

The Doctor's Office Project Final Report



**Created by Michael Chau, Jon Heikkinen, Alen Peter, and
Alejandro Cabral for CS 440
at the
University of Illinois Chicago**

November 2021

Table of Contents

I	Project Description.....	4
1	Project Overview.....	4
2	Project Domain	4
3	Relationship to Other Documents	4
4	Naming Conventions and Definitions	4
4a	Definitions of Key Terms.....	4
4b	UML and Other Notation Used in This Document	6
4c	Data Dictionary for Any Included Models.....	8
II	Project Deliverables	9
5	First Release.....	9
6	Second Release	10
7	Comparison with Original Project Design Document.....	13
III	Testing	13
8	Items to be Tested	13
9	Test Specifications	14
10	Test Results.....	18
11	Regression Testing	20
IV	Inspection.....	20
12	Items to be Inspected.....	20
13	Inspection Procedures.....	21
14	Inspection Results	21
V	Recommendations and Conclusions.....	22
VI	Project Issues	22
15	Open Issues	22
16	Waiting Room.....	23

17	Ideas for Solutions.....	24
18	Project Retrospective.....	24
VII	Glossary	25
VIII	References / Bibliography	25
IX	Index.....	26

I Project Description

1 Project Overview

This project is a desktop application that will make it easier for doctors and patients to view and schedule appointments. Doctors can view their schedule and patient information and patients will be able to see/schedule appointments.

2 Project Domain

To create this project the team used a MYSQL schema. Within the schema the team created three tables that each held information about the different types of users and the appointment that links the two together. Users were separated between patient and doctor or medical professional, where once a patient scheduled an appointment then the appointments table would hold who scheduled the appointment and which doctor was chosen to conduct it.

3 Relationship to Other Documents

This was the previous group's final report for the development of this product. The documentation contained key features and ideas that the previous group had thought of and was implemented throughout the team's application. While some features were cut, there were many more taken from this documentation to hopefully achieve a similar product as the previous group envisioned.

Initial Design UML:

One of the documents that was used relating to the project is the database initial design uml diagram. The initial design allowed the team to properly visualize what objects would be possibly needed with what variables they would hold. The UML was one of the foundations of the project and was a core starting point for the team.

Scenarios:

Another documentation used for this project was the Scenario diagrams that have been created. The documentation had scenarios that the users would possibly go through to reach a result. This documentation gave the team an idea of what needed to be done to reach the result and clearly see the end goal of each feature/scenario.

4 Naming Conventions and Definitions

4a Definitions of Key Terms

User:

The user represents a person that will be defined by one of two groups within the application, where the user can be either a patient or a doctor. The application will not know which group the user is in until they log in/create an account.

Patient:

The patient represents a user who is the primary service receiver from the Doctor's Office. The application provides a separate program space for the patient users containing all the services required by this type of application user such as appointment booking, doctor location visibility, and so on.

Doctor:

The doctor represents a type of user who can view appointments made by patients and see individual patient information within the program.

Appointment:

The appointment represents a link between the two types of users, where the patient will schedule an appointment with a doctor on a specific date, time, and location.

