



MONASH
University

FIT3164 Data Science Project Part 2

User Guide

Automated Health Information System

Team MDS2

Foo Kai Yan | 33085625 | kfoo0012@student.monash.edu

Alicia Quek Chik Wen | 33045240 | aque0004@student.monash.edu

Eunice Lee Wen Jing | 33250979 | elee0075@student.monash.edu

Jesse Yow San Gene | 32794649 | jyow0001@student.monash.edu

Supervised by:

Dr Muhammad Fermi Pasha

Word Count: 5001

Date: 11th October 2024

Table of Contents

Part 1: End User Guide.....	3
1.1. Web Application.....	3
1.1.0. Access Software Application.....	3
1.1.1. User Log In.....	3
1.1.2. Register New Patient.....	5
1.1.3. Navigating Medication, and Physician.....	10
1.1.4. Navigating Appointment, Encounter, Diagnosis, and Prescription.....	15
1.1.5. Navigating Banner, Navigation Menu, and Navigation Bar.....	25
1.1.6. Web Application: Limitation and Constraints.....	26
Part 2: Technical User Guide.....	27
2.1. Web Application.....	27
2.1.1. Download and Installation.....	27
2.1.2. Run Web Application.....	30
2.1.3. Set up MongoDB Database.....	31
2.1.4. Set up User Accounts.....	34
2.1.5. Troubleshooting.....	38
2.2. HTR and ICROP Model.....	39
2.2.1. Install Dependencies and Set up Model Configurations.....	39
2.2.2. Run ICROP Model.....	42
2.2.3. Run HTR Model.....	44
2.2.4. Software Limitations and Restriction.....	46

Part 1: End User Guide

This section of the guide provides step-by-step instructions on how to navigate and use the automated health information system and its functions.

1.1. Web Application

1.1.0. Access Software Application

Once the setup and configuration is completed as guided in [Part 2: Technical User Guide](#), and the web application is up and running, the web-based application can be accessed in the browser through this URL: <http://localhost:8080/home#/>. The users will be greeted with a system login page.

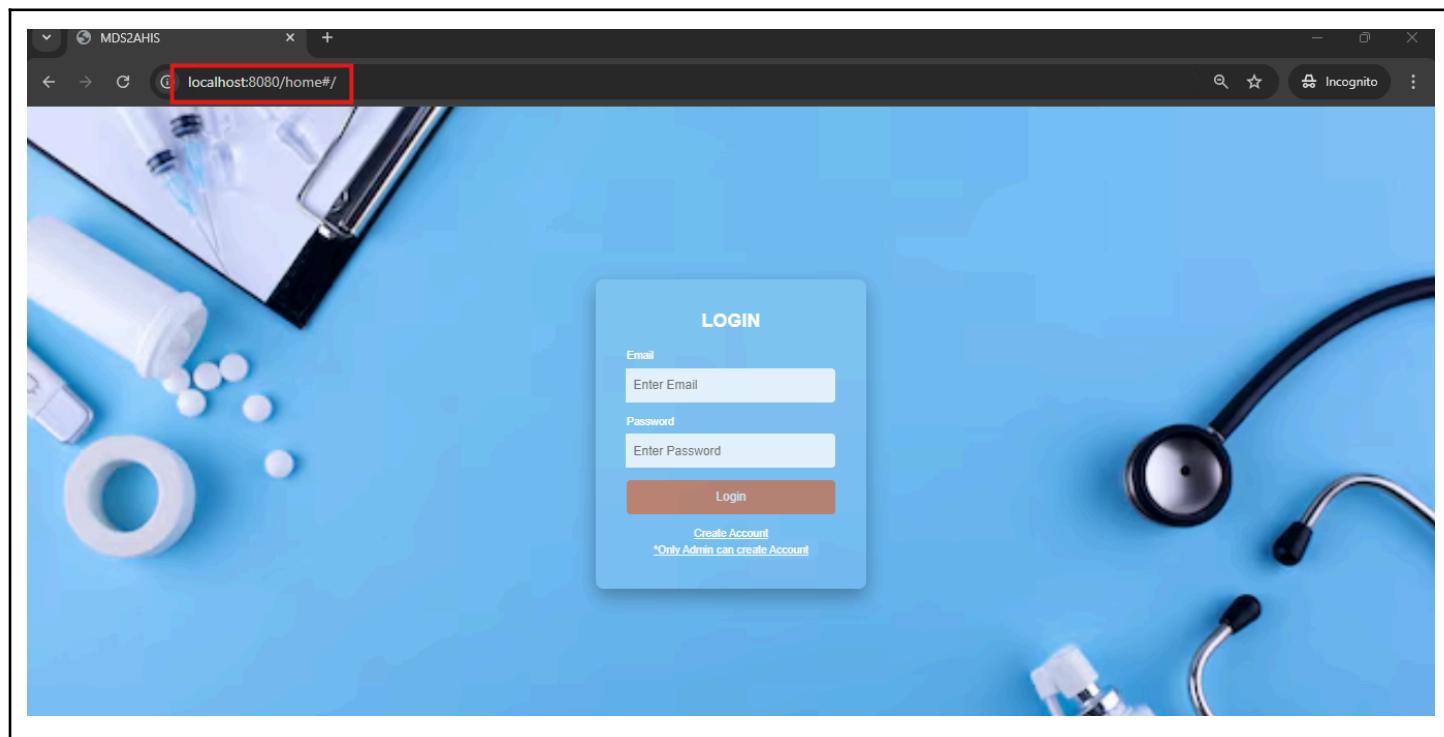


Table 1.1.0.1. Screenshots of the login page with annotation.

1.1.1. User Log In

To log in to the system, users must enter their login credentials to proceed to the system and use any of its functionality. Once a user login successfully, a success confirmation message will appear, else a failed confirmation message will be displayed.

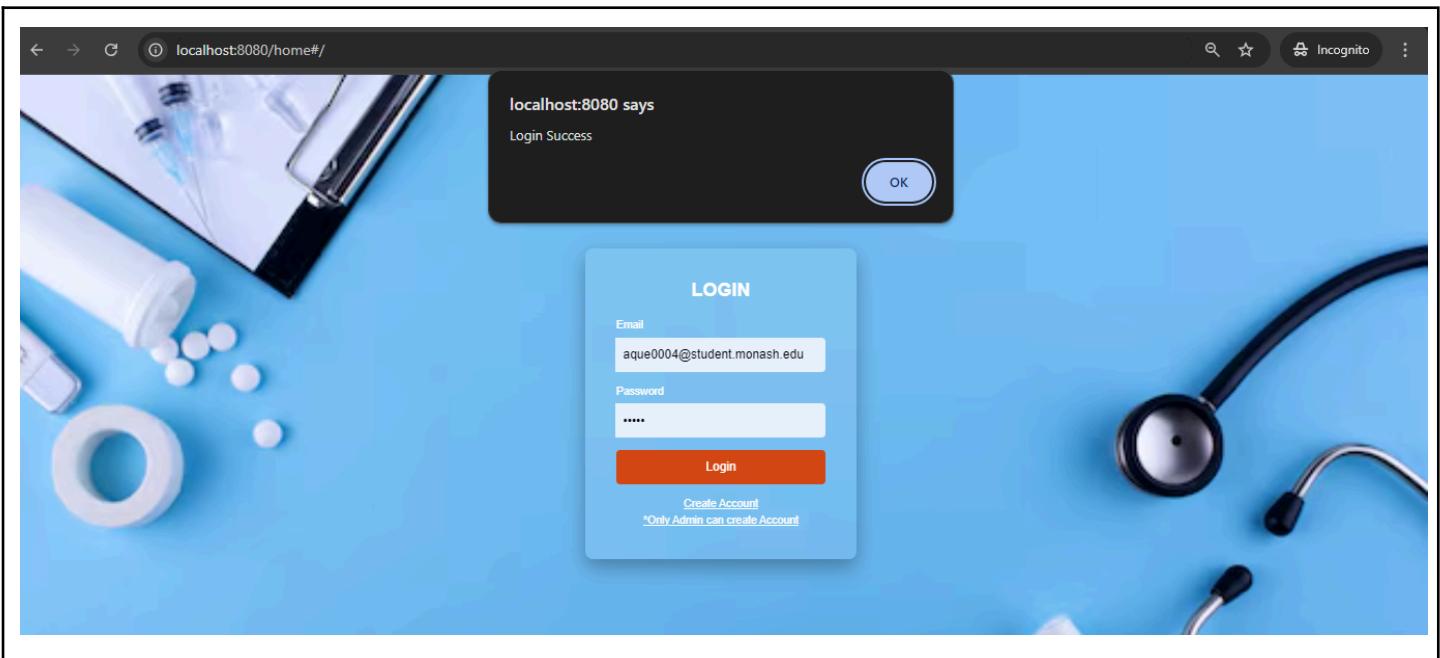


Table 1.1.1.1. Screenshots of login page with popup success confirmation message.

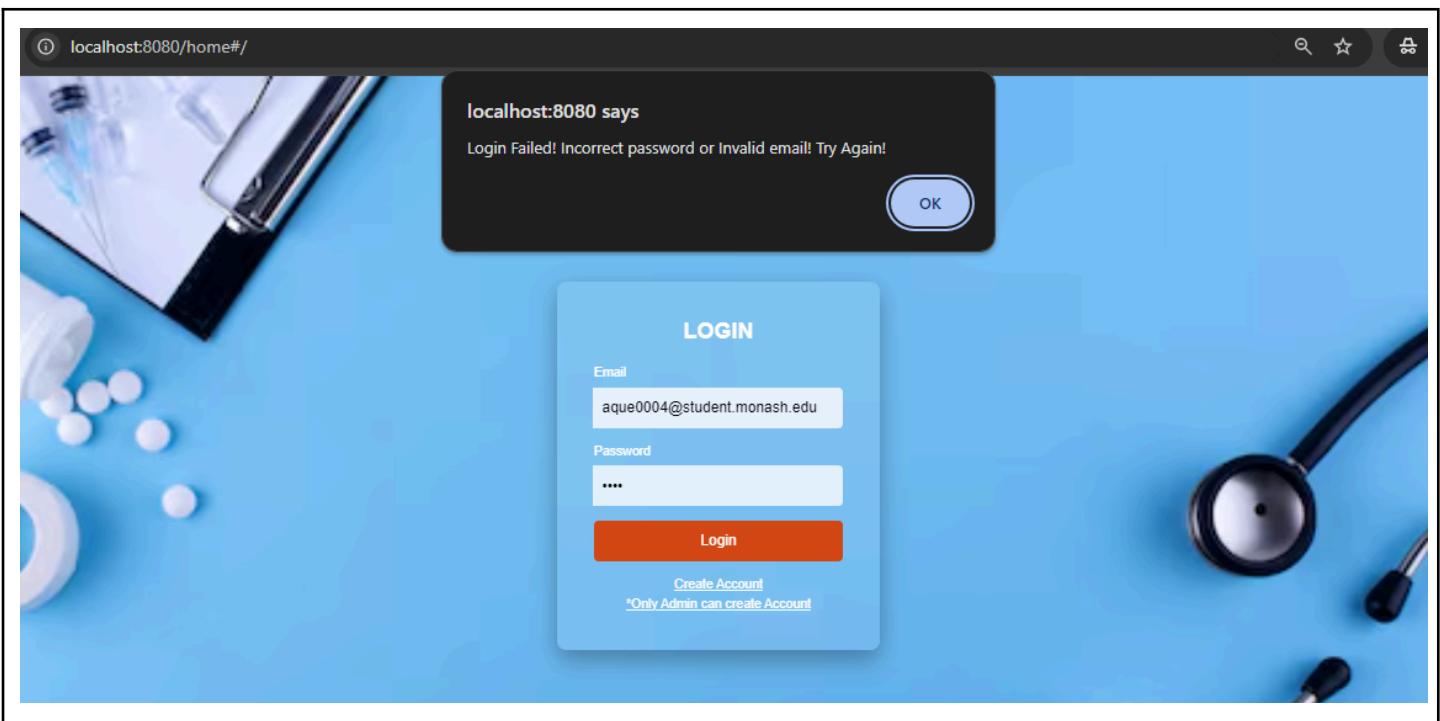


Table 1.1.1.2. Screenshots of login page with popup failed confirmation message.

Once the users dismiss the popup confirmation message, the system directs the users to the system homepage.

