ClientCentral

Group 4

Owen Powell, Jordan Sharkey, Jennifer Lewis, Andrew Kuziel

**Software Design Specification**

**Document**

**Version: 1** **Date: 11/10/2022**

Table of Contents

[1 Introduction 4](#_heading=h.gjdgxs)

[1.1 Goals and objectives 4](#_heading=h.30j0zll)

[1.2 Statement of system scope 4](#_heading=h.2dlolyb)

[2 Architectural design](#_heading=h.tyjcwt) 5

[2.1 System Architecture](#_heading=h.3dy6vkm) 5

[2.2 Design Rational 5](#_heading=h.1t3h5sf)

[3 Key Functionality design](#_heading=h.4d34og8) 6

[3.1 Appointment Scheduling](#_heading=h.2s8eyo1) 6

[3.1.1 Scheduling Use Cases](#_heading=h.17dp8vu) 6

[3.1.2 Processing sequence for Appointment Scheduling](#_heading=h.3rdcrjn) 6

[3.1.3 Structural Design for Appointment Scheduling](#_heading=h.26in1rg) 6

[3.1.4 Key Activities](#_heading=h.lnxbz9) 6

[3.1.5 Software Interface to other components](#_heading=h.35nkun2) 6

[3.2 The User Profile 6](#_heading=h.1ksv4uv)

[3.2.1 User Profile Use Cases 6](#_heading=h.44sinio)

[3.2.2 Processing sequence for User Profile 6](#_heading=h.2jxsxqh)

[3.2.3 Structural Design for User Profile 6](#_heading=h.z337ya)

[3.2.4 Key Activities 6](#_heading=h.3j2qqm3)

[3.2.5 Software Interface to other components](#_heading=h.1y810tw) 7

[3.3 Logging In](#_heading=h.4i7ojhp) 7

[3.3.1 Logging In Use Cases](#_heading=h.2xcytpi) 7

[3.3.2 Processing sequence for Logging In](#_heading=h.1ci93xb) 7

[3.3.3 Structural Design for Logging In](#_heading=h.3whwml4) 7

[3.3.4 Key Activities](#_heading=h.2bn6wsx) 7

[3.3.5 Software Interface to other components](#_heading=h.qsh70q) 7

[4 User interface design](#_heading=h.3as4poj) 7

[4.1 Interface design rules](#_heading=h.1pxezwc) 7

[4.2 Description of the user interface](#_heading=h.49x2ik5) 7

[4.2.1 Navigation](#_heading=h.2p2csry) 8

4.2.1.1 Screen Images8

4.2.1.2 Objects and Actions 8

[4.2.2 Login Page](#_heading=h.32hioqz) 8

4.2.2.1 Screen Images 9

4.2.2.2 Objects and Actions 9

[4.2.3 Patient Booking Page](#_heading=h.2grqrue) 10

4.2.3.1 Screen Images 10

4.2.3.2 Objects and Actions 10

[4.2.4 Patient/Doctor List Page](#_heading=h.vx1227) 10

4.2.4.1 Screen Images 11

4.2.4.2 Objects and Actions 11

[4.2.5 User Profile Page](#_heading=h.3fwokq0) 11

4.2.5.1 Screen Images 11

4.2.5.2 Objects and Actions 12

[5 Restrictions, limitations, and constraints](#_heading=h.4f1mdlm) 12

[6 Testing Issues 1](https://docs.google.com/document/d/1SzcMx2Ps085WT4wCQBX6nm4024mI3MLc/edit#heading=h.2u6wntf)2

[6.1 Classes of tests 1](https://docs.google.com/document/d/1SzcMx2Ps085WT4wCQBX6nm4024mI3MLc/edit#heading=h.19c6y18)2

[6.2 Expected software response 1](https://docs.google.com/document/d/1SzcMx2Ps085WT4wCQBX6nm4024mI3MLc/edit#heading=h.3tbugp1)3

[6.3 Performance bounds 1](https://docs.google.com/document/d/1SzcMx2Ps085WT4wCQBX6nm4024mI3MLc/edit#heading=h.28h4qwu)3

[7 Appendices 1](https://docs.google.com/document/d/1SzcMx2Ps085WT4wCQBX6nm4024mI3MLc/edit#heading=h.nmf14n)3

[7.1 Packaging and installation issues 1](https://docs.google.com/document/d/1SzcMx2Ps085WT4wCQBX6nm4024mI3MLc/edit#heading=h.37m2jsg)3

[7.2 User Manual 1](https://docs.google.com/document/d/1SzcMx2Ps085WT4wCQBX6nm4024mI3MLc/edit#heading=h.2lwamvv)3

[7.3 Open Issues 1](https://docs.google.com/document/d/1SzcMx2Ps085WT4wCQBX6nm4024mI3MLc/edit#heading=h.111kx3o)4

[7.4 Lessons Learned 1](https://docs.google.com/document/d/1SzcMx2Ps085WT4wCQBX6nm4024mI3MLc/edit#heading=h.3l18frh)4

[7.4.1 Design Patterns 1](https://docs.google.com/document/d/1SzcMx2Ps085WT4wCQBX6nm4024mI3MLc/edit#heading=h.206ipza)4

[7.4.2 Team Communications 1](https://docs.google.com/document/d/1SzcMx2Ps085WT4wCQBX6nm4024mI3MLc/edit#heading=h.4k668n3)4

[7.4.3 Task Allocations 1](https://docs.google.com/document/d/1SzcMx2Ps085WT4wCQBX6nm4024mI3MLc/edit#heading=h.2zbgiuw)4

[7.4.4 Desirable Changes 1](https://docs.google.com/document/d/1SzcMx2Ps085WT4wCQBX6nm4024mI3MLc/edit#heading=h.1egqt2p)5

[7.4.5 Challenges Faced 1](https://docs.google.com/document/d/1SzcMx2Ps085WT4wCQBX6nm4024mI3MLc/edit#heading=h.3ygebqi)5

# 1 Introduction

The industry of healthcare is one of the most important subsections of the economy in the world. It is an area in which people’s lives are literally at hand. With labor shortages on the rise, we need less person-hours dedicated to purely clerical tasks, and more time allocated to what matters. Furthermore, much of the clerical and administrative work that needs to be completed gets put on the back burner. One of the biggest time inefficiencies of any office is the process of scheduling appointments for clientele, or in this case, patients.

