Final Project Report – My A.I. Doctor (MAID)

Group #4

Ryan Jones, Iye Kargbo, John Sommers

December 12th, 2023

University of Maryland Global Campus (UMGC)

CMSC495 Capstone in Computer Science

Section 7381

Dr. Majid Shaalan

# Introduction to *My A.I. Doctor*

**Bumper Video Link:** <https://www.youtube.com/watch?v=SDyU3FLo5qA>

# Overview & Summary of Individual Contributions

Ryan Jones, Iye Kargbo, and John Sommers joined forces to form Group #4, which undertook the work of developing the *My A.I. Doctor*, or *MAID*, application designed to integrate Artificial Intelligence into a Healthcare system with the goal of allowing users to identify, diagnose, and potentially treat illness and ailments. The applications consists of several elements, including User Account Management, Personal Information Managements, Health Records Managements, and A.I. Integration. These functions were implemented through an intuitive GUI interface and underscore by SQL Database Management.

Ryan Jones was responsible for the Personal Information Management part of the system, focusing his efforts on developing *ViewPerson.java* and *UpdatePerson.java* to allow users to access and amend their personal information. He also developed the underlying framework to run and handle the program using the *RunApp.java, LandingPage.java,* and *MainPage.java.* Ryan assisted with some of the Account Management and Health Records Management functions with *ViewAccount.java,* *UpdateAccount.java,* and *ViewHealthConditions.java.* He also helped with the SQL functions and A.I. integration through *SQL.java, submitMAID.java,* and *MAIDResponse.java.*

Iye Kargbo was responsible for the User Account Management aspects of the application, dedicating her time to develop *Account.java, Regitration.java,* and *Login.java*, which were integral to serve as the first part of the program where users are able to register and log into their accounts. She also provided assistance to the SQL functions and other account management functions with *SQL.java, ViewAccount.java,* and *UpdateAccount.java.*

John Sommers was responsible for the Health Records Management and A.I. Integration portions of the program. He created *HealthConditions.java,* which created the framework for medical records management and fueled the *ViewHealthConditions.java* class. John also created the initial *ChatGPTClientTest.java* class, which was used to develop the A.I. integration for MAID to include *SubmitMAID.java* and *MAIDResponse.java* classes to allow integration into the rest of the MAID framework.

# Project Plan

Our original project plan consisted of the following components 9 components; excerpts of our original project plan are highlighted below (Jones et al., 2023):

# System Design Document (SDD)

The requirements of our System Design were to create a ChatBot powered by Artificial Intelligence, or AI, that will allow users to diagnose medical conditions. This project was inspired by the research article *Self-Diagnosis through AI-enabled Chatbot-based Symptom Checkers: User Experiences and Design Considerations* authored by Yue You and Xinning Gui. The My AI Doctor program, or MAID, will help patients have illnesses diagnosed by cross-referencing current symptoms with an individual’s medical history. The system is designed to be integrated into existing medical system such as MyChart to allow seamless integration with a patient’s existing medical information.

# System Development Document

An important aspect of the development phase is identifying and understanding how data flows from the user through the application. To highlight these, we outlined each entity, node, and process involved in the MAID system. Figure 3 highlights this flow of information, detailing how information flows back and forth, and in and out of the MAID system.

**Figure 1***MAID System Flowchart*

*A diagram of a medical procedure

Description automatically generated  
Note.* Jones, 2023.

# Security Plan

Given the sensitive nature of medical data, our web application will ensure the utmost security by being compliant with the Health Insurance Portability and Accountability Act of 1996, otherwise known as HIPAA. Hence, each patient can access their account by through a unique username and password, and they will be prompted to verify their email. Data should also be stored securely, encrypting data at rest.

# Data Retention Plan

MAID’s data retention will follow the HIPAA retention policy for healthcare applications, which requires data to be stored for a minimum period of 6 years. Account information including, a user’s email address and username, will be stored until the account is prompted for deletion or if the account is inactive for a period of 10 consecutive years. Patient’s past medical health history and family health history will also be stored until the account is deleted or until the retention period is over. This allows us to dispose of outdated data and to accommodate more storage space.