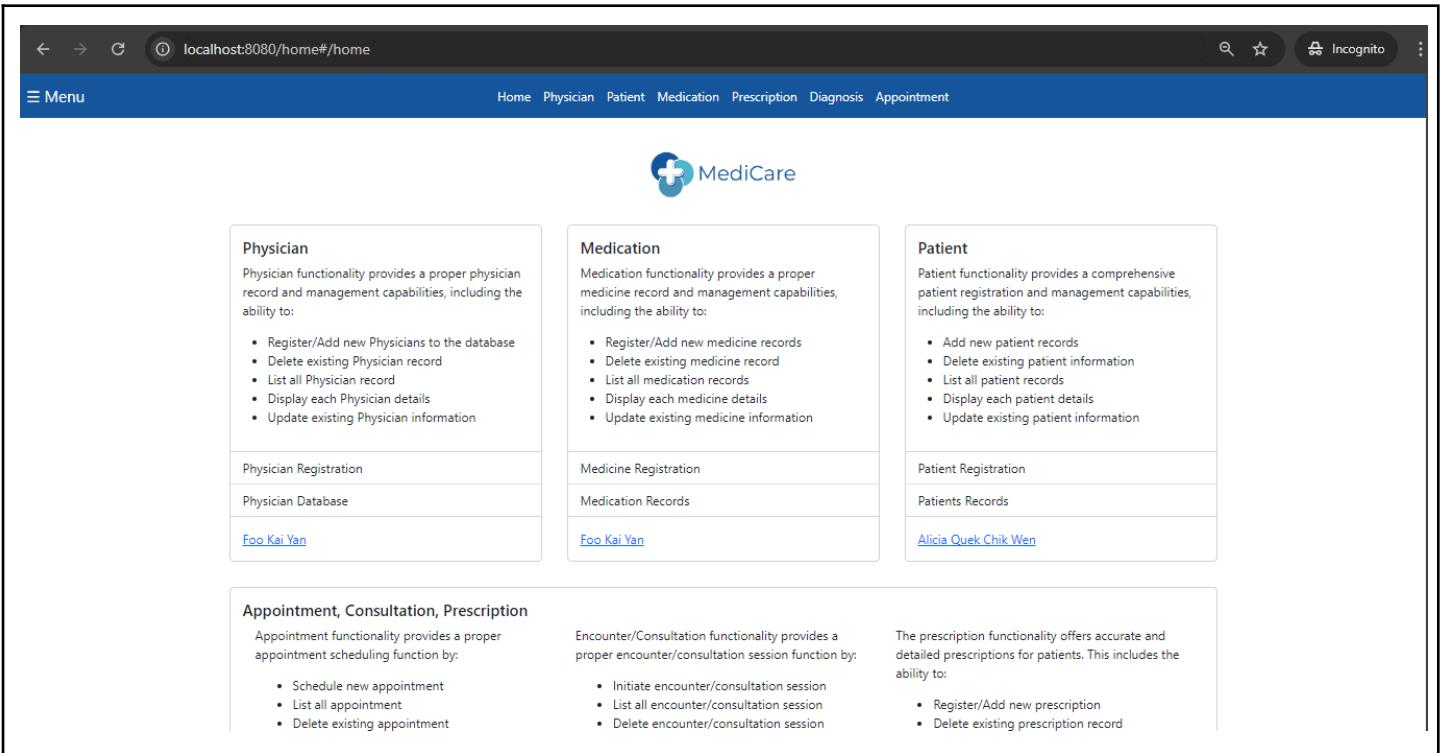
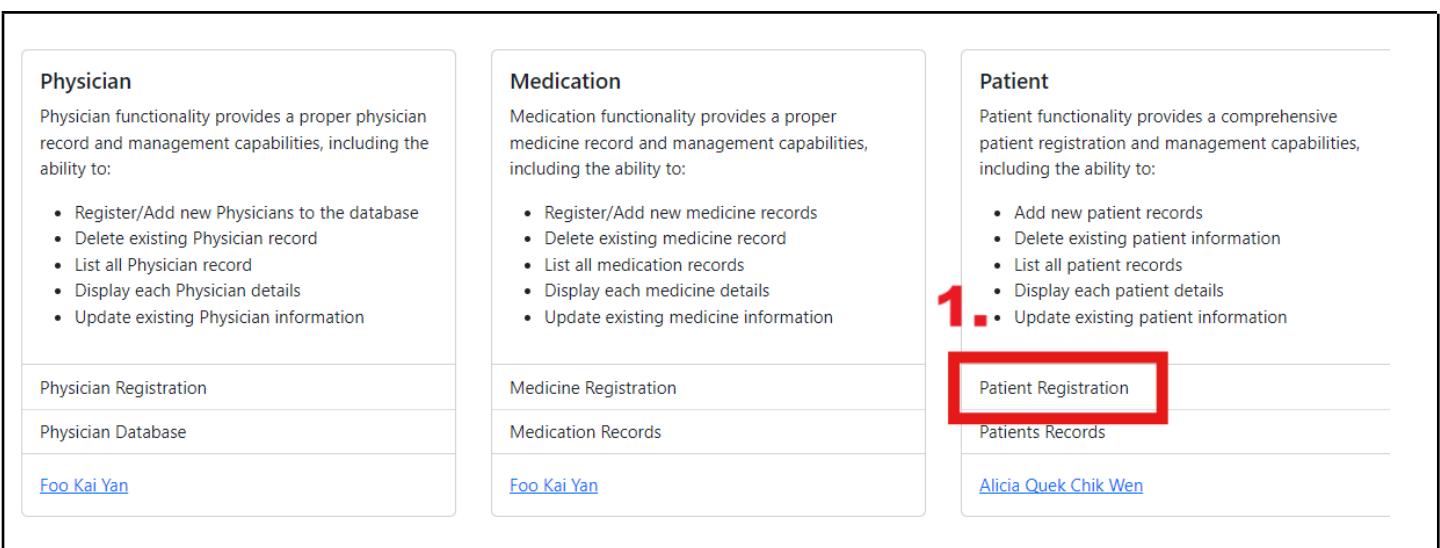


Table 1.1.1.3. Screenshots of the homepage.

1.1.2. Register New Patient

Once the user has logged into the web application, the user can initiate the patient registration process by navigating to the "Patient" section and clicking the "Patient Registration" button. There are two options available for users to add new patients, either by entering their information manually or by uploading a customized patient registration form that automatically fills in the form fields. In this section, we will guide you on both methods of enrolling a new patient.



2.

Patient Registration Form

Fill in the Patient details and Click Submit to register a new patient.

Name	
Enter first name	Enter last name
Date of Birth	
dd/mm/yyyy	Gender <input type="radio"/> Female <input type="radio"/> Male
Nationality	Identification Number (IC)
Enter nationality	Enter Identification Number
Marital Status	
<input type="text"/>	
Contact Number	E-mail
Example: 0123456789	myname@example.com
Address:	
<input type="text"/>	
City	State
Enter City	Enter State
Postal / Zip Code	
<input type="text"/>	
Emergency Contact	
Name	
Enter first name	Enter last name
Relation	Contact Number
Enter Relationship	Example: 0123456789

Upload Form Image for Automated Extraction

Choose File No file chosen Upload

Submit

Name

Enter first name

Enter last name

Date of Birth

dd/mm/yyyy

Gender

Female Male

Nationality

Enter nationality

Identification Number (IDC)

Enter Identification Number

Marital Status

Contact Number

Example: 0123456789

E-mail

myname@example.com

Address:

Enter Street Address

City

Enter City

State

Enter State

Postal / Zip Code

Enter postal/zip code

Emergency Contact

Name

Enter first name

Enter last name

Relation

Enter Relationship **3.**

Contact Number

Example: 0123456789

Upload Form Image for Automated Extraction

Choose File **3.** No file chosen

Upload Form Image for Automated Extraction

Choose File Form.jpg **4.**

Image Upload successfully.

Patient Registration Form

Fill in the Patient details and Click Submit to register a new patient.

Name

Date of Birth

Gender

Nationality

Identification Number (IC)

Marital Status

Contact Number

E-mail

Address:

City

State

Postal / Zip Code

Emergency Contact

Name

Relation

Contact Number

Upload Form Image for Automated Extraction

*IMPORTANT: Extraction Is Not 100% Accurate. Proceed with Caution and Edit accordingly.

 Form.jpg

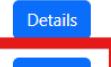
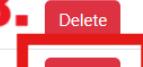
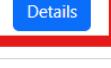
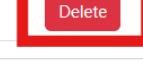
Extraction Complete.

*NOTE: Double Check Inputs before Click Submit.
 **Submit**

5.

Patient Record List

Add

ID	First Name	Last Name	Contact No	View	Edit	Delete
PTD-7903	EUNICE	LEE	0126681311	6.  Details	 Update	 Delete
PET-6679	JESLYN	LEE	012-2346052	 Details	 Update	 Delete

9.

Patient Details

Patient Details

Patient ID: PET-6679 **NRIC No:** 01119141196
Patient Name : Jeslyn Lee **Home Address:** Kota Damansara, 1811 Petaling Jaya, selangor
DOB : 19-12-1002 **Email Address:** jeslynlee@1219egmail.com
Age : 1021 **Tel/Mobile No:** 012-2346052
Gender : female **Admission Date:** 10-10-2024
Marital Status: Single
Nationality: malaysian

Emergency Contact Details

Name: Lily Chan
Relation: Friend
Tel/Mobile No: 016-8881811

Appointment History

Appointment ID	Appointment Date Time

Diagnosis History

Diagnosis ID	Physician ID	Description	Date

Table 1.1.2.1. Screenshots of web application with annotations

Label Number	GUI Element	Description
1.	Patient Registration Button	When the user clicks on the “Patient Registration” button, they are taken to the Patient Registration Form. Users can either manually input the patient details or use the Handwritten Text Recognition (HTR) Model to automatically populate the fields.
2.	Patient Registration Form	On this page, users can manually input the patient's information into each designated field.
3.	Upload file Button	Users can upload a completed patient registration form by selecting the “Choose file” button and then select the “Upload” button to submit the file.
4.	Extract Button	To retrieve patient details, users can click the “Extract” button, which will automatically populate the fields.
5.	Submit Button	After ensuring all fields are correctly filled and error-free,

		the user can click the “Submit” button located at the bottom left to finalize the registration.
6.	View Registered Patient Detail Button	In order to see the details of the added patient, users need to click on the “Details” button, which will direct them to the relevant page on the web application.
7.	Update Patient Detail Button	In order to update the details of the added patient, users need to click on the “Update” button, which will direct them to the relevant page on the web application.
8.	Delete Patient Button	In order to delete a specific patient, users need to click on the “Delete” button, which will remove the selected patient before refreshing the page automatically.
9.	Patient Details	The card displays all the patient details.

Table 1.1.2.2. Description of functionalities of each buttons and GUI Elements

1.1.3. Navigating Medication, and Physician

Users may effectively manage vital healthcare data with the web application's Medication and Physician management functionality. It allows the user to add, update, view, and delete the medication records and physician details within the system. In this section, we will guide you how to navigate the functionality of managing both Medication and Physician records effectively.

Physician
Physician functionality provides a proper physician record and management capabilities, including the ability to:

- Register/Add new Physicians to the database
- Delete existing Physician record
- List all Physician record
- Display each Physician details
- Update existing Physician information

Medication
Medication functionality provides a proper medicine record and management capabilities, including the ability to:

- Register/Add new medicine records
- Delete existing medicine record
- List all medication records
- Display each medicine details
- Update existing medicine information

Patient
Patient functionality provides a comprehensive patient registration and management capabilities, including the ability to:

- Add new patient records
- Delete existing patient information
- List all patient records
- Display each patient details
- Update existing patient information

Medicine Registration **2.**

3. Add/Register Medication

Fill in new medication details & click submit

Name:

Rocephin

Route Of Admission:

intramuscular

Medication Concentration:

10

Medication Volume:

15

Add Medication

4.

Medication Records

ID	Name	Route of Admission	Concentration	Volume	Actions
MLQ-0989	PANADOL	INTRAVASCULAR	4	6	5. View 6. Update 7. Delete
MCI-4451	ROCEPHIN	INTRAMUSCULAR	10	15	View Update Delete

MCI-4451 Details

8.

Object ID: 6708109e7f0712c27b73bf38

Medication ID: MCI-4451

Medication Name: ROCEPHIN

Route of Admission: INTRAMUSCULAR

Concentration: 10

Volume: 15

9.

Physician functionality provides a proper physician record and management capabilities, including the ability to:

- Register/Add new Physicians to the database
- Delete existing Physician record
- List all Physician record
- Display each Physician details
- Update existing Physician information

[Physician Registration](#)

10.

Physician Database

[Foo Kai Yan](#)

Medication

Medication functionality provides a proper medicine record and management capabilities, including the ability to:

- Register/Add new medicine records
- Delete existing medicine record
- List all medication records
- Display each medicine details
- Update existing medicine information

Medicine Registration

Medication Records

[Foo Kai Yan](#)

Patient

Patient functionality provides a comprehensive patient registration and management capabilities, including the ability to:

- Add new patient records
- Delete existing patient information
- List all patient records
- Display each patient details
- Update existing patient information

Patient Registration

Patients Records

[Alicia Quek Chik Wen](#)

11.

Register Physician/Doctor

Fill in the physician/doctor information

First Name:

Wooi King

Last Name:

Soo

Birth Date:

01/01/1983



Assigned Working Department:

DENTIST

Register Physician/Doctor

12.

Physician Records				
ID	First Name	Last Name	Department	Actions
OOH-8587	JOHN	SMITH	GENERAL	13. View 14. Update 15. Delete
OEJ-9681	WOOI KING	SOO	DENTIST	View Update Delete

16.

OEJ-9681 Details

Object ID: 6708163d7f0712c27b73bf3d

Physician/Doctor ID: OEJ-9681

First Name: WOOI KING

Last Name: SOO

Birth Date: 1-1-1983

Department: DENTIST

Table 1.1.3.1. Screenshots of web application with annotations

Label Number	GUI Element	Description
1, 2	Register New Medicine	When the user clicks on the “Medicine Registration” button under the medication card, they are taken to the medication entry form.
3, 4	Add Medicine Button	On this page, users can input the medication details into each designated field. After ensuring all fields are correctly filled and error-free, the user can click the “Add Medication” button located at the bottom left to register the medicine into the database.
5.	View Registered Medicine Detail Button	In order to see the details of the added medicine, users need to click on the “View” button, which will direct them to the relevant page on the web application.
6.	Update Medicine Detail Button	In order to update the details of the added medicine, users need to click on the “Update” button, which will direct them to the relevant page on the web application.
7.	Delete Medicine Button	In order to delete a specific medicine, users need to click on the “Delete” button, which will remove the selected medication before refreshing the page automatically.
8.	Medicine Details	The card displays all the medicine details.

9, 10	Register Physician	When the user clicks on the “Physician Registration” button under the physician card, they are taken to the physician entry form.
11, 12	Register Physician Button	On this page, users can input the physician details into each designated field. After ensuring all fields are correctly filled and error-free, the user can click the “Register Physician/Doctor” button located at the bottom left to register the physician into the database.
13.	View Registered Physician Detail Button	In order to see the details of the added physician, users need to click on the “View” button, which will direct them to the relevant page on the web application.
14.	Update Physician Detail Button	In order to update the details of the added physician, users need to click on the “Update” button, which will direct them to the relevant page on the web application.
15.	Delete Physician Button	In order to delete a specific physician, users need to click on the “Delete” button, which will remove the selected physician before refreshing the page automatically.
16.	Physician Details	The card displays all the physician details.

Table 1.1.3.2. Description of functionalities of each buttons and GUI Elements

1.1.4. Navigating Appointment, Encounter, Diagnosis, and Prescription

After registering the patient and populating the medication and physician database with data, the user can now schedule an appointment between a patient and a physician. Users can start scheduling appointments between patient and physician by navigating to the homepage and selecting the “Schedule Appointment” button within the homepage. Within this section, we will guide you through the whole process of appointment scheduling, encounter initiation, diagnosis writing, as well as prescription assignment within the web application.

