DS-5110-My Doc

Abhinav Nippani
Student, MS in DS, NEU
nippani.a@northeastern.edu

Ayush Patel
Student, MS in DS, NEU
patel.ayushj@northeastern.edu

Shivansh Verma
Student, MS in DS, NEU
verma.shi@northeastern.edu

Abstract

My Doc provides the benefits of streamlined operations, enhanced administration, control, superior patient care, strict cost control, and improved profitability. MyDoc is powerful, flexible, and easy to use. It is designed and developed to deliver real conceivable benefits to hospitals. The aim was to develop an application that is tied to several hospitals and clinics where a user can book an appointment seamlessly and streamline the process from booking an appointment to getting the required diagnosis and viewing their bills. This an application that lets the user choose their desired hospital and appointment date and time to book an appointment. The application also has capabilities for admin and doctors to view their relevant data like appointments and schedules. This application has capabilities for data visualizations for various aspects of data, for instance, hospitals near the user's location and for admin dashboard for any administrator to view the workforce of the hospital graphically. The project 'My Doc' is built on the use of SQL with Python frameworks.

Keywords- Hospital Management System, Booking, Bill, MySQL, Python

Introduction

Booking an appointment in a hospital has always been a tedious task as often users have to navigate through byzantine directories and make several calls to find the best doctor for their ailments. In this day and age, people often shy away from calling, therefore an online-based solution will be easily accessible to everyone. The application allows users to view all the available doctors in the hospitals and allows them to book an appointment with a particular doctor.

The Doctors will have access to view the list of appointments they have along with the previous appointments for a particular user allowing them to analyze the history of a particular patient and perform better diagnosis. After completion of an appointment, the doctor diagnoses the patient by specifying all the prescribed medicines and their doses alongside any required lab tests. All of the charges are then calculated, and the total bill is generated and shown to the user. If the amount of the bill is above $500 then a subsequent 10% discount is provided with the bill.

The application works by leveraging PL/SQL for procedures, functions, and views and advanced SQL queries computing the final bills. The application database also does integrity checks on the data being fed into the database to avoid any discrepancies, incorrect data, or malicious data being entered to avoid any kind of SQL injection attacks.

The application is built on top of:

- SQL: MySQL is an open-source relational database management system. MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access, and facilitates testing database integrity and the creation of backups.
- Python: Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems.

- Mockaroo: It allows you to quickly and easily download large amounts of randomly generated test data based on your own specs which you can then load directly into your test environment using SQL or CSV formats. No programming is required.

Database Design

No.	Table	Relational Schema
1.	users	(user_id,firstname, lastname, gender, date_of_birth, address, city, state, pincode, phone_number, email_id, ssn, password, user_type)
2.	doctors	(doctor_id, license_number, start_of_practice, salary, degree, university, appointment_fees)
3.	doc_schedule	(monday_work, tuesday_work, wednesday_work, thursday_work, friday_work, start_shift, end_shift)
4.	area_of_specialization	(area_of_specialization_id, name)
5.	hospitals	(hospital_id, name, address, city, state, pincode, phone_number, email_id, date_of_establishment)
6.	departments	(department_id, name, department_spec)
7.	appointments	(appointment_id, appointment_date, appointment_time)
8.	card_details	(card_details_id, card_holder_name, card_number, card_type, expiry_date)
9.	prescriptions	(prescription_id, prescription_text)

10.	medicines	(medicine_id, medicine_name, medicine_manufacturer, price_per_unit)
11.	lab_test	(lab_test_id, lab_test_text)
12.	lab_test_description	(lab_text_description_id, description_text, price_per_lab)
13.	payments	(payment_id, payment_date, fees)
14.	contact_us	(contact_us_id, firstname, lastname, email_id, phone_number, questions)
15.	appointment_details	(user_id, doctor_id, hospital_id, appointment_id)
16.	appt_pay	(appointment_id, payment_id)
17.	diagnosis	(appointment_id, prescription_id, lab_test_id)
18.	lab	(lab_test_id, lab_test_description_id)
19.	pres_med	(prescription_id, medicine_id, daily_doses, number_of_days)
20.	doc_dept	(department_id, doctor_id)
21.	doc_hosp	(doctor_id, hospital_id)
22.	specializes	(area_of_specialization_id, doctor_id)
23.	hosp_contact_us	(contact_us_id, hospital_id)
24.	pay_card	(card_details_id, payment_id, appointment_id,cvv)
25.	user_card	(user_id, card_details_id)

