Module Interface Specification for SFWRENG 4G06A

Team #25, RapidCare
Pranav Kalsi
Gurleen Rahi
Inreet Kaur
Moamen Ahmed

January 17, 2025

1 Revision History

Date	Version	Notes
Jan 14, 2025	1.1	Initial Document
Jan 17,2025	1.2	Revised Document Incorporating Feedback

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at SRS document

Symbol	Description
MG	Module Guide
M	Module
MIS	Module Interface Specification
API	Application Programming Interface
MFA	Multi-Factor Authentication

Contents

1 Revision History **i**

2 Symbols, Abbreviations and Acronyms **ii**

3 Introduction **1**

4 Notation **1**
4.1 Primitive Data Types . . . 1
4.2 Imported Data Types . . . 2

5 Module Decomposition **2**

6 MIS of User Authentication Module **4**
6.1 Module . . . 4
6.2 Uses . . . 4
6.3 Syntax . . . 4
6.3.1 Exported Constants . . . 4
6.3.2 Exported Access Programs . . . 4
6.4 Semantics . . . 4
6.4.1 State Variables . . . 4
6.4.2 Environment Variables . . . 4
6.4.3 Assumptions . . . 4
6.4.4 Access Routine Semantics . . . 4
6.4.5 Local Functions . . . 5

7 MIS of Administrator View Module **6**
7.1 Module . . . 6
7.2 Uses . . . 6
7.3 Syntax . . . 6
7.3.1 Exported Constants . . . 6
7.3.2 Exported Access Programs . . . 6
7.4 Semantics . . . 6
7.4.1 State Variables . . . 6
7.4.2 Environment Variables . . . 6
7.4.3 Assumptions . . . 6
7.4.4 Access Routine Semantics . . . 6
7.4.5 Local Functions . . . 7

8 MIS of Patient View Module **8**
8.1 Module . . . 8
8.2 Uses . . . 8
8.3 Syntax . . . 8

8.3.1 Exported Constants . . . 8
8.3.2 Exported Access Programs . . . 8
8.4 Semantics . . . 8
8.4.1 State Variables . . . 8
8.4.2 Environment Variables . . . 8
8.4.3 Assumptions . . . 8
8.4.4 Access Routine Semantics . . . 8
8.4.5 Local Functions . . . 9

9 MIS of Broker Module **10**
9.1 Module . . . 10
9.2 Uses . . . 10
9.3 Syntax . . . 10
9.3.1 Exported Constants . . . 10
9.3.2 Exported Access Programs . . . 10
9.4 Semantics . . . 10
9.4.1 State Variables . . . 10
9.4.2 Environment Variables . . . 10
9.4.3 Assumptions . . . 11
9.4.4 Access Routine Semantics . . . 11
9.4.5 Local Functions . . . 12

10 MIS of Administrator Model Module **13**
10.1 Module . . . 13
10.2 Uses . . . 13
10.3 Syntax . . . 13
10.3.1 Exported Constants . . . 13
10.3.2 Exported Access Programs . . . 13
10.4 Semantics . . . 13
10.4.1 State Variables . . . 13
10.4.2 Environment Variables . . . 13
10.4.3 Assumptions . . . 13
10.4.4 Access Routine Semantics . . . 13
10.4.5 Local Functions . . . 14

11 MIS of Patient Model Module **15**
11.1 Module . . . 15
11.2 Uses . . . 15
11.3 Syntax . . . 15
11.3.1 Exported Constants . . . 15
11.3.2 Exported Access Programs . . . 15
11.4 Semantics . . . 15
11.4.1 State Variables . . . 15

11.4.2 Environment Variables . . . 15
11.4.3 Assumptions . . . 15
11.4.4 Access Routine Semantics . . . 15
11.4.5 Local Functions . . . 16

12 MIS of Transcription Module **17**
12.1 Module . . . 17
12.2 Uses . . . 17
12.3 Syntax . . . 17
12.3.1 Exported Constants . . . 17
12.3.2 Exported Access Programs . . . 17
12.4 Semantics . . . 17
12.4.1 State Variables . . . 17
12.4.2 Environment Variables . . . 17
12.4.3 Assumptions . . . 17
12.4.4 Access Routine Semantics . . . 17
12.4.5 Local Functions . . . 18