A MDS2AHIS localhost:8080/home#/home

Physician Database Medication Records Patients Records

Foo Kai Yan Foo Kai Yan Alicia Quek Chik Wen

Appointment, Consultation, Prescription

Appointment functionality provides a proper appointment scheduling function by:

- Schedule new appointment
- List all appointment
- Delete existing appointment
- Display each appointment details
- Update/Reschedule existing appointment information

1 Schedule Appointment

2 List All Appointment

3 Encounter/Consultation functionality provides a proper encounter/consultation session function by:

- Initiate encounter/consultation session
- List all encounter/consultation session
- Delete encounter/consultation session
- Display encounter/consultation session detail
- Update encounter/consultation session

The prescription functionality offers accurate and detailed prescriptions for patients. This includes the ability to:

- Register/Add new prescription
- Delete existing prescription record
- List all prescription records
- Display each prescription details
- Update existing prescription information

A MDS2AHIS localhost:8080/home#/add-appointment

MediCare

Add/Register Appointment

Fill in new appointment details & click submit

Physician ID:
OSN-1670 - Kai Yan Foo

Patient ID:
2 PZK-9633 - Tong En Lim
PUJ-6809 - Yew Wai How
00/10/2024 10:00 AM

Venue:
TB5001

Appointment Duration (Minutes):
60

3 Add Appointment

© 2024 Built by: MDS02

A MDS2AHIS

localhost:8080/home#/list-appointment

Menu Home Physician Patient Medication Prescription Diagnosis Appointment

MediCare

Appointment Records

ID	Physician ID	Patient ID	Appointment Date Time	Actions	Consultation		
AWL-7476	OSN-1670	PZK-9633	10/30/2024, 12:00:00 PM	View 4	Update 5	Delete 6	Start 7

8

Medical Consultation Form

Physician in charge

Physician Id: _____
Physician Name: Dr. _____

Patient Information

Patient Id: PZK-9633
Patient Name: Tong En Lim

General Physical Examination

Height: _____
Enter height in cm

Weight: _____
Enter weight in kg

General Appearance: _____
Enter Description, e.g Normal/Tired/Pale/Malnourished

Body Temperature: _____
Enter body temperature value in °C

Blood Pressure: _____
Enter blood pressure value in mmHg

Breathing Pattern: _____
Normal/Irregular

Pupil Reflex Condition: _____
Normal/Abnormal

Nerve Reflex Condition: _____
Normal/Abnormal

Other:
Add any additional note if needed, else leave it blank

9

Submit

Screenshot of the MediCare Consultation Records page (localhost:8080/home#/list-consultation).

The page displays a table of consultation records:

ID	Consultation Date Time	Physician ID	Patient ID	View	Edit	Delete
CKG-7639	10/10/2024, 12:58:10 AM	OSN-1670	PZK-9633	Details	Update	Delete

Buttons 10, 11, and 12 are highlighted with red boxes around the "Details", "Update", and "Delete" links respectively.

At the bottom of the page, there is a footer bar with the text "© 2024 Built by MDSQ2".

Screenshot of the Consultation Details page (localhost:8080/home#/view-consultation/CKG-7639).

The page shows the following details:

- Consultation ID:** CKG-7639
- Consultation Date, Time:** 10/10/2024, 12:58:10 AM
- Physician in Charge:**
 - Physician ID: OSN-1670
 - Physician Name: Dr. Kai Yan Foo
- Patient Information:**
 - Patient ID: PZK-9633
 - Patient Name: Tong En Lim
- Consultation Examination Details:**
 - General Appearance: Normal
 - Blood Pressure: 120/100mmHg
 - Body Temperature: 37°C
 - Breathing Pattern: Normal
 - Height: 180cm
 - Weight: 66kg
 - Pupil Reflex Condition: Normal
 - Nerve Reflex Condition: Normal
 - Other: Tired
- Patient Symptoms:**
 - Symptoms: Tired
- Diagnosis:**

ID	Assigned Doctor	Description	Date	Actions
----	-----------------	-------------	------	---------

Button 13, "Add Diagnosis", is highlighted with a red box.

Medical Diagnosis

Physician in charge

Physician ID: OSN-1670

Physician Name: Dr. Kai Yan Foo

Patient Information

Patient ID: PZK-9633

Patient Name: Tong En Lim

Diagnosis

Treatment Description:

Enter Treatment Description

Additional Note:

Enter Additional Notes or NA

Submit 14

Consultation Details

Consultation ID: CKG-7639

Consultation Date, Time: 10/10/2024, 12:58:10 AM

Physician in Charge

Physician ID: OSN-1670

Physician Name: Dr. Kai Yan Foo

Patient Information

Patient ID: PZK-9633

Patient Name: Tong En Lim

Consultation Examination Details

General Appearance: Normal

Blood Pressure:

120/100mmHg

Body Temperature: 37°C

Breathing Pattern: Normal

Height: 180cm

Pupil Reflex Condition:

Normal

Weight: 66kg

Nerve Reflex Condition:

Normal

Other: Tired

Patient Symptoms

Symptoms: Tired

Diagnosis

Add Diagnosis

ID	Assigned Doctor	Description	Date	Actions
----	-----------------	-------------	------	---------

DGQ-7999	Dr. Kai Yan Foo	Consume assigned pills and drink more water as well as rest more	10-10-2024	View 15	Update 16	Delete 17
----------	-----------------	--	------------	----------------	------------------	------------------

Consultation and Diagnosis Details

Consultation ID: CKG-7639

Consultation Date, Time: 10/10/2024, 12:58:10 AM

Prescriptions

18

Add

ID Prescription View

Physician in Charge

Physician ID: OSN-1670

Physician Name: Dr. Kai Yan Foo

Patient Information

Patient ID: PZK-9633

Patient Name: Tong En Lim

Diagnosis Details

Update

Diagnosis ID: DGQ-7999

Date: 10-10-2024

Treatment Description: Consume assigned pills and drink more water as well as rest more

Additional Note: Patient lacks water percentage within body

Consultation Examination Details

General Appearance: Normal

Blood Pressure: 120/100mmHg

Body Temperature: 37°C

Breathing Pattern: Normal

Height: 180cm

Pupil Reflex Condition: Normal

Weight: 66kg

Nerve Reflex Condition: Normal

Other: Tired

Patient Symptoms

Symptoms: Tired

Physician in charge

Physician ID: OSN-1670

Physician Name: Dr. Kai Yan Foo

Patient Information

Patient ID: PZK-9633

Patient Name: Tong En Lim

19

20

Issue Prescription

Fill in the prescription information for the patient

Prescription Name:

Enter prescription name

Medication:

Medication Dosage:

Enter medication dosage

Consumption Frequency:

Enter medication consumption frequency

Issued Date:

dd/mm/yyyy



Consume Before:

dd/mm/yyyy



Special Instructions:

Enter any potential special instructions

Issue Prescription

21

Consultation and Diagnosis Details

Consultation ID: CKG-7639

22

Prescriptions		
ID	Prescription	View
RFN-9301	Water Deficiency	View

Add

23

Consultation Date, Time: 10/10/2024, 12:58:10 AM

24

Diagnosis Details

Update

Diagnosis ID: DGQ-7999

Date: 10-10-2024

Treatment Description: Consume assigned pills and drink more water as well as rest more

Additional Note: Patient lacks water percentage within body

Physician in Charge		
Physician ID:	OSN-1670	
Physician Name:	Dr. Kai Yan Foo	
Patient Information		
Patient ID:	PZK-9633	
Patient Name:	Tong En Lim	

Consultation Examination Details

25

General Appearance: Normal

Blood Pressure: 120/100mmHg

Body Temperature: 37°C

Breathing Pattern: Normal

Height: 180cm

Pupil Reflex Condition: Normal

Weight: 66kg

Nerve Reflex Condition: Normal

Other: Tired

Patient Symptoms

Symptoms: Tired

Table 1.1.4.1. Screenshots of web application with annotations

Label Number	GUI Element	Description
1, 2	Schedule Appointment Button	When the user clicks “Schedule Appointment”, they will be taken directly to the Add Appointment page. On this page, users must enter the appointment details as shown in the second image and labeled as ‘2’ in Table 1.1.4.1. above.
3	Add Appointment Button	To officially add the appointment into the database, users will have to click this “Add Appointment” button.
4	View Appointment Detail Button	In order to see the details of the added appointment, users need to click on the “View” button, which will direct them to the relevant page on the web application.
5	Update Appointment Detail Button	In order to update the details of the added appointment, users need to click on the “Update” button, which will direct them to the relevant page on the web application.

6	Delete Appointment Button	In order to delete a specific appointment, users need to click on the “Delete” button, which will remove the selected appointment before refreshing the page automatically.
7	Initiate Encounter Button	In order to start an appointment, users need to click on the “Start” button, which will direct them to the consultation feature of the web application. When the appointment has started, this appointment will not show in the list appointment anymore as the appointment has been met.
8, 9	Medical Consultation Form, Add Consultation Button	On this page, users will have to fill in the relevant and required details to complete the medical consultation form before submitting the form by clicking on “Submit” labeled as ‘9’ within the fourth image in Table 1.1.4.1. above.
10	View Consultation Detail Button	In order to see the details of the encounter/consultation, users need to click on the “Details” button, which will direct them to the relevant page on the web application.
11	Update Consultation Detail Button	In order to update the details of the encounter/consultation, users need to click on the “Update” button, which will direct them to the relevant page on the web application.
12	Delete Consultation Button	In order to delete a specific encounter/consultation, users would need to click on the “Delete” button, which will remove the selected encounter/consultation before refreshing the page automatically.
13	Add Diagnosis Button	To be directed to the Add Diagnosis page, users would have to click on the “Add Diagnosis” button within the consultation details page.
14	Submit Diagnosis Button	Once the user is directed to the Add Diagnosis page, users would have to fill in the relevant and required information before clicking this button to submit the diagnosis to be stored in the database. <u>Note: Multiple diagnosis can be added for each consultation</u>
15	View Diagnosis Detail Button	In order to see the details of the diagnosis, users need to click on the “Details” button, which will direct them to the

		relevant page on the web application.
16	Update Diagnosis Detail Button	In order to update the details of the diagnosis, users need to click on the “Update” button, which will direct them to the relevant page on the web application.
17	Delete Diagnosis Button	In order to delete a specific diagnosis, users would need to click on the “Delete” button, which will remove the selected diagnosis before refreshing the page automatically.
18	Add Prescription Button	In order to add a prescription for the patient, users would have to click this button to be redirected to the Add Prescription page of the web application.
19	Physician and Patient Information	These 2 cards display the patient and physician ID and name.
20, 21	Prescription Form, Submit Prescription Button	Once the user is directed to the Add Prescription page, users would have to fill in the relevant and required information before clicking the button labeled as ‘21’ to submit the prescription to be stored in the database. <u>Note: Multiple prescription can be added for each diagnosis</u>
22, 23	Prescription Assigned, View Prescription Detail Button	These card display a list of prescription assigned to the patient by a physician for a specific Diagnosis <u>Note: Multiple prescription can be added for each diagnosis</u>
24, 25	Diagnosis Details, Physician Information, Patient Information, Consultation Examination Detail	These 3 cards display diagnosis detail, physician and patient information, as well as consultation examination detail for the user to see and know who wrote this diagnosis, what is the diagnosis written, who was the diagnosis written for, and what examination was done during the physician-patient consultation.

Table 1.1.4.2. Description of functionalities of each buttons and GUI Elements

1.1.5. Navigating Banner, Navigation Menu, and Navigation Bar

If the user intends to navigate to the homepage of the web application from any page within the web application, the user can opt to head to the homepage from 3 different methods:

1. Clicking banner logo image.
2. Selecting the homepage option from the navigation bar.
3. Selecting the homepage option from the navigation menu.

Within this section, we will guide you through the steps to use the Banner Logo, Navigation Bar, and Navigation Menu.

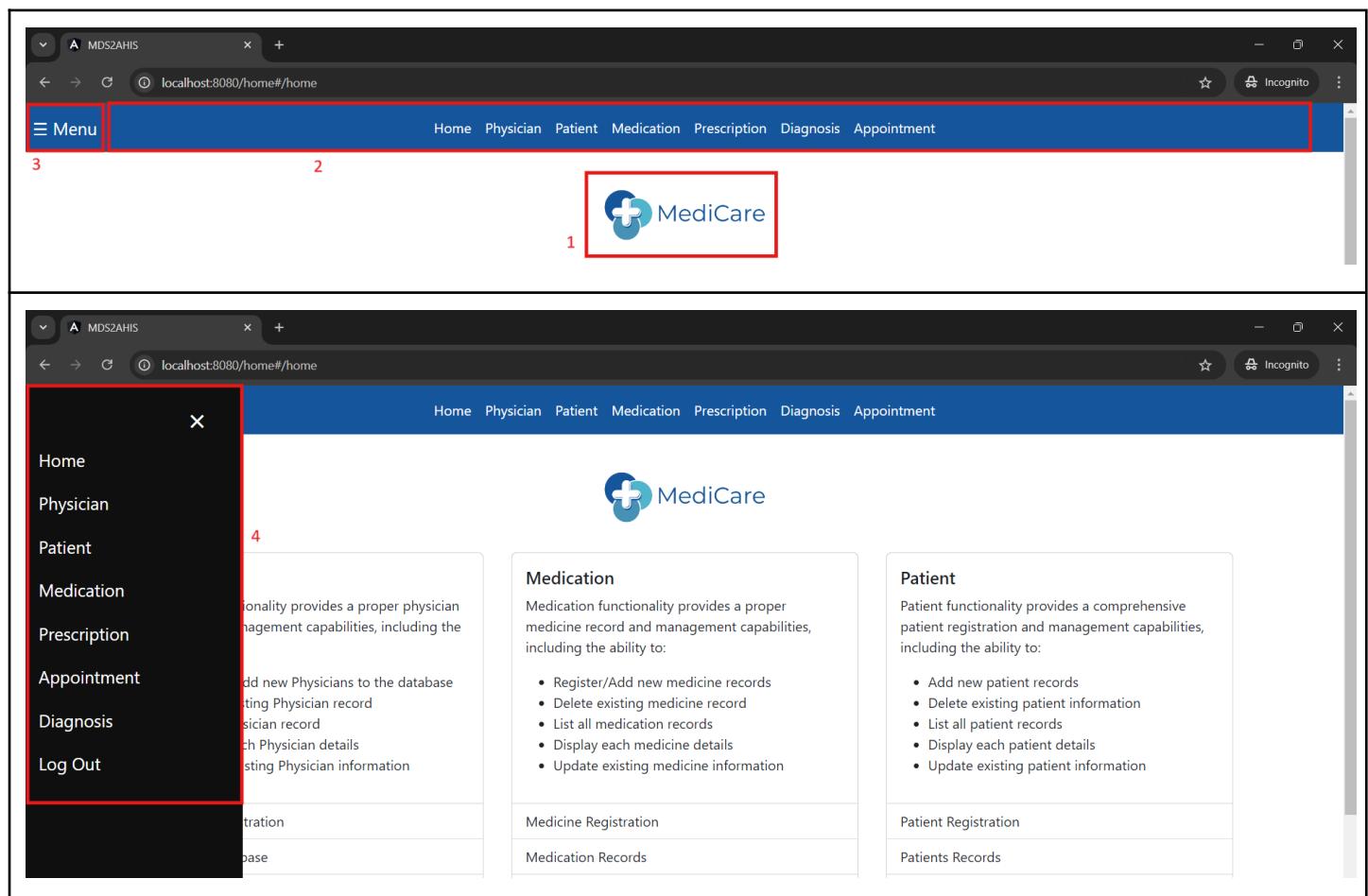


Table 1.1.5.1. Homepage with annotations

The table provided below will help you identify the GUI elements on your screen and explain their functionalities.

Label Number	GUI Element	Description
1	Banner with Logo	Clicking on the logo labeled 'Medicare' serves as a quick access button for users to instantly reach the homepage.