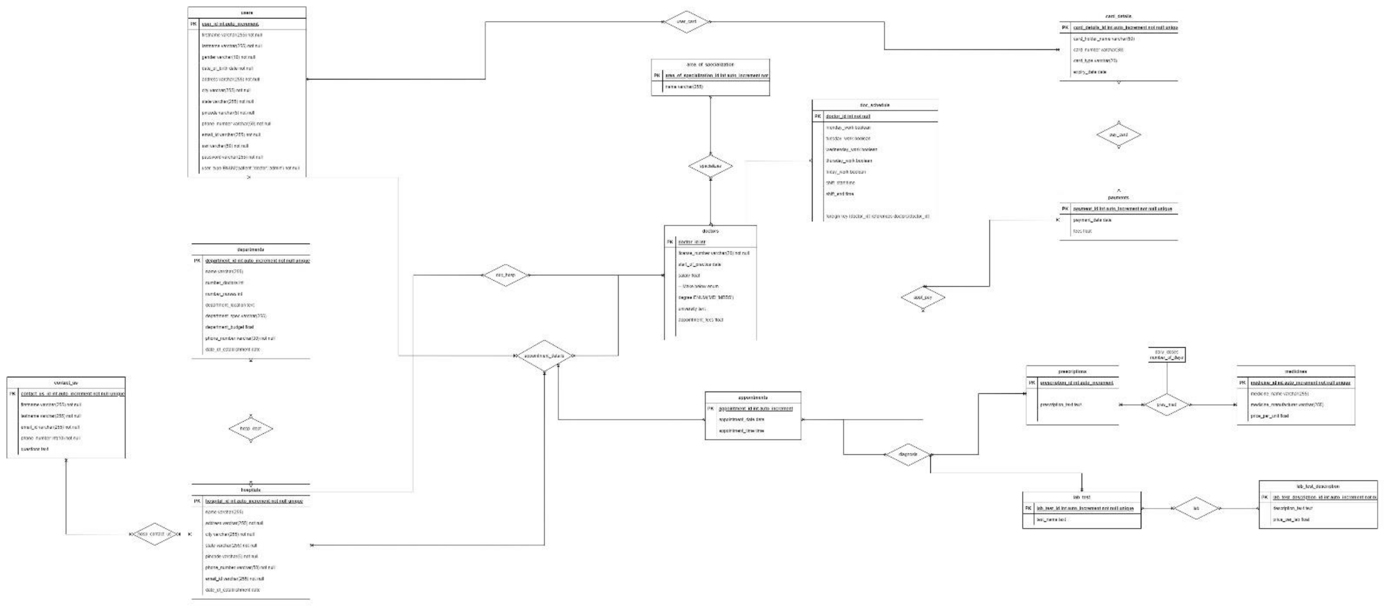


Figure 1 – Entity Relationship Diagram for My Doc

The users table contains the user information and their sensitive information like passwords. This is the master user table which works with a lot of tables like doctors, appointments, and appointment schedules to book an appointment. The doctors table contains the license number and the doctor id which is a foreign key in many other tables and builds the core logic of the application. The doctor's id is referenced in the appointments table which also helps us to get the appointments of the doctor. The appointments table contains all the information about a particular appointment like the appointment id, doctor id, hospital id, and the date and time of the application. This is used to generate a diagnosis and further used to create the total bills.