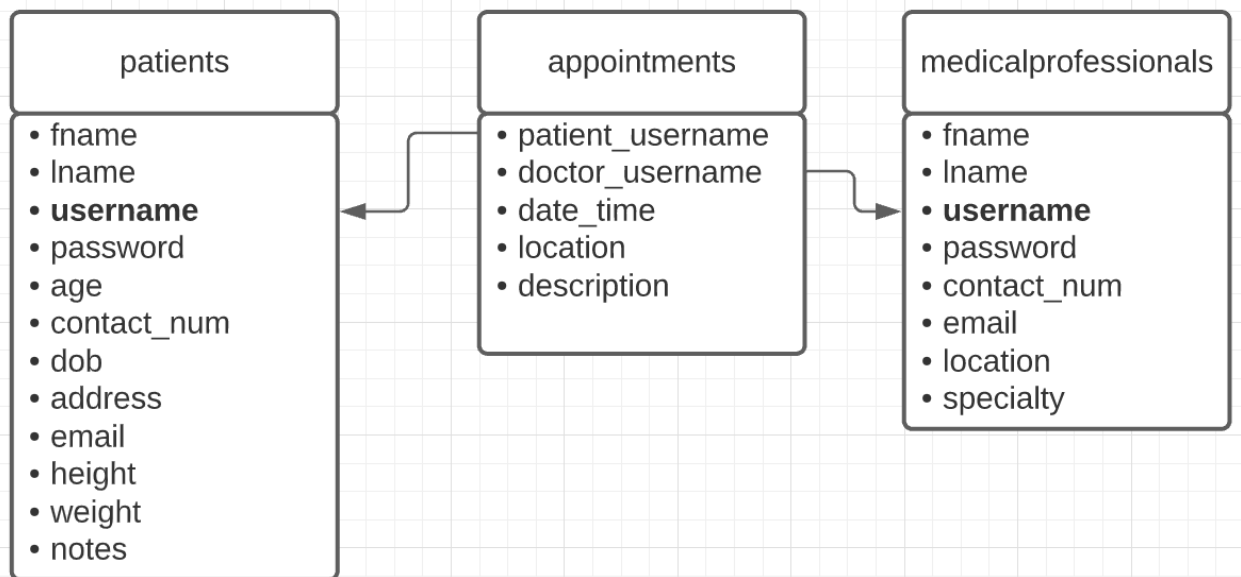
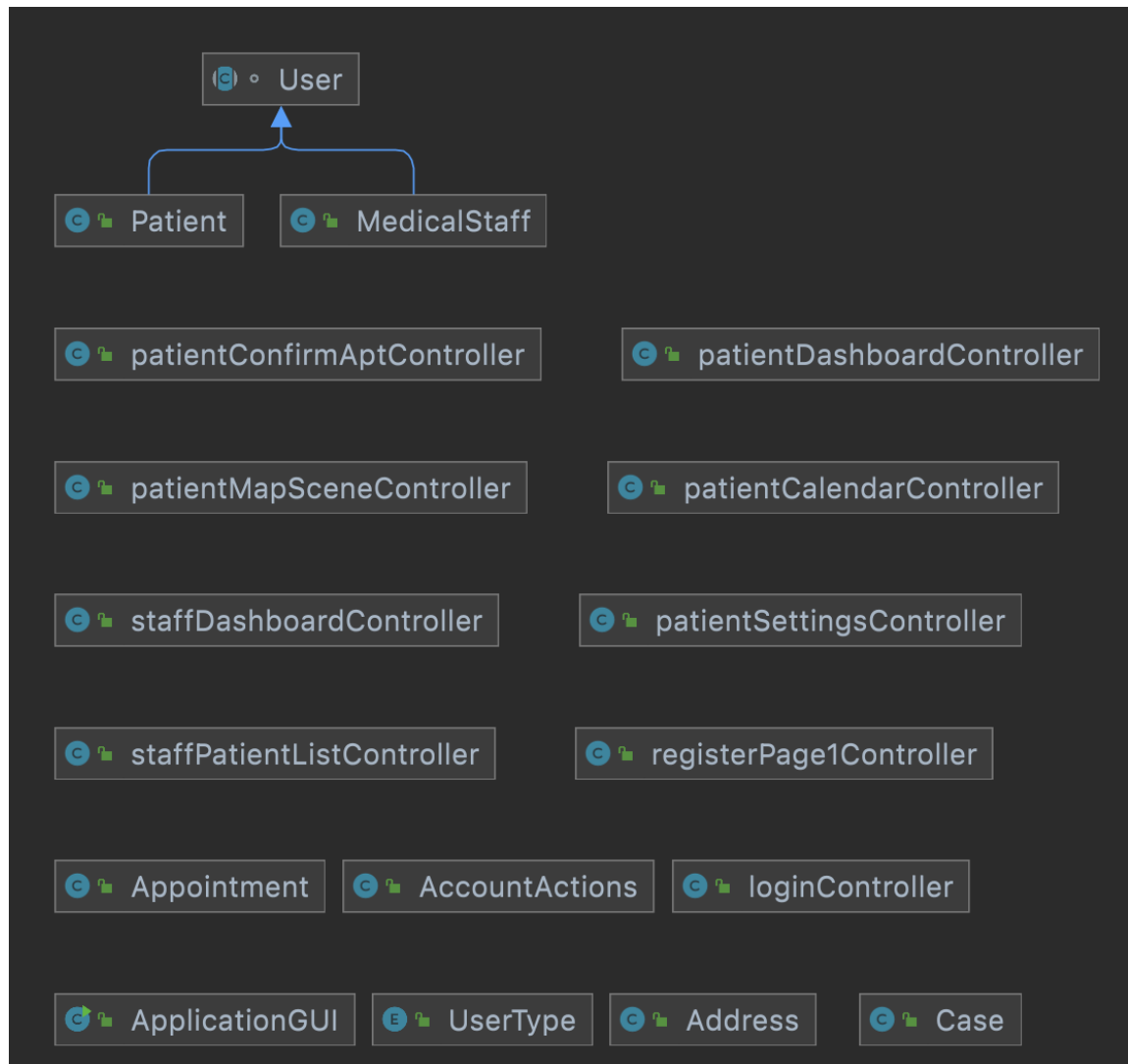
Query:

The query represents a MYSQL command made within the backend of the program to grab the necessary information from the schema's tables to display to the program's graphical interface.

API (Application Programming Interface):

The API represents one of the main components that are being used within the program where the program will communicate with Google's API to grab embedded map location data to display within the graphical interface.

4b UML and Other Notation Used in This Document



4c Data Dictionary for Any Included Models

ApplicationGUI: The ApplicationGUI is the main class in the program which loads the main function and GUI elements of the program. In addition to this in this class, we establish the connection to the database, store/retrieve data through the database, determine the user type and provide the implementation of major functionalities of this program through methods.

User: The user is an abstract class that contains fields/methods common to all users of the program.

Patient: The Patient is a specific type of user of the application which in addition to the User object content, contains fields that are specific to the Patients. For this reason, the Patient Class extends from the abstract class User to include all the fields that are required for a Patient object such as height, weight, dob, patient notes, and so on.

