A logo with black text

Description automatically generated

**NIT6150 Advanced Project**

**Technical Documentation**

**HealthCare Chatbot System**

**A hand holding a heart with a pulse line

Description automatically generated**

**Team Leader:** Jitendra Shrestha (s8104215)

**Team Member:** Pranish Acharya (s8100698)

**Client:** Holroyd Private Hospital

**Supervisor:** Fakhra Jabeen

VU Sydney

**Table of Contents**

[**1.Introduction** 1](#_Toc176733180)

[1.1 Tools and Technologies 1](#_Toc176733181)

[**2. Code Documentation** 2](#_Toc176733182)

[2.1 Frontend Development 2](#_Toc176733183)

[2.1.1 Functions Explanation 2](#_Toc176733184)

[2.2 Backend Developement 3](#_Toc176733185)

[2.2.1 Database Structure 3](#_Toc176733186)

[2.2.2 API 5](#_Toc176733187)

[2.2.3 Folder Structure 6](#_Toc176733188)

[2.2.4 Password Recovery 7](#_Toc176733189)

[2.3 Training the Chatbot 7](#_Toc176733190)

[2.4. Integration 9](#_Toc176733191)

[**3. Troubleshooting Issues** 9](#_Toc176733192)

[**4. Testing** 9](#_Toc176733193)

[4.1 Unit Testing 9](#_Toc176733194)

[4.2 UI Testing 10](#_Toc176733195)

# **1.Introduction**

Healthcare chatbot system is developed with subsystem of user management system, personal health management system and chatbot. Each system is developed alone and integrated at the end.

## Tools and Technologies

|  |  |  |  |
| --- | --- | --- | --- |
| S.N. | Category | Tool/Technology | Description |
| 1. | Programming Language | Python (3.8.0) | Overall language used for the system |
| 2. | Frameworks | Django (4.2.16) | Web framework for building application’s backend and handling requests |
| RASA (3.6.20) | Open-source framework for building conversational AI and chatbots |
| 3. | Testing | Unit Testing | Built-in python module |
| Selenium (4.24.0) | Tool used for web browsers to perform UI testing and ensure the UI behaves correctly |
| 4. | Web Technologies | HTML, CSS, JavaScript | Creating webpages |
| 5. | Libraries | jQuery | Fast JavaScript library to simplify the DOM manipulation and event handling |
| Bootstrap | For responsive design |
| 6. | Database | SQLite | Default Database used by Django |
| 7. | Version Control | Git | Version Controlling all the project and documentation |
| 8. | Other Tools | Trello | We have used Trello for project management. Assigning tasks and tracking the deadlines.  Divided the tasks for each member |
| MindView(9.0) | Tools used for work breakdown structure |

# **2. Code Documentation**

## 2.1 Frontend Development

### 2.1.1 Functions Explanation

|  |  |  |
| --- | --- | --- |
| S.N. | Function Name | Purpose |
| 1. | homepage | Renders the website homepage |
| 2. | patienthome | Renders the patient homepage |
| 3. | aboutpage | Renders the patient about us page |
| 4. | loginpage | Renders the login form page where user can provide username and password to login into the system |
| 5. | createaccountpage | To create the new account for patients |
| 6. | chatbot\_response | Renders the chatroom page |
| 7. | updatepassword | Renders the update password page to change the password of the logged in user |
| 8. | adminaddDoctor | Renders the page to add doctor |
| 9. | adminviewDoctor | Renders the page to view doctor |
| 10. | admin\_delete\_doctor | To delete the doctor |
| 11. | patient\_delete\_appointment | To delete the appointment by patient |
| 12. | adminviewAppointment | Renders the appointment page for admin |
| 13. | Logout | Logout from the system |
| 14. | Logout\_admin | Logout admin from the system |
| 15. | AdminHome | Renders the homepage of admin |
| 16. | Home | Renders the home according to the user category. |
| 17. | profile | Renders the profile according to the user category |
| 18. | MakeAppointments | Renders the page to book the appointments |
| 19. | viewappointments | Renders the page to view the appointments |
| 20. | viewhealthrecords | Renders the page to view the past appointments |
| 21. | contactus | Renders the contact us page |
| 22. | get\_available\_time\_slots | This function will generate the time slot for everyday according to the date chosen , also checks in the system that chosen date has available time slot or not  This can be only selected if there is time slot in that date and chosen date is in weekday. |