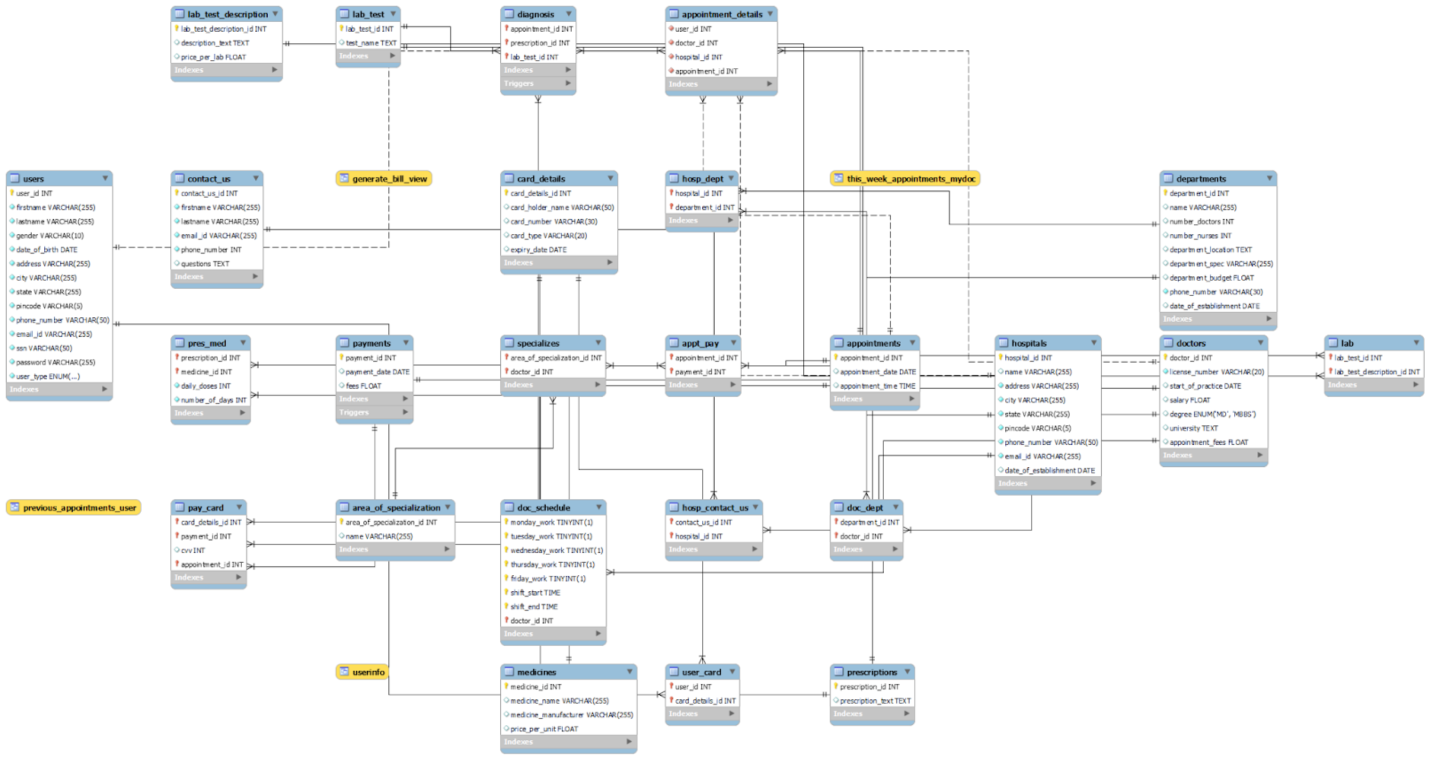


Figure 2 – Relational model

Triggers:

After the doctor performs the required diagnosis on the patient, the payments table is updated. The diagnosis table contains the prescribed medicines and the lab tests to be performed for the user. This trigger also updates the total bill of the user with the updated costs of the medicines and the lab tests which are added to the final bill.

The total bill also has a trigger that updates the user's bill by giving a discount of 10% when the bill exceeds $500. The trigger then generates a discount of 10% on the total bill including the visit charges, medicines, and lab test fees.

Stored Procedures:

- onCall

This stored procedure checks if the doctor is working on a given date and time. This helps the user to book an appointment. This stored procedure starts with finding on which day of the week the given date is on. It then compares it with the schedule of the doctor and checks if the doctor is working on that day. After confirming that the doctor is working on that day the application then analyses whether the doctor is working at the given time or not. If the doctor is working on that particular date and time of the given day, the application notifies the user that the doctor is working.

- numDoc

This stored procedure aids the administrator to find out the number of doctors working on a given date. This stored procedure looks into the doctor’s schedule in the schedule table after finding out the day of the week and uses this to find the count of doctors that would be available, it then shows the list to the users to book an appointment.

- getThisWeek

This stored procedure acts as a helper function as this is involved in a lot of admin and bill generator stored procedures. This stored procedure helps us to find the start and end date of the week corresponding to any date; this stored procedure tells the start date of the week (Sunday) and the corresponding end date of the week (Monday).