MedicalStaff: Like a Patient, a MedicalStaff also acts as a specific user of the application which extends its functionalities from a general User object. Therefore, in addition to fields and methods from the abstract class User (which the MedicalStaff object extends) the MedicalStaff class contains fields that are specifically related to a Medical Staff such as medical id, the staff's office location, and specialty.

Appointment: The Appointment class is an independent class that holds information that are related to a booked appointment between a patient and medical staff.

Address: The address class holds the program data of a patients address.

Controller: All the controller classes such as registerPage1Controller, patientDashboardController and staffDashBoardController add the implementation of response logic for each GUI element in the application.

The database diagram:

Patients: As said previously, patients are a type of user. The patient object holds information as data fields that will be used when an appointment is scheduled.

Appointments: The link between patients and medicalProfessionals where once an appointment is made then the respective data fields are filled as to who scheduled the appointment and who will conduct it from the medical professionals/doctor object's data.

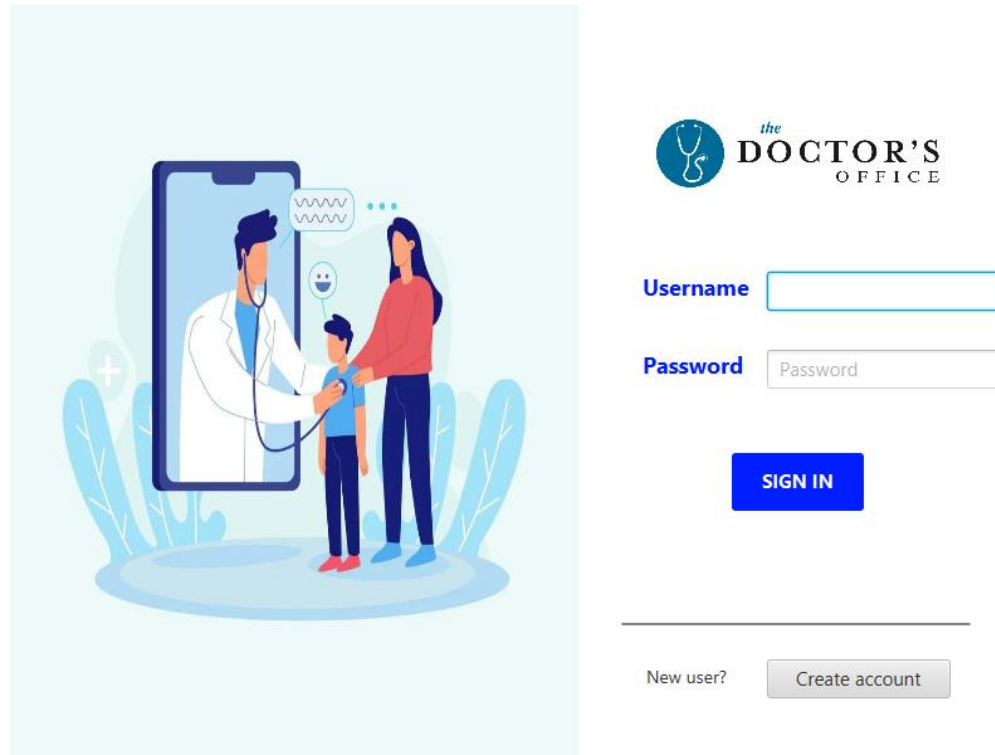
medicalProfessionals: As said previously, medicalProfessionals are a type of user, representing the doctor, who will oversee appointments and be able to view patients.

II Project Deliverables


5 First Release

First release of the project mainly had the login functionality of doctors and patients and ensured they would be brought to different dashboards that had no content in them. The first release also had the functionality of creating user accounts and ensured that the database maintained the information even with the closure of the application


Login page from 10/08/2021



Account creation from 10/08/2021



The illustration shows various medical supplies including a stethoscope, a syringe, a first aid kit with a red cross, a bottle of medicine, a pill bottle, and a pair of gloves, all arranged on a light blue background.

