Group Name: QuinnNSaslowKQuachL

Group Members: Katie Saslow, Nicole Quinn, Linda Quach

HealthEase Patient Portal

Introduction

Our interest in developing a healthcare portal was influenced by the personal experiences of one of our team members, who has worked as a medical ICU nurse for the past three years. This background has cultivated a deep interest in the interplay between healthcare and technology, especially given the inefficiencies exposed by the pandemic in data utilization within the healthcare sector. These inefficiencies are felt not only among healthcare providers, but are even more profoundly experienced by patients, such as you and I.

The complexity of navigating the healthcare system is a challenge we have all personally encountered, whether in the quest to identify appropriate healthcare providers and services, or in managing medical paperwork and bills. Such challenges can deter individuals from effectively managing their health, impacting both physical and mental well-being.

Our project, therefore, aims to address these challenges by integrating essential healthcare functionalities onto a single platform. By doing so, we seek to simplify the patient's experience within the healthcare system. We believe that a unified solution will provide timely access to healthcare resources and enable patients to make more informed decisions regarding their health care. This project represents not just a technological endeavor but also a commitment to enhancing accessibility of healthcare for all individuals.

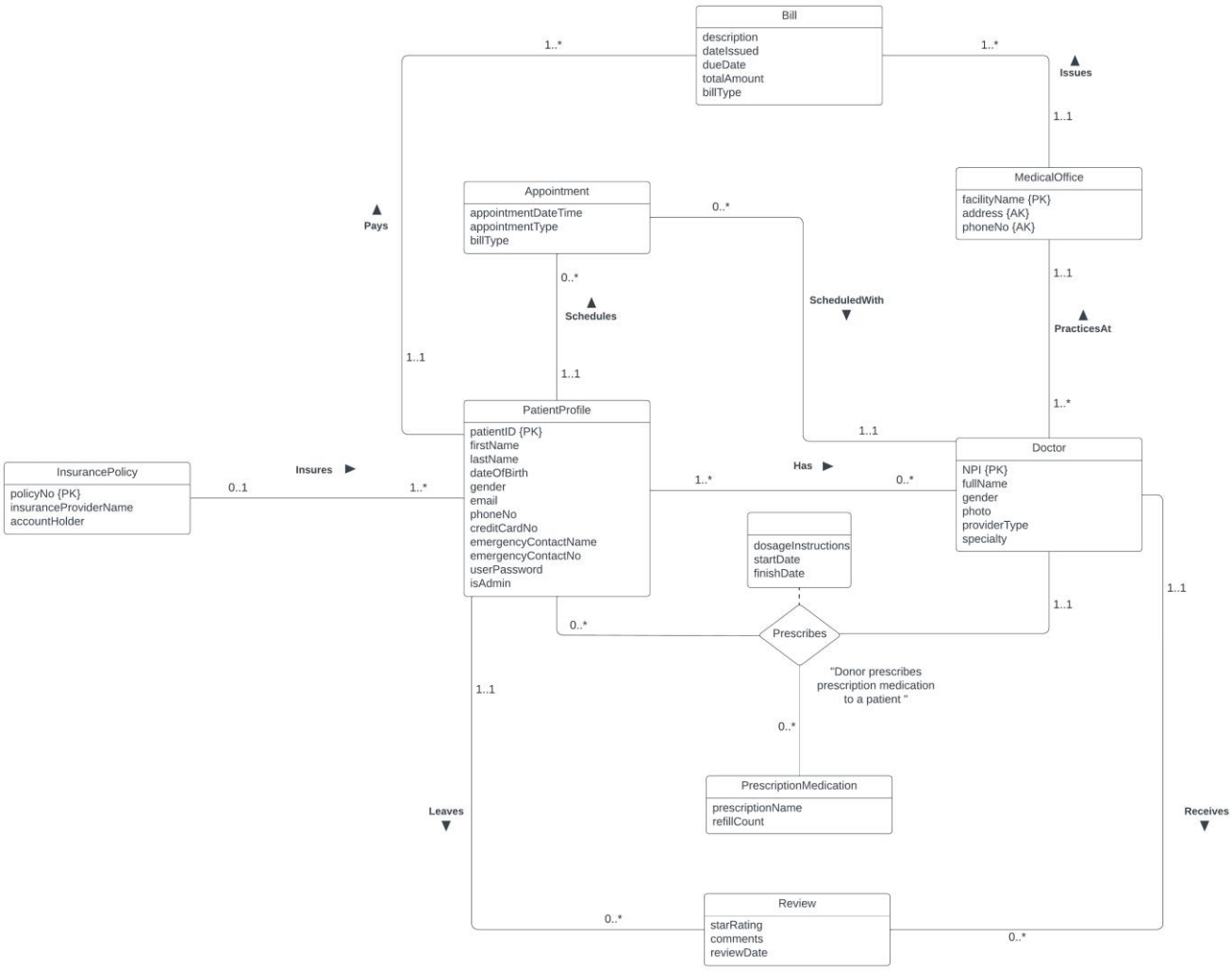
Top-Level Project Description

The objective of this project is to develop a healthcare management app, designed to provide patients with a unified platform for accessing and managing their health-related information and services. The platform encompasses a range of functionalities aimed at simplifying the management of and access to healthcare for patients, including allow users to:

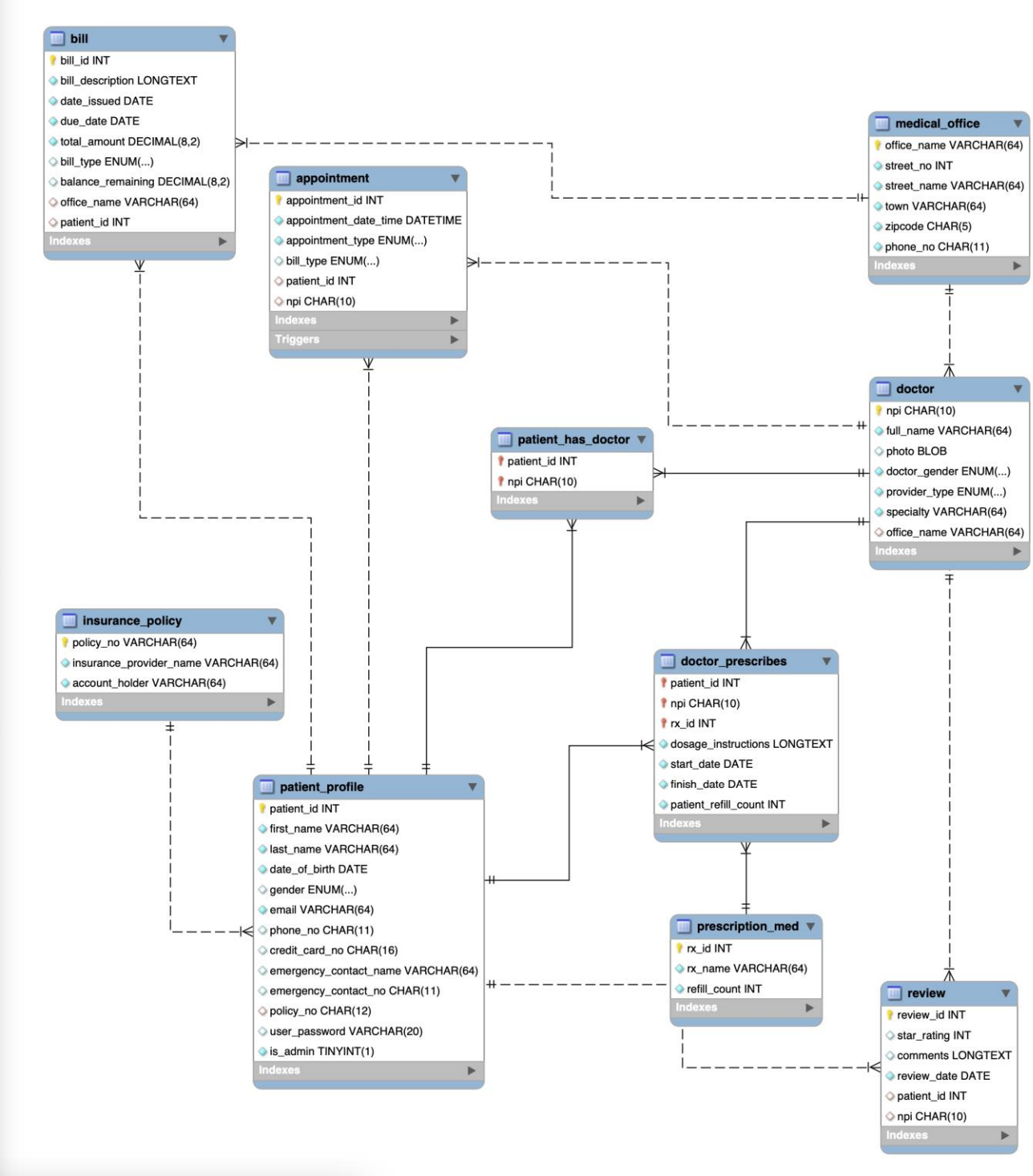
- Create an account
- Log-in to Account
- View Profile
- Update Profile
- Update Insurance
- Find Doctor
- View Doctor Reviews
- Write Doctor Reviews
- Delete Doctor Reviews
- View Appointments

