Table of Contents

Introduction to the Organization

We are a healthcare analytics team committed to improving patient outcomes through predictive insights. Our project is centred on leveraging **shiny web development** and applying machine learning to predict the risk of **Stroke.** Stroke remains a critical public health challenge globally, ranking among the top 10 leading causes of death according to the World Health Organization (WHO). Our initiative aims to develop a web application that identifies individuals at high risk of stroke, enabling timely intervention and potentially saving lives of the risk population.

Scope of the Project

The project involves creating a comprehensive platform for stroke prediction and targeting patients and healthcare professionals. The system allows users to input clinical data to receive the result of stroke risk predictions using a trained **Logistic Regression Model**. After this, the patient can book an appointment or view healthcare tips. For the Admin team, here is a Doctor's dashboard for visualizations, and an integrated chatbot powered by SQL LangChain for database queries and retrieving the necessary information. This aim will facilitate early detection of stroke risk for proactive health management, provide healthcare providers with real-time analytics and empower database administrators with an interactive query system for efficient data access.

Goals of the Project

- **Predictive Analytics**: Making sure that there is a reliable stroke risk prediction system to improve the use of early intervention.
- **Insights and Analytics**: To make sure that the doctors have visualised data on patient management
- **Intelligent Query System**: To empower the administrators and analysts with a chatbot for database queries which effectively have the **C.R.U.D** characteristics
- **Community Health Impact**: Support public health initiatives by generating data and patients who belong to a particular support group.

Conceptual Design (ER Diagram)

This database design has five primary tables: doctor, patient, vital, appointment, and support_group, and their associated relationships. These entities are structured to effectively manage healthcare records, book appointments, and ensure that a patient belongs to support.

Key Entities and Relationships

- Doctor Table: It consists of unique doctor records, connected to patients and appointments "Treats" and "Confirms" respectively
- Patient Table: Serves as the central entity, connecting to appointments, vitals, doctors, and support groups. This will ensure a strict tracking of patient-related data.
- Vital Table: Captures and describes the health metrics for each patient, aiding in the analysis of key indicators such as BMI and glucose levels.
- Appointment Table: It is assigned to a particular Doctor, helping track the status and details of patient visits.

- Support Group Table: This links the patients to a particular community, such as support groups, which will enhance holistic care.

The relationships which were used such as "Treats", "Schedules", "Describes", and "Belongs" define the interactions between entities. This design was chosen to ensure there is modularity and ease of integration with future functionalities, such as adding "prescriptions" entity.

Logical Design

Transformation to Tables

In this design, each entity in the E-R diagram was converted into a table with a unique primary key. For example, The Doctor table uses doctor_id as the primary key and the Patient table uses patient_id as the primary key.

In relationships, they were represented using foreign keys. An example is that the Appointment table includes doctor_id and patient_id as foreign keys to establish connections between doctors and patients. The Vital table includes patient_id as a foreign key to link health metrics to individual patients.

Normalization

The database design adheres to the Third Normal Form (3NF) for efficiency and data integrity. All tables contain atomic data, with unique rows. Attributes like avg_glucose_level and bmi in the Vital table are stored as single-valued attributes.

Partial dependencies were removed by ensuring every non-key attribute depends on the entire primary key. For example, in the Appointment table, attributes such as date_of_appointment and status depend on appointment_id.

Transitive dependencies were eliminated. For example, in the Support Group table, attributes like location_lat and location_long depend solely on support_group_id. Normalization ensures that each table has a clear purpose, reduces redundancy, and minimizes update anomalies, enhancing the overall efficiency and maintainability of the database.

Shiny Application Design

The User Interface and Workflow

My stroke Status

This is the entry point for both patients and healthcare providers. This phase allows users to input their age and clinical data to predict the likelihood of a stroke. It is powered by a Logistic Regression Model, which analyses the data and generates results that indicate the patient's stroke status, “Low risk” or “High Risk”.

Appointment Booking

This section connects patients with healthcare providers where patients select from available doctors and conveniently book their appointments through the platform. Once an appointment is confirmed, doctors receive instant notifications, keeping them updated on their schedules.

Support Groups

This feature fosters a sense of community among patients by enabling them to join groups based on shared health conditions. The platform uses geographical mapping to help users identify and connect with local support groups, creating opportunities for interaction and mutual support.

Health Tips

This feature provides users with valuable healthcare tips related to different types of strokes and answers to general healthcare questions. The results generated after every question are from **Chatgpt** which is taking information from the internet only and provide a solution to the asked question. The primary audience for this feature is patients seeking relevant and accessible health information.