- bookAppointment

The bookAppointment is the most complicated stored procedure as it utilizes complex functionalities of PL/SQL and checks all the possible conditions before confirming an appointment. This procedure also ensures the integrity of our database is maintained before and after booking an appointment. The first check is made to ensure that the selected doctor is working on a particular date and time and then the application proceeds further if and only if the doctor is working on that date and time. If the doctor is working, the application continues to check if the selected doctor has a conflicting appointment on the same date and time, if there are no conflicts the code then automatically updates the appointment and appointment_schedule table with the appointment’s auto-incremented ID. This ensures data integrity across all the tables.

Functions:

- medicine_amt

Calculates the total number of a specific medicine consumed, the number of medicines multiplied by the daily dosage. This then gives the sum for a specific medicine that would be consumed by a user as prescribed by the doctor

- prescription_amt

The prescription is made based on the number of medicines consumed multiplied by the cost of the specific medicine to provide us with the total cost of all the medications given to a doctor.

- lab_test_amt

The lab tests also incur some charges to find the sum total of the lab test charges the application finds all the lab tests required to be completed by the user and provides the sum of all the lab test costs.

- visit_charges

This function calculates the cost of all the visit of a particular appointment and adds it. This function takes into consideration the medicines and their prescription quantities along with the lab test charges to compute a final bill of the appointment. Then final bill is generated based on the above functions.

- numAppointmentsToday

This function returns the number of appointments in a hospital at a given date.

- billTotal

This calculator returns the total bill amount for a particular user id after adding up all their bill totals.

Views:

- previous_appointment_user

Shows the previous appointments of the patient to the doctor on which the doctor performs the diagnosis. This helps the doctor look at the patient's medical history.

- this_week_appointments_mydoc

The doctor can see their current and updated schedule with the help of this view allowing the doctor to analyse their schedule and workweek.

- generate_bill_view

This allows the user to see their final bill with the sum total of all the charges that they have been charged.

- userInfo

This view performs abstraction by hiding sensitive information like SSN and password form the main tables and displays just the information about the user like full name, address and their email providing basic information about the user.

- card_user

While paying the bill the user can view their saved cards and hides the card number and other sensitive details.

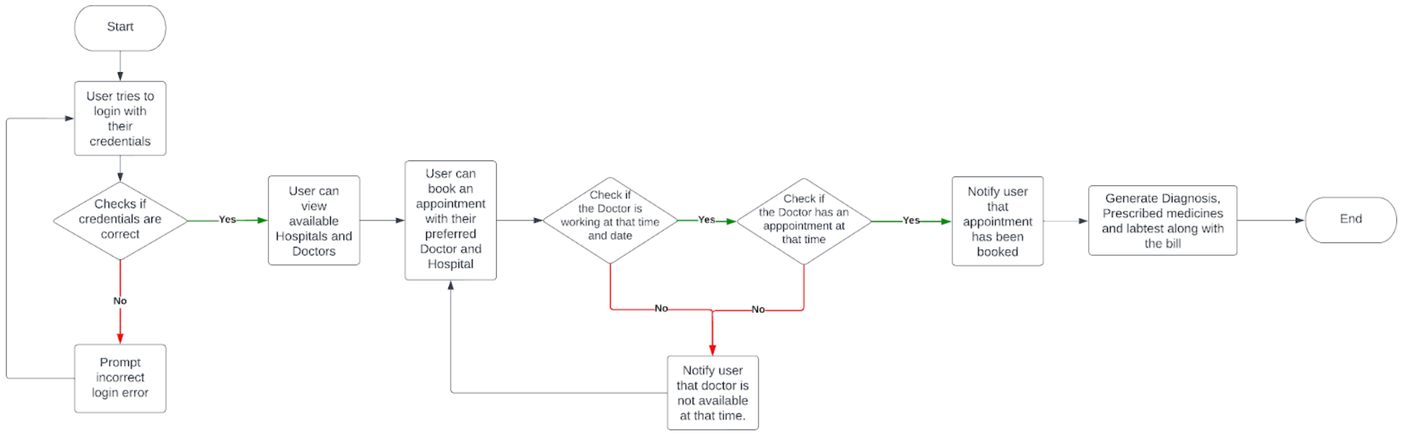
Data Normalization

Due to HIPAA regulations getting access to any category of medical data is highly prohibited, this project relied on mock data generated by Mockaroo which provides simulated mock data with real-world-like authenticity to the data. Since the dataset is mock, we have made sure to filter out any data discrepancies and null values while generating the data but if this application had to be launched into production we still have integrity constraints for all columns in the tables making sure that there aren't any incorrect value is fed into the database. Alongside integrity constraints, the application will have checks and constraints in the user interface to avoid any possibilities of SQL injection attacks. By using Mockaroo we make sure that the data feels authentic and closely resembles real-world data. The mock data can be generated for more than 10,000 rows; enough for us to do any kind of stress testing for our application.