# Unit and Integration Test Plans

# Our test plan breaks down MAID’s functionality into several parts: successful account creation, successful login, correct display of homepage, functional chatbot, accurate diagnosis based on given symptoms, incorporation of patient’s health history, successful logout, and automatic logoff when idle. Specific tests are detailed below.

# Conversion Plans

Currently, there is no projected need to migrate data from an existing system to another. Instead, MAID will be implemented to the existing system where the data already exists natively. Therefore, there is no strategy needed for migrating data. The only relevant strategy is accessing the existing data, which has been covered in other parts of this document.

**Implementation Plan**

As previously noted, MAID will be designed to be implemented with online healthcare systems and connect with the patient's medical history in the software and build a unique profile for them to predict what illnesses are likely to befall them. However, our implementation here will be done using a client-based system for proof of concept. Our other feature enables patients to talk to MAID about symptoms they are experiencing to get a "diagnoses". It is important to note that MAID's diagnoses aren't always 100% accurate, and must be confirmed by a licensed medical professional.

# Operations or System Administration Manual

Due to the fact that MAID will have access to sensitive PHI and PII, it will be necessary to have strict access controls for system administrators. The system will involve a tiered approach, in which the majority of administrators will only have access to administer the frontend system. Next, a smaller group will have access to the backend system, excluding the medical database and patient information. Finally, a select number of administrators will have access to all the systems and data. Using this tiered approach, each group will have different permissions with various vetting requirements, training elements, and distinct safeguards to protect patient information. This issue and approach are less of technical challenge, and more of an administrative hurdle that an organization would need to figure out.

# Training Plan

When training to use MAID, users must keep in mind that MAID is not infallible. False positives can and will occur, along with false negatives. User must take the results from MAID with a grain of salt, and always examine the facts of the patient's condition before making their own diagnosis. Our user guide will provide detailed training instructions.

# Requirements Specifications

The requirements phase began with our class being appointed to design a software application that would mimic something that would be seen in the real world, with the added challenge of coming up with a program that is both new and useful. This led our group to choose the truly challenging task of creating MAID, which would integrate A.I. into the medical treatment process. This project decision is what determined our specific requirements.

**Language**

Our group was not prescribed a specific programming language for our project but was instead given the freedom to choose. For that reason, we had a self-imposed requirement to find a language we all knew well enough to create a new and unique program. Hence, we decided to go with Java with our backup option being Python.

**User Interface**

Our next task was to decide on the type of interface we would use for our project. It was quickly apparent that a Command-Line Interface would not be an appropriate option for a Capstone project nor for our specific application. That left us with the option of a Graphical-User Interface, which could either be a client-based application or web-based application. We opted for a client-based application due to the hardships we’d experienced in previous classes with configuring web-based applications across multiple user platforms. Additionally, our group’s familiarity with Java Swing libraries for building GUIs.

**Development Environment**

Another task was choosing a platform to use for our development environment, which led us to IntelliJ IDEA as it provides extensive resources for assisting in the creation of GUIs through Java’s Swing framework. We needed to ensure we all used the same platform to help provide continuity and consistency across all group members.

**Data Storage**

Our group had a challenge with determining how we would store data for our application. Our initial plan was to start our development using a JSON file to store and load user data for our test environment. However, it quickly became evident that this was not a long-term solution for a production environment, nor was it feasible given the complexity of the data we would need to store. Instead, we decided that we would implement a proper database, which we used MySQL to build databases and tables for our system. This allowed us to use powerful SQL commands to load, store, and edit data for our application.

**Modularity**

Our final requirement was to design a system that was modular and scalable to meet emerging needs. We met this challenge by establishing separate classes for the various pages, tables, and functions we would need to execute our application. This resulted in distinct classes like *ViewAccount.java* and *UpdateAccount.java*, which gave use much more flexibility. The most notable example of modularity was *SQL.java,* which gave our application a central repository of SQL-related functions and commands that could be accessed from throughout the program. This severely minimized the duplication of code throughout our application, as our program relies heavily on SQL to access information stored in the MAID database.