13 MIS of Classification Module **20**
13.1 Module . . . 20
13.2 Uses . . . 20
13.3 Syntax . . . 20
13.3.1 Exported Constants . . . 20
13.3.2 Exported Access Programs . . . 20
13.4 Semantics . . . 20
13.4.1 State Variables . . . 20
13.4.2 Environment Variables . . . 20
13.4.3 Assumptions . . . 20
13.4.4 Access Routine Semantics . . . 20
13.4.5 Local Functions . . . 21

14 MIS of Diagnosis Prediction Module **22**
14.1 Module . . . 22
14.2 Uses . . . 22
14.3 Syntax . . . 22
14.3.1 Exported Constants . . . 22
14.3.2 Exported Access Programs . . . 22
14.4 Semantics . . . 22
14.4.1 State Variables . . . 22
14.4.2 Environment Variables . . . 22
14.4.3 Assumptions . . . 22
14.4.4 Access Routine Semantics . . . 23
14.4.5 Local Functions . . . 23

15 MIS of Medicine Prediction Module **24**
15.1 Module . . . 24
15.2 Uses . . . 24
15.3 Syntax . . . 24
15.3.1 Exported Constants . . . 24
15.3.2 Exported Access Programs . . . 24
15.4 Semantics . . . 24
15.4.1 State Variables . . . 24
15.4.2 Environment Variables . . . 24
15.4.3 Assumptions . . . 24
15.4.4 Access Routine Semantics . . . 25
15.4.5 Local Functions . . . 25

16 MIS of Administrator Account Management Module **26**
16.1 Module . . . 26
16.2 Uses . . . 26
16.3 Syntax . . . 26
16.3.1 Exported Constants . . . 26
16.3.2 Exported Access Programs . . . 26
16.4 Semantics . . . 26
16.4.1 State Variables . . . 26
16.4.2 Environment Variables . . . 26
16.4.3 Assumptions . . . 26
16.4.4 Access Routine Semantics . . . 26
16.4.5 Local Functions . . . 27

17 MIS of Patient Account Management Module **28**
17.1 Module . . . 28
17.2 Uses . . . 28
17.3 Syntax . . . 28
17.3.1 Exported Constants . . . 28
17.3.2 Exported Access Programs . . . 28
17.4 Semantics . . . 28
17.4.1 State Variables . . . 28
17.4.2 Environment Variables . . . 28
17.4.3 Assumptions . . . 28
17.4.4 Access Routine Semantics . . . 28
17.4.5 Local Functions . . . 29

3 Introduction

The following document details the Module Interface Specifications for the RapidCareapplication.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/PKALXI/RapidCare/blob/main/docs/Design/SoftArchitecture/MG.pdf.

4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1|c_2 \Rightarrow r_2|...|c_n \Rightarrow r_n)$.

4.1 Primitive Data Types

The following table summarizes the primitive data types used by RapidCare.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	int	a number without a fractional component in (-∞, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)
boolean	boolean	value of true or false

The specification of RapidCare uses some derived data types: sequences, strings, and tuples, maps. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. Maps contain key-value pairs. In addition, RapidCare uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

4.2 Imported Data Types

Data Type	Notation	Description
FormData	FormData	A built-in browser API object used to construct a set of key/value pairs representing form fields and their values for HTTP requests. The keys and values are arbitrary to the use case.
TensorFlow Sequential Model	tf.sequential	A deep learning model architecture from TensorFlow that allows layers to be stacked sequentially

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	Level 2
Hardware-Hiding	None
Behaviour-Hiding	User Authentication Module Administrator View Module Patient View Module Administrator Model Module Patient Model Module Broker Module Administrator Account Management Module Patient Account Management Module
Software Decision	Transcription Module Classification Module Diagnosis Prediction Module Medicine Prediction Module

Table 1: Module Hierarchy

6 MIS of User Authentication Module

6.1 Module

UserAuthentication

6.2 Uses

Firebase Auth

6.3 Syntax

6.3.1 Exported Constants

isAuthenticated: boolean

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
Auth	-	React.component	RenderError

6.4 Semantics

6.4.1 State Variables

isUserAdmin: boolean

6.4.2 Environment Variables

N/A

6.4.3 Assumptions

N/A

6.4.4 Access Routine Semantics

Auth():

- transition: Renders the login page on the screen.
- output: N/A
- exception: RenderError — Thrown if the component fails to render.