		Upon clicking, the web application will immediately redirect the user to its homepage.
2	Navigation Bar	The navigation bar is the extended blue bar situated at the top of all the functionalities in the web application. Clicking on any of the features will take you to the corresponding feature's list all page. If a user clicks on "Patients", they will be directed to the "List All Patient" page where all patient records in the database will be displayed.
3, 4	Navigation Menu	The menu button, labeled as '3' in the third image of Table 1.1.5.1. above, triggers the display of the Navigation Menu as a sidebar labeled as '4'. The options in the Navigation Menu are identical to those in the Navigation Bar, except for the additional option of a "Log Out" button that takes users to the Login page upon selection.

Table 1.1.5.2. Description of functionalities of GUI Elements

1.1.6. Web Application: Limitation and Constraints

A significant constraint of the web application is the time consumption involved in extracting text from uploaded patient registration forms. Although the HTR model is effective at accurately recognizing handwritten text, it can be time-consuming. This delay impacts the overall efficiency of the system. Additionally, the text recognition feature is only limited to patient registration forms, and other sections of the application, such as medication entry, physician information, and diagnosis details, do not yet utilize HTR functionality. Therefore, expanding the HTR capabilities to these areas could significantly streamline the data entry process and reduce the time spent on administrative tasks.

The web application has not yet been formally deployed, which limits access to its database. This is another significant disadvantage as the database is stored locally on users' machines, which limits data sharing and synchronization. This local storage hinders collaboration, as updates made on one machine are not reflected on others. Without a centralized database, maintaining data consistency is challenging, and the application's effectiveness in a multi-user environment is compromised. Therefore, deploying the application on a server with a centralized database is necessary for real-time updates and seamless data sharing among multiple users.

Part 2: Technical User Guide

This section outlines the setup and configuration needed for the software system that is to be done by system administrator. In order to proceed to the configuration and the set up for the web-based system, the HTR and the Image Cropping (ICROP) model, there are prerequisites that must be met. This includes the following software installation:

1. **Visual Studio Code (VS Code):** This is a source code editor. Use the following [link](#) to download the version that matches your system specifications.
2. **Node.js:** This is a JavaScript backend runtime environment. Use the following [link](#) to download the version that matches your system specifications.
3. **MongoDB Compass:** This is a GUI for MongoDB. Use the following [link](#) to download the version that matches your system specifications.
4. **Angular:** This is a frontend web application framework. To download, run this code `npm install -g @angular/cli` in your terminal.
5. **HTR:** This is the HTR model for recognizing text and extracting text from handwritten input within uploaded images. Use the following [link](#) to download the python version 3.8 to run the HTR model.
6. **ICROP:** This is the ICROP model that preprocesses images by cropping out unnecessary background details and focusing on the fields of the handwritten text for better recognition accuracy. Use the following [link](#) to download the opencv-python version to run the ICROP model.

2.1. Web Application

2.1.1. Download and Installation

Below outline the procedure needed to download and configure the web application.

1. Download the source code from [this GitLab repository](#) by selecting the zip download button.

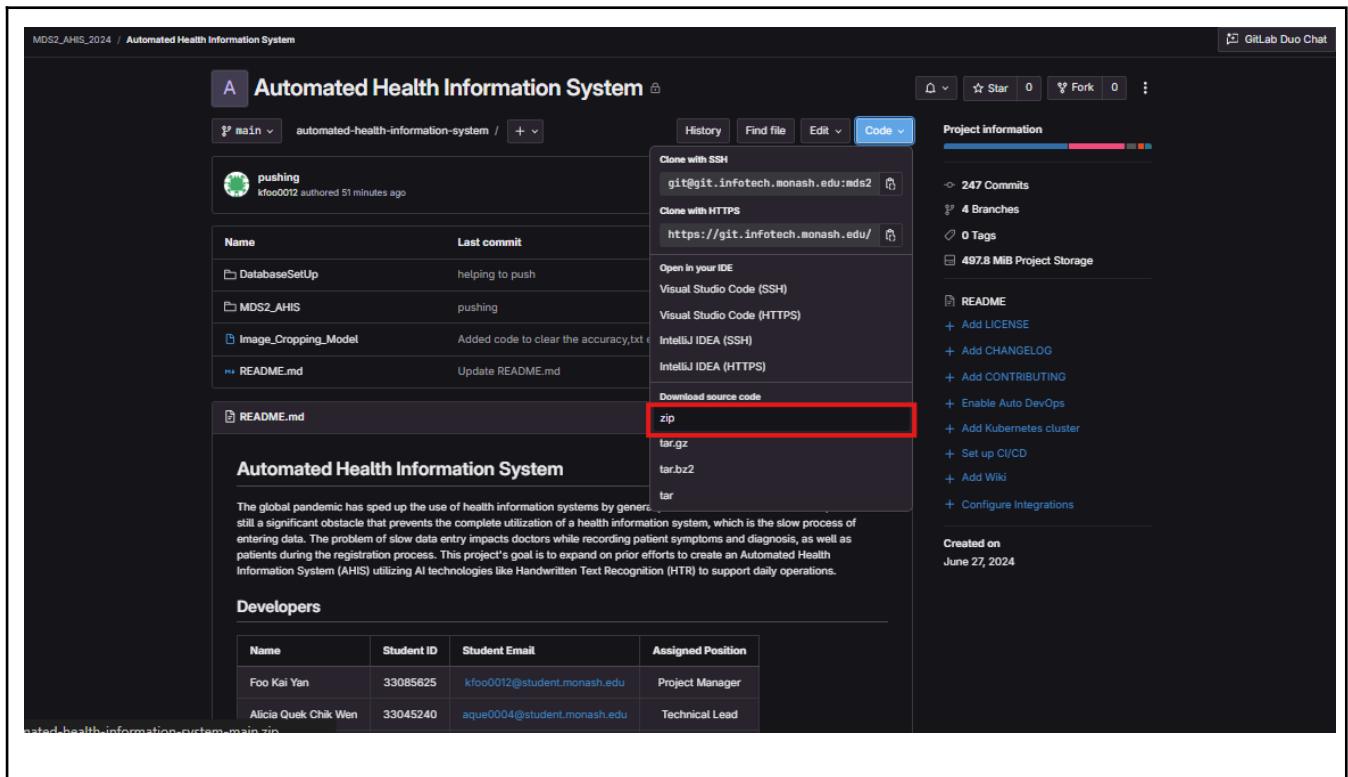


Table 2.1.1.1. Screenshots of the GitLab repository with annotation.

2. Once the zip file is downloaded, unzip the file and save it in the desired destination.
3. Open VS Code and click ‘Open Folder’

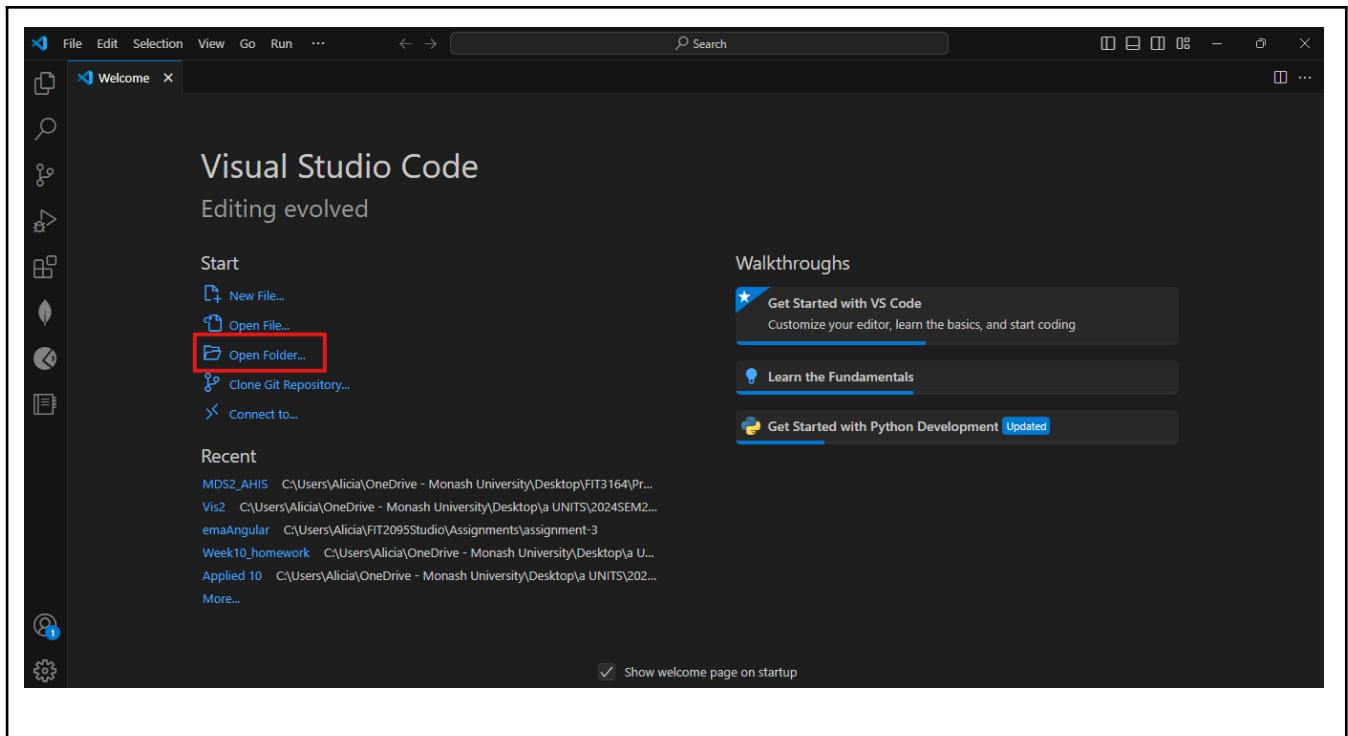


Table 2.1.1.2. Screenshots of the VS Code with annotation.

4. Locate the destination path in which the software source code is stored and click into the folder till you encounter a folder named MDS2_AHIS, in which you select to open.

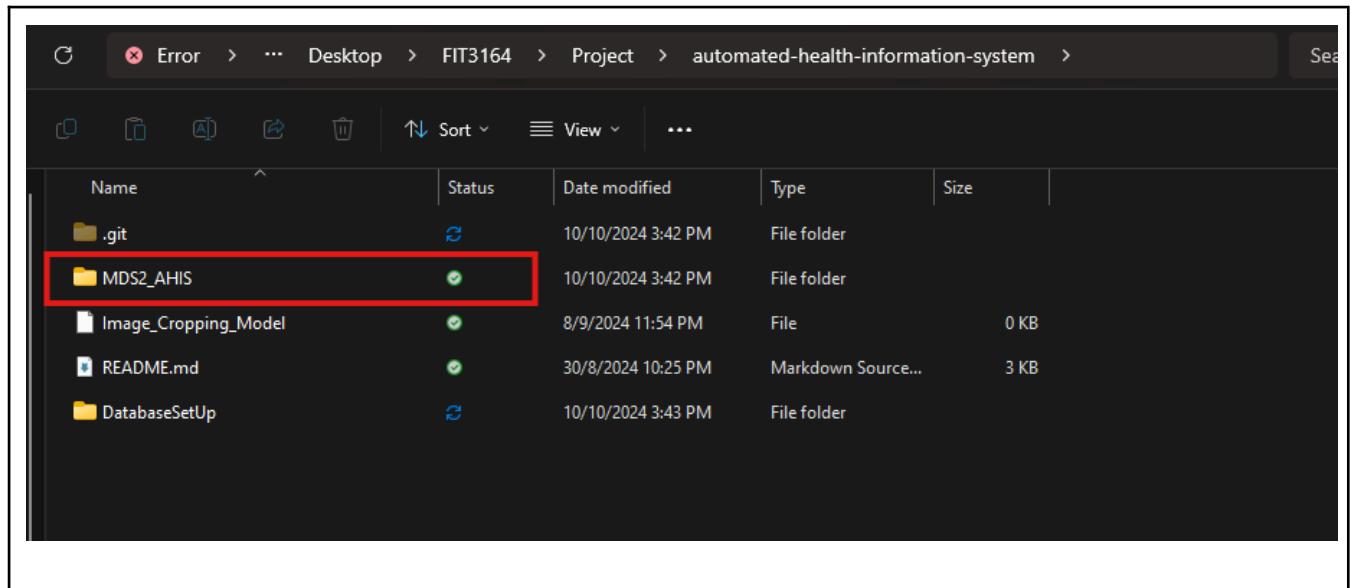


Table 2.1.1.3. Screenshots of the file explorer with annotation.

5. Open a new terminal and enter `npm install`, which will immediately install all the necessary dependencies specified in the project for the web-based system. Once similar output is displayed in the terminal like below, you are ready to proceed to run the web application as guided in [2.1.2. Run Web Application](#).

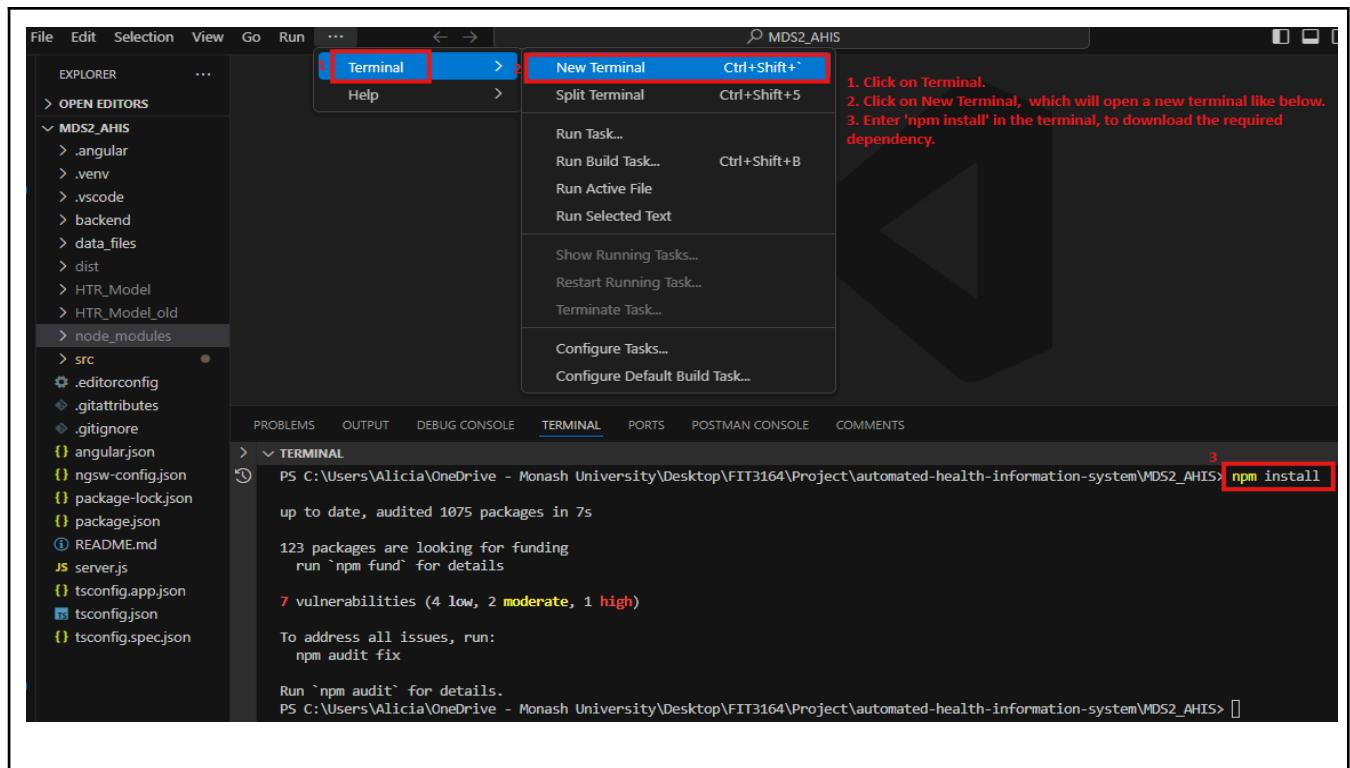


Table 2.1.1.4. Screenshots of the VS Code with annotation.