 **the DOCTOR'S OFFICE**

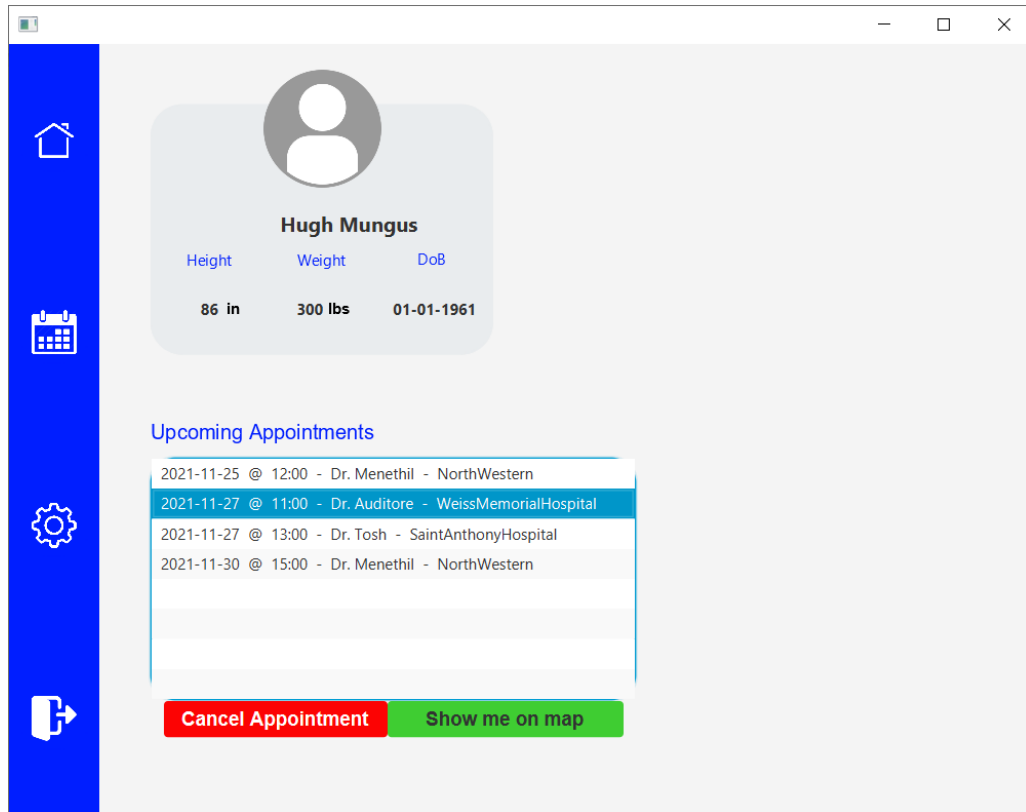
Create your account

First Name	<input type="text"/>	Last Name	<input type="text"/>
Email	<input type="text"/>	Username	<input type="text"/>
Password	<input type="text"/>	Confirm Password	<input type="text"/>
Date of Birth	<input type="text"/>	Address	<input type="text"/>
Insurance Name	<input type="text"/>	Phone Number	<input type="text"/>

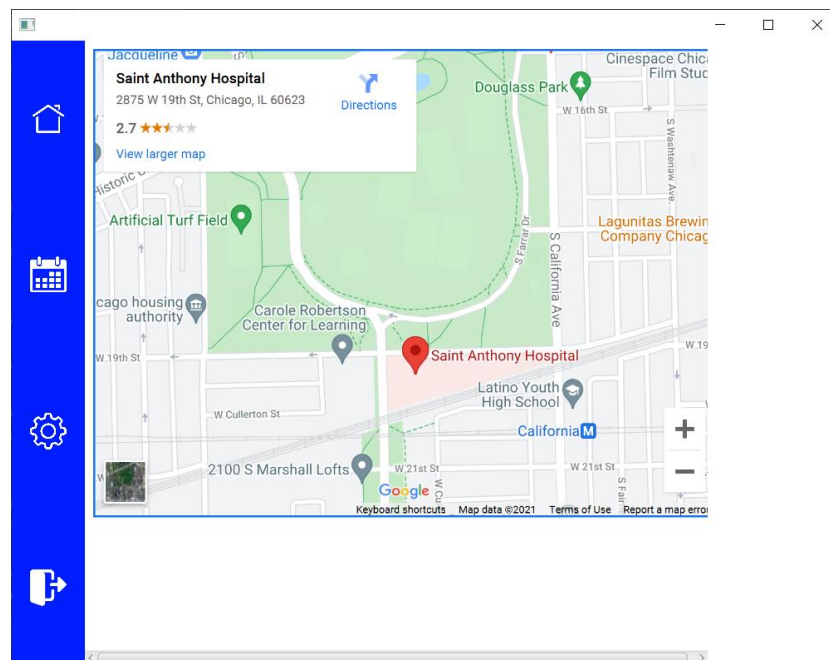
6 Second Release

Second release mainly focused on the different functionalities that the patients and doctors should have respectively. Doctors were able to successfully view scheduled appointments and take notes while patients were able to view current appointments and schedule any necessary. The second release also had the team implementing the google maps embedded API throughout the application to properly display hospital/appointment locations.

Initial patient dashboard 11/05/2021



When patient views location of appointment 11/05/2021



Process of Patient scheduling Appointment 11/05/2021

Schedule an appointment

Find a Specialist

General Physician

Orthopedist

Surgeon

Filter by location

Select the date:

Pick the hour:

Hospital Location

- NorthWestern
- SaintAnthonyHospital
- KindredHospitalChicagoNorth
- UIHealth
- RushHospital
- WeissMemorialHospital
- MountSinaiHospital
- MercyHospital

Doctors

- Deloitte - orthopedist
- Kekington - physician
- Smith - surgeon

CONFIRM APPOINTMENT

Doctor login dashboard with current Appointments 11/05/2021

Welcome, Dr. Menethil

Agenda for:

- 2021-11-25 @ 12:00 - Hugh Mungus - NorthWestern
- 2021-11-26 @ 10:00 - Joe Dirt - NorthWestern
- 2021-11-27 @ 10:00 - Ricky Bobby - NorthWestern
- 2021-11-27 @ 11:00 - Joe Dirt - NorthWestern
- 2021-11-27 @ 13:00 - Joe Dirt - NorthWestern
- 2021-11-30 @ 15:00 - Hugh Mungus - NorthWestern