# System Specifications

Product Name: My A.I. Doctor (MAID)

Supported Operating Systems: Windows, Linux, or MacOS

Software Requirements: IntelliJ IDEA with Java Swing GUI Support

Java Version: JDK 21 (Preferred)

**User Guide**

**Video Walkthrough:** <https://www.youtube.com/watch?v=rTF_QLGb7lw>

**Step 1:** Download the .zip file containing all the requisite files. Download this file and extract the contents to a folder. From there, we recommend you use IntelliJ IDEA and open the aforementioned folder as an IDEA project.

**Step 2:** Once you have opened the project in IDEA, navigate to the RunApp.java file. Run this file by clicking the green run arrow pictured here:



**Step 3:** Once you do this, the application will start, and you will be greeted with this window:

A screenshot of a computer

Description automatically generated

**Step 4:** Click on the Register button. Once you do this, you will be greeted with this window:

A screenshot of a computer screen

Description automatically generated

**Step 5:** Fill out the information in these fields and then click Register. Please be sure that your password meets the following criteria, or else you will be met with this message:

A screenshot of a computer password

Description automatically generated

**Step 6:** Once logged in, you will see this window. From here you can choose which of the five options you would like to do: View Account Info, View Personal Info, View Health Conditions, Check Symptoms with MAID, or Logout.

A screenshot of a computer

Description automatically generated

**Step 7a:** Let’s take a look at View Account Info first. Here you can view the Username, Email Address, or Password for your Account:

A screenshot of a computer screen

Description automatically generated

**Step 7b:** You can also update your name by clicking the update button and this will take you here:

A screenshot of a computer screen

Description automatically generated

**Step 8:** Now let’s take a look at View Personal Info. Here you can view and update your first name, last name, height, age, gender, sex, marital status, and insurance company.

A screenshot of a computer screen

Description automatically generated

**Step 9:** Let’s take a look at View Health Conditions now. Here you can view information related to a health condition such as the onset date, ailment name, whether it is chronic or not, the date it was first reported, whether it is medicated or not, and what medicine you take if it is medicated.

A screenshot of a medical record

Description automatically generated

**Step 10a:** Lastly, we will take a look at Check Symptoms with MAID. Here you can submit the symptoms you are currently experiencing, and MAID will respond with its suggestions as to what might be ailing you.

A screenshot of a medical form

Description automatically generated

**Step 10b:** Here is what its response will look like:

A screenshot of a computer

Description automatically generated

**Step 11:** Once you are all done, click the logout button and the application will close.

**Test Plan**

# Objective:

# Ensure MAID components work individually, as well as an entire system to prove the application to be successful, secure, and helpful, or instead highlight weak areas in security and soundness of medical advice. System components include login functionality, patient updates, and A.I. Chatbot interface, as well as processing and storage of user credentials, patients’ medical information, and A.I. symptom screening.

# Scope:

Testing will include functionality tests of individual application components throughout the development stage, as well as a complete, complex test of the entire application upon completion. Test cases will include those for logging into the system, updating patient information, updating known health conditions, Chatbot integration, logging out, and secure system logging.

# Coverage:

Our goal for our test cases is 100% coverage of individual application functions and methods. The intent is to use standalone test cases for individual components as well as full run-throughs.

# Risk:

The most substantial risk that could occur is the possibility of leaking sensitive data, especially during the test phase. However, this has been mitigated through the use of test data as opposed to authentic patient data, so there is no risk should exposure occur.

# Data:

We will generate sample data using online tools to choose patient information, health conditions, and symptoms for our test data, so no genuine data will be used. This is due to privacy concerns of medical information.

# Originality:

Our test plan will explore and assess our original idea for MAID, but will also use time-honored techniques of unit testing.

