

## Deliverable 3 Report

Team Squad8

### A. Phase 1 Requirements' Status:

#### Requirements:

| Section No. | Requirement Name   | Status     | Description  |
|-------------|--|------------|--|
| 3.1.1       | Front Landing page   | Finished   | Introduces the MedVoyage online appointment booking system   |
| 3.1.2       | About us page  | Finished   | Gives the overview and details of MedVoyage application and its creators   |
| 3.1.3       | Contact us page  | Finished   | Provides contact information and email addresses of the MedVoyage team for any user to contact the team                                  |
| 3.1.4       | Navigation bar   | Finished   | Implementation of a structured navigation bar  |
| 3.1.5       | Mobile user-friendly design                                    | Finished   | Ensures that the platform is mobile friendly   |
| 3.1.6       | Custom Error pages (for HTTP 400, 300, 500 status codes)       | Yet to do  | Designated error pages for specific HTTP error codes   |
| 3.1.7       | Configure the Database   | Finished   | Setup of the MySQL database  |
| 3.1.8       | Unified signup form  | Finished   | Single registration form with essential user details like login-type, first and last names, email, password, contact information         |
| 3.1.9       | Unified Login form   | Finished   | Centralized login portal with user-specific (here patient/user, doctor) login type   |
| 3.1.10      | Handle basic different authorizations for patients and doctors | Finished   | Implement distinct access controls for patients and doctors  |
| 3.1.11      | Authorization DB integration                                   | Finished   | Synchronizing login and signup data with MySQL database  |
| 3.1.12      | Client profile update form                                     | In process | Interface for clients/patients to modify their details   |
| 3.1.13      | Doctor profile update form                                     | In process | Interface for doctors to modify their details  |
| 3.1.14      | Forgot Password Feature  | Finished   | A recovery process for users to change their forgotten passwords by clicking on the link that redirects them to the change password form |

## Deliverable 3 Report

Team Squad8

|        |                      |          |   |
|--------|----------------------|----------|---|
| 3.1.15 | Sign Out page        | Finished | Dedicated logout functionality for users which only appears if a given user is logged in  |
| 3.1.16 | Password change form | Finished | Allows users to update their account password to a new password once they enter the correct answer to the security question answer they entered when signing up |
| 3.1.17 | Webpage Footer       | Finished | Section at the bottom displaying copyright information for MedVoyage  |
| 3.1.18 | GitHub CI/CD work    | Finished | Make use of GitHub actions to run builds for Django Unit testcases  |

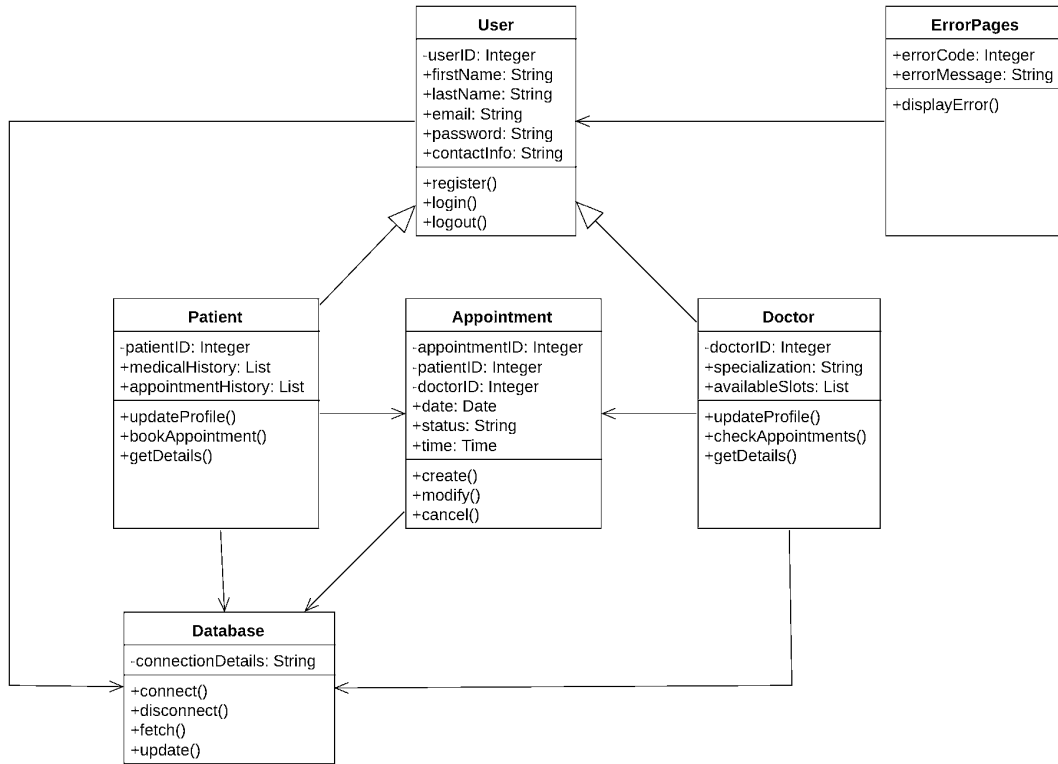
We have largely completed our Phase 1 goals. Key features like the Front Landing Page, About Us, and Contact Us pages are live, along with a mobile-friendly design and a structured Navigation Bar. Backend tasks, including MySQL database setup, user authorization, user sign up, forgot password, and resetting password are also finished. Unified signup and login forms for different user types—patients and doctors—are operational. Some work is still pending, such as custom HTTP error pages and doctor profile update forms. Security features and GitHub CI/CD pipelines to trigger Unit testcases are also in place. Overall, we have made significant progress in both front-end and back-end development. Our significant progress and improvement involve backend and frontend for sign in, log in, sign out, forgot password, and reset/change password, along with MySQL set up that includes changes as suggested by the peer reviews.