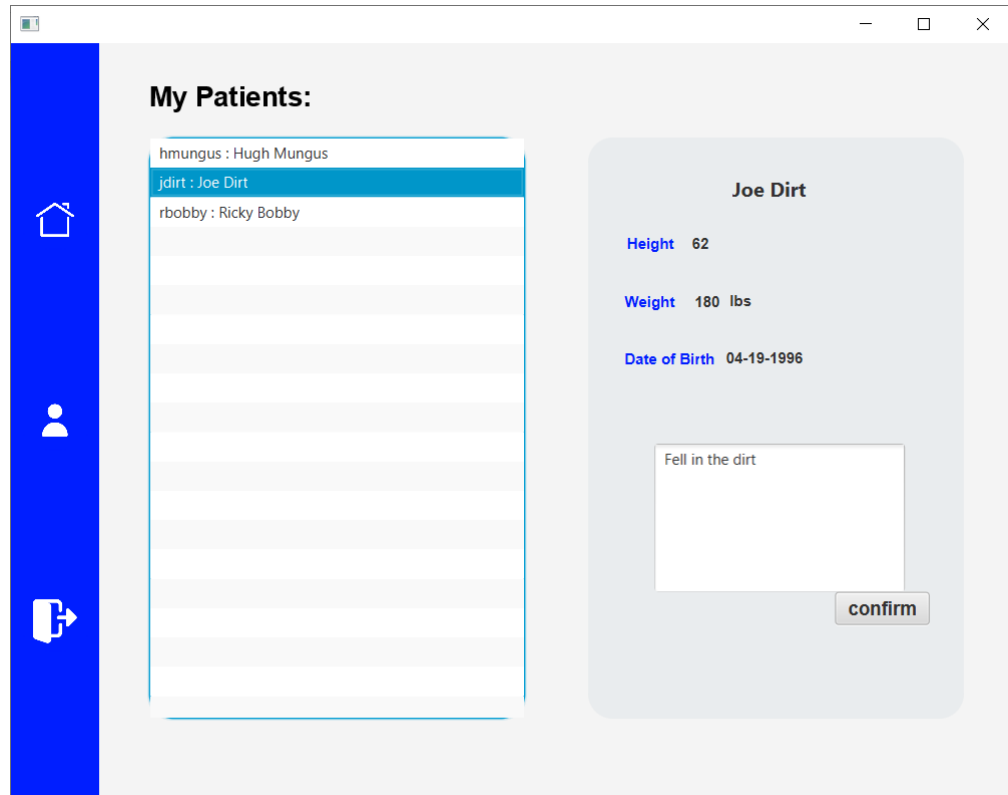
Filter agenda by date:

< November >

< 2021 >

Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4
5	6	7	8	9	10	11

Doctor overview of patients with notes 11/05/2021



7 Comparison with Original Project Design Document

The full project developed by Group 12's Sebastian Greczek, Jayarame Gowde, Thomas Kunik, and Luis Lema Segovia had included the user Nurses to help assist the doctor and patients, had the ability to send information of medicine on stock from the patient's preferred pharmacy, and for patients to receive medical results which were not implemented due to time constraints. However this project, developed by Group 11, had included Google Maps embedded maps API which allowed patients to find hospitals and doctors based on location.

III Testing

8 Items to be Tested

The components tested:

- Correct user type log in
- Account creation correctly inserted into database
- Embedded map html is correct
- Correct user information grabbed from MySQL queries
- Reserved appointments no longer listed

- Removed appointments correctly removed from database

9 Test Specifications

ID#01 - Correct user type log in

Description: Once the user logs in, the program correctly displays information only for that type of user.

Items covered by this test: When a user opens the application, they are given the choice to either log in or create an account. Medical Staff accounts are already created while patients must create an account before they log in if they have not already.

Requirements addressed by this test: Requirements #6

Environmental needs: To test this, once accounts are created then testing each type of user, log into the application and see that the correct pages show up.

Intercase Dependencies: Test ID # 02: Account creation correctly inserted into the database. The user, if a patient, must be able to create an account to be able to log in.

Test Procedures: The testing procedure used was that two user type accounts were created, one patient and one doctor. Upon login for each if the application correctly displayed the information for each user, then the tests passed, if not then they failed.

Input Specification: The necessary input is to have the login information for the user, so a username and a password.

Output Specifications: When the user logs in, they should see the pages meant to view for that user, so for example a patient will only see the pages made for them such as appointment scheduling, appointments upcoming, and the map feature.

Pass/Fail Criteria: The correct pages are displayed upon login.

ID#02 - Account creation correctly inserted into database

Description: User creates an account, and the data is inserted and maintained in the database.

Items covered by this test: Once a patient fills all the information needed in the account registration page and clicks the create account button, the database should correctly insert and maintain the user data even when the application is closed

which is the username, password, email, DOB, Address, Insurance, and phone number.

Requirements addressed by this test: Requirements #2

Environmental needs: Require JDBC and MYSQL to properly run and test the database.