# Communication:

In the real world, we would use the User Acceptance Testing (UAT) process to receive feedback from our main stakeholders: end users. Our target audience, such as patients, will interact with MAID in real-world conditions and give us feedback on the usability of the software and any errors or defects that arise during their use. This enables us to identify and address issues and necessary improvements to achieve user satisfaction and ensure high usability, functionality, and alignment with the real world.

# Usefulness:

This test plan has helped us discover and react to the right problems early. We were able to save time by the early identification and elimination of redundant testing, plan for associated risks and how to mitigate them, and develop an efficient method to generate adequate valid and invalid test data.

# Completeness:

Our test plan will cover all aspects of and functions of MAID, ensuring that nothing is left untested.

# Insightfulness:

Development of our test plan has resulted in our group identify potential issues, solutions and resources that we should keep in mind throughout the development process. We believe our test plan is insightful and shows understanding of what is pertinent, interesting, and challenging in developing and testing the MAID system.

# Test Plan

**Objective:** To ensure that the MAID functions correctly, including the Chat AI feature, and meets the requirements.

**Test Data Preparation**

1. Create test data files with various scenarios for Person and Health Condition entries.
2. Develop test cases based on expected inputs and outputs, including interactions with the Chat AI.

**Test Scenarios**

1. **Login Functionality**

1. Test valid login credentials.
2. Test invalid login credentials.
3. Test login with special characters in the username and password.

2. **Entering Person Information**

1. Verify that a new person can be created with valid data.
2. Test saving a person with missing or invalid data.

3. **Adding HealthConditions**

1. Verify that health conditions can be added to a person.
2. Test adding health conditions with missing or invalid data.

4. **Chatting with the Chat AI**

1. Verify that patients can engage in a conversation with the Chat AI.
2. Test various patient inquiries and responses from the Chat AI.

5. **Viewing and Editing Data**

1. Verify that user information and health conditions can be viewed and edited.
2. Test editing and saving changes for a person's details and health conditions.

6. **Logging Out**

1. Test the logout functionality to ensure the user's session is terminated.

**Test Execution**

1. Execute test cases according to the defined scenarios.
2. Document the results, including any issues encountered.
3. Report any defects to the development team for resolution.

**Test Completion**

1. Ensure that all test cases have been executed and documented.
2. Verify that the system, including the Chat AI feature, functions correctly and meets the requirements.
3. Provide feedback to the development team regarding any issues encountered during testing.

**User Acceptance Testing (UAT)**

1. Engage actual users, including patients, to perform User Acceptance Testing.
2. Gather feedback and ensure that the system, including the Chat AI feature, aligns with their needs and expectations.

**Test Results**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test 1** | **Input** | **Expected Results** | **Results** | **Pass/Fail** |
| MAID Application Display | Run the “RunApp” program | The Landing Page displays correctly in the center of the screen. Login, Register, and Exit Application Buttons are visible. | After running the program, RunApp.java, the result was assessed to determine if it matches the expected results. The panel displayed correctly. | Pass |

**Screenshots for Test 1**

A screen shot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test 2** | **Input** | **Expected Results** | **Results** | **Pass/Fail** |
| Registration functionality | Input 1: Username = iyekargbo; the rest of the fields = blank  Input 2: Username = iyekargbo; Email = iyekargbo@gmail.com; Password = password; Confirm Password = password  Input 3: Username = iyekargbo; Email = iyekargbo@gmail.com; Password = T35t@!yE; Confirm Password = password12345  Input 4: Username = iyekargbo; Email = iyekargbo@gmail.com; Password = T35t@!yE; Confirm Password = T35t@!yE | Input 1: A notification stating, "All fields are required" is shown.  Input 2: A notification of an invalid password and the required password specifications is shown.  Input 3: A notification stating, "Passwords do not match" is shown.  Input 4: The registration is successful, and the user is directed to the login page. The database “maid2” is created and the account information is added to the accounts table. | Input 1: The error message is displayed, and the user is prompted to fill out all the fields.  Input 2: The error message is displayed, and the user is prompted to try again with the required password specifications.  Input 3: The error message is displayed, and the user is prompted to ensure that the passwords match.  Input 4: The user is successfully registered, and their information is successfully added to the database. They are also successfully directed to the login page. | Pass |