6.4.5 Local Functions

login():

- transition: Renders adminstrator view page if user is administrator otherwise renders patient view page.
- output: N/A
- exception: InvalidCredentials - Thrown if the user enters invalid credentials.

ResetPassowrd():

- transition: Sends a reset link to the provided email and renders the login page.
- output: N/A
- exception: InvalidInpuError - Thrown if user input a invalid or unregistered email.

7 MIS of Administrator View Module

7.1 Module

Administrator

7.2 Uses

Broker Module
ReactJS

7.3 Syntax

7.3.1 Exported Constants

N/A

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
Administrator	-	React.component	RenderError

7.4 Semantics

7.4.1 State Variables

isAuthenticated: boolean

7.4.2 Environment Variables

Screen interface
Keyboard
Microphone

7.4.3 Assumptions

User has a functional screen, keyboard, and Microphone.

7.4.4 Access Routine Semantics

Administrator():

- transition: Renders a react component of the administrator view page.
- output: N/A
- exception: RenderError — Thrown if the component fails to render.

7.4.5 Local Functions

handleAdminAccount(id: String, record: FormData, requestType: String):

- transition: Sends an API request to the broker Module to process an add, delete, or update operation in the Administrator Database.
- output: N/A
- exception: InvalidInputError - Thrown if the formData is missing a field or is invalid.

validateInput(InputField: String):

- transition: Renders a success or error message outlining the action performed.
- output: N/A
- exception: InvalidInputError - The input data is incomplete or invalid.

8 MIS of Patient View Module

8.1 Module

Patient

8.2 Uses

Broker Module
ReactJS

8.3 Syntax

8.3.1 Exported Constants

N/A

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
Patient	-	React.component	RenderError

8.4 Semantics

8.4.1 State Variables

isAuthenticated: boolean

8.4.2 Environment Variables

Screen interface
Keyboard
Microphone

8.4.3 Assumptions

User has a functional screen, keyboard, and microphone.

8.4.4 Access Routine Semantics

Patient():

- transition: Renders a react component of the patient view page.

- output: N/A
- exception: RenderError — Thrown if the component fails to render.

8.4.5 Local Functions

handlePatientAccount(id: String, record: FormData, requestType: String):

- transition: Sends an API request to the API Module to process an add, delete, or update operation in the Patient Database.
- output: N/A
- exception: InvalidInputError - Thrown if the formData is missing a field or is invalid.

validateInput(InputField: String):

- transition: Renders a success/error message outlining the action performed.
- output: N/A
- exception: InvalidInputError - The input data is incomplete or invalid.

9 MIS of Broker Module

9.1 Module

Broker

9.2 Uses

- Transcription Module
- Classification Module
- Diagnosis Prediction Module
- Medicine Prediction Module
- Administrator Account Management Module
- Patient Account Management Module

9.3 Syntax

9.3.1 Exported Constants

N/A

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
`getToken`	-	Token : String	InvalidTokenError
`transcribeText`	request : FormData	transcribedText : String	FailedResponseError
`classifyText`	request : FormData	classifiedText : map	FailedResponseError
`predictDiagnosis`	request : FormData	applicableDiagnosis : String	FailedResponseError
`predictedMedicine`	request : FormData	applicableMedicine : String	FailedResponseError

9.4 Semantics

9.4.1 State Variables

- secretKey : String

9.4.2 Environment Variables

N/A

9.4.3 Assumptions

- Requires a stable database connection.
- All end points in distributed systems are up.