# Deliverable 3 Report

Team Squad8

## B. UML design for phase 1:

### 1. Class Diagram for MedVoyage Phase 1 Deliverable:

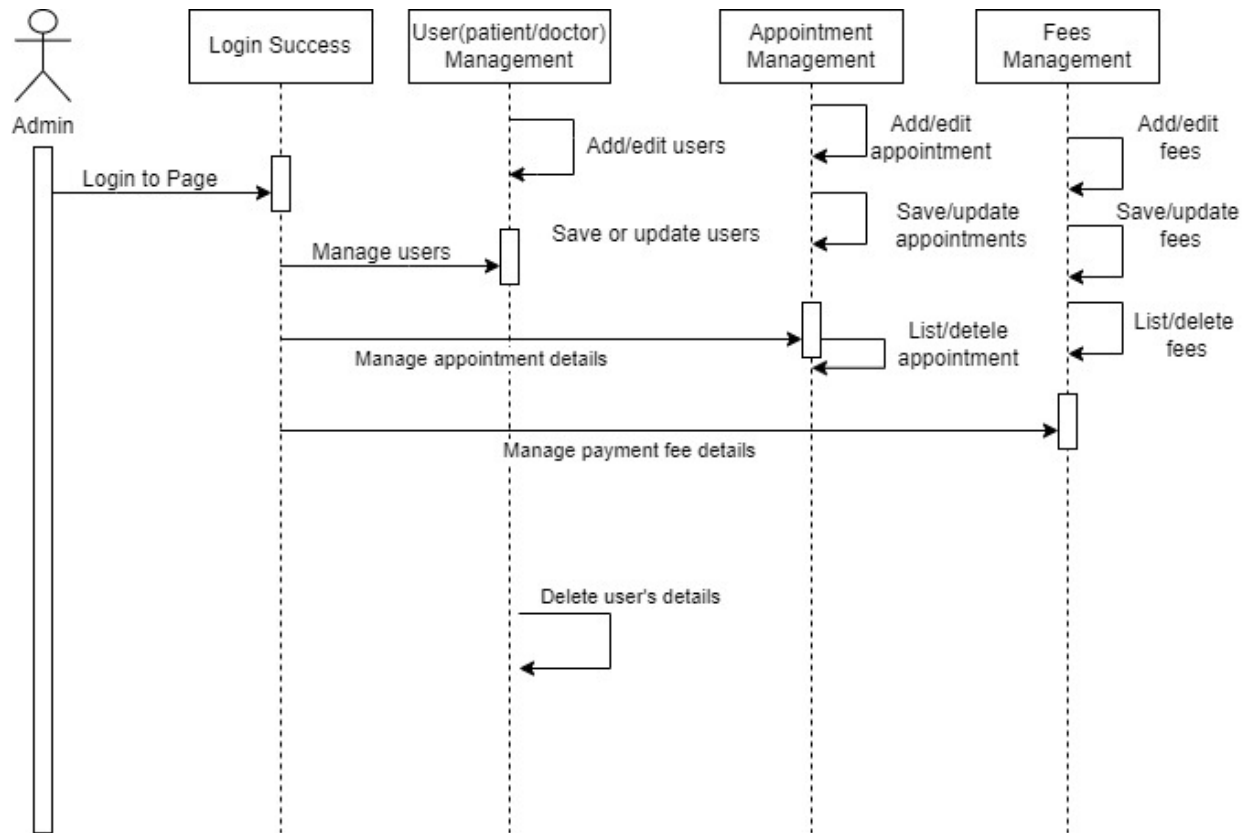


- a. Class Diagram showing showcasing classes, their attributes, operations, and relationship between the various classes