2.1.2. Run Web Application

Before running the web application, ensure that you are in the right directory by using the command `cd MDS2_AHIS` if necessary to move to the directory where the main project files are stored. To run the application, you will need to compile the Angular application. The command `ng build` compiles the project's code and assets into a deployable version, creating essential files and configurations. After the completion of the construction, the next step is to execute the main server file `server.js` by typing: `node server.js`.

The screenshot shows a terminal window with several annotations:

- Annotation 1: A red box highlights the command `cd MDS2_AHIS` in the terminal.
- Annotation 2: A red box highlights the command `ng build` in the terminal.
- Annotation 3: A red box highlights the output line `listening on port 8080` in the terminal.
- Annotation 4: A red box highlights the output line `Connected successfully` in the terminal.
- Annotation 5: A red box highlights the URL `http://localhost:8080/home#/` in a browser address bar.

Terminal Output:

```
PS C:\Users\Owner\Documents\FYP\automated-health-information-system> cd MDS2_AHIS
PS C:\Users\Owner\Documents\FYP\automated-health-information-system\MD52_AHIS> ng build
  ✓ Browser application bundle generation complete.
  ✓ Copying assets complete.
  ✓ Generating index.html... 2 rules skipped due to selector errors:
    .form-floating~label -> Did not expect successive traversals.
    .form-floating~label -> Did not expect successive traversals.
  ✓ Index.html generation complete.
  ✓ Service worker generation complete.

Initial Chunk Files
main.1c2e42a1339ab5f4.js | main | 469.80 kB | 100.29 kB
styles.8296f4a196814c99.css | styles | 218.94 kB | 21.91 kB
polyfills.563625714aab15e4.js | polyfills | 33.03 kB | 10.67 kB
runtime.27a097de2e984900.js | runtime | 1.13 kB | 620 bytes

| Initial Total | 722.89 kB | 133.46 kB

Build at: 2024-10-09T18:13:19.863Z - Hash: c096d96a4920735c - Time: 9506ms

Warning: bundle initial exceeded maximum budget. Budget 500.00 kB was not met by 222.89 kB with a total of 722.89 kB.

PS C:\Users\Owner\Documents\FYP\automated-health-information-system\MD52_AHIS> node server.js
listening on port 8080
Connected successfully
```

Table 2.1.2.1. Screenshots with annotations

After following the steps outlined above and seeing similar output as displayed in the 4th red box on the terminal, it can be confirmed that the web application is running smoothly following a successful connection to the MongoDB server. Hence, the last step is to access the web application itself. If you have installed the web application before then you can directly access the web application else open any browser and access the web application from this link: <http://localhost:8080/home#/> as shown in the second image in Table 2.2.2.1.

2.1.3. Set up MongoDB Database

Before setting up user accounts, the MongoDB database needs to be configured to enable smooth operation on setting up user accounts.

Below outline the procedure needed to configure the MongoDB database.

1. Open MongoDB Compass and add a new connection.

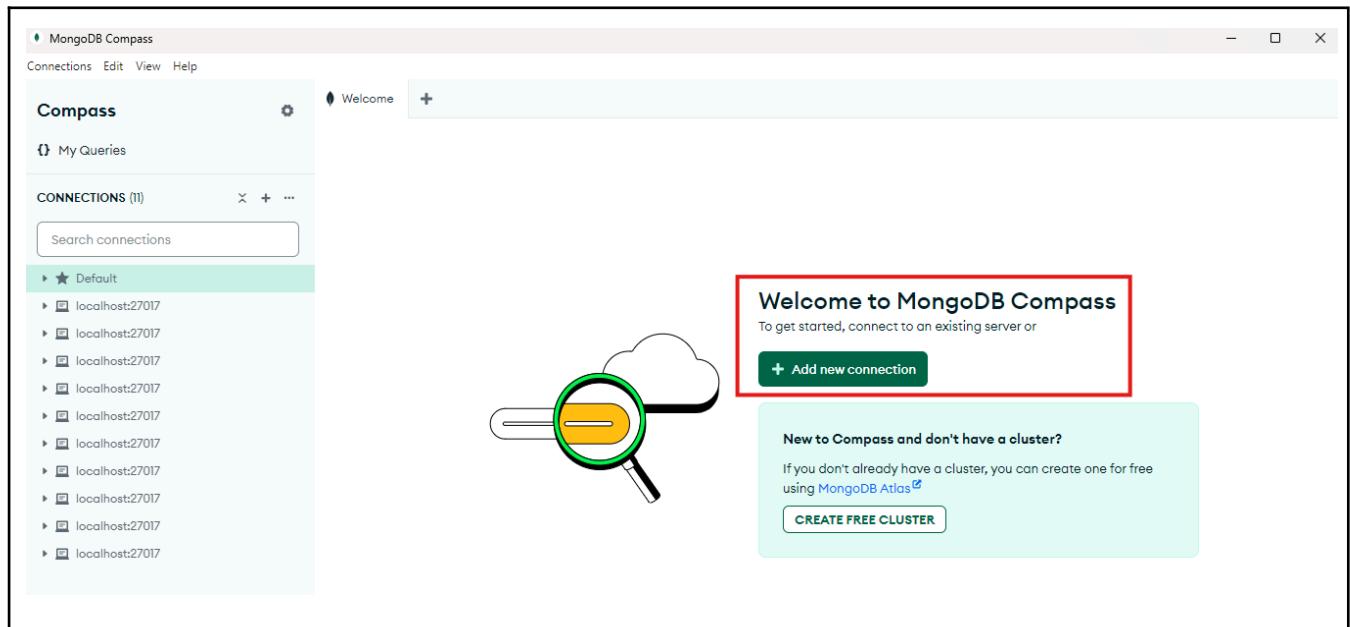


Table 2.1.3.1. Screenshots of the MongoDB Compass with annotation.

2. Using the default MongoDB URI connection string, enter the desired name for the MongoDB Database Server.

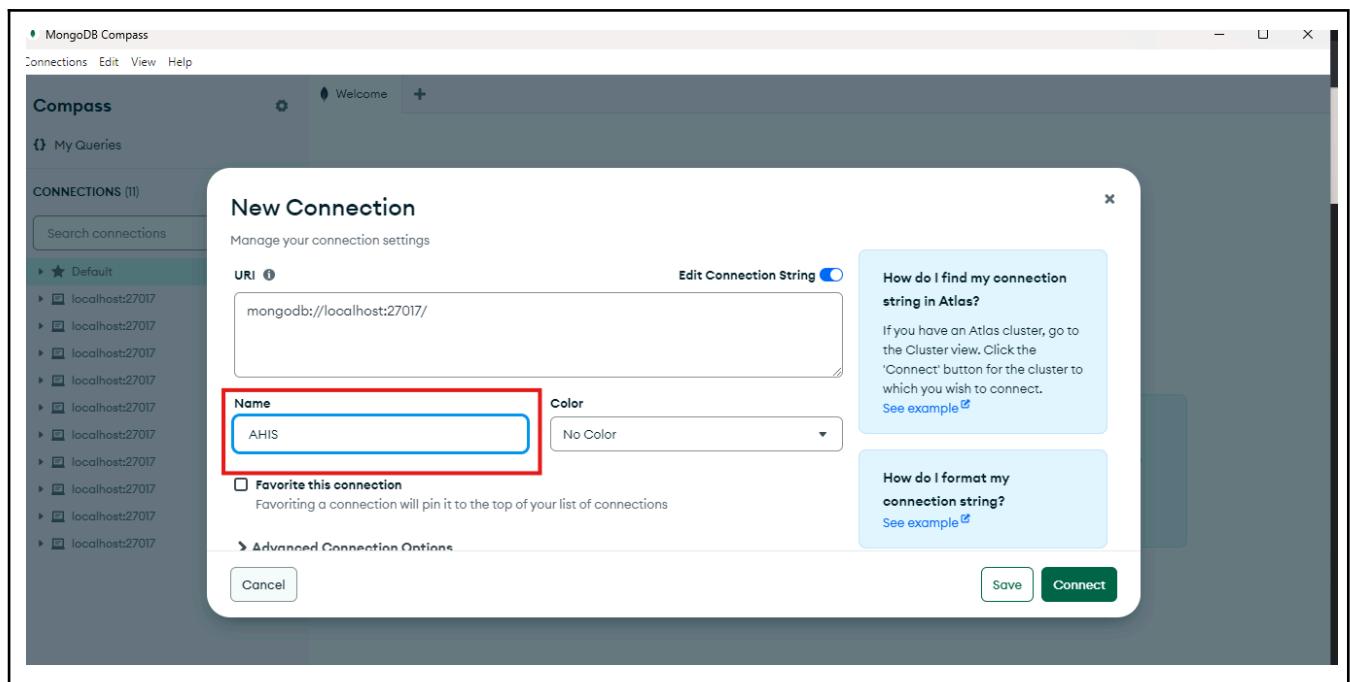


Table 2.1.3.2. Screenshots of the MongoDB Compass with annotation.

3. Upon successful connection:

1. Select a database, named HeathInformationDB.
2. Select a collection, named roles.
3. Select import data to import roles data from HealthInformationDB.roles.csv file in DatabaseSetUp folder.

The screenshot shows the MongoDB Compass interface. On the left, the 'Connections' sidebar lists 'AHIS' with its sub-databases: 'EMA', 'HealthInformationDB' (highlighted with a red box), 'patients', 'users', 'admin', and 'config'. Within 'HealthInformationDB', the 'roles' collection is selected (highlighted with a red box). The main panel displays the 'roles' collection with the message 'This collection has no data'. Below this, a note says 'It only takes a few seconds to import data from a JSON or CSV file.' A green 'Import Data' button is highlighted with a red box. At the bottom of the screen, a file explorer window shows the 'DatabaseSetUp' folder containing two files: 'HealthInformationDB.roles.csv' and 'HealthInformationDB.users.csv', both of which are highlighted with red boxes.

Table 2.1.3.3. Screenshots of the MongoDB Compass and file explorer with annotation.

4. If imported successfully;

1. Popup confirmation message will appear.
2. Roles collection will be populated with role data.

Connections Edit View Collection Help

Compass

My Queries

CONNECTIONS (12)

Search connections

- AHIS
 - EMA
 - HealthInformationDB
 - appointments
 - consultations
 - diagnoses
 - medications
 - patients
 - physicians
 - prescriptions
 - roles
 - users
 - admin
 - config

Documents 2 Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or [Generate query](#)

[ADD DATA](#) [EXPORT DATA](#) [UPDATE](#) [DELETE](#)

25 1 - 2 of 2

2

```
_id: ObjectId('66e7f6fb8cc389831b242e7')
role: "Admin"
createdAt: 2024-09-16T09:14:35.773+00:00
updatedAt: 2024-09-16T09:14:35.773+00:00
__v: 0

_id: ObjectId('66ed595f47eb02f657c65162')
role: "User"
createdAt: 2024-09-20T11:15:43.488+00:00
updatedAt: 2024-09-20T11:15:43.488+00:00
__v: 0
```

Import completed. 2 documents imported.

Table 2.1.3.4. Screenshots of the MongoDB Compass with annotation.

5. Same goes for importing users data from HealthInformationDB.users.csv file in DatabaseSetUp folder.
 1. Select a collection, named users.
 2. Select import data to import users data from HealthInformationDB.users.csv file in DatabaseSetUp folder.

MongoDB Compass - AHIS/HealthInformationDB.users

Connections Edit View Collection Help

Compass

My Queries

CONNECTIONS (12)

Search connections

- AHIS
 - EMA
 - HealthInformationDB
 - appointments
 - consultations
 - diagnoses
 - medications
 - patients
 - physicians
 - prescriptions
 - roles
 - users
 - admin
 - config

Documents 2 Aggregations Schema Indexes 3 Validation

Type a query: { field: 'value' } or [Generate query](#)

[ADD DATA](#) [EXPORT DATA](#) [UPDATE](#) [DELETE](#)

25 0 - 0 of 0

This collection has no data

It only takes a few seconds to import data from a JSON or CSV file.

1 **2** **Import Data**

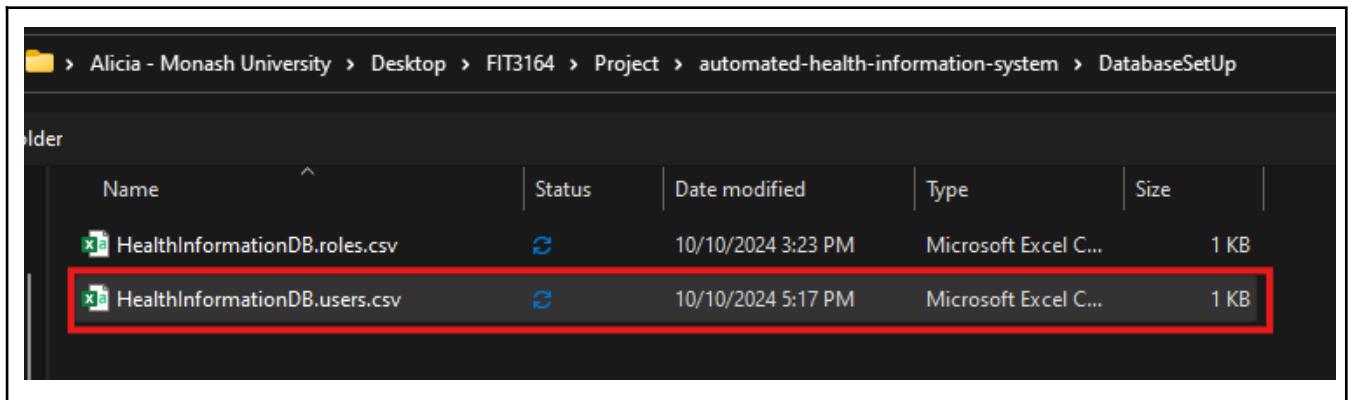


Table 2.1.3.5. Screenshots of the MongoDB Compass and file explorer with annotation.