9.4.4 Access Routine Semantics

getToken()-¿ String:

- **Transition:** N/A
- **Output:** Issues an access token with encoded with the `secretKey`
- **Exception:** InvalidTokenError : Invalid or expired authorization code.

transcribeText(request : FormData)-¿ String:

- **Transition:** N/A
- **Output:** Check if request header has valid token with `authorize`, then return live transcription of audio bytes in the request.body.
- **Exception:** FailedResponseError: The corresponding service/module has returned an error.

classifyText(request : FormData) -¿ map:

- **Transition:** N/A
- **Output:** Check if request header has valid token with `authorize`, then classify the request.text into the fields given in request.chart which represents the medical chart data in FormData form. Return map of text classified.
- **Exception:** FailedResponseError: The corresponding service/module has returned an error.

predictDiagnosis(request : FormData) -¿ String:

- **Transition:** N/A
- **Output:** Check if request header has valid token with `authorize`, then provide stream of diagnosis suggestions based on request.chart which represents the medical chart data in FormData form.
- **Exception:** FailedResponseError: The corresponding service/module has returned an error.

predictMedicine(request : FormData)-¿ String:

- **Transition:** N/A
- **Output:** Check if request header has valid token with `authorize`, then provide stream of medicine suggestions based on request.chart which represents the medical chart data in FormData form.
- **Exception:** FailedResponseError: The corresponding service/module has returned an error.

9.4.5 Local Functions

authorize(header : String) -¿ boolean:

- **Transition:** N/A
- **Output:** This is a header function to make sure the request is authorized, here this function returns whether the request is authorized.
- **Exception:** Invalid client credentials.

10 MIS of Administrator Model Module

10.1 Module

AdminModel

10.2 Uses

N/A

10.3 Syntax

10.3.1 Exported Constants

N/A

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
AdminModel	-	-	-

10.4 Semantics

10.4.1 State Variables

name: String age: int location: String profession: String

10.4.2 Environment Variables

N/A

10.4.3 Assumptions

N/A

10.4.4 Access Routine Semantics

getter(): This is the boiler state of this variable where it will get a certain start and return it.

- transition: N/A
- output: The output depends on the data type of the parameter. It returns the current value of the requested data element.
- exception: N/A

setter(): This is the boiler state of this variable where it will get a certain start and return it.

- transition: Updates the internal state of the data model, either by adding or updating data. This also changes certain state variables.
- output: N/A
- exception: N/A

10.4.5 Local Functions

init(inputField: String):

- transition: N/A
- output: Necessary data structures and connections are made to manage and access data.
- exception: N/A

11 MIS of Patient Model Module

11.1 Module

PatientModule

11.2 Uses

N/A

11.3 Syntax

11.3.1 Exported Constants

N/A

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
PatientModel	-	-	-

11.4 Semantics

11.4.1 State Variables

patientName : String dateOfBirth : date age : int gender : String address : String email : String contactNumber : String allergies : String medicalHistory : String medications : String insuranceInfo : String

11.4.2 Environment Variables

N/A

11.4.3 Assumptions

N/A

11.4.4 Access Routine Semantics

getter(): This is the boiler state of this variable where it will get a certain start and return it.

- transition: N/A
- output: The output depends on the data type of the parameter. It returns the current value of the requested data element.

- exception: N/A

setter(): This is the boiler state of this variable where it will get a certain start and return it.

- transition: Updates the internal state of the data model, either by adding or updating data. This also changes certain state variables
- output: N/A
- exception: N/A

init(inputField: String):

- transition: N/A
- output: Necessary data structures and connections are made to manage and access data.
- exception: N/A

11.4.5 Local Functions

N/A

12 MIS of Transcription Module

12.1 Module

TranscriptionModule

12.2 Uses

N/A

12.3 Syntax

12.3.1 Exported Constants

N/A

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
TranscriptionAudio	byte[]	TranscribedText: String	InvalidInputError

12.4 Semantics

12.4.1 State Variables

N/A

12.4.2 Environment Variables

N/A

12.4.3 Assumptions

N/A

12.4.4 Access Routine Semantics

transMod(audioData: byte[]):

- transition: N/A
- output: Transcribed text transcribed from the audio bytes.
- exception: InvalidInputError - If the bytes could not be converted to text.