- Schedule Appointment
- Reschedule Appointment
- Cancel Appointment
- View Prescriptions
- View Bills
- Pay Bills

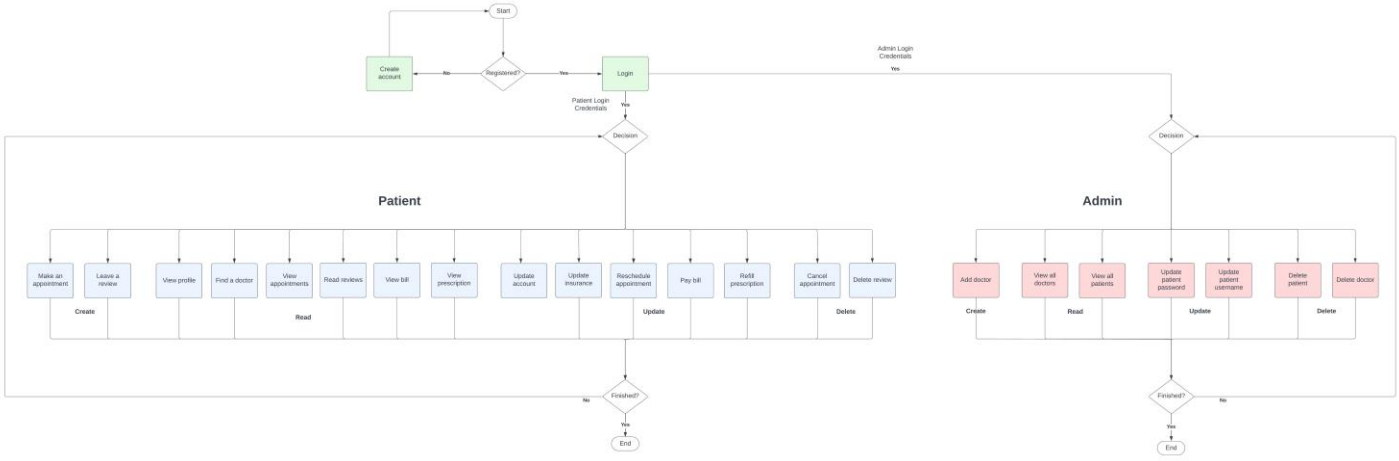
Conceptual Design - UML Diagram



Logical Design - Database Schema



Activity Diagram



User Flow

Detailed Steps:

- **Establish connection to SQL:** First the user is prompted to create a connection with the MySQL server (we used PyMySQL as explained more in the README). Once the connection is successfully created to the database, the user will be shown a homepage, where they will be prompted to either create an account, or log-in to an existing account. Please refer to the README document to see the log-in credentials for existing accounts to be able to experience the full functionality of the application.
- **Create Account:** If the user does not already have an account, they can choose to create one. To successfully create an account, the user needs to enter their first and last name, their birth date (must be at least 18 years old), their email-address (cannot already be in use by another user), and their password. The email and password will be the log-in credentials for the user to access their account. Upon creation of an account, the user is automatically logged into the portal, they will not need to actively log-in. Users are unable to create admin accounts.
- **Log-in:** If the user already has an account, they can log-in to the portal. Depending on the user, they will be directed to the patient menu or the system administrator menu.
    - Patient main menu offers the following functionality:

View Profile	Schedule Appointment
Update Profile	Reschedule Appointment
Update Insurance	Cancel Appointment
Find Doctor	View Prescriptions
View Doctor Reviews (w. data visualization)	Refill Prescriptions
Write Doctor Reviews	View Bills
Delete Doctor Reviews	Pay Bills
View Appointments	Quit

    - System administrator menu offers the following functionality:

View Patients	View Doctors
Update Patient Username	Add Doctor
Update Patient Password	Delete Doctor
Delete Patient	Quit