6. If imported successfully;
 1. Popup confirmation message will appear.
 2. Users collection will be populated with user data.

```

_id: ObjectId('6707809fe917260b6db1ad6e')
userFirstName: "Default"
userLastName: "Admin"
username: "defaultAdmin"
userEmail: "defaultAdmin@gmail.com"
hash: "$2a$10$TyFRu1hPEnqAvxSbfIz.OjpsvK1hRVDVgKDfWRusUE35kl0g8eu"
profileImage: "./src/assets/images/femaleNurse.png"
isAdmin: true
role: "66e7f6fb8cc389831b242e7"
createdAt: 2024-10-10T07:22:07.626+00:00
updatedAt: 2024-10-10T07:22:07.626+00:00
__v: 0
  
```

Import completed. 1 document imported.

Table 2.1.3.6. Screenshots of the MongoDB Compass with annotation.

7. Once done, proceed to setting up accounts as guided in [2.1.4. Set up User Accounts](#).

2.1.4. Set up User Accounts

Before proceeding to use the automated health information system, user accounts need to be set up. In the system, there are two main account roles that can be assigned to users in the system, which are administrator and normal user. Both roles have the same capabilities to access the system core functionalities. The difference between these roles is that the administrator has the capability to

access the register user page to create an account, while normal users can not access the register user page.

Below outline the procedure needed to create an administrator or user account.

1. In the browser, access the system through this URL: <http://localhost:8080/home#>, where you will be greeted with the system login page.

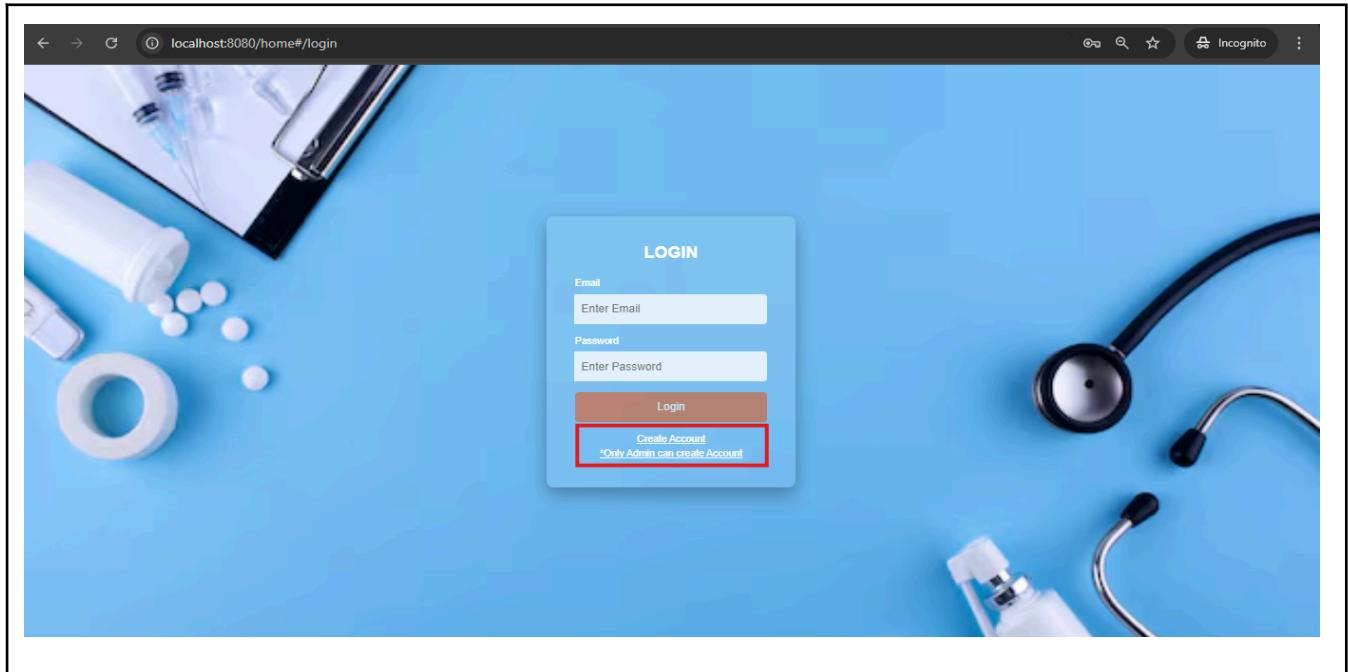


Table 2.1.4.1. Screenshots of the login page with annotation.

Note in the image above, only the administrator has the access to view the register account page and the power to create an account.

2. As a system administrator, click on the 'Create Account' link. This will send you to an administrator authentication page, where you will need to authenticate yourself that you are an authorized administrator.

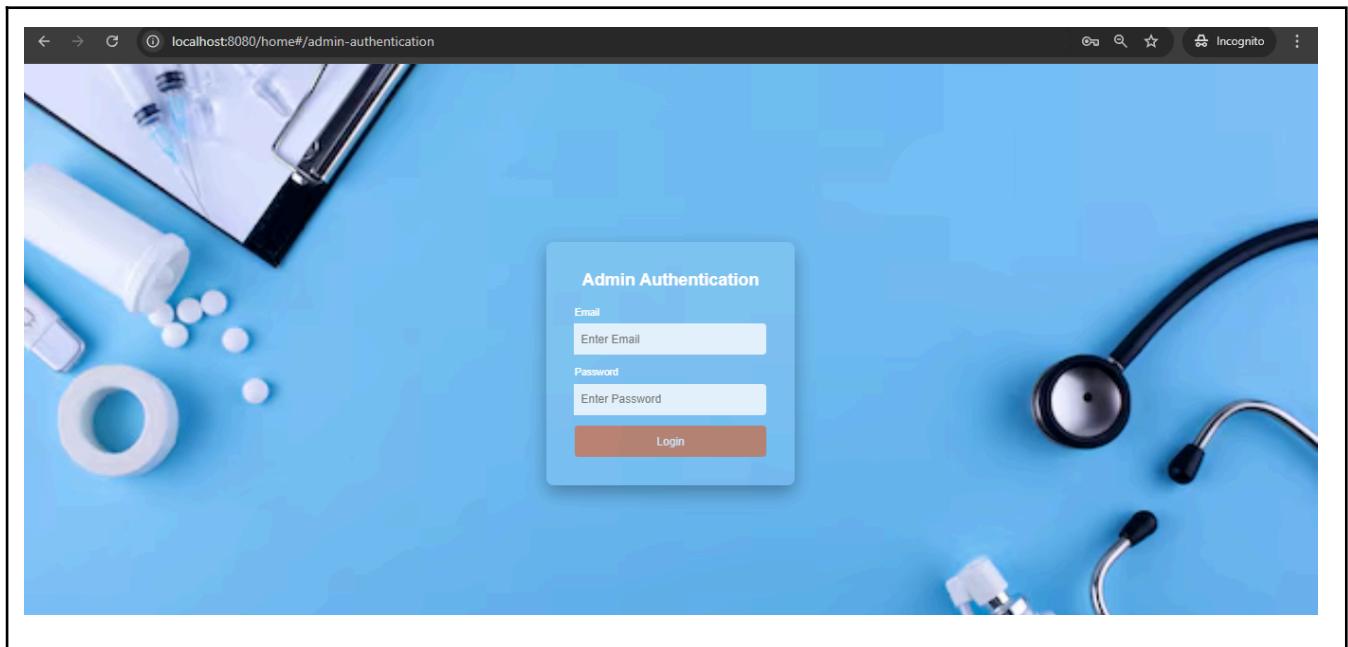


Table 2.1.4.2. Screenshots of the admin authentication page.

However, if you are a new system administrator with no created administrator account, you can enter the default password: 54321 and default email address that can be found in the MongoDB Compass User Database. This default admin account can be removed once an actual administrator account is created.

```

_id: ObjectId('6707809fe917260b6dbiad6e')
userFirstName : "Default"
userLastName : "Admin"
username : "defaultAdmin"
userEmail : "defaultAdmin@gmail.com" highlighted
hash : "$2a$10$yFKuLhPEnyQavxSBf1z.Ojpsvk1hRVDVgKDfWRusUE35k10g8eu"
profileImage : "./src/assets/images/femaleNurse.png"
isAdmin : true
role : ObjectId('66ef6fb8bcc389831b242e7')
createdAt : 2024-10-10T07:22:07.626+00:00
updatedAt : 2024-10-10T07:22:07.626+00:00
__v : 0

```

Table 2.1.4.3. Screenshots of the MongoDB Compass with annotation.

3. Once the authentication form is filled, the ‘Login’ button will be enabled to click. Upon successful administrator authentication, you will be greeted with a popup success confirmation message, else a popup failed confirmation message will appear.

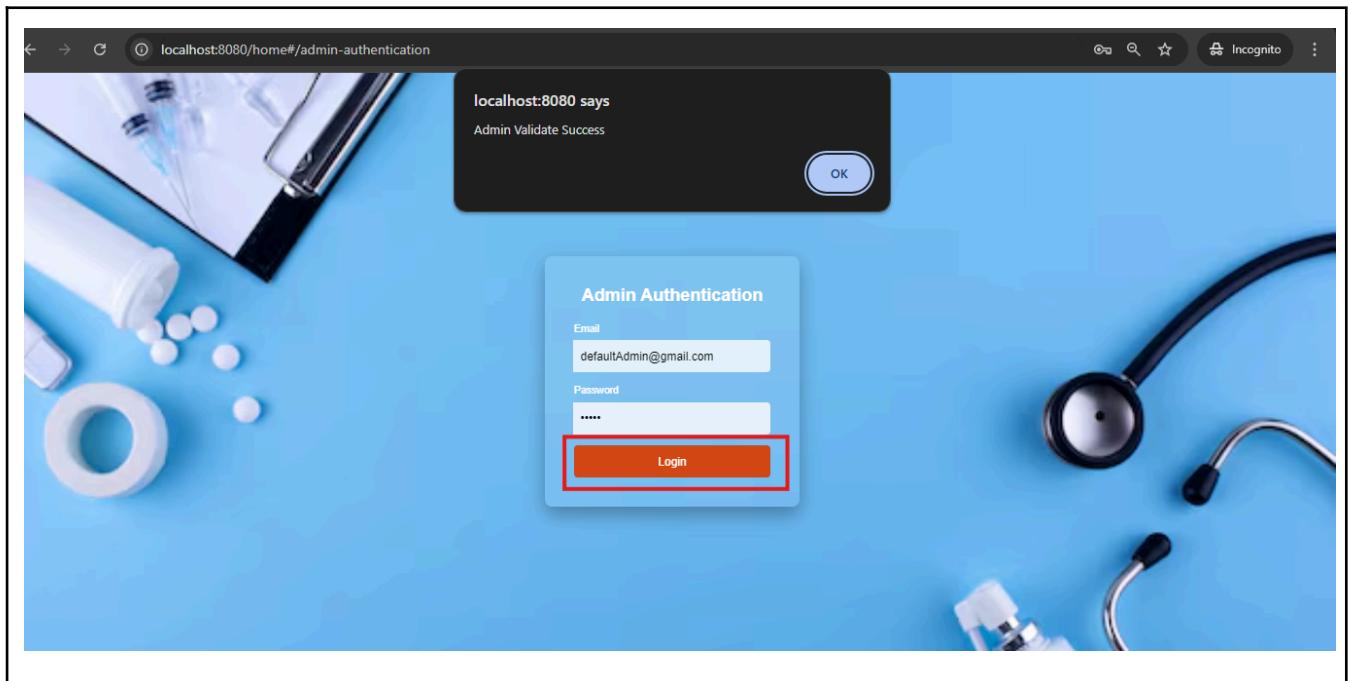


Table 2.1.4.4. Screenshots of authentication page with popup success confirmation message.

4. It will then lead you to a register account page where you will be able to create an administrator or a user account by assigning the role in the 'Assigned Role' field accordingly.

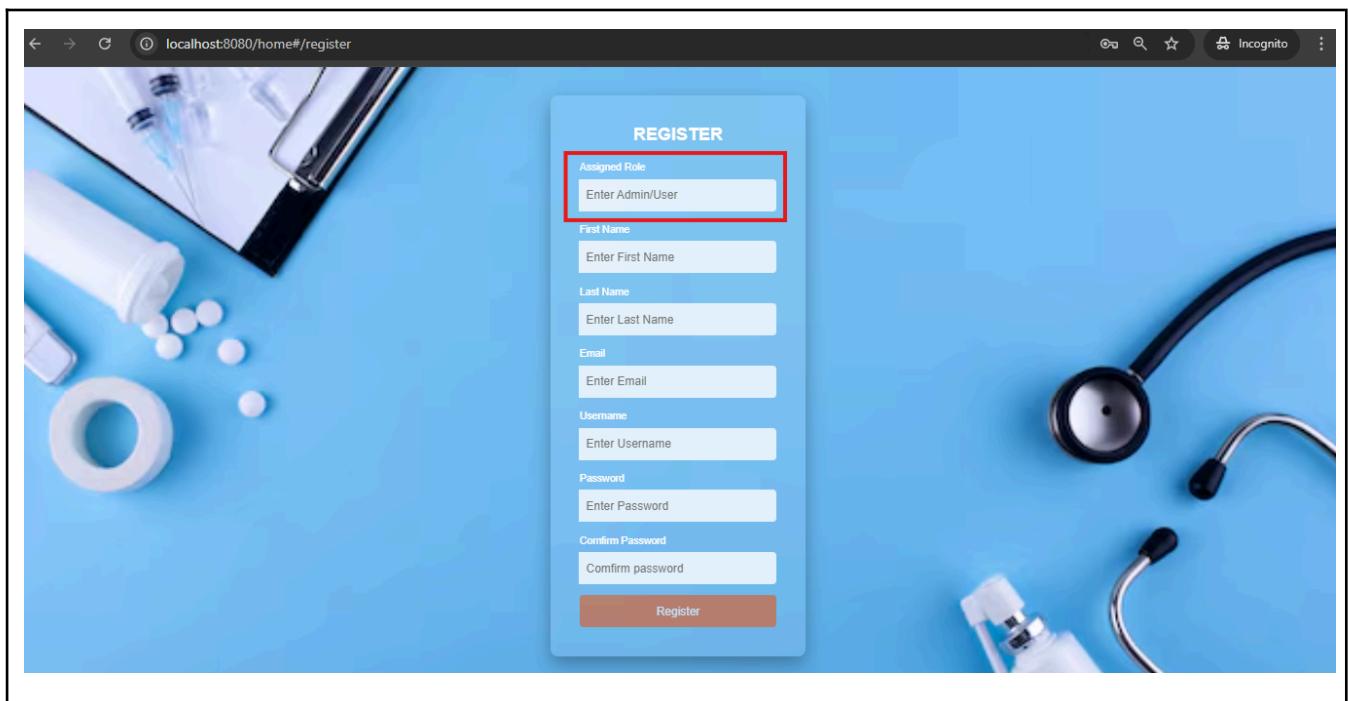


Table 2.1.4.5. Screenshots of register page with annotation.