Many of the current appointment scheduling systems are outdated and incredibly difficult to use. It takes some people months, or even years, to become comfortable with poorly designed systems. What's worse, many tenured clerical workers retired during the pandemic, as they did not want to be exposed to the virus. Herein lies the problem, outdated scheduling technology must be updated, so that doctor’s offices can focus on what matters most, the patient.

ClientCentral is a tool which will be used in doctor’s offices to provide scheduling services and patient data management/storage. The tool is meant to be reliable, stable, efficient, and easy to use for both patients and healthcare providers. It is utilized as a web application, making it easy for most computers to access. It will significantly cut down on training time for office workers, as well as allow patients to schedule their own meetings, furthering progress towards solving labor shortages.

## 1.1 Goals and objectives

To create a tool for patients and healthcare providers to schedule appointments and allowing doctor to view all of the scheduled appointments, even if they aren’t the assigned doctor. We aim to streamline the appointment scheduling process by providing the patient user with more power. Subsequently, fewer mistakes are made by the doctors themselves, allowing office workers to focus on the tasks not easily automated.

## 1.2 Statement of system scope

The software will store user data in an SQL server and retrieve and update that information when it is interacted with. Users will be able to schedule appointments by using the calendar tool to select open dates. The system scope is outlined below in figure 1.1.

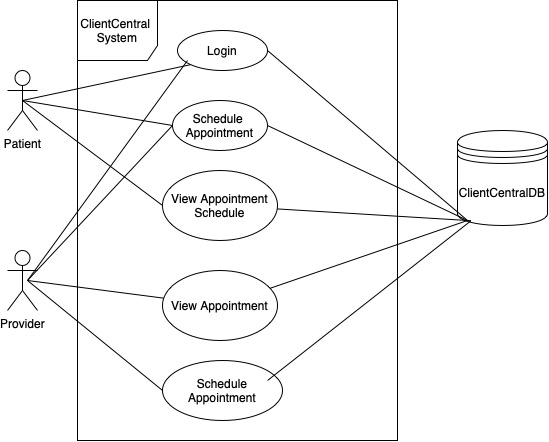


Figure 1.1 - ClientCentral Use Case Diagram

# 2 Architectural design

## 2.1 System Architecture

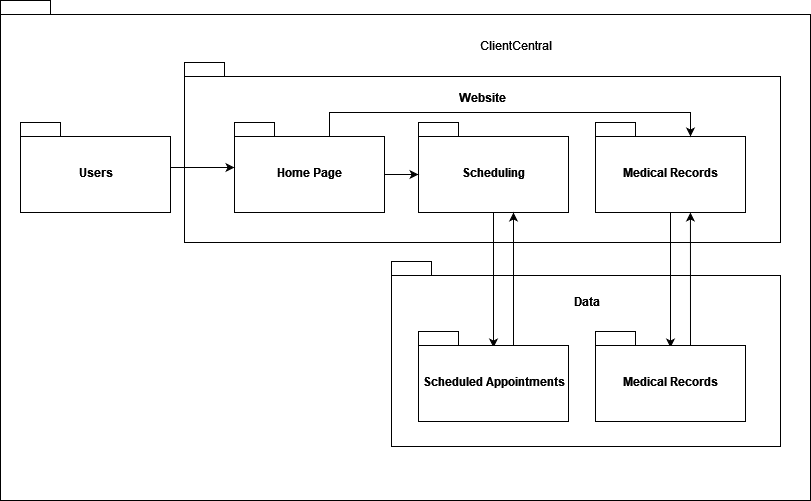


Figure 2.1 - ClientCentral Architectural Design UML package diagram

The UML package diagram is shown in figure 2.1 above. The Users package consists of doctors and patients. The users access the homepage, where they log in and may access scheduling and medical records interfaces. The scheduling interface consists of the calendar and allows the users to access their appointment data. The high-level subsystems include the scheduling system. The role of the scheduling system is to manage appointments and connect the calendars of the patient and the doctor. Figure 2.1 shows the structure of ClientCentral.

## 2.2 Design Rationale

We chose to structure our system this way because we want to keep the scheduling data separate from the user profiles and the booking page itself to maintain privacy.

# 3 Key Functionality design

## 3.1 Appointment Scheduling

### 3.1.1 Scheduling Use Cases

The patient or doctor will navigate to the booking page where a form will be displayed. The form will then be filled out by the patient or doctor, taking in the patient username (doctor fills it in, patient has it auto-filled and readonly), doctor username (patient fills it in, doctor has it auto-filled), date, time, and reasons/symptoms pertaining to the visit. The patient or doctor then clicks submit and sees that their appointment is confirmed. The patient can then see their most recently scheduled appointment in their profile. The doctor will see this in their profile as well, either on the “next three appointments” section or, if the appointment is with them, on the “your next appointment” section.

### 3.1.2 Processing sequence for Appointment Scheduling

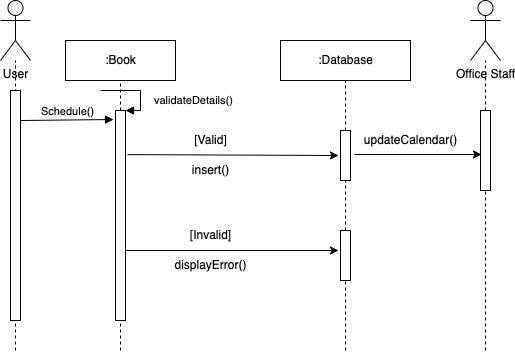


Figure 3.1 - Appointment Scheduling Sequence Diagram

Shown in Figure 3.1, the booking page will display a form that the user will fill out to schedule an appointment. The date and time would be stored in the database and seen in both the specified doctor and patient accounts, as well as the reasons for the visit and the assigned doctor username.