Application description

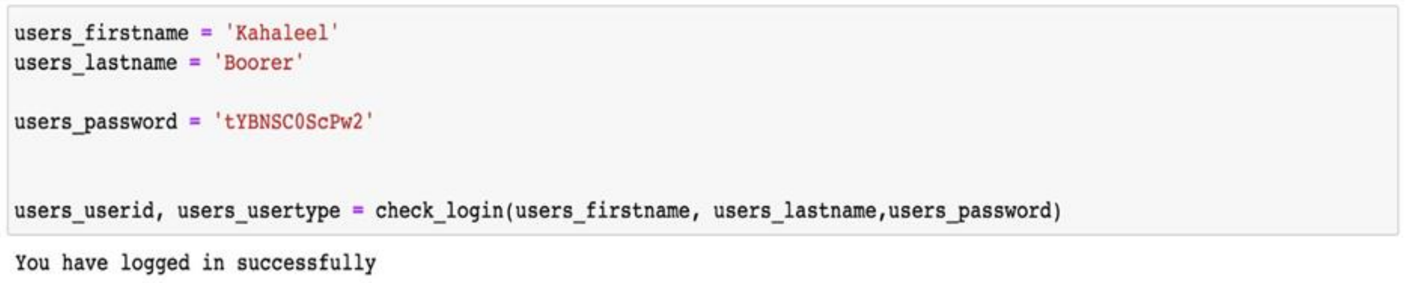
Features:

- User registration and management which includes users, doctors, and admins.
- Booking an appointment.
- Handling medicines and prescriptions.
- Lab reports handling.
- Manage patient bills.



Start
User tries to login with their credentials
Checks if credentials are correct
Yes
No
Prompt incorrect login error
User can view available Hospitals and Doctors
User can book an appointment with their preferred Doctor and Hospital
Check if the Doctor is working at that time and date
Yes
No
Check if the Doctor has an apppointment at that time
Yes
No
Notify user that doctor is not available at that time.
Notify user that appointment has been booked
Generate Diagnosis, Prescribed medicines and labtest along with the bill
End

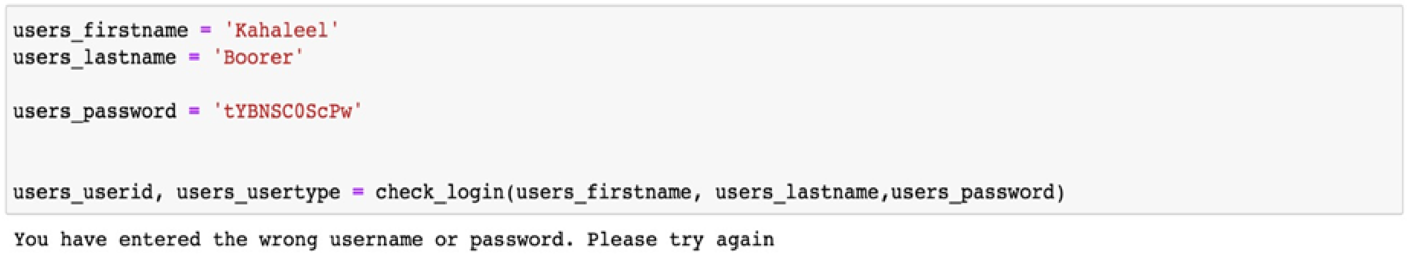
Figure 3 – Flowchart of the application



```
users_firstname = 'Kahaleel'
users_lastname = 'Boorer'

users_password = 'tYBNSC0ScPw2'


users_userid, users_usertype = check_login(users_firstname, users_lastname,users_password)
```

```
You have logged in successfully
```