Intercase Dependencies: Test ID#01 is needed to since the login validation will be used for this test.

Test Procedures:

1. Click “Create account” button
2. Fill in necessary information
3. Click “Create Account” button to confirm
4. Attempt to login with the username and password on the login page and check if MYSQL Database has been properly updated.

Input Specification: First Name, Last Name and Insurance must be a string of characters. Username, Email and Address may be a mixture of characters and numbers. DOB must be in the mm/dd/yyyy format. Phone number may be a string of numbers. Password field must contain at least one uppercase and one number.

Output Specifications: Users should be able to login successfully as a patient and see no appointments available but all the features available.

Pass/Fail Criteria: Upon successful completion a new entry is inserted into the database with the values entered from the account creation page.

ID#03 - Appointment creation correctly inserted into database

Description: Patient schedules an appointment and it's inserted into the database.

Items covered by this test: When a patient schedules an appointment the new entry and all information is inserted as a new value in the database.

Requirements addressed by this test: Requirements #7

Environmental needs: Require JDBC and MYSQL to properly run and test the database.

Intercase Dependencies: Test ID#01 is needed to since the login validation will be used for this test.

Test Procedures: Once logged in as a patient, go to the schedule appointment screen, input all necessary information, and confirm an appointment. Check the database to see if a new entry matches the information just inputted.

Input Specification: All data fields needed to schedule appointment

Output Specifications: In the patient's upcoming appointments a new entry is in the list with the information just added, and the database reflects the new information added as well in the appointment table within the schema.

Pass/Fail Criteria: A new entry in both the upcoming appointments and database are added once a patient schedules an appointment for a pass.

ID#04 - Embedded map html is correct

Description: The html brings forth the correct map on the graphical interface in the program

Items covered by this test: When a patient schedules an appointment, they are able to select a location from the schedule appointment page, if they select a location then the corresponding map should appear.

Requirements addressed by this test: Requirements #5

Environmental needs: Google Embedded Maps API

Intercase Dependencies: Test ID#01 is needed to since the user will need to be logged in as a patient

Test Procedures: Once a patient logs in, the patient goes to schedule an appointment. Patient selects a location, and the location needs to match the location on the map.

Input Specification: Location specified

Output Specifications: Correct map shown upon patient clicking on a location

Pass/Fail Criteria: Clicking on a map yields the correct map shown for a pass

ID#05 - Correct user information grabbed from MySQL queries

Description: Using a query to get a specific user, the information retrieved must be from the corresponding user.

Items covered by this test: Once a user logs in, the backend queries their login information to correctly display their information within the application.

Requirements addressed by this test: Requirements #4

Environmental needs: Require JDBC and MYSQL to properly run and test the database.

Intercase Dependencies: Test ID#01 is needed to since the login validation will be used for this test.

Test Procedures: Once the user logs in, the corresponding dashboard will display the user's information on a card. If the correct information is displayed the test passed.

Input Specification: The necessary input is to have the login information for the user, so a username and a password.

Output Specifications: When the user logs in, the dashboard displays the correct information for the user on a card.

Pass/Fail Criteria: The correct information is displayed on a card upon login.

ID#06 - Reserved appointments no longer listed

Description: No patient should be able to book a time and date of an already booked appointment.

Items covered by this test: When a patient tries to book an appointment, they should only be able to see dates available as opposed to being able to book an appointment at the same time as another one.

Requirements addressed by this test: Requirement #3

Environmental needs: Require JDBC and MYSQL to test the database.

Intercase Dependencies: Test ID#03 and Test ID#05

Test Procedures: Once an appointment is scheduled another patient should attempt to book an appointment at the same time as another one.

Input Specification: User needs to select the same doctor and select the same date as the already booked appointment.

Output Specifications: The listview should not display the time as the previously booked appointment so that no two appointments may be booked at the same time.

Pass/Fail Criteria: The listview for time should not display the time of already booked appointments if the date is the same.

ID#07 - Removed appointments correctly removed from database

Description: Appointments that are canceled by patients should be correctly removed from the appointments database table.

Items covered by this test: Canceled appointments should

Requirements addressed by this test: Requirement #8

Environmental needs: Requires JDBC and MYSQL to test the database.

Intercase Dependencies: Test ID#03 and Test id #05

Test Procedures: Once an appointment has been canceled, that appointment should not appear in the appointments database table

Input Specification: User needs to cancel an appointment on their main portal page.

Output Specifications: The listview should not display that appointment, and that appointments database row should be removed.

Pass/Fail Criteria: The appointment that was canceled should no longer be present in the appointments table.

10 Test Results

ID#01 - Correct user type log in

Date(s) of Execution: 10/05/21

Staff conducting tests: Jon Heikkinen

Expected Results: Doctor account would be logged into doctor dashboard, patient would log into patient dashboard

Actual Results: Both doctor and patient were able to log onto the correct pages respectively.

Test Status: PASS

ID#02 - Account creation correctly inserted into database

Date(s) of Execution: 10/05/21

Staff conducting tests: Alejandro Cabral

Expected Results: Creating an account would correctly be inserted into the MYSQL database.

