UNIVERSITY HEALTH AND COUNSELLING SERVICES

CLINIC MANAGEMENT SYSTEM

Course Name: CS 5200 Database Management System
Group Name: PatelTMohan Raje UrsPArabagatta BasavarajappaP
Group Members: Tejal Patel (NUID: 002763037)
Prathiksha Mohan Raje Urs (NUID: 002711883)
Pranove Arabagatta Basavarajappa (NUID: 002762121)

INDEX

SECTION 1: README

A) Steps to run the project:

1. Unzip the project, and open the project in Visual Studio Code (You can download it from here: https://code.visualstudio.com/download)
2. Download the latest version of python from here: https://www.python.org/downloads/)
3. Open the terminal in Visual Studio Code and run the command ***"pip3 install -r requirements.txt".*** If this command doesn't work, try ***"pip install -r requirements.txt".*** This will install all the required libraries and dependencies.
4. Using MySQL Workbench, create a database named *"dbmsprod"* and import the dump file ".sql" into the database.
5. In the file *app.py*, change the value of the database connection credentials to your own database credentials. Set value of db to "dbmsprod" and set the value of user and password to your own database credentials.
6. Run the command ***"python3 app.py"*** or ***"python app.py"*** to run the project.
7. Open the browser and go to the URL ***"http://localhost:5000"*** to view the project.

B) Libraries required to run the project:

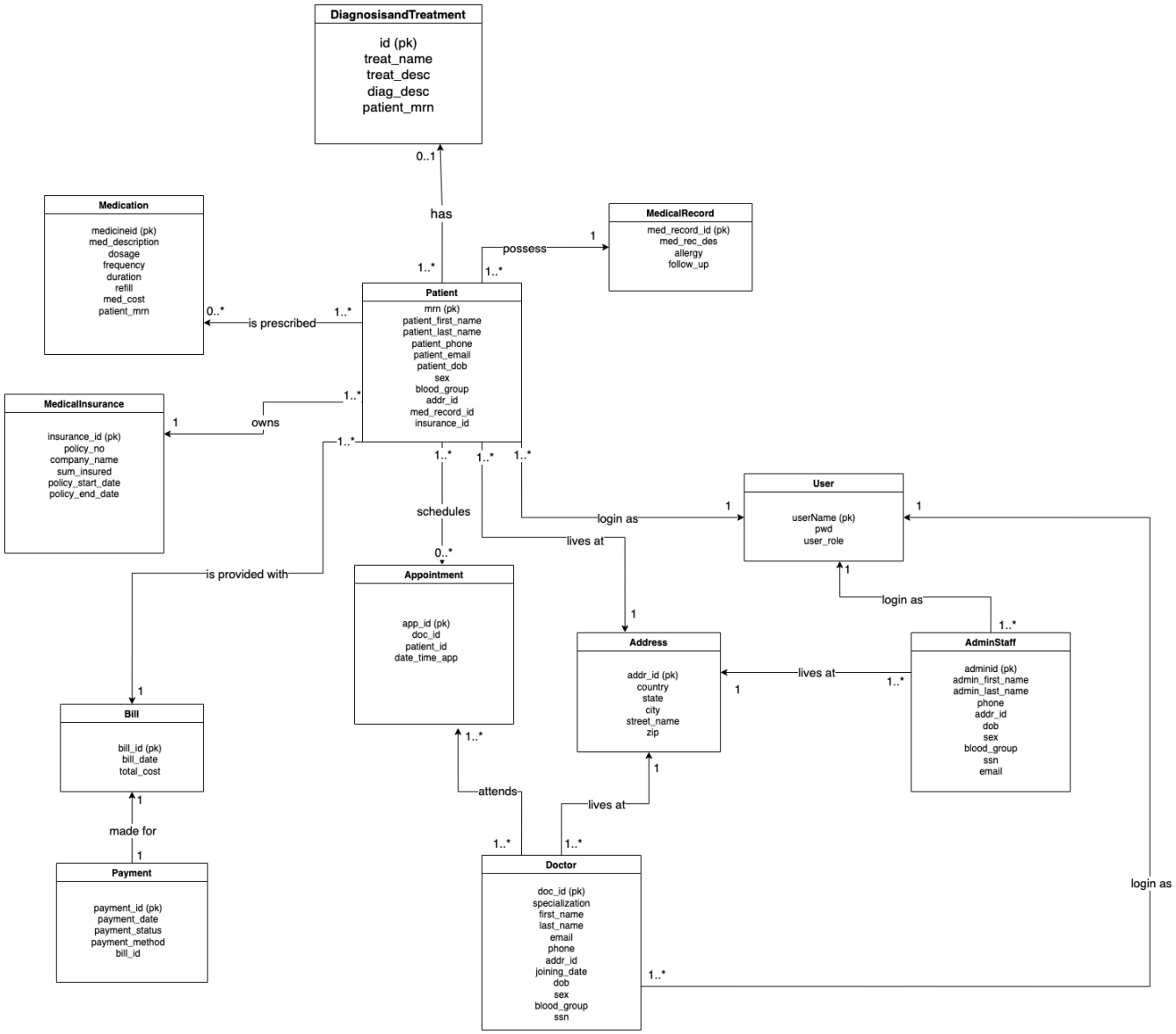
1. blinker (version 1.4)
2. click (version 8.1.3)
3. colorama (version 0.4.6)
4. Flask (version 2.2.2)
5. Flask-Login (version 0.6.2)
6. Flask-Mail (version 0.9.1)
7. greenlet (version 2.0.1)
8. itsdangerous (version 2.1.2)
9. Jinja2 (version 3.1.2)
10. MarkupSafe (version 2.1.1)
11. passlib (version 1.7.4)
12. PyMySQL (version 1.0.2)
13. Werkzeug (version 2.2.2)
14. WTForms (version 3.0.1)