## 2.2 Backend Developement

### 2.2.1 Database Structure

|  |  |  |  |
| --- | --- | --- | --- |
| S.N. | Model Name | Fields | Description |
| 1. | Doctor | |  |  | | --- | --- | | name | Stores the name of max characters 50 | | email | Stores the email and unique | | licenseNo | Stores the license number of authorized doctor | | gender | Stores the gender either male or female | | phonenumber | Stores the phone number of characters 10 | | address | Stores the address of characters up to 100 | | specialization | Specifies the specialization of doctor | | |
| 2. | Patient | |  |  | | --- | --- | | name | Stores the name of max characters 50 | | email | Stores the email and unique | | gender | Stores the gender either male or female | | phonenumber | Stores the phone number of characters 10 | | |
| 3. | Contact | |  |  | | --- | --- | | name | Stores the name of max characters 50 | | email | Stores the email and unique | | phonenumber | Stores the phone number of characters 10 | | message | Stores the message provided by the user | | |
| 4. | Appointment | |  |  | | --- | --- | | doctorname | Stores the name of max characters 50 | | doctoremail | Stores the email of doctor | | patientname | Stores the patient name | | patientemail | Stores the patient email | | appointmentdate | Stores the date of the appointment | | followupdate | Stores the date of follow up | | symptoms | Stores the symptoms provided by patient when booking appointment | | status | Specifies means the doctor has completed prescription or not | | prescription | Prescription for the client by doctor | | appointment\_time | Appointment time slot for the chosen date | | |

### 2.2.2 API

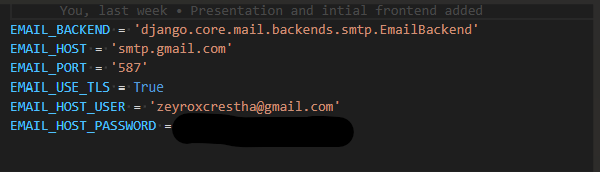
|  |  |  |
| --- | --- | --- |
| URL Path | View Function | Description |
| '' | views.homepage | Main landing page |
| 'home/' | views.home | User's home page after login |
| 'about/' | views.aboutpage | Information about the healthCare |
| 'contact/' | views.contactus | Contact information or form |
| 'profile/' | views.profile | User profile page |
| 'chatroom/' | chatviews.chatroom | Chatbot feature |
| 'login/' | views.loginpage | User login page |
| 'logout/' | views.logout | User logout functionality |
| 'adminlogout/' | views.logout\_admin | Admin-specific logout |
| 'createaccount/' | views.createaccountpage | New patient registration |
| 'patienthome/' | views.patienthome | Dashboard for patients |
| 'doctorhome/' | views.doctorhome | Dashboard for doctors |
| 'adminhome/' | views.adminhome | Dashboard for administrators |
| 'adminadddoctor/' | views.adminadddoctor | Admin interface to add new doctors |
| 'adminviewdoctor/' | views.adminviewdoctor | Admin interface to view doctor list |
| 'admindeletedoctor/int:pk/str:email/' | views.admin\_delete\_doctor | Admin interface to remove a doctor |
| 'adminviewappointment/' | views.adminviewappointment | Admin interface to view all appointments |
| 'makeappointment/' | views.makeappointments | Interface for booking appointments |
| 'viewmypatients/' | views.viewmypatients | Doctor's interface to view their patients |
| 'viewallrecords/' | views.viewallrecords | View all medical records (likely admin or doctor) |
| 'PatientDeleteAppointment/int:pid/' | views.patient\_delete\_appointment | Allow patients to cancel appointments |
| 'get-available-time-slots/' | views.get\_available\_time\_slots | API to fetch open appointment slots |
| 'updatepassword/' | views.updatepassword | Interface for users to change password |
| 'reset\_password/' | auth\_views.PasswordResetView.as\_view() | Initiate password reset process |
| 'reset\_password\_sent/' | auth\_views.PasswordResetDoneView.as\_view() | Confirmation of reset email sent |
| 'reset/<uidb64>/<token>/' | auth\_views.PasswordResetConfirmView.as\_view() | Page to enter new password |
| 'reset\_password\_complete/' | auth\_views.PasswordResetCompleteView.as\_view() | Confirmation of password reset |