### 3.1.3 Structural Design for Appointment Scheduling

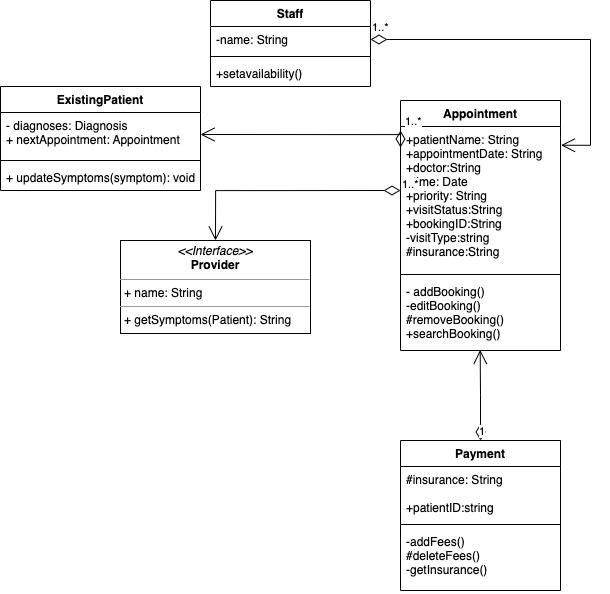


Figure 3.2 - Appointment Scheduling Class Diagram

Figure 3.2 shows the class structure of the booking tool. The booking tool will primarily use the Appointment class, which holds information about an appointment, such as the date and time of the appointment, the patient’s name who scheduled it, the doctor attending the appointment, and so on.

3.1.4 Key Activities

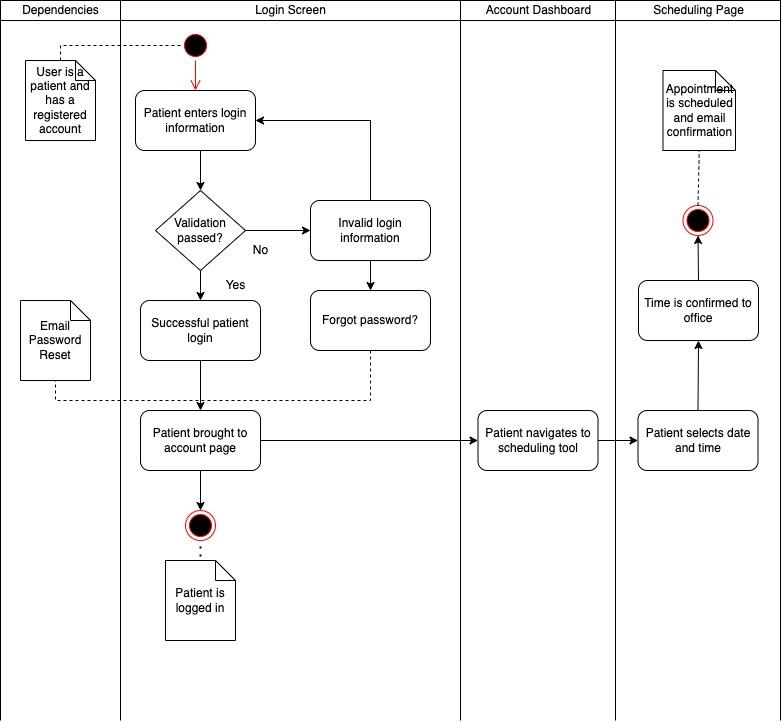


Figure 3.3 - Appointment Scheduling Activity Diagram

The main activities for scheduling an appointment, shown in Figure 3.3, start with a user logging in. With a successful login, they navigate to the booking page, select a date, select a time, input the reasons for the appointment, and confirm they wish to schedule the appointment for that date and time. The appointment is confirmed and displayed on both appropriate accounts.

### 3.1.5 Software Interface to other components

Only the booking tool will interface with the appointment database table.

3.2 The User Profile

### 3.2.1 User Profile Use Cases

A user (either doctor or patient) can navigate to their profile and see their personal information, as well as their upcoming appointments. A doctor can see their next appointment’s details and the next three appointments’ details for any doctor. A patient can see their next upcoming appointment details.

## 3.2 Logging In

### 3.2.1 Logging In Use Cases

There are two types of users who can log in: patients and doctors. To log in, a user must enter a username and a valid password. If this information is valid, they will be logged in. Passwords will be hashed in the database for security purposes. Patient accounts have the ability to view only their own personal information, while doctors can see the non-sensitive health information of their patients.

### 3.2.2 Processing sequence for Logging In

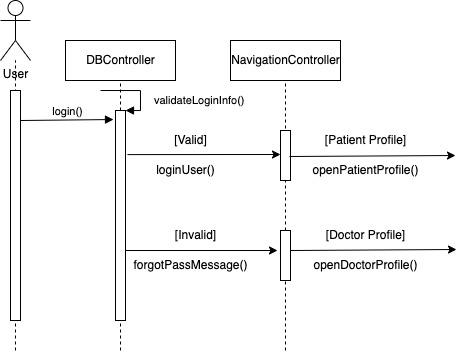


Figure 3.7 - Login Sequence Diagram

Figure 3.7 shows the sequence for a user logging in. When a patient visits the ClientCentral site for the first time, they will be shown the ClientCentral homepage. In the navigation bar, there will be options for register and login. On the login page, they can enter their username or email and their password. If this is authenticated, the user will be logged in. Patients will be brought to their profile page where they can navigate to other sections of the site. Doctors will have a separate login available from a link on the login page. Once they log in, they will be brought to their page, which will show upcoming appointments.