Actual Results: When an account was created it properly was reflected in the MYSQL database.

Test Status: PASS

ID#03 - Appointment creation correctly inserted into database

Date(s) of Execution: 10/05/21

Staff conducting tests: Michael Chau

Expected Results: When an appointment is scheduled, the database should have the correct information about the appointment added into the MYSQL database.

Actual Results: Scheduling an appointment successfully creates and retains the information throughout the MYSQL database.

Test Status: PASS

ID#04 - Embedded map html is correct

Date(s) of Execution: 11/03/21

Staff conducting tests: Alen Peter

Expected Results: When going through appointment locations, the correct locations when clicking the “Show me Location” should be displayed in the embedded map.

Actual Results: Pressing all the maps was working properly with the correct location appearing in the embedded maps.

Test Status: PASS

ID#05 - Correct user information grabbed from mysql queries

Date(s) of Execution: 10/20/21

Staff conducting tests: Alejandro Cabral

Expected Results: When logging on the user’s correct information and appointments should be reflected through the dashboard.

Actual Results: Logging onto the user correctly displayed all of the information and appointments.

Test Status: PASS

ID#06 - Reserved appointments no longer listed

Date(s) of Execution: 11/01/21

Staff conducting tests: Michael Chau

Expected Results: When an appointment is already booked for a specific doctor, other patients should not be able to book an appointment at the same date and time.

Actual Results: When a user attempts to book an appointment that is already booked, the option for the time is not displayed so it is impossible to do so.

Test Status: PASS

ID#07 - Removed appointments correctly removed from database

Date(s) of Execution: 10/20/21

Staff conducting tests: Alen Peter

Expected Results: When a patient cancels an appointment the MYSQL database should be updated to remove the appointment and the dashboard should no longer list the appointment.

Actual Results: Cancelling an appointment deletes the appointment from the list view and the MYSQL database properly.

Test Status: PASS

11 Regression Testing

Logging between users: The program was regressively tested for any errors/bugs for multiple logins and access of program functionalities between multiple users of a type.

Multiple appointment booking: The program was tested repeatedly for accurate information on multiple booking/cancellations of appointments by a patient.

IV Inspection

12 Items to be Inspected

- Google Maps Embedded API functionality when displaying appointment location
- Account Registration
- Filter by location method
- Current Appointments Display

13 Inspection Procedures

To inspect the procedures, the team made sure that each member has submitted one function they wrote to be tested. Then all the members who did not write the code would then come up with a test case and then see the result. The reason why this method was used is because it allowed all members to handle code even if they did not originally write it. Meetings were then held at campus where we would discuss how the test case implemented may or may not have gone the way planned and the team would discuss the reason of why the result occurred.

14 Inspection Results

Google Maps Embedded API functionality when displaying appointment location

(11-01-21) 12:00pm

Michael is the one who originally submitted this function to be tested and when Alejandro tested the google maps to see if Mercy Hospital was properly displaying on an appointment, he noticed that the map was displaying the Weiss Memorial Hospital location instead. The team discussed and reviewed the code to find out that the HTML referenced in the code was the wrong filename hence the wrong place was being displayed. This was quickly resolved by inputting the correct html that was already in the directory.

Account Registration

(11-02-21) 12:30pm

Jon submitted the function he wrote for account registration to be tested and Alen was testing the methods of creating accounts and found out that the restrictions the team had in place for the password (requiring at least one uppercase letter and a number) was not correctly working. The team debugged through the method together to find out that when the user typed too quickly, the check would not be fast enough to act so the method was changed to check the entire string at the end when “Create an Account” is pressed rather than each time a character was pressed.

Filter by Location

(11-02-21) 12:30pm

Alejandro submitted the function he wrote for the Filter by Location button in the calendar page to the team. Michael found a strange bug that would not allow the user to continue at Mercy Hospital when it was selected, and the map preview was not working properly as well. When the team reviewed the code, it was found that the string was searching for the “&” sign in well that is originally in Mercy Hospital & Medical Center. The team decided to remove the string after & and shorten the string overall since it felt overly long anyways.

Current Appointments Display

(11-03-21) 12:00pm

Alen originally created the method to display the ListView that would show the current appointments throughout the application. Jon was the one who found a strange problem where the current appointments which relied on parsing the string from the listview was not working. The team discussed and thoroughly tried to figure out what was happening and eventually found that the military time in the appointments string was causing issues. This was resolved by having a check thrown in the necessary places that would detect if the time is over 12:00 and properly put it in the correct 1-12 time.

V Recommendations and Conclusions

- Google Maps Embedded API functionality when displaying appointment location
 - Passed once the html issue was quickly resolved, now properly shows the correct map when showing the location of the hospital.
- Account Registration
 - Passed so it now properly allows creation of a patient which updates the database and stores the correct information entered in the account registration page.
- Filter by Location
 - Inspection passed and the feature properly allows the patient to sort doctors by location and will also show a quick overview of the hospital in the google embedded map.
- Current Appointments Display
 - Passed so the function will now properly show the current appointments for the patient when they log onto the dashboard and use that listview when necessary.