### 2.2.3 Folder Structure

* System
  + Frontend
  + HealthBot
  + Healthcarechatbotsystem
  + Manage.py

### 2.2.4 Password Recovery

This function is implemented by using Django SMTP email backend. For now, we have used our own email address by generating the app password. Later on, we can replace with the client one. We have use Gmail so se the host as GMAIL.



## 2.3 Training the Chatbot

We have used RASA Framework for the chatbot. For the chatbot we have divided the file to train the chatbot.

|  |  |  |
| --- | --- | --- |
| S.N. | Name | Description |
| 1. | nlu.yml | This file contains the user intents that means the input expected from the user and categorized into single topic.  Example:  - intent: thank\_you    examples: |      - thank you      - thanks      - thnx      - thank you very much      - awesome      - wow  By this we can know user can say thank you in many ways.  Like this we have created more intents. |
| 2. | rules.yml | Rules contains specific criteria whenever we get something unique than expected.  Example:  - rule: Say goodbye anytime the user says goodbye    steps:    - intent: goodbye    - action: utter\_goodbye  By this we can see any time user say bye it will response by responses in between chats also. |
| 3. | stories.yml | This file contains the user story where user can go through the patterns one after another.  Example:  - story: gas stomach    steps:      - intent: stomach\_symptom      - action: utter\_day      - intent: day      - action: utter\_stomach\_types      - intent: affirm      - action: utter\_gas\_response      - action: utter\_did\_that\_help  By this we can see intent represent input from user and action is for response from bot.  If anything matches it will select from multiple stories. |
| 4. | domain.yml | This file contains the collections of intents, responses and text message for the user . |

## 2.4. Integration

As we have separately developed the sub systems. We have integrated all using the available servers running on different port. Below is the detail breakdown of the integration between servers.

|  |  |  |
| --- | --- | --- |
| S.N. | Host URL | Description |
| 1. | Localhost:8000 | From this server the Django frontend and backend both is handled |
| 2. | Localhost:5005 | From this server the RASA farmwork will be running and replying to the response to the user. |

# **3. Troubleshooting Issues**

For issues related to Django it will be seen in the logs through the terminal from where it is being start up.

For issues related to chatbot it will be seen in the logs through the terminal from where it is being start up. Also we can enable debug logs for chatbot from which every step for AI.

# **4. Testing**

## 4.1 Unit Testing

* We have written and executed unit tests for each function using built-in python module. These tests ensure that individual components of our code are functioning correctly in isolation.
* Each unit test is designed to verify specific functionality and edge cases, helping us identify and fix issues at the function level before integration.

## 4.2 UI Testing

* For user interface testing, we utilized Selenium, a powerful tool for automating web browsers. Selenium allowed us to create automated test scripts that interact with the application's user interface.
* Our UI tests cover various scenarios, including form submissions, navigation, and user interactions, to ensure that the application behaves as expected from an end-user perspective.
* These tests help us verify that the user interface is both functional and user-friendly, catching any issues that may arise in real-world usage.

Together, these testing strategies provide a comprehensive approach to verifying the reliability and quality of our project, from individual functions to the overall user experience.

A screenshot of a computer program

Description automatically generated

***Screenshot of all UI testing and Unit testing***