### 3.2.3 Structural Design for Logging In

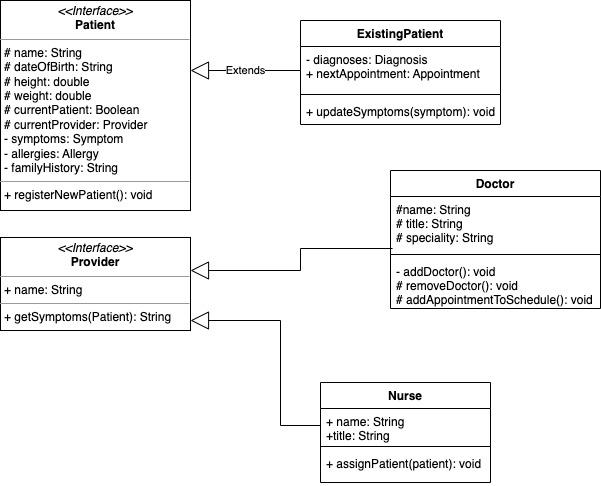


Figure 3.8 - Login Class Diagram

The login page will utilize the Patient and Doctor database tables. The class structure is shown in Figure 3.8.

### 3.2.4 Key Activities

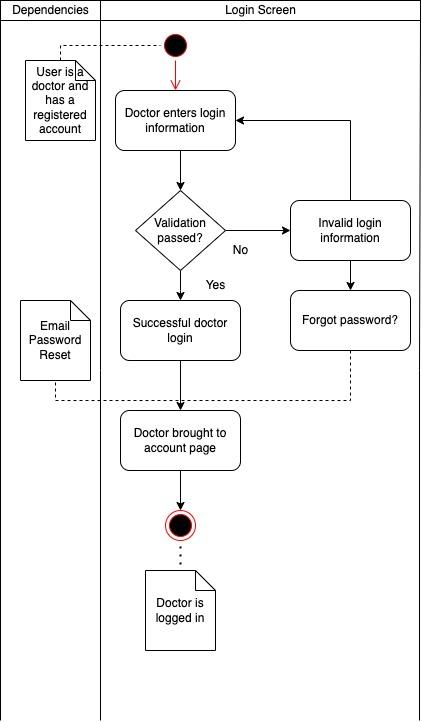


Figure 3.9 - Login Activity Diagram

As shown in Figure 3.9, the patient enters their login information. The authentication.php file communicates with the patient accounts database table or the doctor accounts database table to confirm the authenticity of the login credentials. If authenticated, the user is logged in.

### 3.2.5 Software Interface to other components

This module will interact with the database via the authenticate.php file.

# 4 User interface design

## 4.1 Interface design rules

The standards that we are following are to have a simple user interface for universal usability. The interface will have a minimalist design that allows users the ability to book and view appointment(s) and view their symptoms provided by the patient.

## 4.2 Description of the user interface

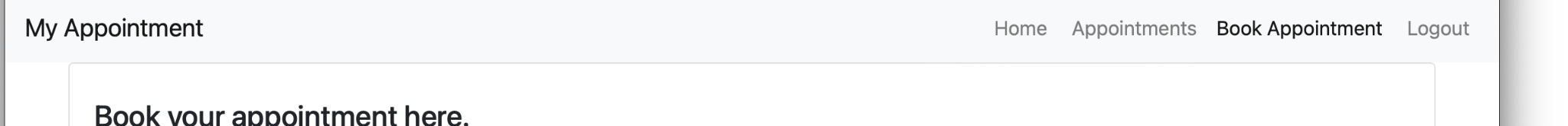
The user interface (UI) provides a visual front-end to our client's databases and queries. It provides visibility and communication between the accounts created by the doctor and the patient’s account. There are seven distinct user interface designs in the following: home page, login, register, patient booking, doctor booking, patient table, and user profile.

### 4.2.1 Navigation

The main menu page allows a user to navigate to their relevant pages. In the doctor account, the user will be able to navigate to a list of all their patients in the database, the other is to view their appointments that were scheduled by the patient users. They’re also able to view the last three appointments booked in the system, whether those appointments are relevant to that doctor. In the patient account, the user will have the option to navigate to their booking page and appointment list, which will be set as the patient’s home page.

#### 4.2.1.1 Screen Images

Shown here are the interface screens of the web application.