**Screenshots for Test 2, Input 1**

A screenshot of a computer

Description automatically generated

**Screenshot for Test 2, Input 2**

A screenshot of a computer

Description automatically generated

**Screenshots for Test 2, Input 3**

A screenshot of a computer

Description automatically generated

**Screenshots for Test 2, Input 4**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test 3** | **Input** | **Expected Results** | **Results** | **Pass/Fail** |
| Login Functionality | Input 1: Email/Username = iyekargbo; Password = blank  Input 2: Email/Username = iyekargbo; Password = Password$123  Input 3: Email/Username = iyekargbo; Password = T35t@!yE | Input 1: A notification stating, "All fields are required" is shown.  Input 2: A notification stating "Invalid email/username or password" is shown.  Input 3: The login is successful, and the user is directed to the main page. | Input 1: The error message is displayed, and the user is prompted to fill out all the fields.  Input 2: The error message is displayed, and the user is prompted to try again with their valid credentials.  Input 3: The user is successfully logged in, and they are successfully directed to the main page. | Pass |

**Screenshot for Test 3, Input 1**

A screenshot of a computer

Description automatically generated

**Screenshot for Test 3, Input 2**

A screenshot of a computer

Description automatically generated

**Screenshots for Test 3, Input 3**

A screenshot of a computer

Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test 4** | **Input** | **Expected Results** | **Results** | **Pass/Fail** |
| View and Edit Account Information | Input 1: Click on “View Account Info”  Input 2: Update username from iyekargbo to iye | Input 1: A panel is displayed, showing the user’s username, email, and password.  Input 2: The username updates from iekargbo to iye. | Input 1: The panel was successfully displayed, and the user’s information is correct.  Input 2: The username is successfully updated, and the database demonstrates the change. | Pass |

**Screenshot for Test 4, Input 1**

A screen shot of a form

Description automatically generated

**Screenshots for Test 4, Input 2**

A screen shot of a computer screen

Description automatically generated

A screen shot of a form

Description automatically generated

A screenshot of a computer

Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test 5** | **Input** | **Expected Results** | **Results** | **Pass/Fail** |
| View and Edit Personal Information | Input 1: Click on “View Personal Info”  Input 2: Update first name to Iye | Input 1: A panel is displayed, showing the user’s personal information.  Input 2: The first name along with other personal information can be edited. | Input 1: The panel was successfully displayed.  Input 2: The user is prompted to enter their first name and confirm the change. | Pass |

**Screenshot for Test 5, Input 1**

A screenshot of a computer screen

Description automatically generated

**Screenshot for Test 5, Input 2**

A screenshot of a computer screen

Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test 6** | **Input** | **Expected Results** | **Results** | **Pass/Fail** |
| View and Edit Health Information | Click on “View Health Conditions” | A panel is displayed, showing the user’s health record and a button to add a new condition. | The panel was successfully displayed. | Pass |

**Screenshot for Test 6**

A medical record form with text

Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test 7** | **Input** | **Expected Results** | **Results** | **Pass/Fail** |
| Chatting with the AI | Click on “Check Symptoms with MAID” | A panel is displayed, allowing the user to enter their current symptoms and submit. Due to time constraints, MAID notifies the user of a failed operation and ask them to try again. | The expected results match the actual results. The user can successfully enter their symptoms and submit, and they are successfully notified. | Pass |

**Screenshots for Test 7**

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

**Design & Alternate Designs**

Our primary design was to use a client-based GUI application using the Java language with Java Swing libraries. We had considered other ideas for our application, including implementing a web-based application with either Java or Python, although more-likely choosing Python and HTML. Another design consideration we had was using a JSON file for storing system and user-related data, however eventually opted for a MySQL database. We did not pursue these alternative design options due to the requirements laid out in the previous *Requirements Specifications* section, which made these alternative designs unrealistic for our project. All other design options were consistent through the development phase.