# Deliverable 3 Report

Team Squad8

## 2. Sequence diagram for overall MedVoyage :

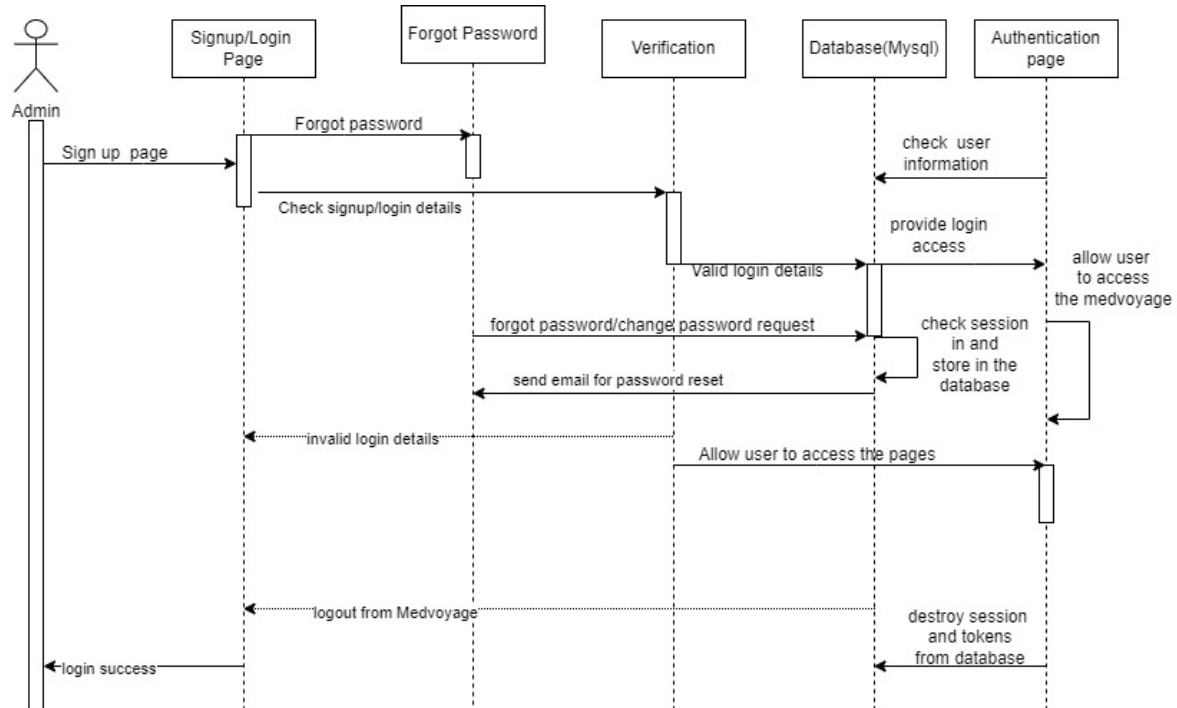


b. Sequence Diagram showing the time-ordered interactions between objects using messages.