****

**Figure 3 - Navigation**

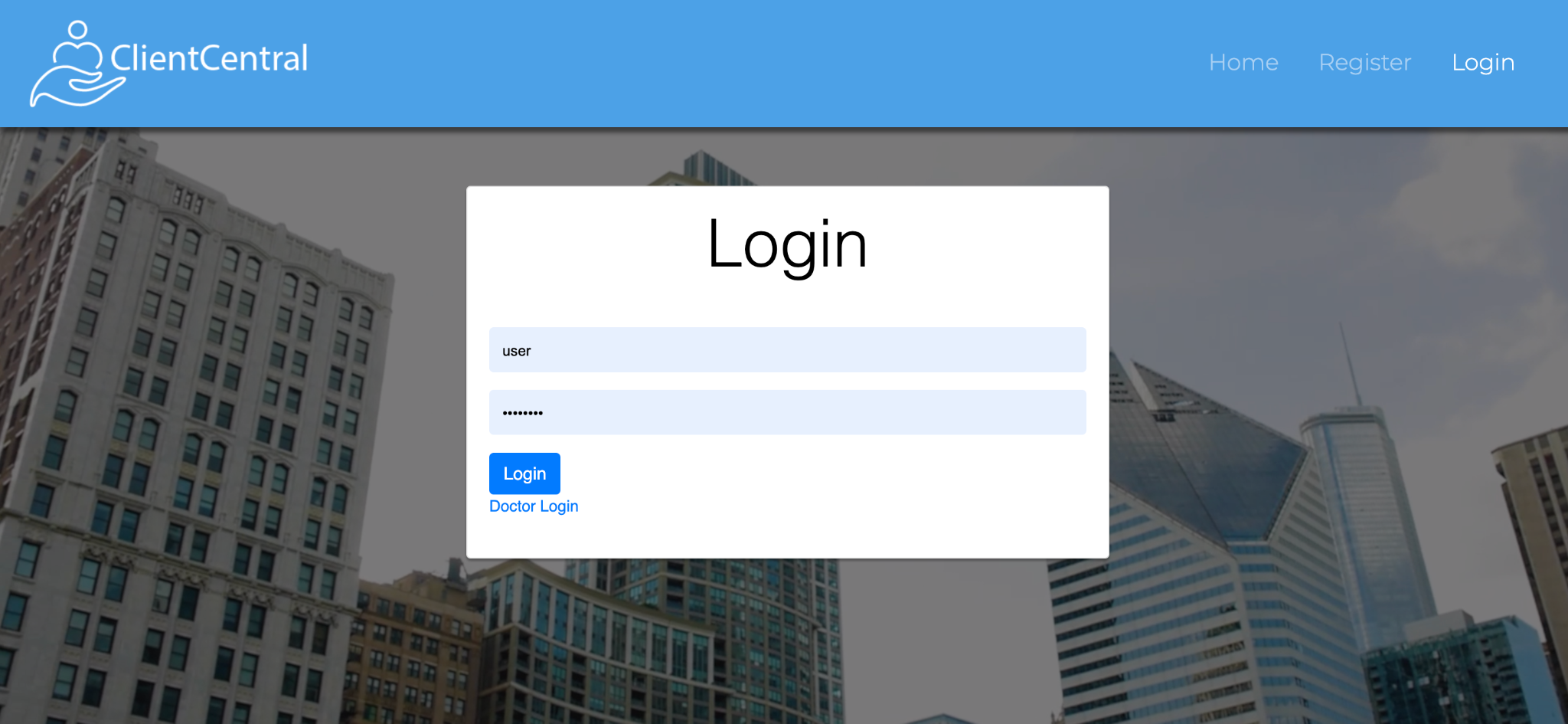
#### 4.2.1.2 Objects and Actions

The main object on all of the pages is the navigation menu that will appear on the top of each page. It allows the user to return to the user’s home page, view patients or doctors list pages, book an appointment page, and review the user profile. Depending on the account, there will be different options on the menu to correlate with the user’s objective. There will also be a log-out option to end access to the user’s account and destroy the session on the server.

### 4.2.2 Login Page

The login page initially prompts the user to select between two different accounts, “doctor” and “patient.” It also contains two input sections for the user to input their email and password. At the bottom, there’s a button the user can select called “login” to gain access to their account. However, if the user hasn’t set up an account with the database, the user may fill out a form to create a new account. Then, the user can add their credentials to be authenticated.

*4.2.2.1 Screen Images*

****

**Figure 4 - Login**

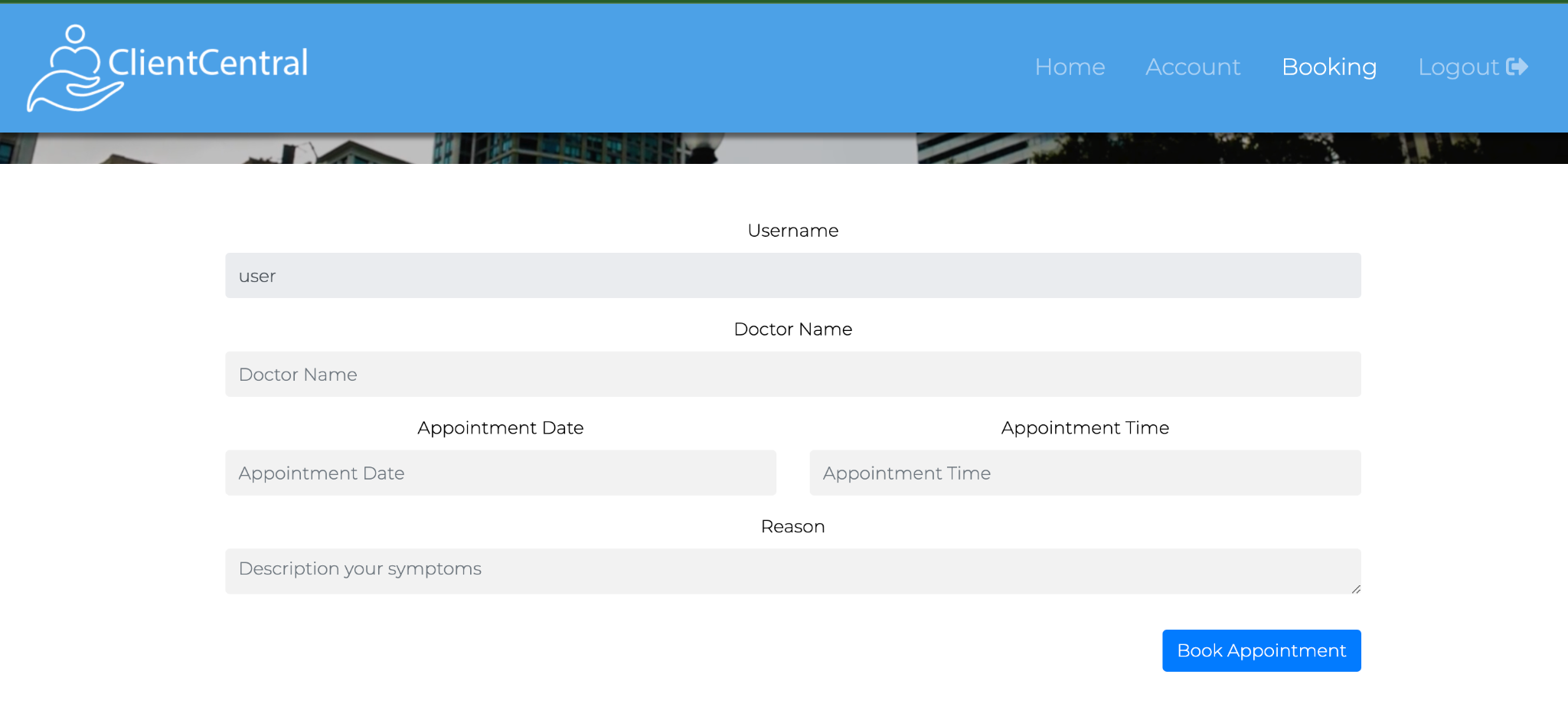
#### 4.2.2.2 Objects and Actions

The purpose of the login page is to allow the user to access their account according to their roles. There are two types of accounts: doctor and patient; the user will have to select their account type before seeing the login page. The login page contains two input sections and a button to log in. There’s also an option to register a new account, where the user will have to fill out a form.

