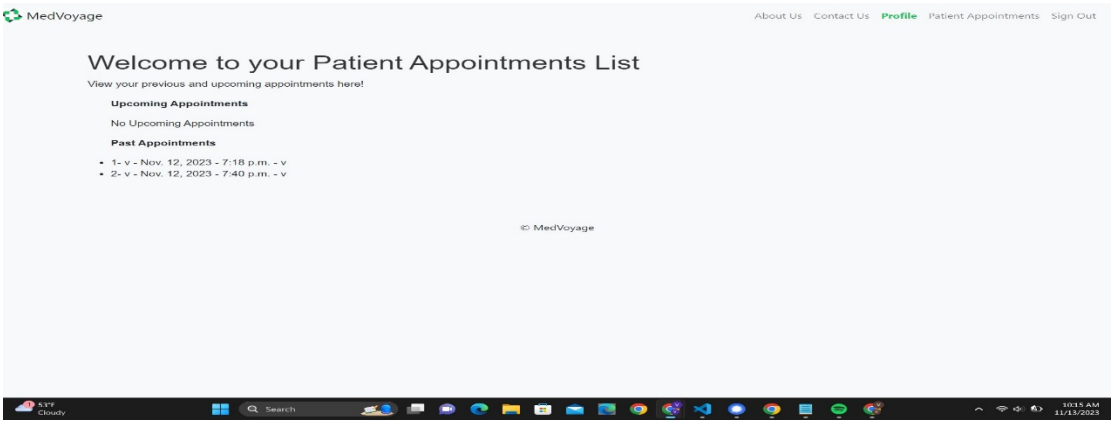
Deliverable 4 Report

Team Squad 8

This image illustrates the “Patient Appointments” button in the top right corner only displaying on the screen once a user (patient) has logged in.



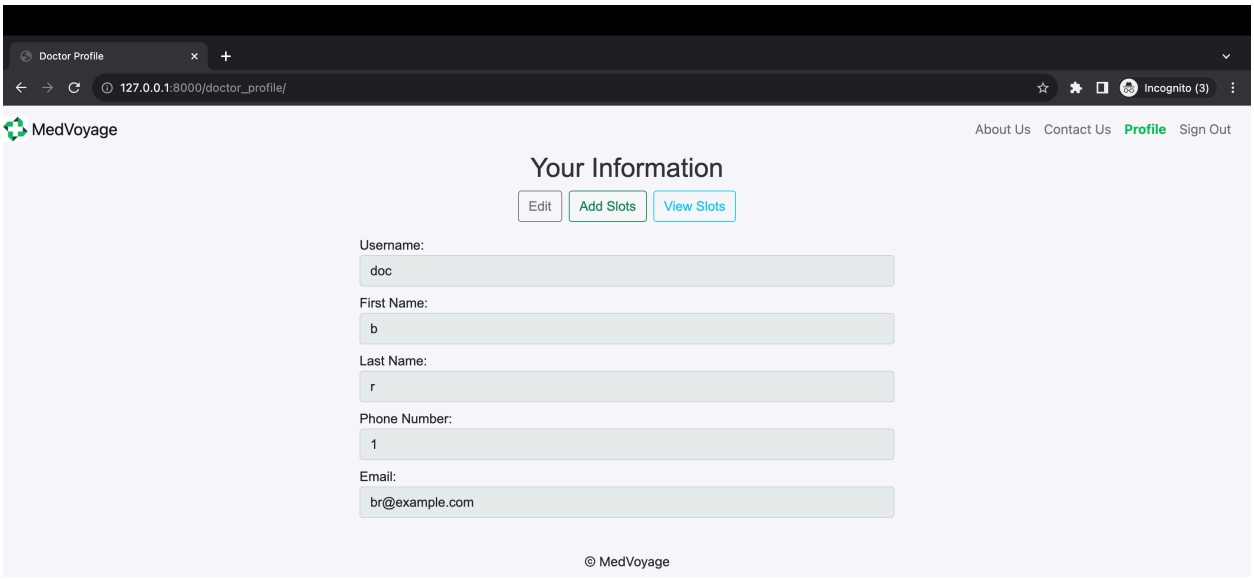
As the patient clicks on the “Patient Appointments,” the below page is displayed that lists only the logged in and no other user’s past and upcoming appointments. For example, to test this, a patient account, “v” was made in the image below and then “v” made two appointments on November 12 on those selected times listed with the doctors also named “v” just to test this feature. Now, the patient names, dates, times, and doctor’s names are listed under “Past Appointments” because these appointments in the picture were made on November 12, and this screenshot of the screen has been taken on November 13 (bottom right corner). This confirms storing, saving, and listing past and upcoming appointments for the given patient, along with the display of “No Upcoming Appointments” if the patient hasn’t made any future appointments. Once the user signs out, the “Patient Appointments” button disappears