# Deliverable 3 Report

Team Squad8

## 3. Sequence diagram for the phase 1 MedVoyage System:

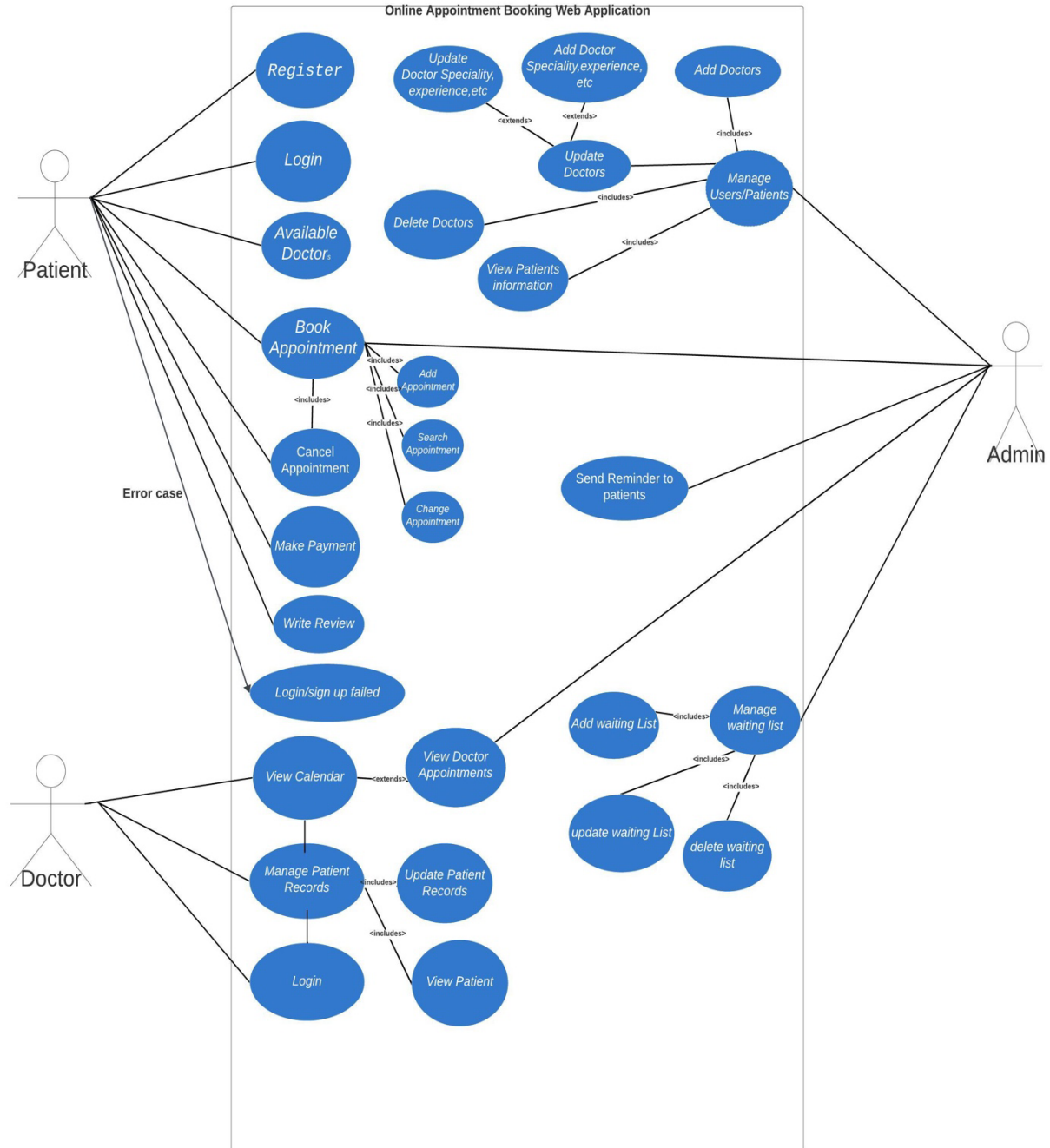


- c. Sequence Diagram showing the time-ordered interactions between objects using messages for the Overall MedVoyage System.

# Deliverable 3 Report

Team Squad8

## 4. Use Case Diagram for MedVoyage - Online Appointment Booking Web Application.



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

## Deliverable 3 Report

Team Squad8

### 5. Different use cases for MedVoyage application :

#### Use case 1:

|                                    |   |
|------------------------------------|---|
| Use Case Number                    | UseCase-01  |
| Use Case Name                      | Registration / Sign up page for Patients and Doctors  |
| Overview                           | The Users (Patients/Doctors) shall start entering the registration information on the MedVoyage web application   |
| Pre-condition(s)                   | <p>The Users have arrived at the MedVoyage web application through the URL</p> <ul style="list-style-type: none"><li>• User database is properly configured. A separate table for patients and a separate table for doctors</li><li>• Web server is open, and the host is waiting for registrations of patients/doctors.</li><li>• The new users have access to the MedVoyage web application via url</li></ul>   |
| Scenario Flow Main (success) flow: | <ol style="list-style-type: none"><li>1.User (patient/doctor) selects the signup option instead of login.</li><li>2.The MedVoyage system prompts the user(patient/doctor) for their personal information, including: Username, Password, and Email.</li><li>3.The user(patient/doctor) enters the requested information.</li><li>4.MedVoyage system verifies if the requested information is within the pre-specified constraints.<ol style="list-style-type: none"><li>a.If constraints are violated, Display error message</li><li>b.Return to Step 2</li></ol></li><li>5.User(patient/doctor) inputs the information.</li><li>6.MedVoyage system verifies the information.<ol style="list-style-type: none"><li>a.If information is invalid, display an error message specifying instructions.</li><li>b.Return to Step 6</li></ol></li><li>7.MedVoyage system displays the confirmation of signup</li></ol> |
| Alternate Flows                    | <ol style="list-style-type: none"><li>1)The user confirms the MedVoyage system suggestion.</li><li>2)The MedVoyage system returns to step (7).</li></ol>  |
| Post Condition                     | New user(patient/doctor) information has been successfully added to the MYSQL database.   |