5. Once the register form is filled, the 'Register' button will be enabled to click. A popup confirmation message will be displayed once the account is created.

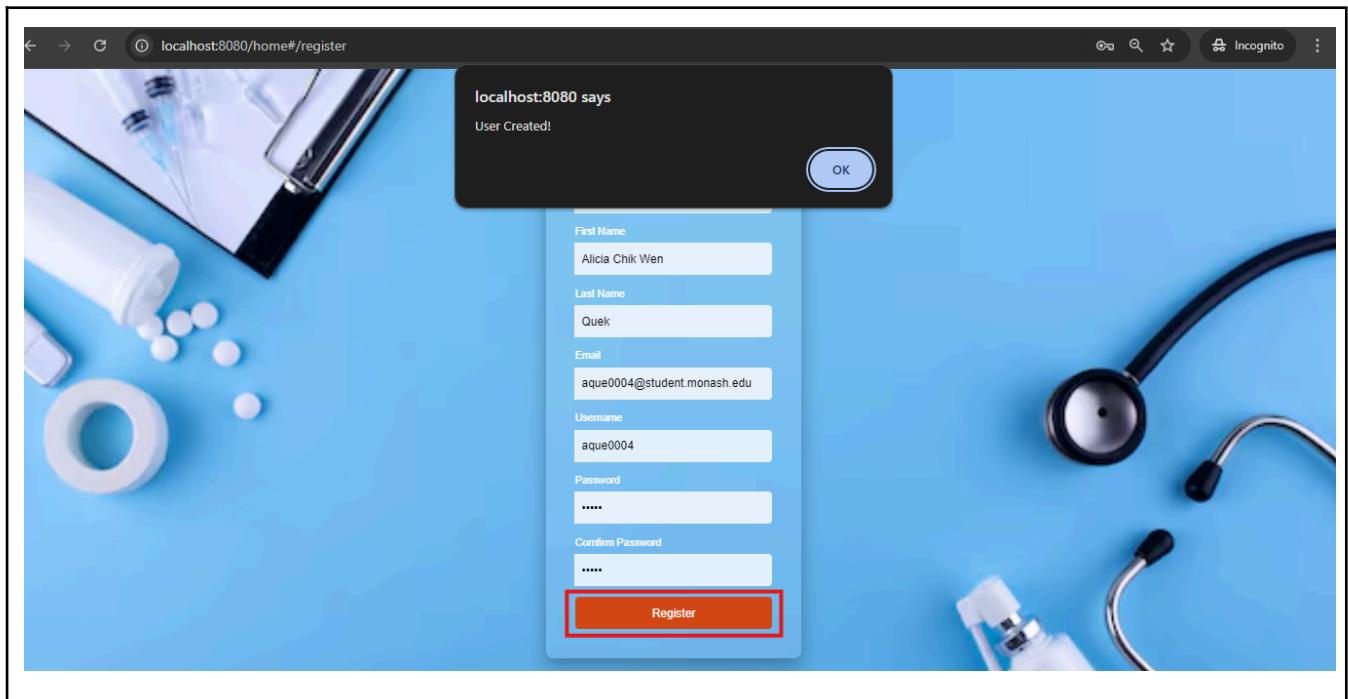


Table 2.1.4.6. Screenshots of register page with popup confirmation message.

2.1.5. Troubleshooting

- 1. Verify Installed Dependencies:** Users should ensure all dependencies for the web application are installed correctly and are also up-to-date. Dependencies are essential to the web application as they consist of libraries and frameworks that your software needs in order to work properly.
- 2. Check Console Errors:** The web application has to be built before being able to execute. Therefore, if the web application cannot be accessed because the building process failed, the user should seek assistance from a technical expert to review error messages in the terminal, browser console, or server logs for identifying problems in the building process. The technical expert can review the build process for any errors or warnings that might indicate issues and solve it accordingly.
- 3. Check MongoDB Connection:** It should be known that the web application relies heavily on the connection to the MongoDB database to send data to the database for safe storage so if by any chance the application would not run after building then the user might need to check if the web application is correctly connected to MongoDB.
- 4. Clear Irrelevant Cache:** Users can opt to clear the browser cache and cookies to ensure the latest version of the application is being served when the web application is loaded. Cached files and cookies have the potential to retain obsolete data, resulting in the browser loading a previous version. Clearing cache will force the browser to fetch fresh, up-to-date content from the server, solving display or functionality issues caused by old data.
- 5. Verify Devices' Execution Policy:** If users were to encounter `FullyQualifiedErrorId : UnauthorizedAccess` error when executing the web application, users can opt to run any terminals

like Command Prompt and Windows PowerShell as Administrator and run the following line to change the execution policy: `Set-ExecutionPolicy RemoteSigned -Scope CurrentUser` to allow local scripts to be executed without issues.

2.2. HTR and ICROP Model

2.2.1. Install Dependencies and Set up Model Configurations

After installing python version 3.8, it is essential to ensure that all the required dependencies for the HTR and ICROP models are installed. This can be done by running the following commands in your terminal or Visual Studio Code (VSC) environment.

Below outline the procedure needed to install the necessary dependencies for the HTR model.

1. Open your Visual Studio Code (or terminal).
2. Click on ‘Open Folder’ and open where you have stored in your desired location mentioned in 2.2.1 step 2.
3. Open the terminal in VSC by navigating to ‘Terminal’.

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists project files including 'SIMPLEHTR-MASTER', 'src', and various image files like 'PRF_Template.jpg', 'PRF_Cursive.jpg', etc. The Code Editor tab shows a Python script named 'AHIS_CropModel1.py' with the following code:

```

1 import cv2
2 import numpy as np
3 import os
4
5
6 def crop():
7     # Load the template image (reference form)
8     template_image_path = r"D:\\\\Uni\\\\Y4S1\\\\FIT 3164 - DS Project 2\\\\Project\\\\SimpleHTR-master\\\\src\\\\PRF_Template.jpg"
9     # print(template_image_path)
10    # "D:\\\\Uni\\\\Y4S1\\\\FIT 3164 - DS Project 2\\\\Project\\\\SimpleHTR-master\\\\src\\\\PRF_Template.jpg"
11    template_image = cv2.imread(template_image_path, cv2.IMREAD_GRAYSCALE)
12    if template_image is None:
13        raise ValueError("Template image not found at {template_image_path}")
14
15    # Load the filled form image (scanned form from patient)
16    filled_image_path = r"D:\\\\Uni\\\\Y4S1\\\\FIT 3164 - DS Project 2\\\\Project\\\\SimpleHTR-master\\\\src\\\\PRF_Cursive.jpg"
17    # "D:\\\\Uni\\\\Y4S1\\\\FIT 3164 - DS Project 2\\\\Project\\\\SimpleHTR-master\\\\src\\\\PRF_Red.jpg"
18    filled_image = cv2.imread(filled_image_path, cv2.IMREAD_GRAYSCALE)
19    if filled_image is None:
20        raise ValueError("Filled image not found at {filled_image_path}")
21
22    # Resize filled image to match template dimensions
23    template_height, template_width = template_image.shape
24    filled_image = cv2.resize(filled_image, (template_width, template_height))
25
26    # Detect keypoints and descriptors using ORB detector
27    orb = cv2.ORB_create()
28    keypoints_template, descriptors_template = orb.detectAndCompute(template_image, None)
29    keypoints_filled, descriptors_filled = orb.detectAndCompute(filled_image, None)
30
31    # Match descriptors using BFMatcher
32    bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
33    matches = bf.match(descriptors_template, descriptors_filled)

```

The Terminal tab at the bottom shows a PowerShell prompt: PS D:\\\\Uni\\\\Y4S1\\\\FIT 3164 - DS Project 2\\\\Project\\\\SimpleHTR-master\\\\src\\\\> .

Table 2.2.1.1. Screenshots of the terminal tab in VSC.

4. For HTR model - Use pip to install all required packages for the HTR model. Below are the following required packages for the HTR model. Repeat pip install for all other required packages.

```

1 editdistance==0.5.2
2 lmdb==1.0.0
3 matplotlib==3.2.1
4 numpy==1.19.5
5 opencv-python==4.4.0.46
6 path==15.0.0
7 tensorflow==2.4.0

```

Table 2.2.1.2. Screenshots of the requirement for HTR model

The screenshot shows a code editor interface with the following details:

- EXPLORER:** Shows a project structure for "SIMPLEHTR-MASTER" with folders like "data", "doc", "dump", "model", "selfTrainedmodel", and "src".
- FILES:**
 - "main.py" (selected)
 - "AHIS_CropModel1.py" (closed)
 - "requirements.txt" (selected)
- AHIS_CropModel1.py Content:**

```

6 def crop():
7     # Log file to store cropped field paths
8     log_file_path = os.path.join(save_directory, 'cropped_fields_log.txt')
9     with open(log_file_path, 'w') as log_file:
10        # Loop through the coordinates and crop each field
11        for field_name, (x, y, w, h) in fields_coordinates_adjusted.items():
12            aligned_image = aligned_images[y:y + h, x:x + w]
13            cropped_image_path = os.path.join(save_directory, f'{field_name}.jpg')
14            cv2.imwrite(cropped_image_path, aligned_image, [int(cv2.IMWRITE_JPEG_QUALITY), 95])
15
16            # Log the path of the saved image
17            log_file.write(f'{field_name}: {cropped_image_path}\n')
18
19    print(f'Cropping completed successfully. Images saved in directory: {save_directory}')
20 else:
21     print("Not enough matches found to align images.")
22
23 # Run the crop function
24 if __name__ == "__main__":
25     crop()
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97

```
- TERMINAL:** Shows the command: PS D:\Uni\Y4S1\FIT 3164 - DS Project 2\Project\SimpleHTR-master\src> pip install editdistance == 0.5.2

Table 2.2.1.3. Screenshots of the installation packages for the HTR model.

Note: Long path must be enabled when installing path 15.0.0. Follow this [link](#) to youtube on how to do it.

5. For ICROP model -pip install required packages opencv-python, numpy and os for the ICROP model. The latest version of these packages are sufficient for the ICROP model to run.

```

EXPLORER ... main.py AHIS_CropModel1.py requirements.txt
src > AHIS_CropModel1.py > ...
1 import cv2
2 import numpy as np
3 import os
4
5
6 def crop():
7     # Load the template image (reference form)
8     template_image_path = r"D:\\Uni\\Y4S1\\FIT 3164 - DS Project 2\\Project\\SimpleHTR-master\\src\\PRF_Template.jpg"
9     # print(template_image_path)
10    template_image = cv2.imread(template_image_path, cv2.IMREAD_GRAYSCALE)
11    if template_image is None:
12        | raise ValueError(f"Template image not found at {template_image_path}")
13
14    # Load the filled form image (scanned form from patient)
15    filled_image_path = r"D:\\Uni\\Y4S1\\FIT 3164 - DS Project 2\\Project\\SimpleHTR-master\\src\\PRF_Cursive.jpg"
16    # "D:\\Uni\\Y4S1\\FIT 3164 - DS Project 2\\Project\\SimpleHTR-master\\src\\PRF_Red.jpg"
17    filled_image = cv2.imread(filled_image_path, cv2.IMREAD_GRAYSCALE)
18    if filled_image is None:
19        | raise ValueError(f"Filled image not found at {filled_image_path}")
20
21    # Resize filled image to match template dimensions
22    template_height, template_width = template_image.shape
23    filled_image = cv2.resize(filled_image, (template_width, template_height))
24
25    # Detect keypoints and descriptors using ORB detector
26    orb = cv2.ORB_create()
27    keypoints_template, descriptors_template = orb.detectAndCompute(template_image, None)
28    keypoints_filled, descriptors_filled = orb.detectAndCompute(filled_image, None)
29
30
31    # Match descriptors using BFMatcher
32    bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
33    matches = bf.match(descriptors_template, descriptors_filled)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\\Uni\\Y4S1\\FIT 3164 - DS Project 2\\Project\\SimpleHTR-master\\src> pip install numpy

Table 2.2.1.4. Screenshots of the installation packages for the ICROP model.

- Verify installation of all required packages. Ensure that all the required packages are listed, and their versions match the specified requirements for the HTR and ICROP models.

```

EXPLORER ... main.py AHIS_CropModel1.py requirements.txt
src > AHIS_CropModel1.py > ...
1 import cv2
2 import numpy as np
3 import os
4
5
6 def crop():
7     # Load the template image (reference form)
8     template_image_path = r"D:\\Uni\\Y4S1\\FIT 3164 - DS Project 2\\Project\\SimpleHTR-master\\src\\PRF_Template.jpg"
9     # print(template_image_path)
10    template_image = cv2.imread(template_image_path, cv2.IMREAD_GRAYSCALE)
11    if template_image is None:
12        | raise ValueError(f"Template image not found at {template_image_path}")
13
14    # Load the filled form image (scanned form from patient)
15    filled_image_path = r"D:\\Uni\\Y4S1\\FIT 3164 - DS Project 2\\Project\\SimpleHTR-master\\src\\PRF_Cursive.jpg"
16    # "D:\\Uni\\Y4S1\\FIT 3164 - DS Project 2\\Project\\SimpleHTR-master\\src\\PRF_Red.jpg"
17    filled_image = cv2.imread(filled_image_path, cv2.IMREAD_GRAYSCALE)
18    if filled_image is None:
19        | raise ValueError(f"Filled image not found at {filled_image_path}")
20
21    # Resize filled image to match template dimensions
22

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\\Uni\\Y4S1\\FIT 3164 - DS Project 2\\Project\\SimpleHTR-master\\src> pip list

Package	Version
absl-py	0.15.0
astunparse	1.6.3
cachetools	5.5.0
certifi	2024.7.4
charset-normalizer	3.3.2
colorama	0.4.6
contourpy	1.1.1
coverage	7.6.1
cycler	0.12.1
editdistance	0.8.1
exceptiongroup	1.2.2
flatbuffers	1.12

Table 2.2.1.5. Screenshots of the list of packages for the HTR and ICROP model.