Figure 4 – Successful login of the user

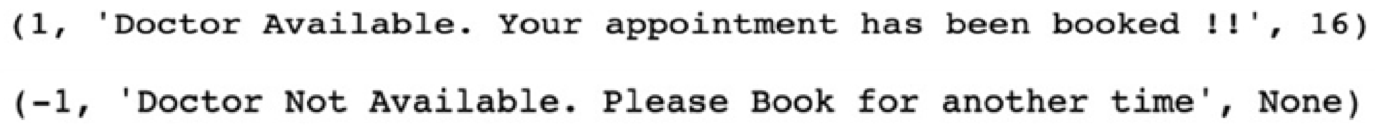


```
users_firstname = 'Kahaleel'
users_lastname = 'Boorer'

users_password = 'tYBNSC0ScPw'


users_userid, users_usertype = check_login(users_firstname, users_lastname,users_password)
```

```
You have entered the wrong username or password. Please try again
```

Figure 5 - Unsuccessful login attempt

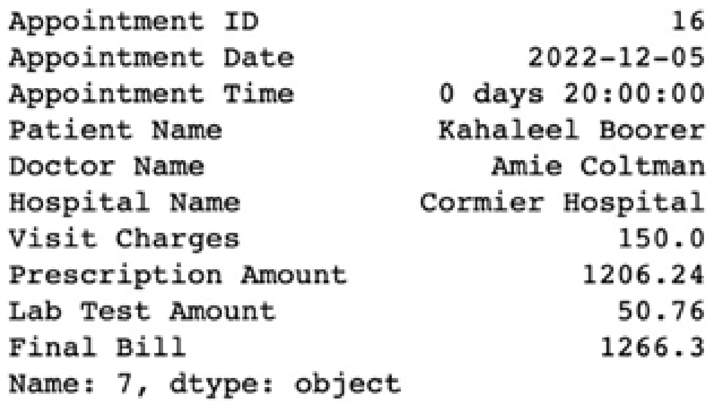
For any user logging in, the application checks whether the credentials they are using are correct. To begin with, the application checks their credentials against the encrypted passwords they have in the database this allows us to safely keep the user passwords to comply with the governance and data protection laws. After the user has successfully logged in, the users can view all the available doctors in all the available hospitals for their desired date and time for their preference. The users can also see the doctors and their respective departments and specializations to book their desired doctor and appointment times.



```
(1, 'Doctor Available. Your appointment has been booked !!', 16)

(-1, 'Doctor Not Available. Please Book for another time', None)
```

Figure 6 - Prompts for successful and failure for booking of appointment

To book the appointment complex workflows have been implemented that make sure that ACID properties are not violated, and the booking flow is consistent. After selecting the doctor, time, and the hospital, a check is made that if the doctor is working at the requested time or not. If the doctor is working, then the application proceeds with the next transaction block; if the doctor is not working then the user is notified that the doctor is not working and goes back to selecting the doctor and user. If the doctor is working at the given time then the next check is made, whether another user has an appointment as this time frame or not. This logic requires the code to check appointments and appointment-schedule tables to ensure that the data being input is consistent with the appointment IDs. If the doctor does not have a conflicting slot and is available, the appointment is booked and is reflected the appointment and appointment-schedule tables which display the appropriate messages to the user. The doctor's schedule has also been updated so that the doctor can view their schedule. The admin has the authority to make changes to the user's appointment and doctor's schedule. The above figure returns the prompt texts as well as the appointment ID.



```
Appointment ID                        16
Appointment Date              2022-12-05
Appointment Time         0 days 20:00:00
Patient Name             Kahaleel Boorer
Doctor Name                  Amie Coltman
Hospital Name           Cormier Hospital
Visit Charges                      150.0
Prescription Amount              1206.24
Lab Test Amount                    50.76
Final Bill                        1266.3
Name: 7, dtype: object
```

Figure 7 - A sample bill is generated

After their appointment gets booked and processed the doctor then generates the diagnosis for the patient. This diagnosis would contain the prescribed medicines, lab tests, and bills. To generate the final bill the visit charges, the cost of each medicine (i.e., the type of medicine prescribed multiplied by the number of days and daily dosage), and lab test costs are added. Finally, the bill is displayed to the user with all the total charges that were charged to the user for their visit. Subsequently, if the bill amount is greater than $500 the application then provides a 10% discount to the user.