## Deliverable 3 Report

Team Squad8

### Use case 2 :

|                                    |  |
|------------------------------------|--|
| Use Case Number                    | UseCase-02   |
| Use Case Name                      | Login  |
| Overview                           | The users(doctors/patients) shall login with their username and password.  |
| Pre-condition(s)                   | The user(doctors/patient) has already completed registration/sign up page  |
| Scenario Flow Main (success) flow: | <p>Main (success) flow:</p> <ol style="list-style-type: none"><li>1.The user(doctor/patient) connects to the MedVoyage system.</li><li>2.The user(doctor/patient) selects the option to login</li><li>3.MedVoyage System prompts the user(doctor/patient) to enter credentials.</li><li>4.The user(doctor/patient) enters credentials.</li><li>5.MedVoyage System validates credential information is correct.</li><li>a.If the information is incorrect, display an error message.</li><li>b.MedVoyage System returns to step (3)</li><li>6.Med voyage System redirects users to About us page.</li></ol> |
| Alternate Flows                    | <p>At any step, the user(doctor/patient) can instead click the cancel button.</p> <ol style="list-style-type: none"><li>1)Return to homepage(About Us page )</li></ol>   |
| Post Condition                     | MedVoyage System redirects users(doctors/patients) to their Manage User Account.   |



## Deliverable 3 Report

Team Squad8

### C. Test Cases (unit tests) for phase 1:

| Test Function Name   | Test Description  | HTTP URL  | HTTP Method | Expected HTTP Response Code | Expected Outcome   |
|----------------------|---|-----------|-------------|-----------------------------|--|
| test_home_title      | Tests if the homepage title is correctly rendered.                    | /         | GET         | 200                         | Contains: "MedVoyage  Home"                                  |
| test_home_nav        | Tests if the navbar on the homepage contains the correct fields.      | /         | GET         | 200                         | Contains navbar fields                                       |
| test_footer          | Tests if the footer content is present on the homepage                | /         | GET         | 200                         | Contains "&copy; MedVoyage"                                  |
| test_content_display | Tests specific content display on the homepage                        | /         | GET         | 200                         | Contains 'MedVoyage' and the specific journey-to-health text |
| test_about_title     | Tests if the About Us page title is correctly rendered.               | /about/   | GET         | 200                         | Contains: "MedVoyage  About Us"                              |
| test_about_nav       | Tests if the navbar on the About Us page contains the correct fields. | /about/   | GET         | 200                         | Contains navbar fields                                       |
| test_aboutus_content | Tests if the About Us content is correctly rendered on the page.      | /about/   | GET         | 200                         | Contains About Us content                                    |
| test_about_dev_list  | Tests if the developer list and their details on the About Us page.   | /about/   | GET         | 200                         | Contains developers' details                                 |
| test_contact_title   | Tests if the Contact Us page title is correctly rendered.             | /contact/ | GET         | 200                         | Contains: "MedVoyage  Contact Us"                            |

## Deliverable 3 Report

### Team Squad8

|                                     |   |           |      |     |                                |
|-------------------------------------|---|-----------|------|-----|--------------------------------|
| test_contact_nav                    | Tests if the navbar on the Contact Us page contains the correct fields.         | /contact/ | GET  | 200 | Contains navbar fields         |
| test_contact_footer                 | Tests if the footer on the Contact Us page is correctly rendered.               | /contact/ | GET  | 200 | Contains footer copyright      |
| test_contact_details                | Tests if the Contact Us details are correctly rendered on the page.             | /contact/ | GET  | 200 | Contains Contact Us details    |
| test_contact_message                | Tests if the message form fields are correctly rendered on the Contact Us page. | /contact/ | GET  | 200 | Contains message form fields   |
| test_signup_form_rendered_correctly | Tests if the Signup form is rendered correctly.                                 | /signup/  | GET  | 200 | Rendered 'signup.html'         |
| test_form_labels_and_placeholders   | Tests if labels and placeholders for the Signup form are rendered correctly.    | /signup/  | GET  | 200 | Contains labels & placeholders |
| test_form_errors_displayed          | Tests if the form displays errors when provided with incorrect data.            | /signup/  | POST | 200 | Contains form errors           |
| test_form_valid_submission          | Tests if the form redirects after a valid submission.                           | /signup/  | POST | 302 | Redirect action                |
| test_login_type_field_rendered      | Tests if the login type fields are rendered correctly on the Signup page.       | /signup/  | GET  | 200 | Contains login type fields     |
| test_submit_button_rendered         | Tests if the Signup form submit button is rendered correctly.                   | /signup/  | GET  | 200 | Contains submit button         |