12.4.5 Local Functions

N/A

13 MIS of Classification Module

13.1 Module

ClassificationModule

13.2 Uses

N/A

13.3 Syntax

13.3.1 Exported Constants

N/A

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
Classification	TranscribedText: String	ClassifiedText: String	-

13.4 Semantics

13.4.1 State Variables

N/A

13.4.2 Environment Variables

N/A

13.4.3 Assumptions

- It is assumed that the transcribed text is in English language.

13.4.4 Access Routine Semantics

ClassifyModule(TranscribedText: String):

- transition: N/A
- output: Classified text generated for report generation.
- exception: N/A

13.4.5 Local Functions

N/A

14 MIS of Diagnosis Prediction Module

14.1 Module

DiagnosisPred

14.2 Uses

- Tensorflow
- Scikit-Learn
- Flask
- Flask-CORS

14.3 Syntax

14.3.1 Exported Constants

predictedDiagnosis : String

14.3.2 Exported Access Programs

Name	In	Out	Exceptions
diagnosePatient	request:FormData	Prediction of possible Diagnosis : String	InputDimError

14.4 Semantics

14.4.1 State Variables

- model: tf.sequential

14.4.2 Environment Variables

N/A

14.4.3 Assumptions

- Patients are not making up symptoms and all input features are accurate.

14.4.4 Access Routine Semantics

diagnosePatient(request : FormData):

- transition: N/A
- output: Returns the predicted diagnosis for the patient based on preprocessed input given by `preProcessData`. FormData is expected to contain the key-value format of (past medical history:String, symptoms:String, user characteristics including age:int and weightLint, and physician confirmed diagnosis:String)
- exception: InputDimError - The expected request body items were not received or were in the wrong format.

14.4.5 Local Functions

preProcessData(pastHistory: String, symptoms: String, user -¿ {age, weight}):

- transition: N/A
- output: Preprocess the text using TF-ID and normalize the continuous inputs and return the preprocessed data.
- exception: InputDimError - The expected arguments were not received or were in the wrong format.

15 MIS of Medicine Prediction Module

15.1 Module

MedPred

15.2 Uses

- Tensorflow
- Flask
- Flask-CORS
- Scikit-Learn

15.3 Syntax

15.3.1 Exported Constants

possibleMedicine : String

15.3.2 Exported Access Programs

Name	In	Out	Exceptions
medicatePatient	request : FormData	Prediction of possible Medicine : String	InputDimError

15.4 Semantics

15.4.1 State Variables

- model: tf.sequential

15.4.2 Environment Variables

N/A

15.4.3 Assumptions

- All input features are accurate.

15.4.4 Access Routine Semantics

medicatePatient(request : FormData):

- transition: N/A
- output: preprocess the request body (FormData) which contains the has the key-value format of (past medical history:String, symptoms:String, user characteristics including age:int and weightLint, and physician confirmed diagnosis:String)
- exception: InputDimError - The expected request body items were not received or were in the wrong format.

15.4.5 Local Functions

preProcessData(diagnosis: String, pastHistory: String, symptoms: String, user -¿ {age, weight}):

- transition: N/A
- output: Preprocess the text using TF-ID and normalize the continuous inputs, finally the diagnosis will be encoded using a LabelEncoder then return the preprocessed data.
- exception: InputDimError - The expected arguments were not received or were in the wrong format.

16 MIS of Administrator Account Management Module

16.1 Module

AdministratorAccountManagement

16.2 Uses

N/A

16.3 Syntax

16.3.1 Exported Constants

N/A

16.3.2 Exported Access Programs

Name	In	Out	Exceptions
addHealthcareProfessional	id:String, value: FormData	-	InvalidInputError
deleteHealthcareProfessional	id:String	-	IdNotFound
updateHealthcareProfessional	id:String, value: FormData	-	InvalidInputError

16.4 Semantics

16.4.1 State Variables

dbConnection : Database connection point.

16.4.2 Environment Variables

N/A

16.4.3 Assumptions

Database containing the account information for healthcare professionals exists.

16.4.4 Access Routine Semantics

addHealthcareProfessional(id:String, record: FormData):

- transition: Adds a new document with the provided details to the database.