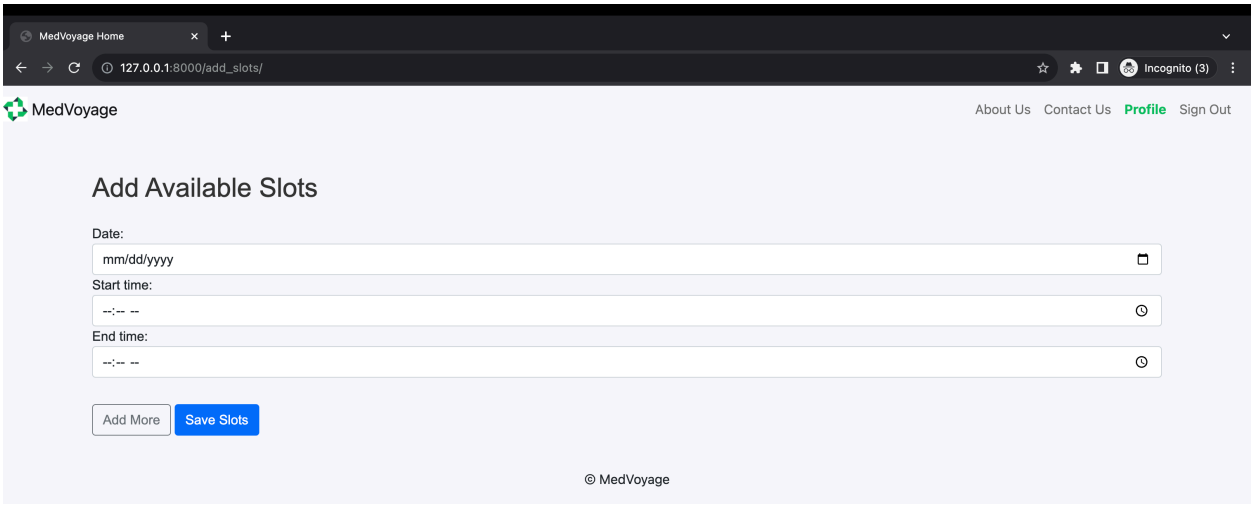


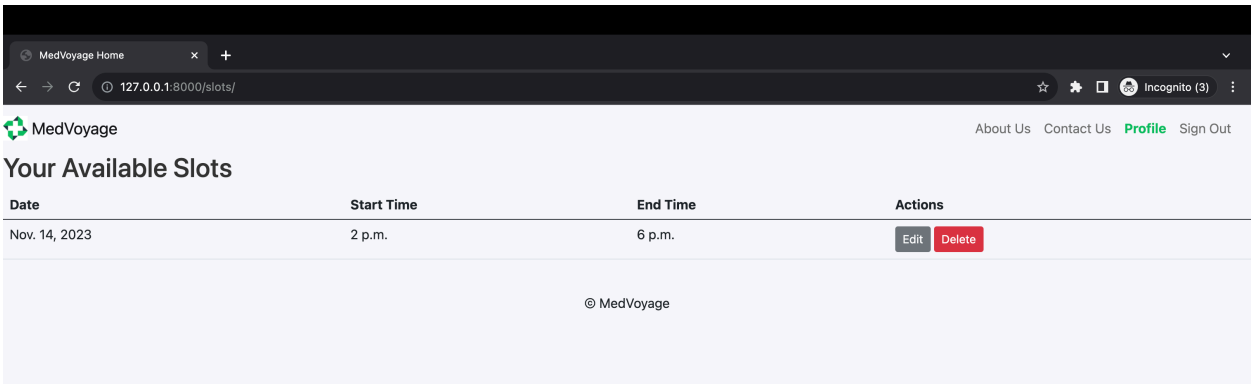
Doctor Availability Tracking: This feature allows doctors to add, edit, and delete available slots for appointments. This includes specifying the date, start time, and end time for each slot. The system feature is intended to streamline the process of booking and managing appointments, providing a clear and organized view of availability. Also, the inclusion of success messages and confirmations for actions like adding or deleting slots enhances the user experience by providing immediate feedback on actions taken within the system. The below image illustrates the mentioned features:

Deliverable 4 Report

Team Squad 8

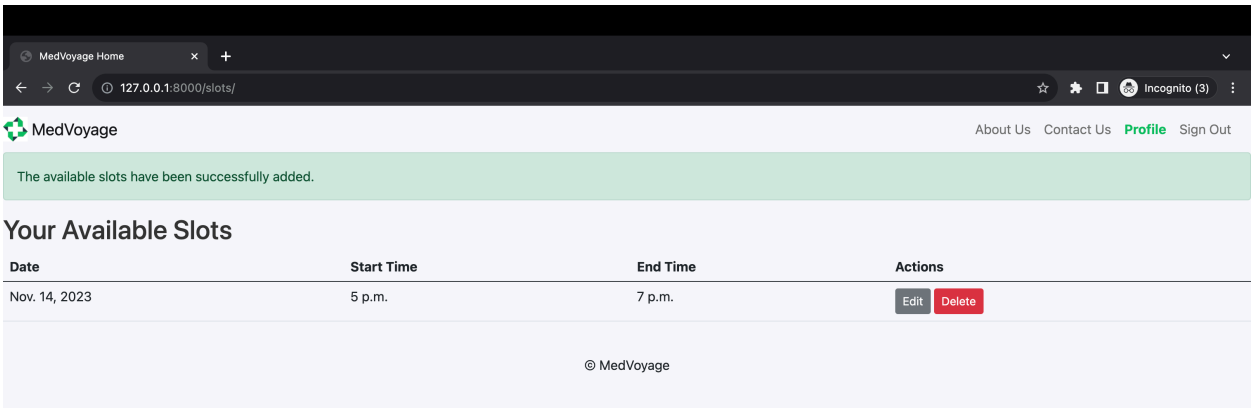


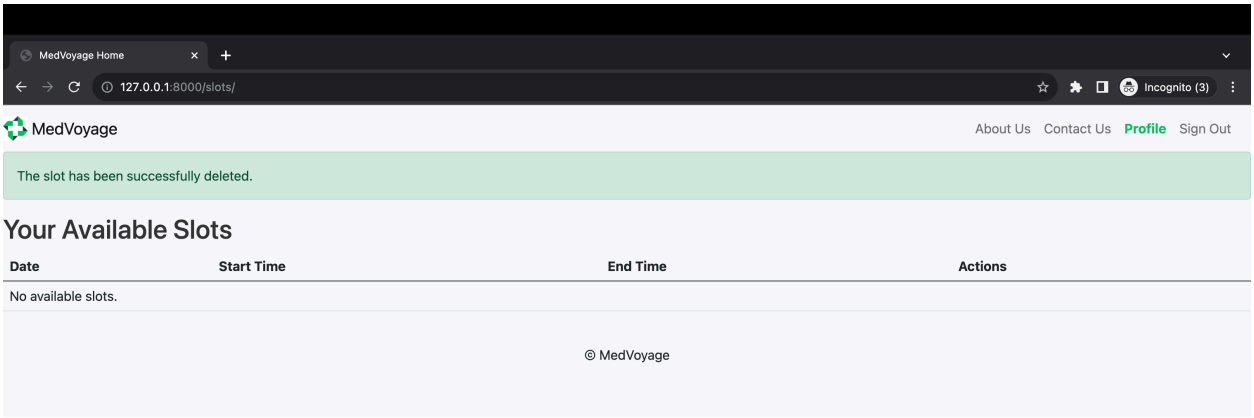




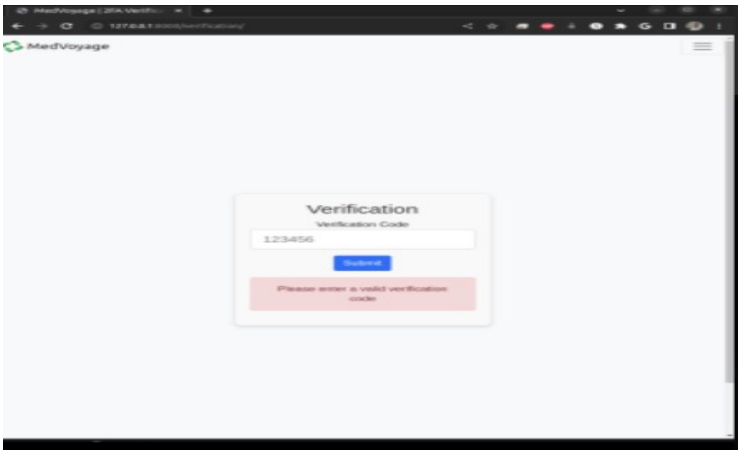
Deliverable 4 Report

Team Squad 8



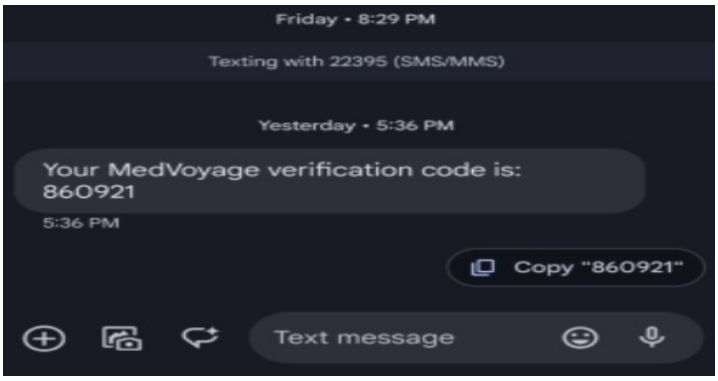


Backend Two-Factor Authorization integration: The below images display the integration of a verification system through text. This is turned on by switching the flag switch in the settings.py file as on. This turns on the two-factor verification system that sends a text message code. The verification system only accepts the correct code received by the given user. Once the correct verification code is inserted in the page, the user is verified. Overall, the system serves as an additional tool for privacy and security where important data such as patient appointment data is stored.



Deliverable 4 Report

Team Squad 8



F. Peer Review Feedback:

During the code inspection workshop, we reviewed each other's code in this meeting and discussed the approaches to try implementing them within our team. During this inspection, we discussed the phone number field, contact us page enquiry, Developer section enquiry. We were suggested that in our code in the phone field of patient model, we had blank=true; they indicated that it would be better not to keep Blank=true in the code. The second topic that was discussed was about the Developer's section that was mentioned in the code, where all the team members were mentioned. We were also confronted about the Contact Us page regarding the details like phone numbers and emails that the users could contact. According to them, rest of the part of code's functionality was good and up to the mark.

Actions Based on Feedback:

According to our team, the input received from the other team mostly came as clarifications rather than any modifications or changes. The inputs that we received were about the phone number field, according to what we discussed after the input that this was a part of non-functional requirement, so we decided to push it to the upcoming phase. The other two discussions were concerns related to contact us and developer's section where we thought the users might want to know about the developers all the team members of squad-8 were mentioned, so we clarified that all team members were given developer roles as all are contributing to the development of the MedVoyage application. For the phone number and email part, we mentioned integrating emails and seeing how it goes further.

Accepted:

We accepted the review to modify our phone number field that we plan to implement in the next phase of implementation,

Rejected:
There are no suggestions that were rejected.