## Deliverable 3 Report

Team Squad8

|                            |   |   |          |          |   |
|----------------------------|---|---|----------|----------|---|
| test_login_content         | Validates the content of the login page, including navbar, footer, and login form fields.       | /login/   | GET      | 200      | Page content matches expected elements and fields.              |
| test_login_success         | Tests successful user registration, login, and sign-out process.                                | /login/   | POST     | 302      | Redirect to /home/ after successful login.                      |
|                            |   | /signup/  | POST     | 302      | Redirect after successful signup.                               |
|                            |   | /home/  | GET      | 200      | Page content matches expected elements and fields after login.  |
|                            |   | /signout/                                       | GET      | 302      | Redirect to /home/ after signing out.                           |
| test_login_failure         | Tests login process with an incorrect password.   | /login/   | POST     | 200      | Page content matches expected elements and fields with failure. |
| test_login_without_signup  | Tests login process without prior signup.   | /login/   | POST     | 200      | Page content matches expected elements and fields with failure. |
| test_resetpassword_success | Tests successful password reset flow including signup, login, logout, reset password, and login | Various URLs [signup, login, home, signout, and | POST/GET | 302, 200 | Navigation and authentication success                           |

## Deliverable 3 Report

Team Squad8

|                                      |   |   |          |          |   |
|--------------------------------------|---|---|----------|----------|---|
|                                      |   | reset_password]   |          |          |   |
| test_reset_password_content          | Tests content of the reset password page                | /reset_password/  | GET      | 200      | Proper content displayed on reset password page |
| test_resetpassword_wrong_secquestion | Tests password reset with wrong security question       | /reset_password/  | POST     | 200      | Login with new password should fail             |
| test_resetpassword_short_success     | A shorter version of the successful password reset flow | Various URLs [signup, login, home, signout, and reset_password] | POST/GET | 302, 200 | Navigation and authentication success           |

**D. A user manual that tells us how to install/use your program. This is meant for the end-user of the software. (more of the user manual is under “*How to use the basic functionalities of the MedVoyage Web-Application*” section)**

### 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

# Deliverable 3 Report

Team Squad8

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>.

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:

## Deliverable 3 Report

Team Squad8

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.

```
PS C:\Users\Manushree> docker exec -it mysqlinstance mysql -uroot -p'pa$$w0rd'
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 19
Server version: 8.1.0 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> |
```

```
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
```

## Deliverable 3 Report

Team Squad8

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.

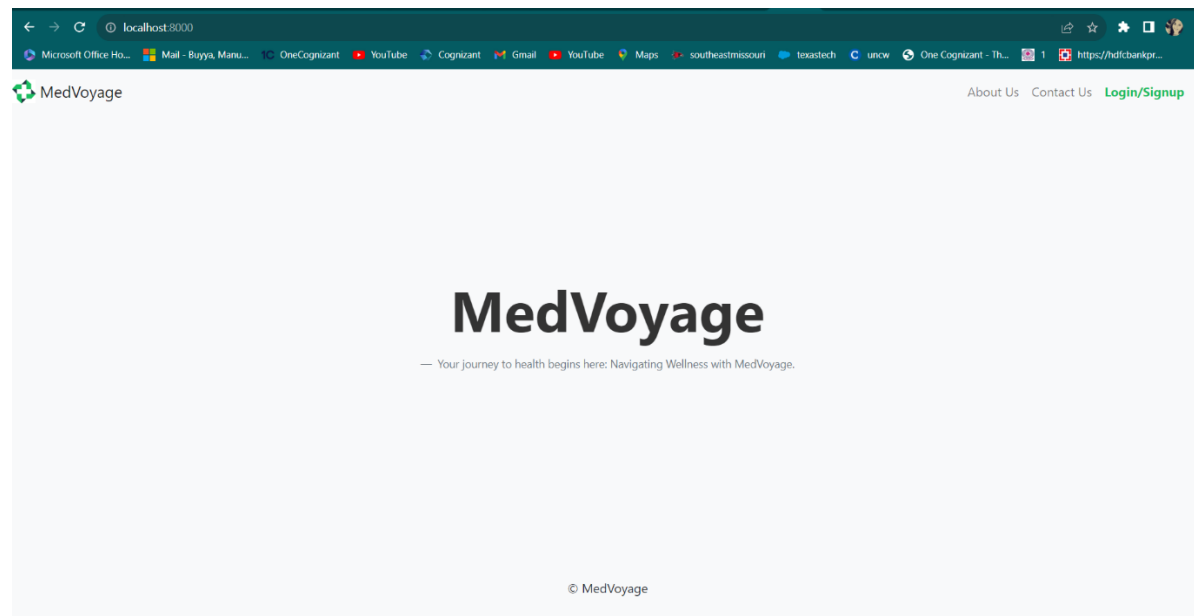
### Running the Django webserver:

`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

```
PS C:\Users\Manushree> cd C:\Users\Manushree\Desktop\SE\MedVoyage\src
PS C:\Users\Manushree\Desktop\SE\MedVoyage\src> python .\manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
October 22, 2023 - 22:30:37
Django version 4.2.6, using settings 'medvoyage.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```



### Running unit testcases

# Deliverable 3 Report

Team Squad8

`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``.



# Deliverable 3 Report

Team Squad8

## Running the Program:

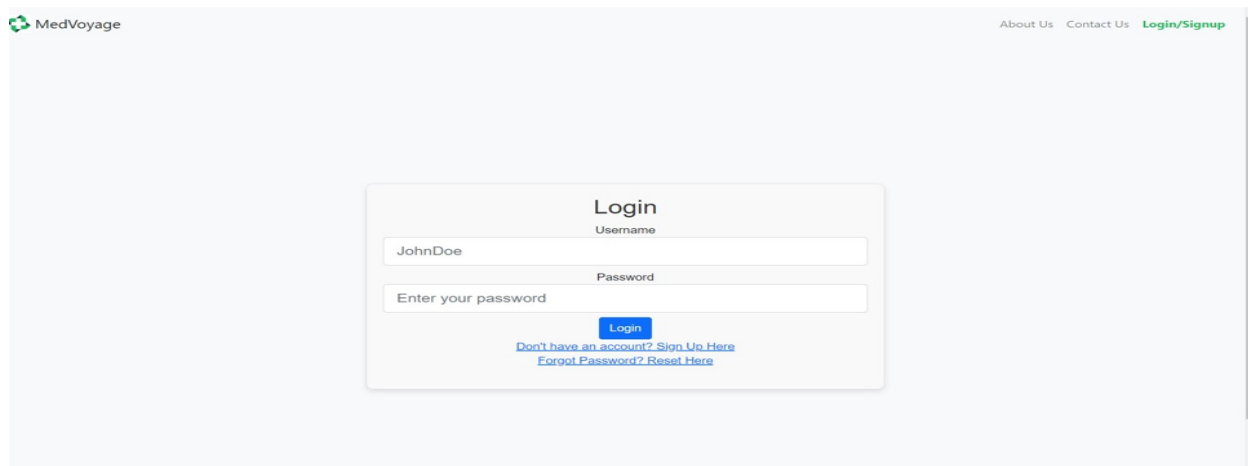
13. To launch the Django web server, use the command: `python src/manage.py runserver`. This starts the server, and we can view the app at `localhost:8000/`.

## 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 MedVoyage Web-Application from phase one of implementation:

**Log In Page:** Users can insert username and password, and if they don't have an account, they can click on the sign-up page link below the login button. If they don't remember their password and have already signed up for the web-application, they can click on the forgot password link below the don't have an account link. The sign out button is not seen on the log in page until the user signs in to prevent confusion and allow the website to be user-friendly. As a user cannot sign out until they have logged in, the sign out button should not be there in the initial logging in at the top right corner.



MedVoyage

About Us Contact Us Login/Signup

### Login

Username

JohnDoe

Password

Enter your password

Login

[Don't have an account? Sign Up Here](#)

[Forgot Password? Reset Here](#)

# Deliverable 3 Report

## Team Squad8

**Sign Up Page:** Users can insert their desired username, password, their personal info, and answer the security question to change their password in case they forgot their sign in password.

MedVoyage About Us Contact Us Login/Signup

### Sign-Up

Username:  
JohnDoe

First Name:  
John

Last Name:  
Doe

Email:  
John@email.com

Phone:  
555-555-5555

Security Question:  
Enter what city you grew up in

Password:  
Enter password

Confirm Password:  
Confirm password

☐ Patient  
☐ Doctor  
☐ Staff

Sign-Up

© MedVoyage

**Reset/Change Password Page:** Users can click on the forgot password feature on the sign in page to be redirected to this change password page. This allows users to enter their username, new password they would like to change their account to, and the security question answer that the user entered while signing up to ensure the correct user has access and can change the password. Once clicked submit, the user's password is changed to the new one and the page redirects to the log in page, where the user can easily sign in with the new password, they reset utilizing this form.

MedVoyage About Us Contact Us Login/Signup

### Reset Password

Username  
JohnDoe

New Password  
Enter your new password

Security Question  
Enter your security question answer

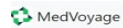
Reset

[Don't have an account? Sign Up Here](#)

**Sign Out:** this feature is only available in the top right corner when the user logs in as it does not serve a purpose when no log in has happened. Once the user logs in, the sign out button pops up allowing successful sign out and redirection to the log in page.

# Deliverable 3 Report

Team Squad8



[About Us](#) [Contact Us](#) [Login/Signup](#) [Sign Out](#)

## MedVoyage

— Your journey to health begins here: Navigating Wellness with MedVoyage.

© MedVoyage

### **F. Peer Review Feedback:**

During the peer review session, we reviewed each other's features to be implemented and discussed approaches to implement them. Email verifications during registration, automatic session timeouts, and website language accessibility features were discussed. We appreciated adhering to the documentation format. We discussed queries about how doctors manage appointments, session timeouts, and tools for performance testing. We received suggestions for implementing an email verification system, version control details, and language support for non-English speakers.

### **Actions Based on Feedback:**

Our team believes in a state like Texas, most spoken languages include English and Spanish. Since our web application needs to start off just within the Dallas- Fort Worth region, we think implementing a feature that accepts these two languages would be the most ideal. For this, we plan to research various ways we can implement different languages and make it as user-friendly as possible. We implemented a system that allows a security question in the sign-up form that saves the given input, and when the user clicks on the forgot password feature, the page is redirected to a change password form where the users can enter their username, new password, and the answer to the security question they entered in the sign-up form. All these responses are saved, and the password is successfully changed to the desired new password the user sets up.