- output: N/A
- exception: InvalidInputError - The input data is incomplete or invalid or a duplicate document already exists.

deleteHealthcareProfessional(id:String):

- transition: Deletes the corresponding document from the database.
- output: N/A
- exception: IdNotFound - Provided id is invalid or does not exist.

updateHealthcareProfessional(id:String, record: FormData):

- transition: Update document with the provided details in the database.
- output: N/A
- exception: InvalidInputError - The input data is incomplete or invalid.

16.4.5 Local Functions

N/A

17 MIS of Patient Account Management Module

17.1 Module

PatientAccountManagement

17.2 Uses

N/A

17.3 Syntax

17.3.1 Exported Constants

N/A

17.3.2 Exported Access Programs

Name	In	Out	Exceptions
createPatientRecord	id:String, value: FormData	-	InvalidInputError
deletePatientRecord	id:String	-	IdNotFound
updatePatientRecord	id:String, value: FormData	-	InvalidInputError

17.4 Semantics

17.4.1 State Variables

dbConnection : Database connection point.

17.4.2 Environment Variables

N/A

17.4.3 Assumptions

Database containing the account information for the patients exists.

17.4.4 Access Routine Semantics

createPatientRecord(id:String, record: FormData):

- transition: Adds a new document with the provided details to the database.
- output: N/A

- exception: InvalidInputError - The input data is incomplete or invalid or a duplicate document already exists.

deletePatientRecord(id:String):

- transition: Deletes the corresponding document from the database.
- output: N/A
- exception: IdNotFound - Provided id is invalid or does not exist.

updatePatientRecord(id:String, record: FormData):

- transition: Update document with the provided details in the database.
- output: N/A
- exception: InvalidInputError - The input data is incomplete or invalid.

17.4.5 Local Functions

N/A

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Problem Analysis and Design.

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?
   This document has let us build more on the semantics and uses of each module by module decomposition. While going through the outline of this document, we were able to decompose semantics into different variables as well as assumptions that each module will have.

2. What pain points did you experience during this deliverable, and how did you resolve them?
   Every team project has challenges that must be solved in order to move forward successfully. We had to create a plan to gurantee smooth operations. To ensure that all of our plans are in line with the system, we need to plan user-hierarchy diagram that complement our project. Along with this, we also needed to decide how the modules will be decomposed in the best way possible. In order to contribute to the document and evaluate each other's work as effectively as possible, we also needed to set up a schedule.

3. Which of your design decisions stemmed from speaking to your client(s) or a proxy (e.g. your peers, stakeholders, potential users)? For those that were not, why, and where did they come from?
   After having conversation with our client, we created data model modules for both administrator and patient. In order to create state variables for both administrator and patient profile, our supervisor gave us a run down of the general attributes from which we selected the ones that are relevant to our system. Rest of the decision decisions came up by deciding as a team and were not stemmed from our client.

4. While creating the design doc, what parts of your other documents (e.g. requirements, hazard analysis, etc), it any, needed to be changed, and why?

While creating this design document, we had to edit Software Requirement Specification (SRS) document to modify CI/CD implementation strategy from Jankins to GitHub actions. This is because unlike Jenkins, GitHub Actions offer excellent scalability and reliability for our CI/CD pipelines. It also offers some security features such that code scanning and vulnerability alerts. We also updated the Hazard Analysis documentation for modifying the data layer to include a medicine prediction database and diagnosis prediction database for the system to predict treatment based on the symptoms.

5. What are the limitations of your solution? Put another way, given unlimited resources, what could you do to make the project better? (LO_ProbSolutions)
   One of the limitations of this project is that if not trained properly, data model may provide incorrect prediction suggestions. If given unlimited resources, we would invest in some of the experts to prioritize easy interoperability and data exchange through connection with other healthcare systems.

6. Give a brief overview of other design solutions you considered. What are the benefits and tradeoffs of those other designs compared with the chosen design? From all the potential options, why did you select the documented design? (LO_Explores)
   An additional design decision that we considered was incorporating a chatbot tool within our existing system that would help the healthcare professional to pull up a summary of patient's information. It's beneficial as it will ease the healthcare professional's work and save time. However, due to time constraints and complexity, this design decision was dropped.