Analytics Dashboard

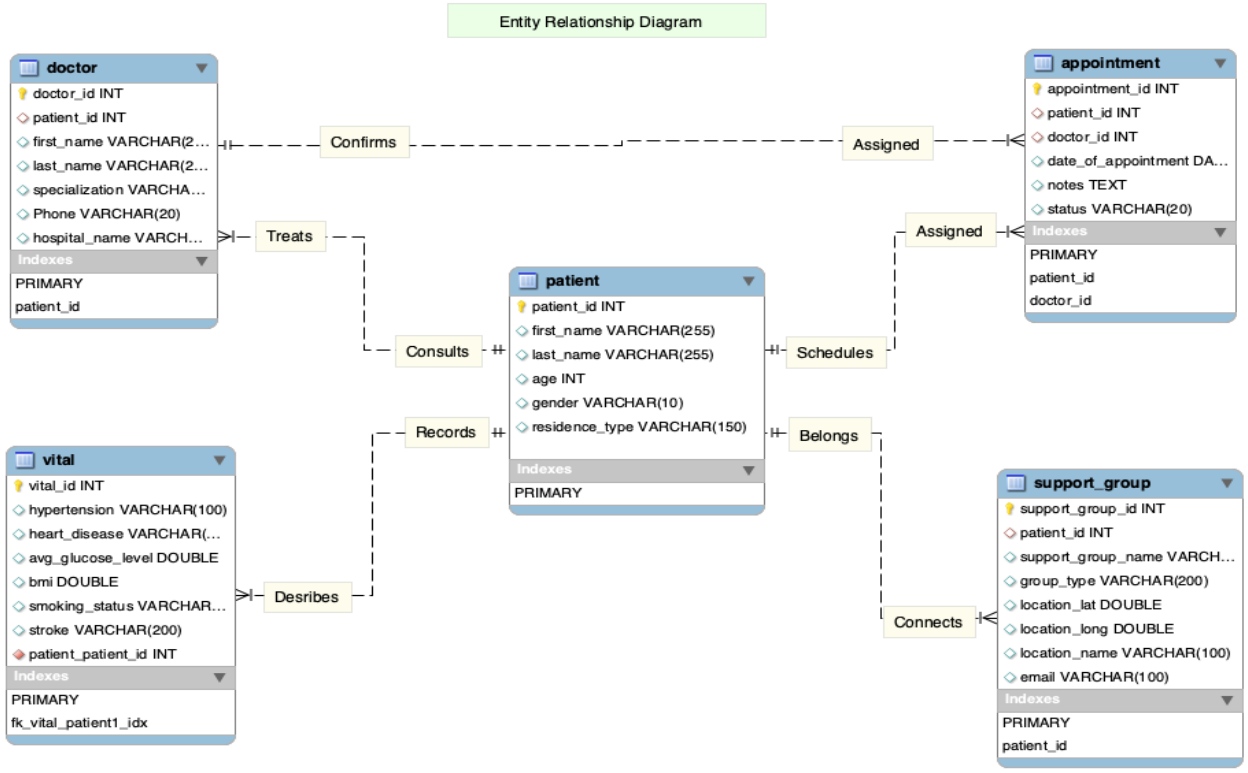
This is designed to empower doctors with data-driven insights into their patient data. With the provision of visual representations of key metrics, such as the number of patients with hypertension, stroke cases, and other essential health statistics, this tool helps healthcare providers identify trends, monitor patient outcomes, and make informed clinical decisions.

Reports

The platform features a robust Chatbot Integration powered by **SQL LangChain**, enabling admin users to interact with the system through natural language queries. This intelligent chatbot facilitates seamless database queries, providing instant access to relevant information. For example, Internal data analysts can ask, *"List or How many patients have hypertension?"* and receive real-time data for analysing. This feature enhances accessibility and ensures that healthcare providers can obtain critical information effortlessly.

APPENDIX SECTION

Appendix A: ERD Diagram



Appendix B: Relationships in Words

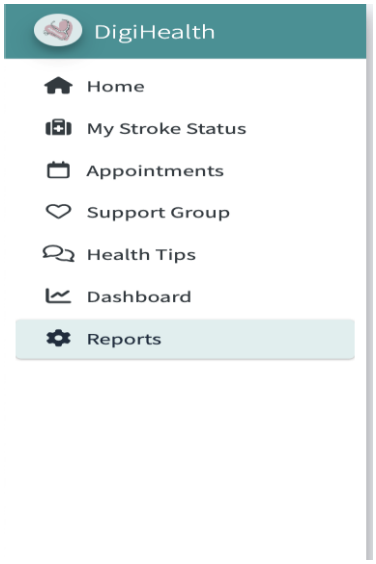
Entity/Table	Attributes	Primary Key	Foreign Keys	Relationship Example
Doctor Table	doctor_id, first_name, last_name, specialization, phone, hospital_name	doctor_id	None	"Confirms": Doctors confirm appointments. "Treats": Doctors treat patients.
Patient Table	patient_id, first_name, last_name, age, gender, residence_type	patient_id	None	Central entity related through: "Schedules", "Records", "Belongs" relationships. Links to Vitals, Appointments, etc
Vital Table	vital_id, hypertension, heart_disease, avg_glucose_level, bmi, smoking_status, stroke	vital_id	patient_id	- "Describes": Vital metrics are linked to patients via patient_id

Appointment Table	appointment_id, date_of_appointment, notes, status	appointment_id	patient_id, doctor_id	Connects patients and doctors for appointments.
Support Group Table	support_group_id, support_group_name, group_type, location_lat, location_long, email	support_group_id	patient_id	Connects Patients are linked to support groups via patient_id.