### 4.2.3 Patient Booking Page

The patient booking page mainly consists of a secure form that the patient user will fill out with their personal information. The user will choose a day and time for their appointment. The user will also select a doctor. When the form is submitted, the page is returned to a list of their appointments.

#### 4.2.3.1 Screen Images



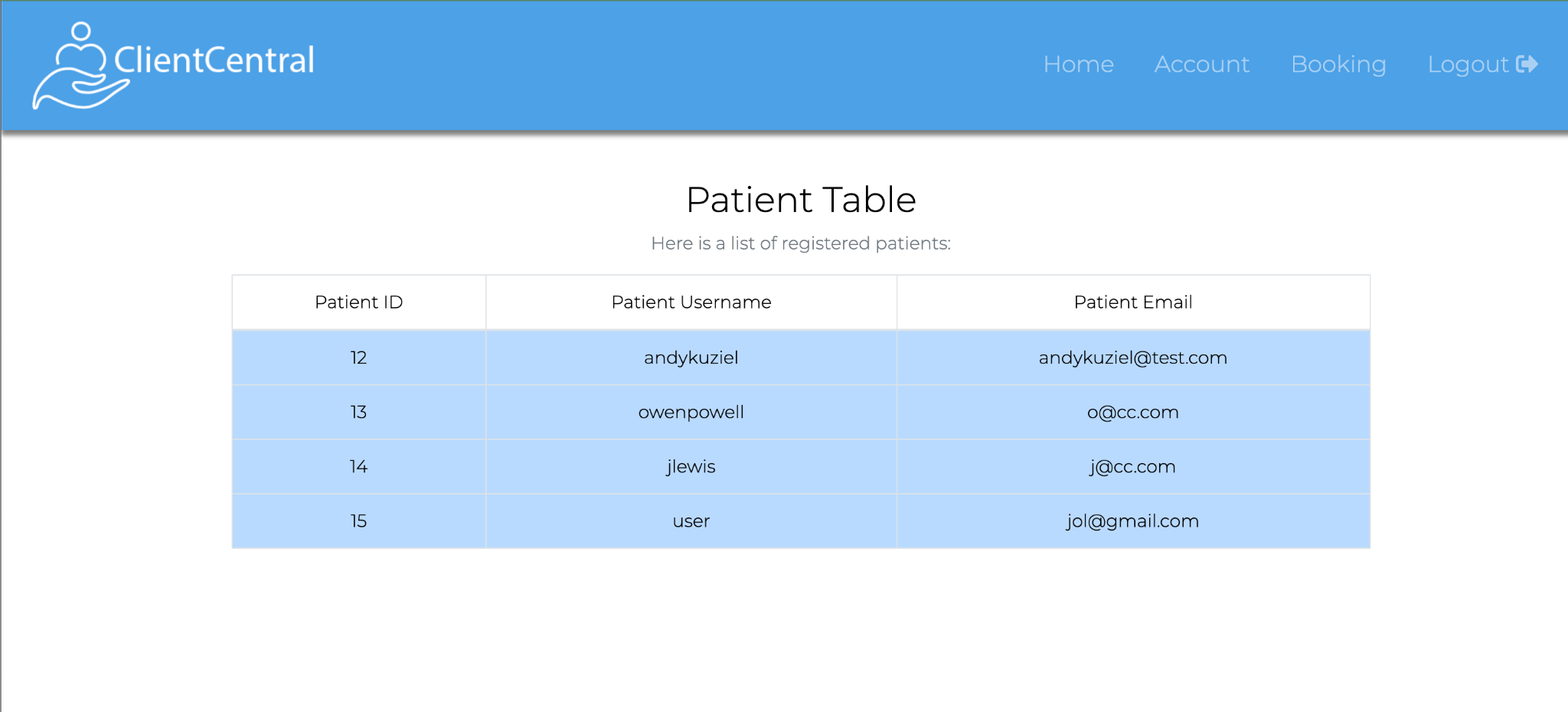
#### 4.2.3.2 Objects and Actions

The form leads to the Appointment object in the database. The user will be required to input information that contains the following: name, phone number, appointment time and date, select doctor, and (a text area for ) symptoms. When the form is complete, the user selects “book appointment”. This action will display the appointment on the patient’s and doctor’s appointment pages.

### 4.2.4 Patient/ Doctor List Page

The patient list shows all the patient accounts in the database and some of their information. This page can only be accessed via the doctor’s account. However, the patient can see the doctor’s list, which displays a list of the doctors in the database or the users that created a doctor account. The doctor list and patient list pages are similar and give the users the ability to view the account users that are opposite to their current one.