Exploratory Data Analysis

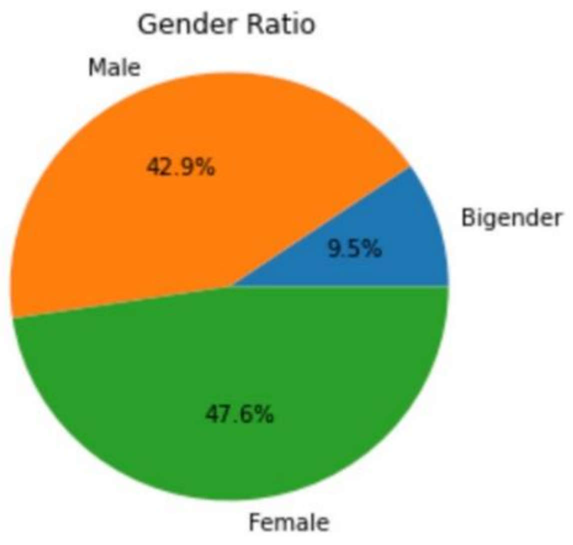


Figure 8 – Pie chart for gender classification

The gender of users in the applications allows the admin to find the current demographic of the users.

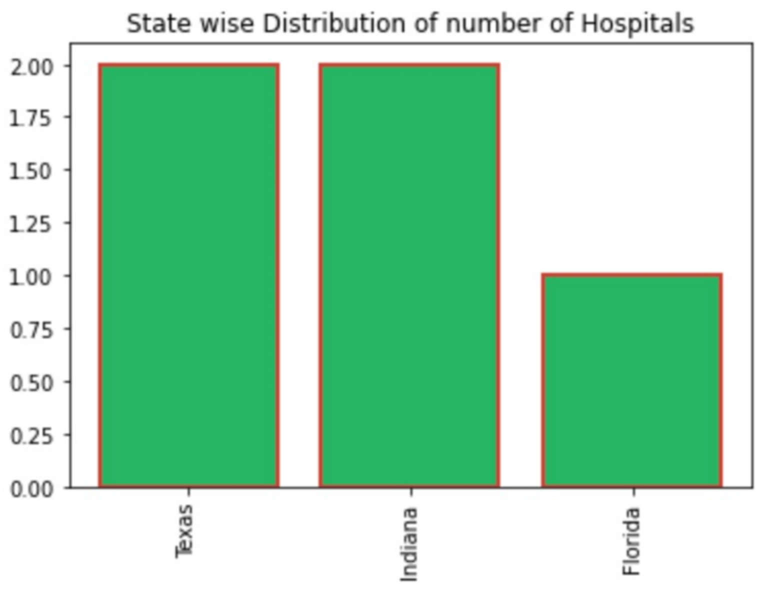


Figure 9 – Bar chart for state-wise distribution of hospitals

This graph shows the number of hospitals in various states allowing the users to see how many hospitals are there in their locality.