### **Accepted:**

Email verification during registration. This is to be done in the future deliverables, as some elements of this process may require monetary expense. We plan to research the best and optimum ways to ensure that we are successfully implementing these features.

### **Rejected:**

Details on doctors managing appointments, performance testing tools, language support, and in-depth functionalities. We believe these changes can be outside of the scope of the semester product, as these may require other technologies, equipment, pay, time, etc. that we currently do not possess. However, if these

# Deliverable 3 Report

Team Squad8

changes are possible in the last stage of our project where we have done enough research, have time, these changes are accepted by Dr. Do., and are feasible, we will implement them.

## **G. Report Reflection**

We've made good progress implementing most of the features for phase 1 requirements, including Home, About, contact pages, User Sign Up and log in, Forgot Password, Reset Password, and Sign Out. Throughout the first phase, the things that went well were- more participation and task assignment in the beginning of the phase to every member that resulted in this progress, but this could be further improved in the later phases by better communication, collaboration, and division of tasks based on equivalent difficulty levels. We're still working on the Client profile update form and need to implement the Doctor profile update form or email verification. We faced initial obstacles due to dependent features but overcame them by prioritizing tasks. Specifically, creating a sign-up page and login page was the initial obstacle as it took about a week to resolve. Many other programs of the same difficulty were halted because of this program which resulted in less time (one day) being dedicated to implementing other programs such as sign-out, forgot password feature, and changing password feature. In the future, it would be best to efficiently communicate and collaborate and help other developers as much and as fast as possible so other programs of the same or greater difficulty can be implemented properly and receive the same amount of time to implement. It is also important to divide tasks of the same difficulty among every developer so that no single developer is overwhelmed with more difficult tasks compared to the rest. This helps in avoiding halts in the project and smooth implementation. We're dedicating resources to developing the Client profile update form and identifying a plan for the Doctor profile update form. We're also researching email verification. We have researched ways that allow email verification when a user signs up, but this seems to have a change in some cases, so we will be figuring out cost-free ways of doing it in the future.

## Deliverable 3 Report

Team Squad8

### 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        | -Implemented Basic Django Work Setup, Navigation Bar, MySQL DB Configuration, GitHub CI/CD integration, Unit testcases for most features<br>-Peer reviewed all PRs and handled code conflicts for merging<br>-Fixed some ad-hoc bugs<br>-Updated Code Inspection Report with descriptions<br>-Provided assistance as needed for documentation   |
| Yasmeen Haleem        | Did the front end of forgot password and front landing page; Drew use case diagram and sequence diagrams for phase 1, uploaded meeting minutes and assisted in documentation and writing test cases<br>Designed MedVoyage Logo and handled Meeting minutes  |
| Bhavani Rachakatla    | -Implemented About Us, Contact Us pages<br>-Compiled whole Code Inspection Report<br>-Documentation:<br>In Deliverable 3 Report wrote<br>Section A – Requirements List<br>Section B – drew Phase 1 Class Diagram<br>Section C –Test Cases (unit tests)<br>Section E – Instructions on how to compile/run the program  |
| Manushree Buyya       | Implemented the unified signup form<br>Wrote patient model Class<br>Documentation – Section D User Manual   |
| Pravallika Bollavaram | Provided table schema for patient model,<br>Wrote unit test cases for Home and Footer Pages<br>Working Client Update Form (In progress)<br>Working on Doctor Update Form (In progress)<br>Formatted the whole document  |
| Vidhi Bhatt           | -Implemented frontend and backend for the forgotten password functionality and link, reset/change password form allowing users to change their old forgotten password, added a security question input by updating the sign-up page, and implemented a sign out functionality. These can be viewed under GitHub history.<br>-Answered major questions for the peer review feedback on actions based on feedback section such as explained why inputs were rejected in the rejected section, and why certain inputs were accepted in the accepted section, finished what could be improved section, and what went well/didn't in the report reflection section.<br>-Finally, inserted how to utilize/navigate basic functionalities with descriptions as well as images for user friendly use. |
| Demir Altay           | Implemented unit tests for the signup page, attempted to build signup page and ultimately clean up the existing sign-up page and fix some bugs  |
| Emmie Abels           | Implemented the login page and login functions  |