#### 4.2.4.1 Screen Images



#### 4.2.4.2 Objects and Actions

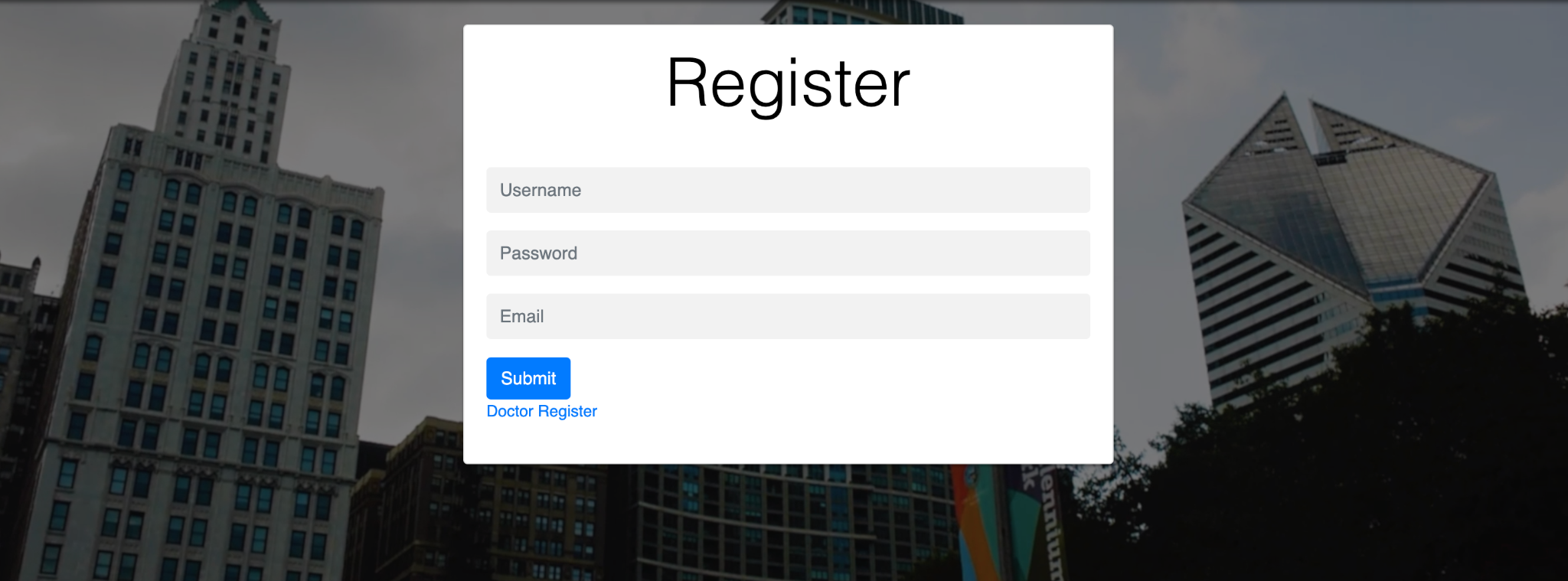
The patient list displays the users with a patient account. The page is available under a doctor’s account. The user has the ability to access the user profile by selecting the patient’s name. The user also has the option to book an appointment on this page by selecting a date and time, which is seen next to each patient’s name. The appointment is booked after clicking “book appointment.”

### 4.2.5 User Profile Page

On the patient’s home page, they can easily navigate to both the booking page, as well as the account page. On the account page, the user will be able to view their upcoming appointments. On the booking page, they can book future appointments. In the doctor’s account, the doctor user can view the patient list page and then book an appointment if necessary.

#### 4.2.5.1 Screen Images

Below is an image of the user profile.



#### 4.2.5.2 Objects and Actions

Both accounts will have a user profile that only the user can access. The doctor’s user profile will have the option to view all the patients in the system. The user profile is the home page for the patient account.

# 5 Restrictions, limitations, and constraints

* HTML, CSS, PHP, MySQL will be the main languages used
* MySQL is used to run the development server
* Xampp was used to run MySQL server
* The information store will be a MySQL database

# 6 Testing Issues

Test strategy and preliminary test case specification are presented in this section. Additional test cases are located in the Appendix section in spreadsheet format.

## 6.1 Classes of tests

* Performance Test – ensure that the response time for information retrieval is within an acceptable range.
* Accuracy Test – Scheduling data, as well as patient information, must be correct 100% of the time.
* User Interface Test – Many users may be less technologically adept, so it must excel in terms of simplicity.
* Security test – Patient and Doctor log-ins must provide separate access to data.
* Repeatability Test – Inquiries should always provide the latest data.

## 6.2 Expected software response

* Performance Test – There must not be any fatal crashes.
* Accuracy Test – All data must be correctly cataloged and accessed as needed.
* User Interface Test – Patients will not be trained on how to use the interface, so they must be able to make it through the scheduling process within 5 minutes.
* Security Test – Patients can only access their information, while Doctors can access patient information and Doctor information.
* Repeatability Test – Only the latest data should be visible when repeating inquiries.

## 6.3 Performance bounds

* Must be compatible with Google Chrome on desktop computers.
* Registration process should be under 2 minutes.
* Scheduling process should be under 3 minutes.
* Pages must load within 10 seconds.
* Web pages must be responsive.
* Previous sessions should be “destroyed” to prevent a web browser back button being used to access privileged information.

# 7 Appendices

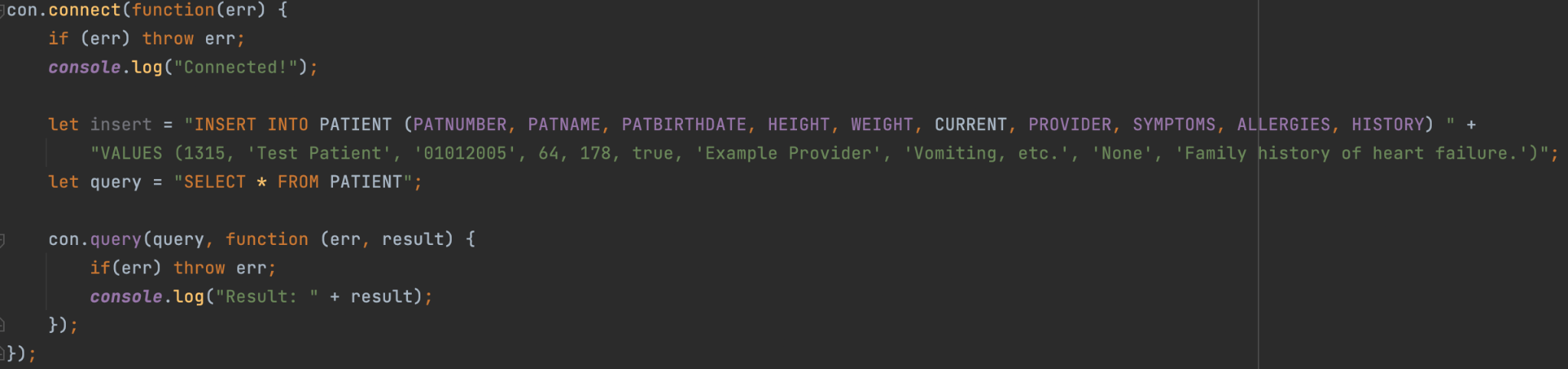
## 7.1 Packaging and installation issues