Appendix C: Data Dictionary

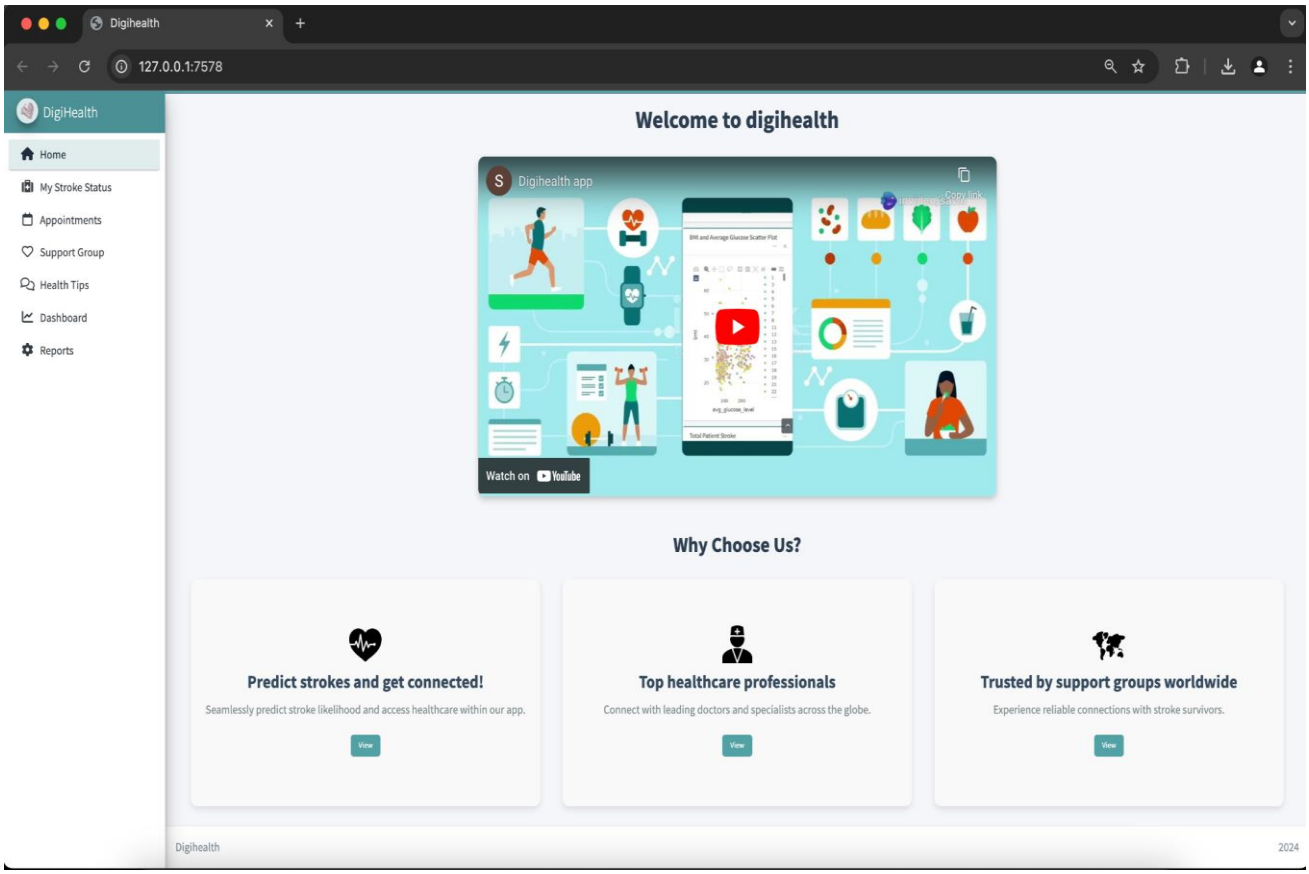
Entity	Description
Doctor	○ doctor_id: Unique ID for the doctor. ○ first_name, last_name: Personal details. ○ specialization: Doctor's area of expertise. ○ hospital_name: Associated hospital.
Patient	○ patient_id: Unique ID for the patient. ○ age, gender, residence_type: Demographic details. ○ work_type, smoking_status: Lifestyle information.
Vital	○ vital_id: Unique ID for the record. ○ bmi, avg_glucose_level: Key health metrics. ○ hypertension, heart_disease: Medical history indicators.
Appointment	○ appointment_id: Unique ID for the appointment. ○ date_of_appointment, diagnosis, treatment: Consultation details.
Support Group	○ support_group_id: Unique ID for the group. ○ support_group_name, location_lat, location_long: Group information.

Appendix D: Screenshots of Application Workflow



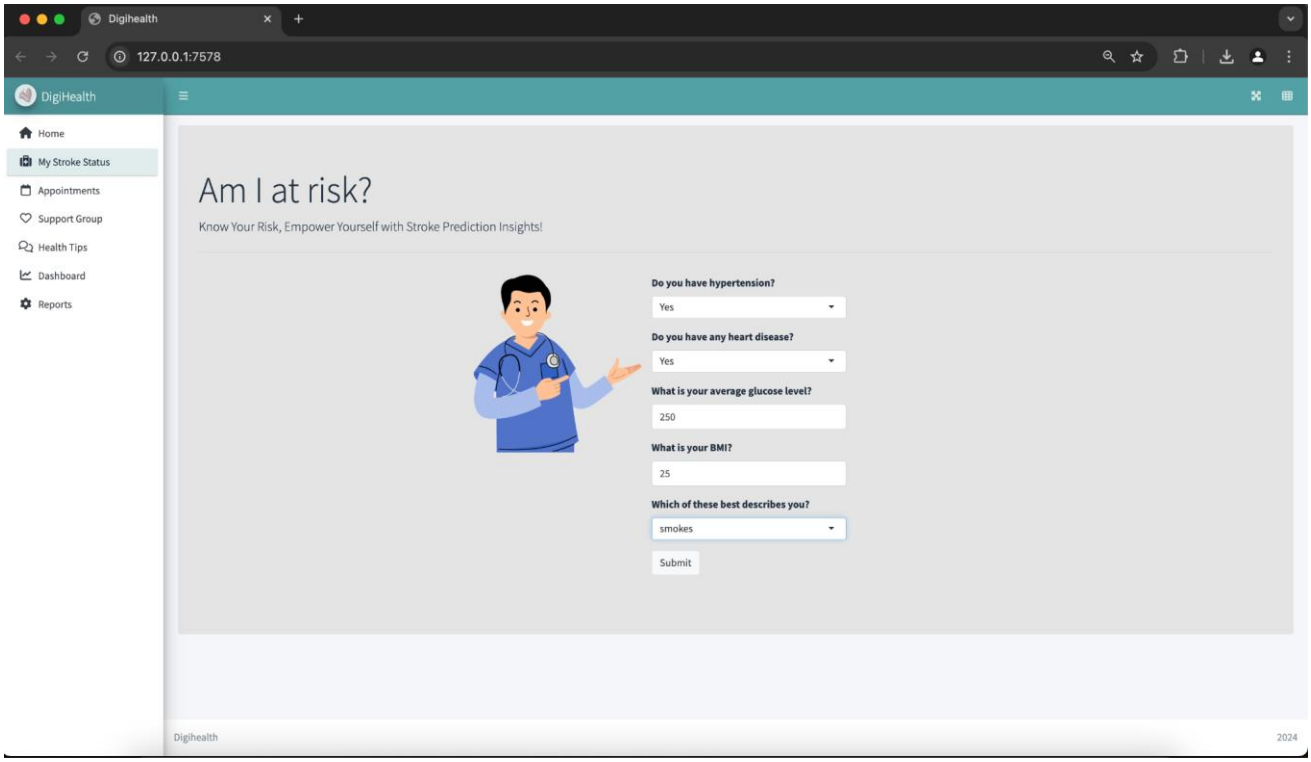
Home Page:

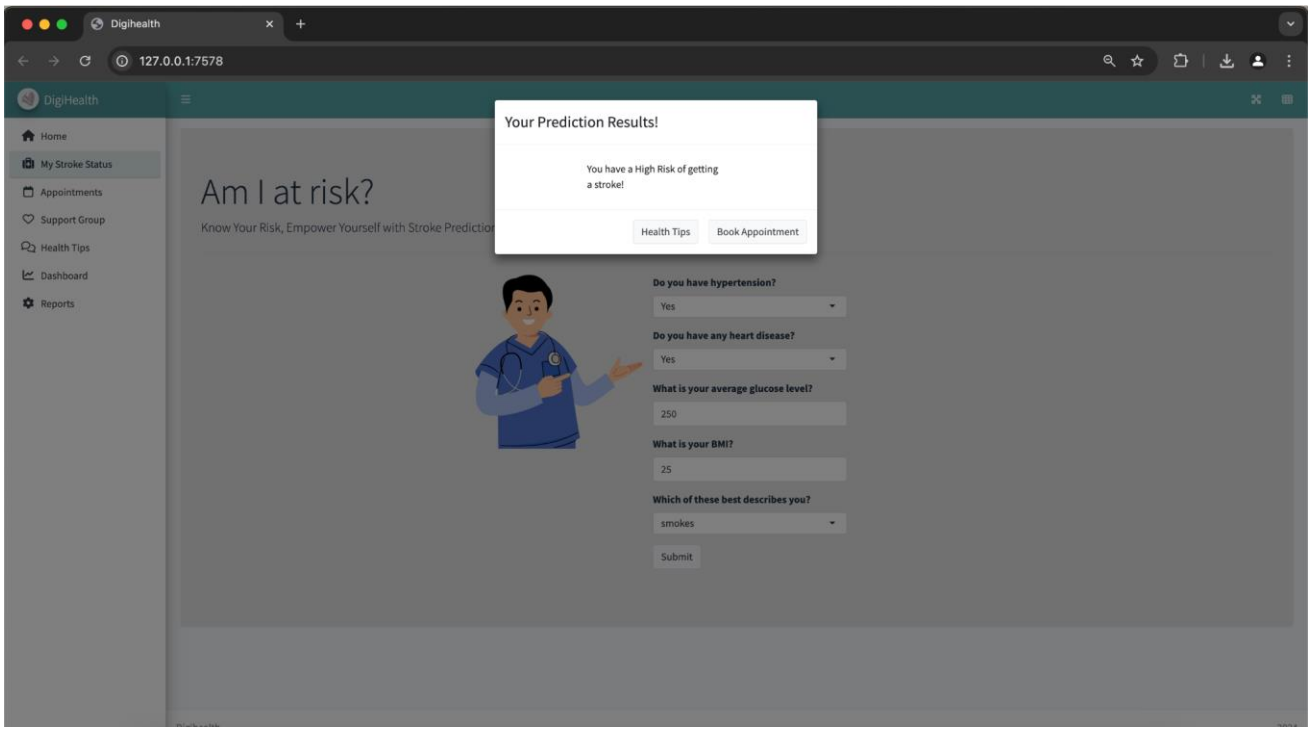
The user can view a video of the benefits of our application and how it works before booking and appointment.



My Stroke Status:

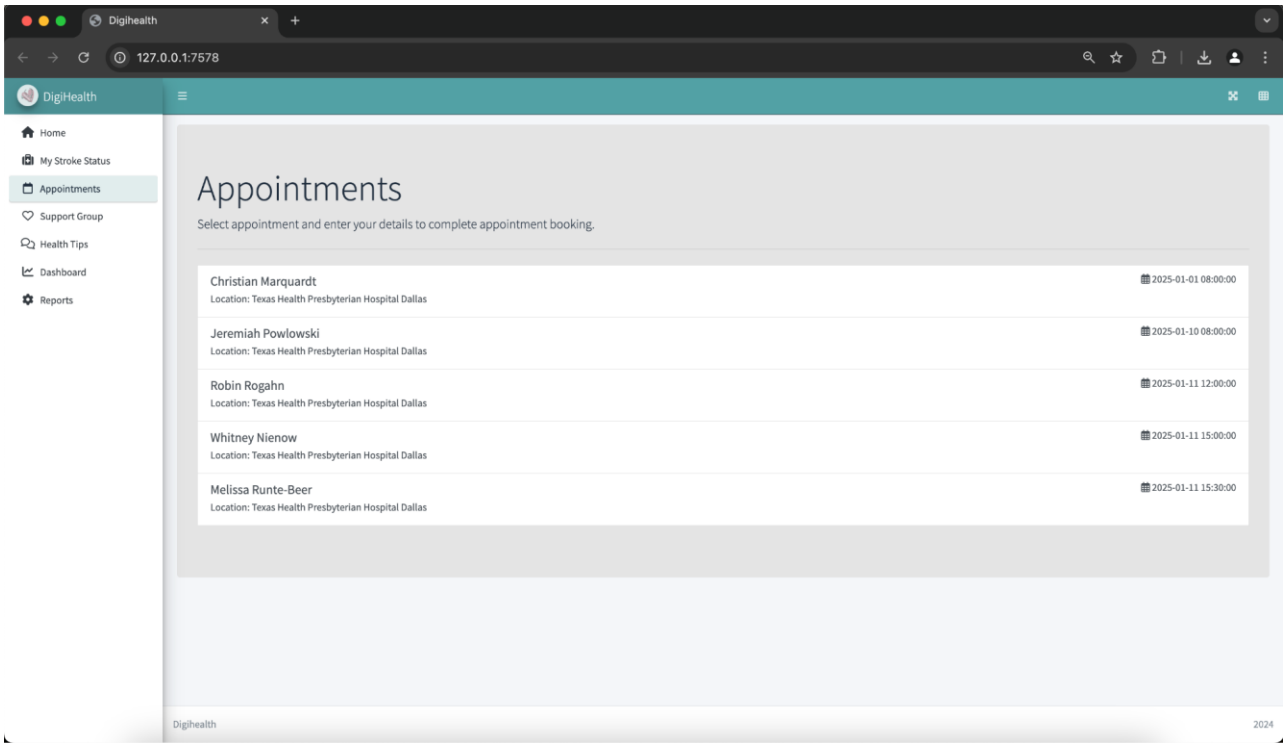
The user is allowed to do a stroke prediction to see the current health status. The predictions are shown in the screenshot below after the information is captured





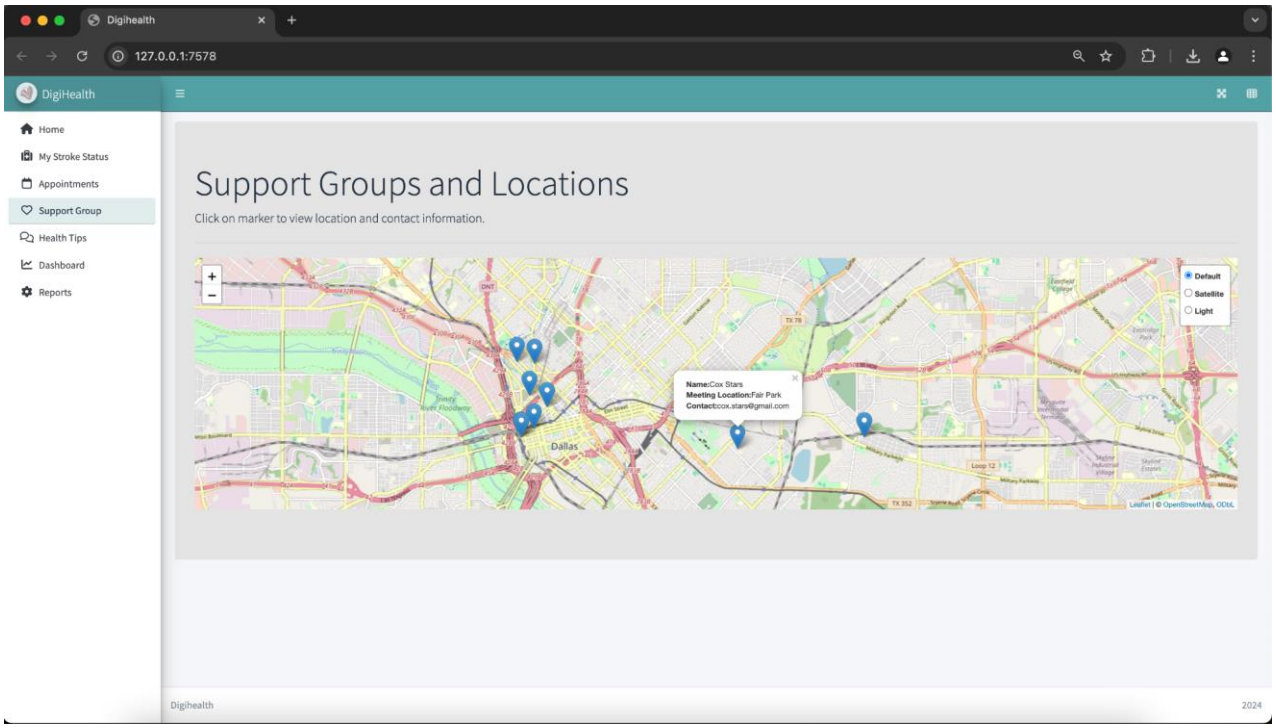
Appointment:

Depending on the available Doctors, the member can book the appointment in this section.