Report Reflection:

Throughout the implementation phase, the team has mostly made good progress on the features planned to implement. MedVoyage has implemented a two-factor authorization, Doctor, and Patient Profile, that the

team was unable to implement on the last deliverable was finished in this deliverable phase. Also, they have successfully implemented patients being able to view their past and upcoming appointments lists that include date, time, patient, and doctor names, along with email integration, data table addition for appointments, and doctor slots addition and deletion. The planning stage for these features revolved around utilizing Trello and attempting to divide tasks as evenly and smoothly as possible. MedVoyage's progress was quite significant and valuable from the last deliverable to this one, as the team implemented features that permit user integration to the web application. As the complexity of some of the features increased from the last phase, some features are still currently in progress. The team recognizes it is important to finish assigned feature implementation, and MedVoyage plans to significantly improve this in the next phase by enhanced collaboration. The team hopes to increase communication and establish a strong background for Django as currently many of the developers are beginners. In addition to this, the team is dedicating resources to learning and helping any other developers in need through either peer programming or aiding in debugging, so that all our features in the next phase are finished in a timely manner. MedVoyage hopes to learn various Django frameworks and successfully implement and apply them to the web application. Overall, MedVoyage is committed to continually working on their technical skills and communication to smoothly develop in the next phases.

H. Member contribution table (should describe who wrote what components or classes of the system and what parts of the report).

Member Contribution Table:

Member	Contribution and Overall / Contribution(percentage)
Kalyan Cheerla	Implementing the appointment tracker and calendar widget and worked on the backend of the appointment booking system and worked on domain name for MedVoyage. Collaborated with Pravallika to develop the Doctor Model. Worked on setting-up GCP instance and domain settings for our MedVoyage webapp. Reviewed almost all Pull-requests and worked alongside team to streamline the project changes. Collaborated with Manushree on Doctor profile changes.
Yasmeen Haleem	Worked in implementing calendar widget and appointment booking and drew the use case diagrams, class diagram and sequence diagrams and uploaded the meeting minutes.
Bhavani Rachakatla	Implemented the Doctor Availability Feature that has the sub features of Adding one or more time slots, Edit/Update the time slots, Deletion of already recorded time slots. Have also implemented a feature called View Slots that shows all the recorded Slots for a Doctor. Have contributed to the DB Table Schema and added Doctor Availability feature's test cases.
Manushree Buyya	Implemented the doctor profile page and basic dashboard page for doctors Also implemented the profile update form for doctors within the profile page. Drafted few sections in the report.
Pravallika Bollavaram	Implemented the Patient Dashboard which involves profile page for patients and update form for modify.

Deliverable 4 Report

Team Squad 8

	On the backend, I contributed to the development of database schema and collaborated with Kalyan to develop the Doctor Model .
Vidhi Bhatt	-Finished implementation of a "Patient Appointments" button that only renders once the user (patient) has signed up and logged in, ensuring privacy. -Finished implementation of a page for patients to view and store their past appointments lists with important details of the appointment and introduced a date and time model to ensure past/future appointments created by the user are saving under the correct appointment list -Finished implementation of a URL on the same page to view logged in patient's upcoming appointments lists enhancing user accessibility -Finished implementation and created an admin portal until the appointment booking system feature is finished to create test appointments to ensure that appointments result in successful storing and display output -Finished implementation of unit test cases for both upcoming and past appointments storing and display -Completed user manual instructions for email integration, patient past/upcoming appointment display/store integration, 2FA integration, report reflection, edited formatting and inserted more test cases in the document
Demir Altay	Worked on figuring out a way to do two factor authentication, implemented it using a 3rd party called twilio. Tested it extensively with different configurations and phone numbers. Helped fix some bugs and build a table for the cancel appointment page as well
Emmie Abels	Worked on creating the cancel appointment page and basic functionality and implemented the doctor view schedule page with most of its functionality. Also implemented a basic appointment booking schedule that works with the cancel appointment and view schedule features.

Percentage contribution table for Deliverable 4:

Member	**Overall / Contribution(percentage)**
Kalyan Cheerla	12%
Yasmeen Haleem	11.5%
Bhavani Rachakatla	13%
Manushree Buyya	13%
Pravallika Bollavaram	14%
Vidhi Bhatt	13%
Demir Altay	12%
Emmie Abels	11.5%