2.2.2. Run ICROP Model

The ICROP model is already integrated into the web-based system providing a streamlined experience for users. However, if you want to run the ICROP model independently, you can follow these steps:

1. Open Visual Studio Code (VSC)
2. Navigate to the folder where the ICROP model is stored. Open the ‘AHIS_CropModel1.py’ python file.
3. Ensure that the path to the form template exists and is correct.

```

    import cv2
    import numpy as np
    import os

    def crop():
        # Load the template image (reference form)
        template_image_path = r"C:\Users\Alicia\OneDrive - Monash University\Desktop\FIT3164\Project\automated-health-information-system\MDS2_AHIS\HTR_Model\SimpleHTR-master\src\Form_Template"
        template_image = cv2.imread(template_image_path, cv2.IMREAD_GRAYSCALE)
        if template_image is None:
            raise ValueError(f"Template image not found at {template_image_path}")

        # Load the filled form image (scanned form from patient)
        filled_image_path = r"C:\Users\Alicia\OneDrive - Monash University\Desktop\FIT3164\Project\automated-health-information-system\MDS2_AHIS\data_files\patientRegistrationForm.jpg"
        filled_image = cv2.imread(filled_image_path, cv2.IMREAD_GRAYSCALE)
        if filled_image is None:
            raise ValueError(f"Filled image not found at {filled_image_path}")

        # Resize filled image to match template dimensions
        template_height, template_width = template_image.shape
        filled_image = cv2.resize(filled_image, (template_width, template_height))

        # Detect keypoints and descriptors using ORB detector
        orb = cv2.ORB_create()
        keypoints_template, descriptors_template = orb.detectAndCompute(template_image, None)
        keypoints_filled, descriptors_filled = orb.detectAndCompute(filled_image, None)

        # Match descriptors using BFMatcher
        bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
        matches = bf.match(descriptors_template, descriptors_filled)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\Uni\Y4S1\FIT 3164 - DS Project 2\Project\SimpleHTR-master\src> pip list

Package	Version
absl-py	0.15.0
astunparse	1.6.3

powerShell Python

Table 2.2.2.1. Screenshots of the template path for the ICROP model.

4. Ensure that the path to the form that you want to crop.

```

    import cv2
    import numpy as np
    import os

    def crop():
        # Load the template image (reference form)
        template_image_path = r"C:\Users\Alicia\OneDrive - Monash University\Desktop\FIT3164\Project\automated-health-information-system\MDS2_AHIS\HTR_Model\SimpleHTR-master\src\Form_Template"
        template_image = cv2.imread(template_image_path, cv2.IMREAD_GRAYSCALE)
        if template_image is None:
            raise ValueError(f"Template image not found at {template_image_path}")

        # Load the filled form image (scanned form from patient)
        filled_image_path = r"C:\Users\Alicia\OneDrive - Monash University\Desktop\FIT3164\Project\automated-health-information-system\MDS2_AHIS\data_files\patientRegistrationForm.jpg"
        filled_image = cv2.imread(filled_image_path, cv2.IMREAD_GRAYSCALE)
        if filled_image is None:
            raise ValueError(f"Filled image not found at {filled_image_path}")

        # Resize filled image to match template dimensions
        template_height, template_width = template_image.shape
        filled_image = cv2.resize(filled_image, (template_width, template_height))

        # Detect keypoints and descriptors using ORB detector
        orb = cv2.ORB_create()
        keypoints_template, descriptors_template = orb.detectAndCompute(template_image, None)
        keypoints_filled, descriptors_filled = orb.detectAndCompute(filled_image, None)

        # Match descriptors using BFMatcher
        bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
        matches = bf.match(descriptors_template, descriptors_filled)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\Uni\Y4S1\FIT 3164 - DS Project 2\Project\SimpleHTR-master\src> pip list

Package	Version
absl-py	0.15.0
astunparse	1.6.3

powerShell Python

Table 2.2.2.2. Screenshots of the filled form path for the ICROP model.

5. Ensure that the path to cropped images is to the crop_images folder.

```

EXPLORER          ...   main.py   AHIS_CropModel1.py src   model.py   AHIS_CropModel1.py C:\Uni\X   requirements.txt

C:\> Uni > AHIS_CropModel1.py > crop
1  import cv2
2  import numpy as np
3  import os
4
5
6  def crop():
7      # Load the template image (reference form)
8      template_image_path = "C:/Users/Alicia/OneDrive - Monash University/Desktop/FIT3164/Project/automated-health-information-system/MDS2_AHIS/HTR_Model/SimpleHTR-master/src/Form_Template"
9      template_image = cv2.imread(template_image_path, cv2.IMREAD_GRAYSCALE)
10     if template_image is None:
11         raise ValueError(f"Template image not found at {template_image_path}")
12
13     # Load the filled form image (scanned form from patient)
14     filled_image_path = "C:/Users/Alicia/OneDrive - Monash University/Desktop/FIT3164/Project/automated-health-information-system/MDS2_AHIS/data_files/patientRegistrationForm.jpg"
15     filled_image = cv2.imread(filled_image_path, cv2.IMREAD_GRAYSCALE)
16
17     if filled_image is None:
18         raise ValueError(f"Filled image not found at {filled_image_path}")
19
20     # Resize filled image to match template dimensions
21     template_height, template_width = template_image.shape
22     filled_image = cv2.resize(filled_image, (template_width, template_height))
23
24     # Detect keypoints and descriptors using ORB detector
25     orb = cv2.ORB_create()
26     keypoints_template, descriptors_template = orb.detectAndCompute(template_image, None)
27     keypoints_filled, descriptors_filled = orb.detectAndCompute(filled_image, None)
28
29     # Match descriptors using BFMatcher
30     bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
31     matches = bf.match(descriptors_template, descriptors_filled)
32
33     # Draw matches
34     draw_params = dict(matchColor=(0, 255, 0),
35                         singlePointColor=(255, 0, 0),
36                         matchesMask=None,
37                         flags=cv2.DrawMatchesFlags_DEFAULT)
38
39     result = cv2.drawMatches(template_image, keypoints_template, filled_image, keypoints_filled, matches, None, **draw_params)
40
41     # Save the result
42     cv2.imwrite("cropped_result.jpg", result)
43
44     return result
45
46
47 if __name__ == "__main__":
48     crop()

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\Uni\Y4S1\FIT 3164 - DS Project 2\Project\SimpleHTR-master\src> pip list

Package	Version
absl-py	0.15.0
astunparse	1.6.3

powershell Python

Table 2.2.2.3. Screenshots of crop_images path for the ICROP model.

6. Run the AHIS_CroppedModel1.py file.
7. A number of jpg files will be stored in the crop_images folder.



Table 2.2.2.4. Screenshots of cropped images with their respective names in crop_images folder

2.2.3. Run HTR Model

The HTR is already integrated into the web-based system allowing seamless interaction through the web interface. However, if you wish to run the HTR model independently, follow the steps below:

1. Open Visual Studio Code (VSC)
2. Navigate to the folder where the HTR model is stored.
3. Open the “data files” folder. Ensure that the image you want the HTR to run is present in the crop_images folder.

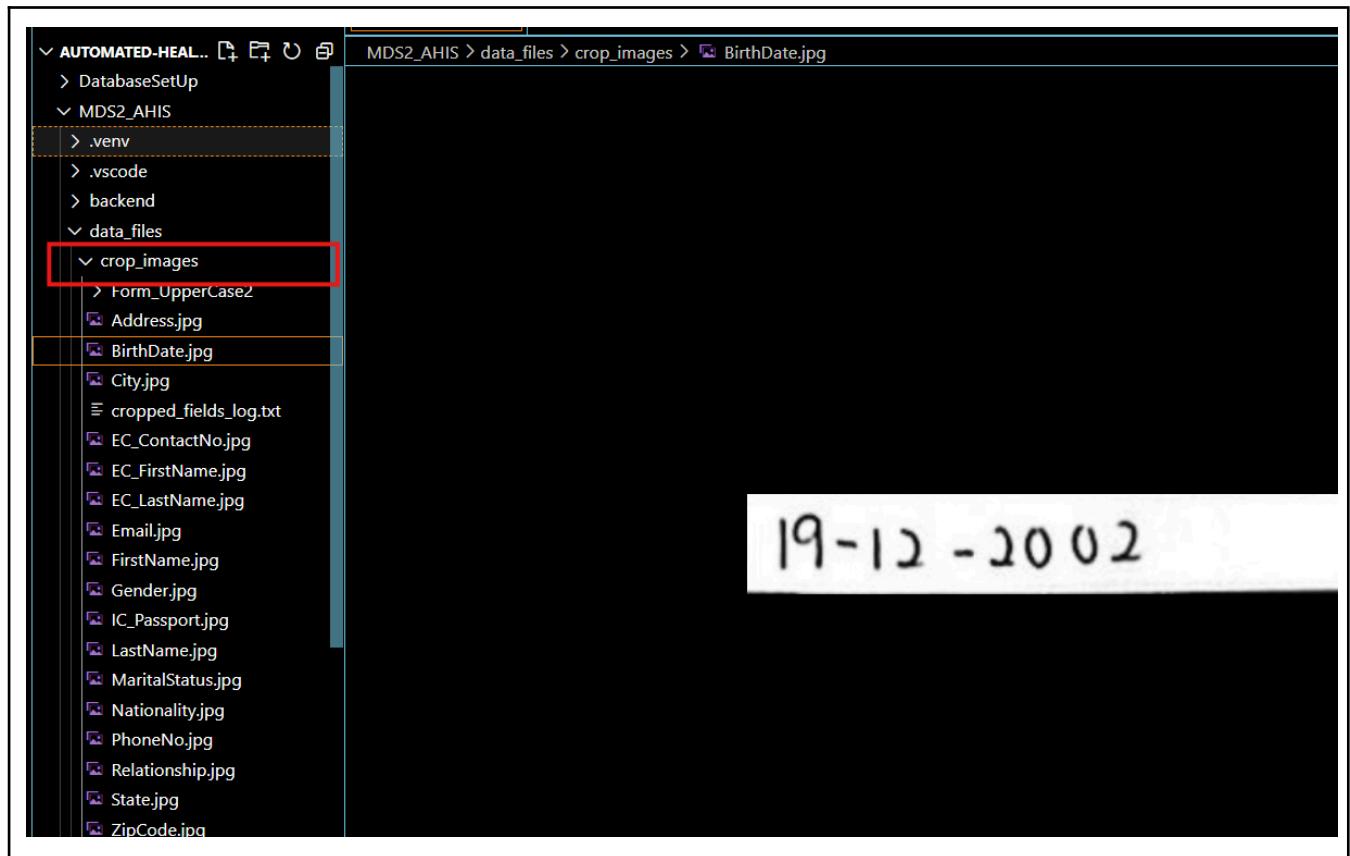


Table 2.2.3.1. Screenshots of the crop_images folder containing images for the HTR model.

4. Run the python file either by terminal or pressing the run button on the top right in VSC.

A screenshot of the Visual Studio Code interface. The top bar shows tabs for 'CropModel1.py' and 'requirements.txt'. On the right side of the top bar, there is a red rectangular box highlighting the green 'Run' button icon. The main editor area contains Python code related to cropping images. Below the editor is a terminal window showing the command 'pip list'. A sidebar on the right lists 'powershell' and 'Python'.

Table 2.2.3.2. Screenshots of the run button in VSC.

5. A “patient_data.json” file is created to store the patient data of the HTR model results. Open the json file to see the results.

A screenshot of the Visual Studio Code interface. The left sidebar shows a tree view of project files under 'AUTOMATED-HEALTH-INFORMATION...'. The 'EXPLORER' tab is selected. In the center, the 'patient_data.json' file is open in the editor, displaying its JSON content. The content is as follows:

```

1  {
2     "streetAddress": "Kota Damansara",
3     "birthDate": "19-12-1002",
4     "city": "Petaling Jaya",
5     "emgcyTel": "012-333 5555",
6     "emgcyFname": "rai ran",
7     "emgcyLname": "oo",
8     "email": "eunicelee 1219@gmail.com",
9     "firstName": "Eunice",
10    "gender": "female",
11    "ICPassport": "0111914 1196",
12    "lastName": "Lee",
13    "maritalStatus": "singre",
14    "nationality": "malaysian",
15    "tel": "012-6881311",
16    "emgcyRelationship": "Friend",
17    "state": "selangor",
18    "postalCode": "179is"
19 }

```

Table 2.2.3.3. Screenshots of the patient_data.json file containing results recognized by the HTR model.

2.2.4. Software Limitations and Restriction

1. Performance Variation:

The performance of the HTR and ICROP models can vary significantly based on the quality and clarity of the input images. If the images are poorly lit, blurred or have low resolution, the models may struggle to accurately recognize text or execute cropping tasks. Therefore, users should ensure that the images fed into the system are of high quality to achieve optimal results. Additionally, users should consider factors such as background noise or distracting elements in the image as these can further complicate the models' ability to deliver accurate outputs.

2. System Configuration Requirements:

Both the HTR and ICROP models require specific system configurations and dependencies to function correctly. Any deviations or changes made to the environment such as updates to operating systems, libraries or frameworks may adversely affect the models' performance. Users must ensure that their systems are properly configured and maintain the specified dependencies to prevent any disruptions in functionality. Additionally, it is advisable for users to document their configuration settings and maintain a backup of previous versions to facilitate troubleshooting in case of any issues that may arise.

3. Complex Layout Limitations:

The HTR and ICROP models may encounter difficulties when dealing with highly complex layouts or forms that feature overlapping text. In scenarios where text is densely packed or arranged in unconventional patterns, the models may not perform as effectively resulting in potential inaccuracies. Users should be cautious when using the models for documents with intricate layouts, as they might not achieve the desired output in such cases. Furthermore, the presence of graphical elements or non-textual components in the layout can further complicate the recognition process and lead to degraded performance.

4. Resource Limitations:

The software's performance can be significantly influenced by the hardware specifications of the machine on which it runs. This makes resource limitations a critical consideration for users. Those operating on machines with limited system resources may encounter slower processing times or reduced responsiveness particularly when handling large amounts of image files. This performance degradation can impact user experience and hinder the efficiency of workflows especially in scenarios where quick data retrieval and processing are essential.

5. Limited Input Formats:

The HTR model is specifically designed to work primarily with certain image formats such as JPG, JPEG and PNG which are widely used for their compatibility and quality. Users should ensure that their input images conform to these formats to prevent errors during processing as unsupported formats can lead to failed operations or inaccurate results. In addition to adhering to the required image formats, users must also ensure that the ICROP model receives forms that are properly formatted. If the input forms do not meet the expected formatting standards, the cropping results may not align with user intentions that leads to potential inaccuracies or omissions in the final output.