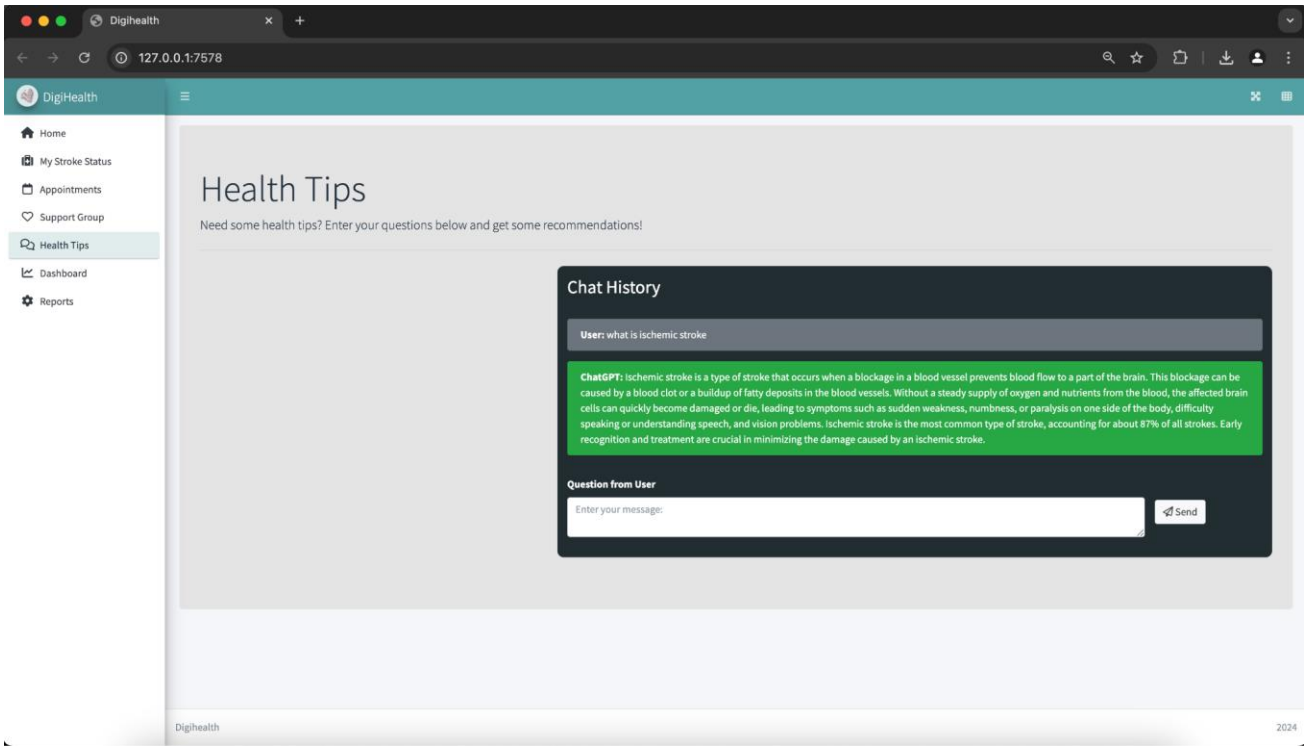
Support Groups:

Users can find a support group which is closer to them and connect basing on the geographical location.



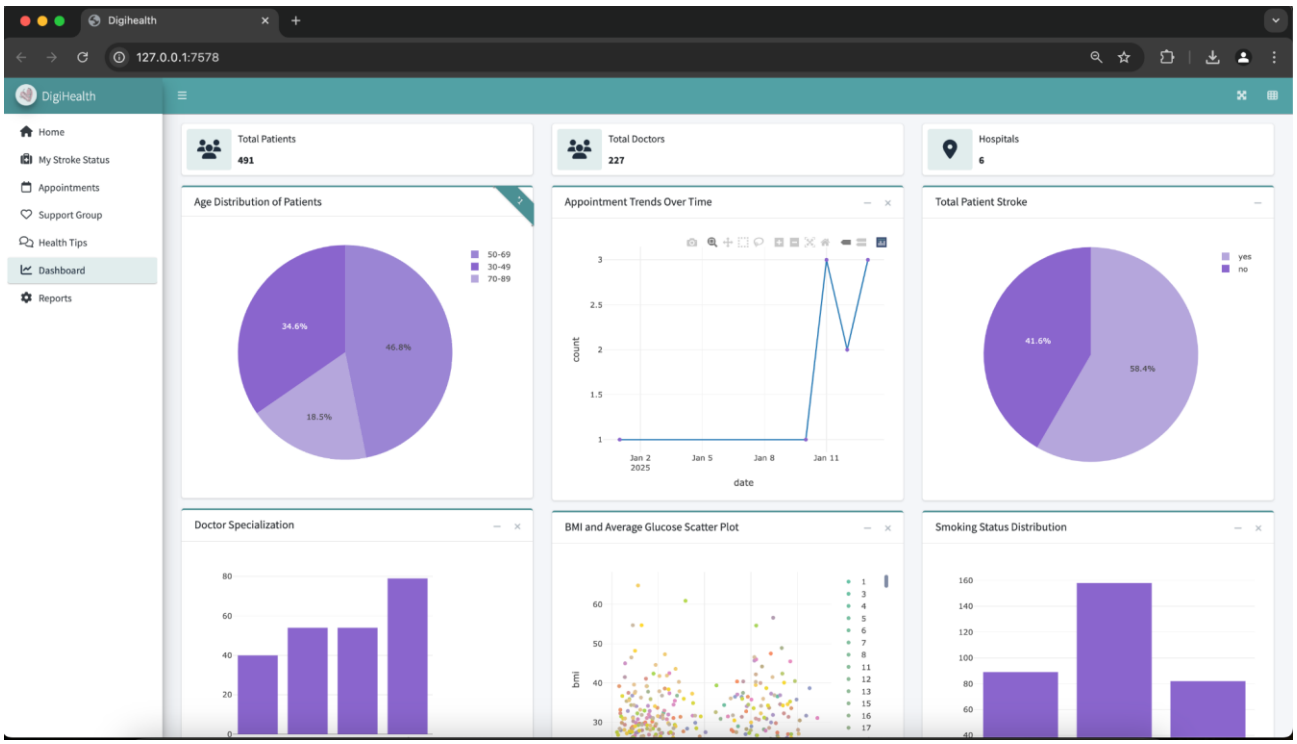
Health Tips:

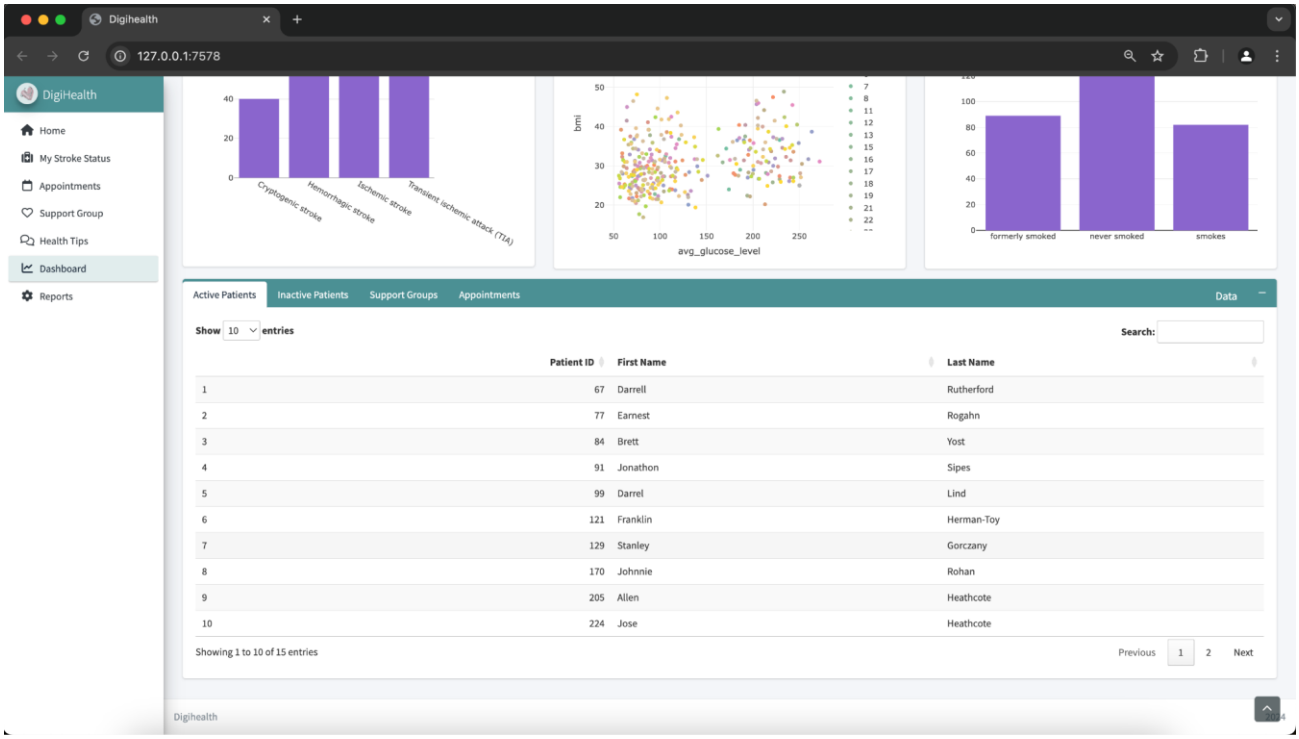
This a general Ask section which provides insights on the Healthcare tips depending on the question by the User, It gives general information about the Healthcare status. The user can ask what stroke is and get information about it.



Dashboard (Analytics)

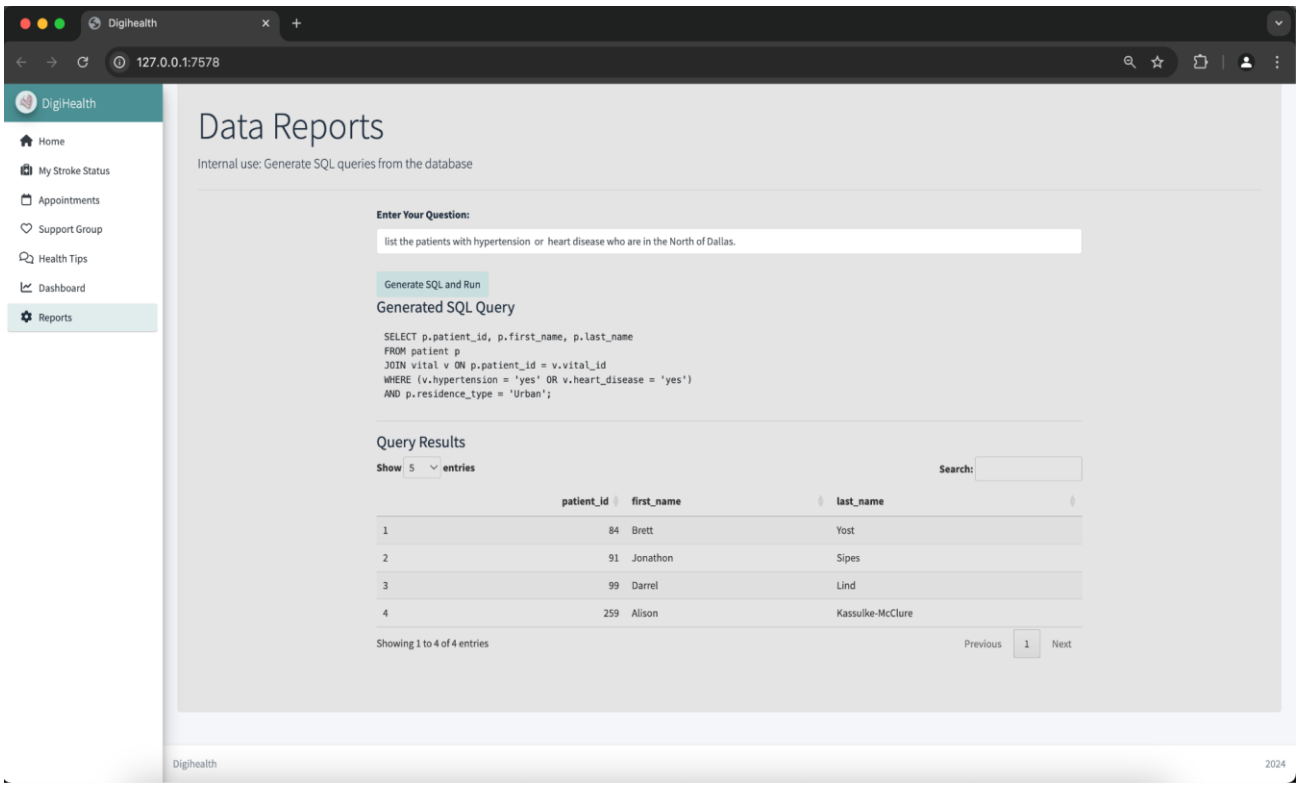
This is the Doctors section whereby we have so analytics view for the dashboard and analysing the trends for the available patients. There is also options to delete appointments