SECTION 2: TECHNICAL SPECIFICATION

Tech stack used:

1. The project is built using Flask, a micro web framework written in Python.
2. The database used is MySQL.
3. Programming language used: Python
4. Libraries used: Flask framework, pymysql to connect to MySQL, passlib to hash passwords, Flask-Login to manage user sessions, Flask-Mail to send emails, Flask-WTForms to create dynamic web forms, Jinja2 to render templates, blinker to send signals.
5. MySQL Workbench to create the database and to visualize the database.
6. HTML and CSS to create the front-end.
7. FullCalendar.io to implement the appointment calendar.
8. Visual Studio Code IDE to write and execute the project.

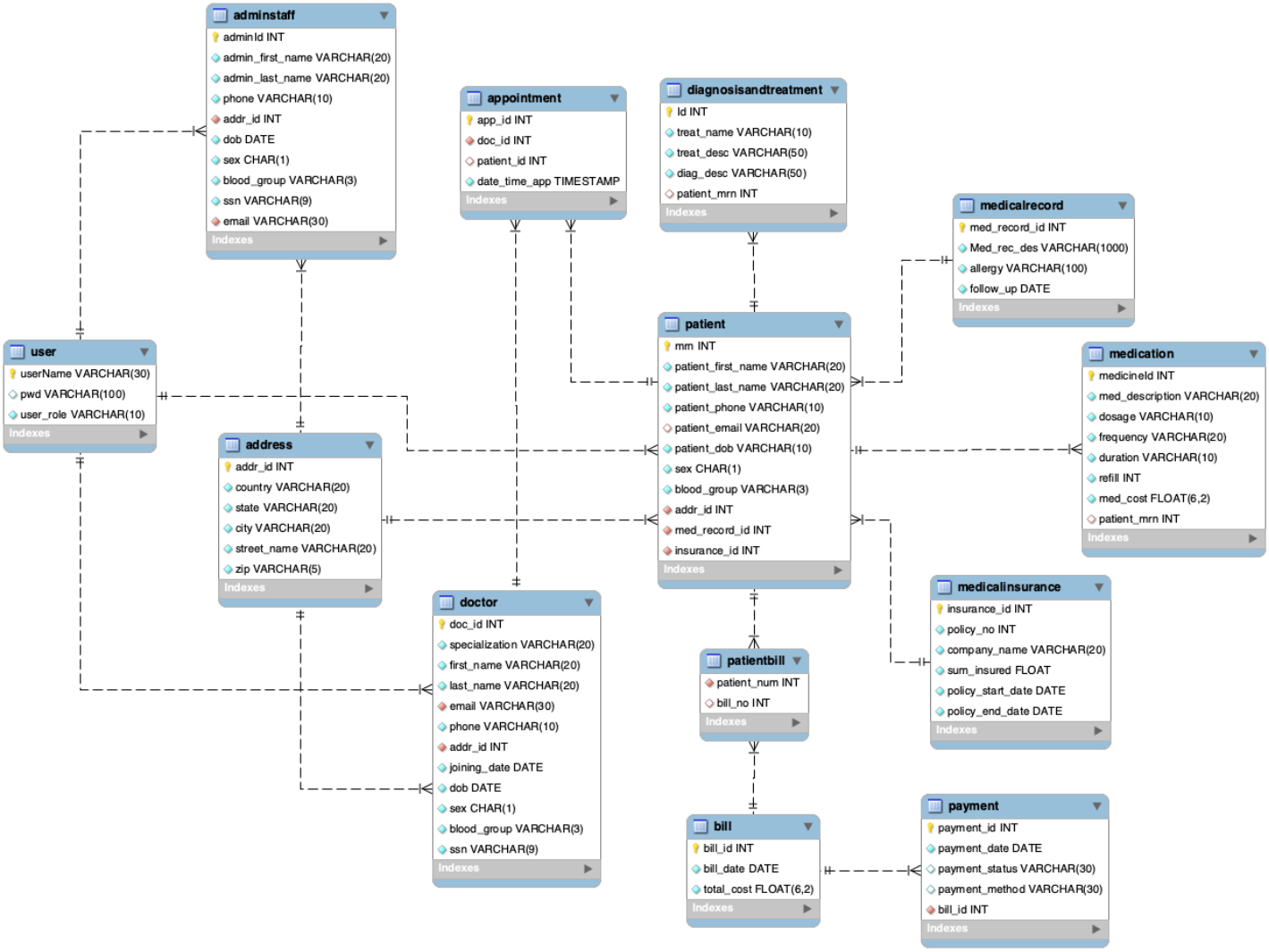
SECTION 3: UML DIAGRAM



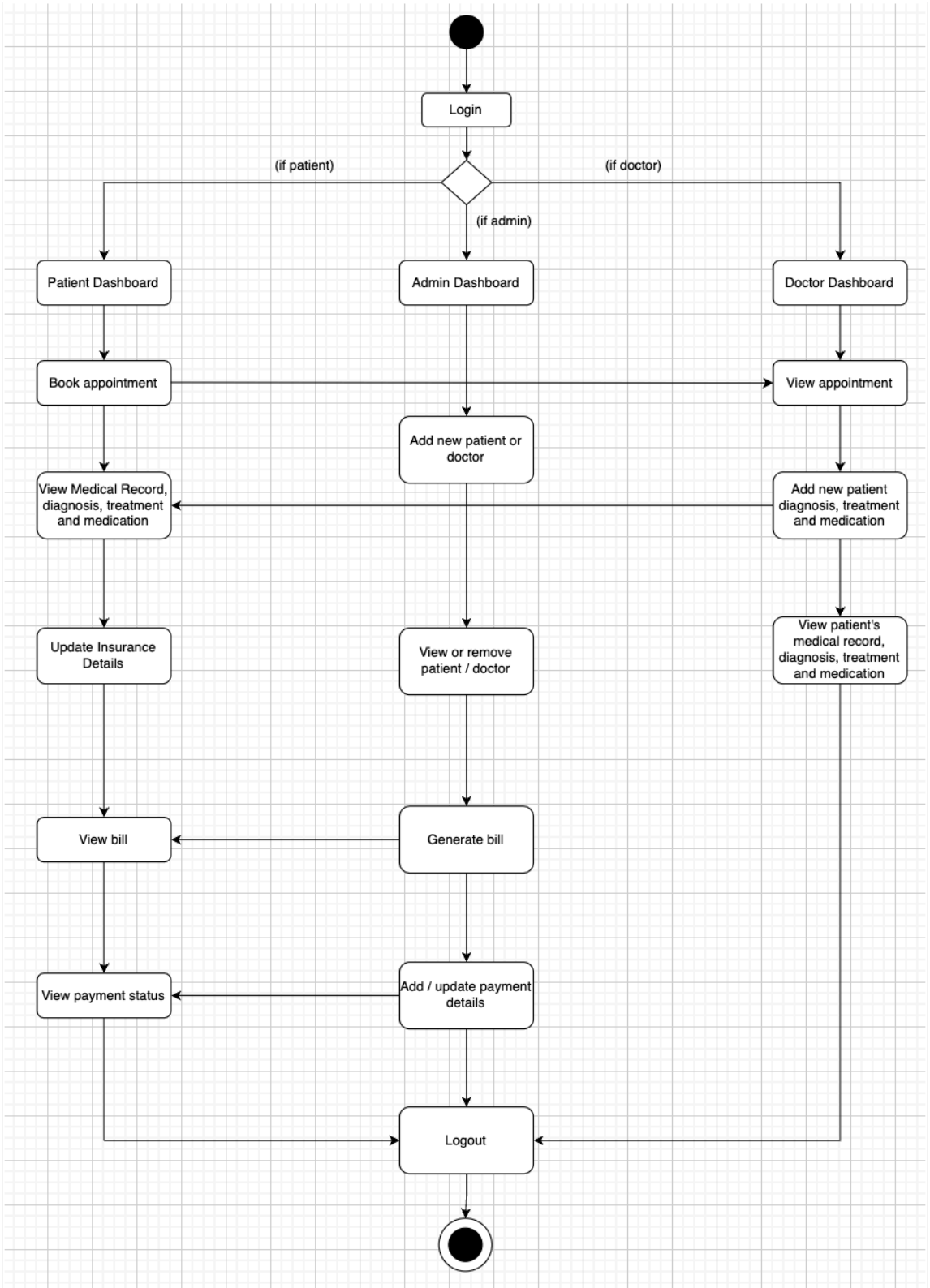
UML Diagram Description:

- Clinic Management System has three users namely Admin, Doctor, and Patient.
- All the users are required to login using their username (email) and password.
- AdminStaff table has attribute *"adminid"* as primary key
- AdminStaff table is connected to:
    - user - 1 to many admins can have 1 user name
    - Address - 1 to many admins can have 1 address
- Patient table has attribute *"mrn"* as primary key.
- Patient table is connected to various tables like:
    - Address – 1 to many patients can have 1 address
    - User - 1 to many patients can have 1 user name
    - Appointment - 1 to many patients can have 1 to many appointments
    - Medication - 1 to many patients can have 0 to many medications
    - Bill - 1 to many patients can have 1 bill (per visit to doctor)
    - DiagonsisandTreatment - 1 to many patients can have 1 to many DiagonsisandTreatment
    - MedicalRecord - 1 to many patients can have 1 medical record
    - MedicalInsurance - 1 to many patients can have 1 medical insurance
- Doctor table is connected to:
    - User – 1 to many doctors can have 1 user name
    - Address – 1 to many doctors can have 1 address
    - Appointment – 1 to many doctors can have 1 to many appointments
- Doctor table has attribute doc_id as primary key
- User table has attribute *"username"* as primary key
- User table is connected to Patient, Doctor, and AdminStaff
- Appointment table has attribute *"app_id"* as primary key
- Appointment table is connected to Patient, Doctor and AdminStaff tables
- Address table has attribute *"addr_id"* as primary key
- Address table is connected to: Patient, Doctor, and AdminStaff tables
- Bill table has attribute *"bill_id"* as primary key
- Bill table is connected to patient and payment tables
- Payment table has attribute *"payment_id"* as primary key
- There is 1..1 relation between bill and payment table
- MedicalRecord has attribute *"med_record_id"* as primary key
- DiagnosisandTreatment has attribute *"id"* as primary key
- Medication has attribute *"medicineid"* as primary key
- MedicalInsurance has attribute *"insurance_id"* as primary key

SECTION 4: EER DIAGRAM (Crow's Foot Notation)



SECTION 5: USER FLOW (ACTIVITY DIAGRAM)



SECTION 6: LESSONS LEARNED

- We learned how to use Flask to create a web application.
- We learned how to use MySQL to store data.
- We learned how to use Flask-Login to manage user sessions.
- We learned to use ngrok to expose a local web server to the internet, so that all the team members can access the web application to create data and test the application.
- ngrok is a cross-platform application that enables developers to expose a local development server to the Internet with minimal effort.
- It is useful for testing webhooks and also for creating a temporary public URL for a web application that is running on a local machine.

Technical Expertise Gained:

- We gained experience on how to use Flask to build scalable web applications, manage user sessions.
- We gained thorough knowledge on how to use MySQL to store and interact with data.
- We gained hands-on experience on how to write modular and production-ready code.
- Learned to develop dynamic web forms using Flask-WTForms, receive and validate user input, and store it in the database.
- Learned to write complex CREATE, SELECT, UPDATE and DELETE statements, and procedures in mySQL, and how to execute them using python.

Insights:

- The project was a good learning experience for us. We learned how to use Flask to create a web application and how to use MySQL to store and interact with data.
- The data domain for the project was a good choice, but the scope of the project was too large. We had to reduce the scope of the project to make it feasible.

Realised:

- We realised that the initial plan of creating a web application to manage appointments for a hospital was not feasible. We had to change our plan to create a web application to manage appointments for a doctor's clinic.
- We realised that the scope of the project was too large. We had to reduce the scope of the project to make it feasible.
- We realised that we had to change the database design to make it feasible and to make it simpler to submit in time for the deadline.
- We realised that using CLI to make the project was not intuitive and we had to change the project to a web application.

SECTION 7: FUTURE SCOPE

- Scheduling of appointments by synchronizing the calendar of the doctor and the patient.
- Implementing a feature to email the patient when a new patient is registered by the admin.
- Adding a feature to email the patient when a new appointment is scheduled.
- Setting up online payment gateway.
- Expanding to include rooms and hospital department data.