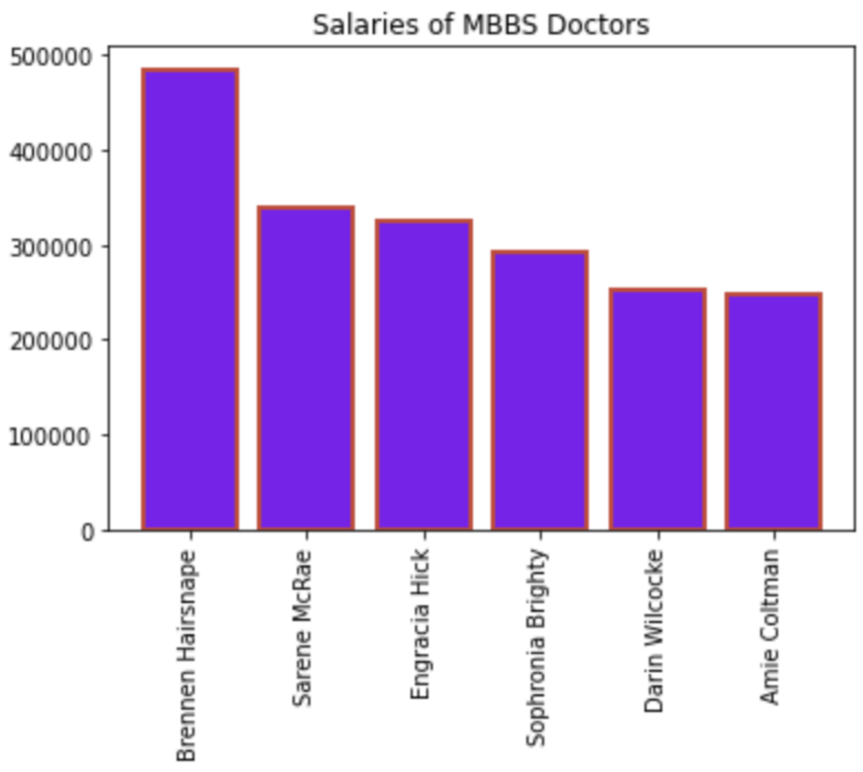


Figure 10 – Bar chart which shows the highest-paid doctors with their names

This graph shows the admin the top salaries of doctors who are qualified as MBBS

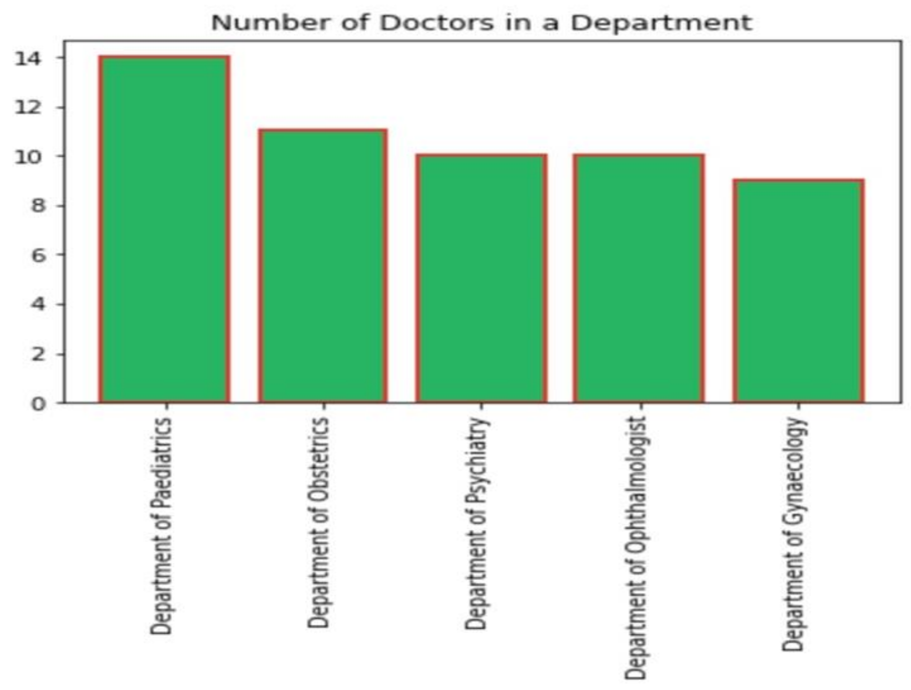


Figure 11 – Bar chart for ranking the highest number of doctors in a department

The above graph shows the number of doctors in a given department allowing us to visualize the headcounts and strengths of various departments.

Future Scope and Conclusion

My Doc is an application that can register a patient. A patient can log in, can book an appointment as per their required date and time. The doctors' schedule is then updated, and they can view the patients' previous appointments. The doctor can then diagnose which consists of prescribing medicines or suggesting lab tests. After which the final bill is generated which can be seen by the patient. The application can be further modified by the introduction of the insurance claim. If the patient has medical insurance, they can utilize it which covers a certain part of their bill. Reviews of each doctor can be added. Before booking an appointment, the user can see the reviews of the doctors posted by previous users. Further access can be given to doctors where they can make changes to their appointments. In the future, the application can be further updated by adding payment gateways and completely deploying the code on cloud services after developing an easy user interface. There is no machine learning side to our application, we can embed LSTM and Fourier Transformation to predict the peak hours in hospitals and the medicinal supply is efficient after the COVID pandemic which portrayed the shortcoming of hospital management systems.

My advice to future DS 5110 students is that always get a team, you will understand how to coordinate with other people, and always be hands-on during lectures. This course has a lot of new software applications that we need to learn, it is always good to practice.

References

1. https://www.zocdoc.com/
2. https://www.mfine.co/
3. https://www.mockaroo.com/
4. https://doi.org/10.1016/j.jhin.2009.03.031
5. https://www.practo.com/
6. https://www.netmeds.com/
7. https://gloriumtech.com/hospital-management-software-development-key-features-and-benefits/