Due to a lack of budget, Client Central will be hosted on a local MySQL database, and a text file will be included with the project with the necessary SQL commands to populate the database. The following links give a tutorial on installing a MySQL database on multiple OSs:

MacOS: <https://dev.mysql.com/doc/refman/8.0/en/macos-installation-launchd.html>

Windows: <https://dev.mysql.com/doc/refman/8.0/en/windows-installation.html>

The DB controller module can run SQL commands for inserting and retrieving, such as the ones below:



## 7.2 User Manual

Give a step-by-step description of using the key features of the system.

**Login Instructions**

Step 1: Select “Patient” or “Doctor” based on the user’s role.

Step 2: If you already have account credentials, skip to Step 3. Otherwise, Select “Create a New Account”.

Step 3: Input your email and password inside the designated text areas. If you forgot your password, skip to step 4. Otherwise, continue to step 5.

Step 4: Click the “Forgot password” link under that password input text area. This will lead to a page where you need to input your email and await for an email to reset your password. Create a new password then, start at Step 1.

Step 5: Click “Login” to access your account. This will return your account home page.

**Patient Book Appointment Instructions**

Step 1: Select “Book Appointment”

Step 2: User must input their required information in the designated text area. The user will be asked about their date of birth, symptoms, doctor, appointment time and date, and insurance information.

Step 3: After the form is completed, the user can book their appointment by clicking the “Book Appointment” button at the bottom of the form.

**Doctor Book Appointment Instructions**

Step 1: In the doctor’s account, select “Book Appointment”.

Step 2: A list of patient users will be displayed. Find the desired patient user.

Step 3: Select an appointment time and date.

Step 4: Click “Book Appointment on the line you were setting up the appointment.

## 7.3 Open Issues

The only feature we have planned but not implemented is a forgotten password system. We have an email notification system somewhat functional, but not fully completed.

## 7.4 Lessons Learned

We have learned a lot in regards to proper design. While many people can sort of “throw together” a program, building reliable software requires foresight and planning. The biggest lesson learned is how critical adhering to system design can be, and how it shapes a project entirely.

### 7.4.1 Design Patterns

We’re using a client-server model in our structural design. This allows a more reliable system and helps our team to collaborate in our implementation process. The singleton pattern is used for the scheduling tool, where only a single instance of the calendar will exist at once. For users, the design pattern is a general hierarchy. Each type of user shares similar properties, with some differences, with the Patient and Provider classes being the superclasses.

### 7.4.2 Team Communications

How team communications were conducted and where could you improve in the future?

Team communications were conducted on a weekly basis via zoom, as well as several times a week on a group messaging app. This was more than adequate, though a weekly project deliverable system may have helped the team stay focused on the objective and tasks ahead.

### 7.4.3 Task Allocations

How your team allocate tasks and responsibilities and where could you improve in the future?

We allocated tasks and responsibilities based on both skillsets, and areas of opportunity for learning. We wanted to play to our strengths, but also learn a lot along the way. An area for improvement would be to cross the bridge into an overly constructive manner. If we had mapped out our whole project week by week, that may have created a better environment for work generated. Still, a more agile approach is important when building this kind of project.

### 7.4.4 Desirable Changes

Assume that you have another month to work on the project, what aspects of the system you would like to improve? What are the additional features you want to add to the system? [Each student should use a separate paragraph to respond to the questions]

Jordan: The ability for the system to email forgotten usernames and passwords to the proper parties would be very useful. This would be accomplished through the Node.JS using Nodemailer for gmail accounts. The usage of emails would really elevate this aspect of the software, allowing patients to make mistakes without being punished.

Jennifer: We would like to add security features to test the HIPPA compliance of the project. Many of the features that are needed to make the data secure require the website to be hosted, such as SSL certificate, HIPPA business associate agreement with web hosting service, and multi-factor authentication. Another aspect would be adding a search function to the Patient List page under the doctor’s account. This will make it easier for a doctor to easily find the patient in a large dataset.

Owen: As Jennifer mentioned above, security would be the first thing I would improve, as the program deals with sensitive personal data. Additionally, having a more secure password encryption service would be very beneficial. If we had a budget available for this project, I would like to use Google Cloud, AWS, or a similar service to host the website and the MySQL server so we could avoid the clunky process of setting up a local server on each machine. Unfortunately, hosting services cost money and can take some time to learn and configure properly.

### 7.4.5 Challenges Faced

Among requirements specification, system design, and system implementation, which one you think is the hardest task? Why? [Each student should use a separate paragraph to respond to the questions]

Jordan: System design is the most difficult aspect of this software, as it has been a challenge to not only find the right design, but to really understand it. Every system design has its challenges and benefits, and retrospect always serves as a reminder to why another design may have been better. Regardless, system design is about weighing pros and cons, and sticking to a decision once it is made.

Jennifer: The hardest task for this project would be integration. Although we have extensive documentation and the members of the team understand the requirements for the project, there is still room for functionality flexibility. This could allow sudden changes of a component to overlap with others, especially if different members are working on the same use case. These issues could increase production time and leave less time to test the final project.

Owen: I found system design to be the most challenging part of this process. Planning for a system without being able to do hands-on work is difficult for me, as there are changes that needed to be made in the implementation process that sometimes were completely different from what we planned. This made it harder to use the designs we already made, as they started to look different from the actual system. Most of the programming work I have done so far has been without intense planning such as in this course, but the experience has taught me a lot about system design and I have found it very valuable.