Reports(Chatbot Query Result):

This is the LLM sector which is for admin and analyst team within the organisation where they can generate reports based on what they need. It captures information from the database and it also research from the Internet to integrate with the data set.



Appendix E: SQL To Create Blank Tables

```sql
-- ********************************************************************************************
-- This script  describes all sql queries used in the Digihealth Application SQL queries --*
-- ********************************************************************************************
-- Creates a new schema (or database) named "digihealth"
 Run | New Tab
CREATE SCHEMA digihealth;
-- Sets the current schema/database to "digihealth" so that all subsequent commands will operate within this schema
 Run | New Tab
USE digihealth;

-- Create Patient table to store patient information----
 Run | New Tab | Copy
CREATE TABLE `patient` (
  `patient_id` int NOT NULL AUTO_INCREMENT,
  `first_name` varchar(255) DEFAULT NULL,
  `last_name` varchar(255) DEFAULT NULL,
  `age` int DEFAULT NULL,
  `gender` varchar(10) DEFAULT NULL,
  `residence_type` varchar(150) DEFAULT NULL,
  PRIMARY KEY (`patient_id`)
)

-- Create doctor table to store doctor information----
 Run | New Tab | Copy
CREATE TABLE `doctor` (
  `doctor_id` int NOT NULL AUTO_INCREMENT,
  `patient_id` int DEFAULT NULL,
  `first_name` varchar(255) DEFAULT NULL,
  `last_name` varchar(255) DEFAULT NULL,
  `specialization` varchar(200) DEFAULT NULL,
  `Phone` varchar(20) DEFAULT NULL,
  `hospital_name` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`doctor_id`),
  KEY `patient_id` (`patient_id`),
  CONSTRAINT `doctor_ibfk_1` FOREIGN KEY (`patient_id`) REFERENCES `patient` (`patient_id`)
)
```

```sql
-- Create appointment  table to store appointment information----
CREATE TABLE `appointment` (
  `appointment_id` int NOT NULL AUTO_INCREMENT,
  `patient_id` int DEFAULT NULL,
  `doctor_id` int DEFAULT NULL,
  `date_of_appointment` datetime DEFAULT NULL,
  `notes` text,
  `status` varchar(20) DEFAULT 'active',
  PRIMARY KEY (`appointment_id`),
  KEY `patient_id` (`patient_id`),
  KEY `doctor_id` (`doctor_id`),
  CONSTRAINT `appointment_ibfk_1` FOREIGN KEY (`patient_id`) REFERENCES `patient` (`patient_id`),
  CONSTRAINT `appointment_ibfk_2` FOREIGN KEY (`doctor_id`) REFERENCES `doctor` (`doctor_id`)
)
-- Create vital table to store vital information for patients----
CREATE TABLE `vital` (
  `vital_id` int NOT NULL AUTO_INCREMENT,
  `patient_id` int DEFAULT NULL,
  `hypertension` varchar(100) DEFAULT NULL,
  `heart_disease` varchar(100) DEFAULT NULL,
  `avg_glucose_level` double DEFAULT NULL,
  `bmi` double DEFAULT NULL,
  `smoking_status` varchar(200) DEFAULT NULL,
  `stroke` varchar(200) DEFAULT NULL,
  PRIMARY KEY (`vital_id`),
  KEY `patient_id` (`patient_id`),
  CONSTRAINT `vital_ibfk_1` FOREIGN KEY (`patient_id`) REFERENCES `patient` (`patient_id`)

-- Create support_group table to store support group for which patients belongs to -----
▷ Run | New Tab | Copy
CREATE TABLE `support_group` (
  `support_group_id` int NOT NULL AUTO_INCREMENT,
  `patient_id` int DEFAULT NULL,
  `support_group_name` varchar(200) DEFAULT NULL,
  `group_type` varchar(200) DEFAULT NULL,
  `location_lat` double DEFAULT NULL,
  `location_long` double DEFAULT NULL,
  `location_name` varchar(100) DEFAULT NULL,
  `email` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci DEFAULT NULL,
  PRIMARY KEY (`support_group_id`),
  KEY `patient_id` (`patient_id`),
  CONSTRAINT `support_group_ibfk_1` FOREIGN KEY (`patient_id`) REFERENCES `patient` (`patient_id`)
```