**Development History**

**Phase 1**

Our first phase began with developing the basic look we wanted for MAID, along with its key functionalities. This included developing classes for creating and viewing accounts, as well classes for viewing personal information and health records. This is where the project initially took on its form.

**Phase 2**

This second phase saw a lot of added functionality, primarily through the establishment of our database to store user information, along with the necessary SQL commands needed to access that information. This phase is also where we successfully implemented our registration and login classes, as well as the functionality necessary to update the account information and health records noted in the previous phase.

**Phase 3**

This third and final phase is where our project came together, allowing integration of all the elements from the previous weeks, while also developing the framework needed for the artificial intelligence integration through OpenAI’s ChatGPT application. During this time we also added functionality to the program landing page and user’s homepage in addition to expanding the SQL functionality.

**Conclusion**

**Lessons Learned**

Our primary lessons learned were to be realistic with our project goals and to communicate continuously. We wouldn’t necessarily call our project unrealistic, however there were a lot of unknowns going into our project that made it extremely difficult to execute, due to current technology constraints. We would have been better off to not include A.I. integration into a short-term project but in the end, we learned a lot about A.I. despite the limitations that will be covered below. Another key lesson in this project involved learning how to communicate effectively. Specifically, we had to assess our natural team dynamic to determine what proper communication would look like. In the end we got into a rhythm of communicating via instant messaging and group calls to discuss the project and move forward.

**Design Strengths**

Our greatest strength may have been our greatest flaw, which was the originality of our idea. While it was great in theory, it proved to be more difficult to implement A.I. than we had anticipated. Regardless, our project design was really good in that it developed a framework where all the integral aspects were combined to support the A.I. integration. This included developing individual user accounts, which in turn were linked to both personal and medical information about the corresponding user. These features alone proved to be a challenge for us in terms of creating functionality, but it ended up being a good representation of real-world systems found in the healthcare industry.

**Limitations**

Our two biggest limitations were time and accessibility to necessary APIs for our project. As was likely the case with other groups, time was our biggest limiting factor when creating our application, as we didn’t realize how quickly our 8 weeks would fly by. It may have partially been a lack of thorough planning on our part, but I think the bigger issue was not foreseeing the challenges we would face, and the time required to troubleshoot and correct those problems. We ended up devoting time to troubleshoot functionality problems that took away time from other security issues. For instance, one of the considerable challenges we had was figuring out an efficient way to create a session for the current user that would track the user’s identity throughout the program to allow SQL queries. We ended up spending so much time finding a solution for our given project that we ran out of time to devote to additional input validation to increase security. For our group that was especially frustrating, given the necessity of sound and secure coding practices, which would definitely be addressed in a production environment.

Our other challenge that we had was finding the appropriate APIs for integrating the artificial intelligence components of our project. We had originally envisioned an A.I. framework geared specifically towards to medical use; however, did not find one in our research and knew the impossibility of developing one internally for our project. In reality, investing in this project would also require investment in developing the A.I. framework to support it. For our application, we ended up settling on OpenAI’s ChatGPT framework, which initially appeared like it would allow us to integrate its APIs into our program for free. In the end, we were unable to integrate it fully due to cost/licensing issues. Instead, we ended up developing the code for integration and providing sample output. This felt like a huge hit to our project initially, but in the end our group is happy for our originality and providing a proof of concept on a new idea, as well as created a functioning healthcare management platform.

**Suggestions**

The only suggestion from our group in hindsight would be reminding students early on that their project idea doesn’t necessarily need to be lofty nor ground-breaking, but rather should be something that is realistic given both the time constraints and technology constraints. I would also urge that students carefully assess what their strengths and weaknesses and identify that when forming groups. Our group was fortunate in that we had a good contrast between our individual strengths and weaknesses, so we were able to make up for each other’s weaknesses, which was helpful in carrying our project as far as we did.

**References**

Jones, R. (2023) *MAID flowchart* [Diagram]. University of Maryland Global Campus.

Jones, R., Kargbo, I., Sommers, J. (2023) *HW2: Project plan*. University of Maryland Global Campus.