VI Project Issues

15 Open Issues

OS Compatibility Issues

As of now the program seems to run into more issues when a windows user is using the application as opposed to Apple users. This has been narrowed down to the WebView feature and will need more attention before the product is released to the public.

Threading

Using a multithreaded application would have been more ideal due to the nature of how many people may be using the application at once. This can be resolved by converting the application methods to be more friendly towards a threaded oriented application and would simply take time.

Communications with Hospitals

This application requires the hospitals that are used in the application to agree and allow either of us to access their current database of medical staff and patients or agree to halt using any used applications such as MyChart and Epic databases and instead use our product when it comes to storing patient data in order to maximize the amount of features implemented in this product.

Communication with Pharmacies

One of the next planned features was being able to communicate with pharmacies in order for doctors to distribute prescriptions to patients. Given more time, it would be required to wire up a communication between our application and pharmacies within the network.

16 Waiting Room

Video calling feature

This feature enhances virtual consultations between doctors and patients in situations like pandemics. It also enables meetings to be held virtually between medical staff. This saves time and travel requirements that need to be fulfilled by medical staff and patients in physical presence in the doctor's office.

Chat support

A medical staff could provide chat support for patients or enquirers requesting information such as account details and prescription details with proper verification if needed. Also, chat support can guide a user in using the application with Natural Language Processing feature.

Online form submission

The program can extend the form submissions to other forms like job applications to the doctor's office, clinical practice requests, and other related services. With this option, a person won't need to be physically present in the Doctor's office to just submit applications.

Accessibility of medical records

Keeping the medical records on a tech platform and accessibility of these records through the application helps both doctors and patients access these records at ease

when needed. Also, since it is saved as a soft copy, it will be safe especially in the long term.

17 Ideas for Solutions

Video calling feature

The video calling feature can be added to the application by embedding or routing the application to an already existing third-party video calling software like Zoom or Skype. This redirection should be made available from both patients and medical staff's portals because both parties will be accessing this feature for virtual appointments.

Chat support

This can be accomplished using multithreading following the client-server model. This is available in programming languages like java. Under the client-server model, a representative from the Doctor's Office acts as a server and the patient follows a client. The server controls the chat system and decides who all will have access to the chat system. The client requests a chat service to the server and once the server provides access to the client, the client can start a conversation with the server to resolve questions through the chat.

Online form submissions

These forms can be implemented in the program like the implementation of the registration form in the current program, but the database needs to be modified.

Accessibility of medical records

In the medical staff's portal, an option could be made available to store or update medical records of a patient in a local database or cloud server having proper security features. Then in both the medical staff and patient's portal an option can be added to access the medical records with proper authentication.

18 Project Retrospective

The things that worked out well throughout this project was the ability to easily use a database throughout this project. MYSQL was a database that made it easy to implement into java due to the JDBC library which heavily assisted in the communication between the java application and the SQL language. The use of XML in the java file also made the front end much easier. Throughout the project the team used FXML files that were equivalent to XML language for java and it made programming the front end functionality much easier with the use of proper controllers. Lastly the Google Embedded Maps API really worked out well for this project simply because it had all the features the team was looking for when it came to displaying location information.

Things that did not work out well for this project would definitely be the WebView that was used to display the Google Embedded Maps API due to the bugs that were affiliated with the windows computers. The Webview sometimes did not feel as responsive as it should have and using a different tool to display the embedded map would have been more ideal. Lastly, one thing that did not go as well is the lack of threading throughout the project. The project felt limited in the features due to the lack of threading and making this a threaded project would have opened the amount of opportunities much more when it came to features.

Overall team thinks this project could have been more successful with more time devoted to having discussions about the project's features but the team is overall quite satisfied with the end result of this project.

VII Glossary

JDBC: External library that allows communication between the MYSQL server and the Java language and sql queries.

MYSQL: MySQL workbench is a application that uses the sql language and used to maintain schemas

Front-end: The visual graphical interface that the users will be able to see on the application.

Back-end: The commands that are done behind the scenes that update the front-end for the user and/or communicate with the database.

ArrayList: A type of data structure that was used within the backend to store relevant data for the users.

FXML: FXML is an XML-based user interface markup language that is meant to be used with the JavaFX framework.

WebView: A browser engine contained within the application used to display google maps throughout this project.

VIII References / Bibliography

[1] Group 12 Sebastian Greczek, Rakshitha Jayaram Gowde, Thomas Kubik, Luis Lema Segovia, Doctor's Office Project Report.

[2] J. Bell, "SE Coding_Project_Report_Template v2.0", 2018.

[3] "The Maps Embed API Overview." Google, Google, <https://developers.google.com/maps/documentation/embed/get-started>.

[4] “Lesson: JDBC Basics.” Lesson: JDBC Basics (The Java™ Tutorial & JDBC Database Access), <https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html>.

IX Index

Comparison to original project – 15

Data dictionary – 9-10

Deliverables – 10-14

Inspection procedures - 23

Inspection results – 23-24

Items to be inspected – 23

Open issues – 25

Project description – 6-10

Project retrospective 26-27

Table of contents - 3

Test results 20-22

Test specs – 15-19

UML and other notations – 8-9

No index entries found.