- **Using the DB Application:** Once successfully logged-in, the user (depending on the user's credentials and user role) can interact with the database in the ways highlighted in the menu. The user simply enters the number of the desired operation and will be prompted by the command-line and given more specific instructions for what to do.
- **Quitting the Application**: If the user wishes to exit the DB application, they can 'exit' out of the operation they selected, which will navigate them back to the main menu, where they can press 'Q' or 'q' and terminate the program.
- **Track Updates Made:** If the user conducts a create, update, or delete operation from the command line, the corresponding read operations ("View ___" on the main menu) can always be examined to make sure that the update was made successfully. All changes will be made in the database, making them visible in the view options on the menu immediately after the user makes the change in the CLI (e.g. if the user chooses "Schedule Appointment" and the CLI says the appointment was scheduled successfully, the user can choose "View Appointments" to ensure that the newly scheduled appointment shows up).

Lessons Learned

Technical Expertise Gained:

- **Python and MySQL**: We developed a stronger foundation in integrating Python together with MySQL and navigating the data validations needed to ensure our data correctly inputs into the database, as well as providing a friendlier user experience when inputting data. To reduce the redundancy of code between python and SQL, we took advantage of error handling from the SQL code to then pass along to the python code.
- **Database Design**: While working through combining python and SQL code together, it allowed us to think about utilizing procedures and triggers on the SQL side to integrate our functionalities more efficiently onto the python side. We made changes to the database tables to better organize certain data more efficiently.

Insights:

- **Project Management**: As a group of 3, we learned the importance of breaking down complex tasks and being mindful of keeping our code decoupled so that any changes made by one of us would not break the code for the others working on a different functionality. Overall, we learned the importance of task division, code modularity, and collaboration in a team setting, ensuring smooth progression and minimal disruption under a tight timeframe.
- **Healthcare Data Handling**: We gained insights into the complexity of handling sensitive healthcare data and the importance of privacy and security. To keep the project more user-friendly to those grading the assignment, the user's passwords are still displayed on the command line interface, however in a real-world application, this would not be the case.

Alternative Approaches:

- **Web-based Interface or GUI**: We considered a web-based platform or a Graphical User Interface for enhanced user accessibility and experience, however due to our limited experience in web

development and time constraints, we decided to stay with a simpler command line interface approach.

Code Challenges:

- **Importance of data validation:** A significant portion of development was dedicated to robust data validation to ensure system resilience against incorrect user inputs within our CLI. We had to think about how to best handle exceptions and catch errors and where to best do it: either with SQL or Python.
- **User error**: even with a thorough and clear CLI that prompts users with specific input requirements, you of course cannot guarantee that users will enter input in the format or data type you want, so a good chunk of our time went to ensuring that the program ran despite user error and bad input.

Bugs or Non-Working Code:

- We were able to handle most error handling through the combination of SQL and python, and we have not discovered any major bugs that render any of our functionalities unfunctional.

Future Work

Expansion Plans:

- **Graphical User Interface (GUI):** although we all wanted to implement a GUI for this project, we decided it made more sense to have a complete and "pretty" CLI before trying to make a half-baked GUI. Our functions were tightly coupled to the CLI format, so turning it into a GUI would have made our extensive data validations within the CLI moot. For that reason, a GUI for our database application is the next thing we would like to tackle within our database application.
- **More user roles:** While our current application supports a patient and a system administrator, we would like to ultimately roll-out functionality for more users as well. The application could be useful for medical offices (manage doctors, manage appointments, send bills/invoices/send prescriptions to pharmacies/etc.), doctors, and more.
- **Add photos to DB:** We intend to add functionality for uploading and storing photos, enhancing personalization for both doctors and patients.

Bonus Features

- Complicated Schema
    - Our project features complex queries utilizing multi-joins, and data pulling from multiple tables for certain complicated operations. For example, one such operation is scheduling appointments, where we are checking the doctor's availability, patient's current appointments, and then tracking appointment history to only allow reviews to be left by a patient only if the patient has that doctor during an appointment.
- Visualization of Data
    - For visualizing data, our project utilizes matplotlib and seaborn libraries to pull data from our database to allow users to visually see the average review score or star ratings of all the doctors. On top of the basic CLI functionality of seeing individual doctor review scores under the **"5. View Doctor Reviews"**, we have the option for data visualization

which allows users to have a more quick glance at the highest rated doctors in a bar graph format. When selecting the option for **"(2) visualize all reviews"** users get to see the pop-out window with the bar chart. In a more extensive database with more doctors, this feature has potential to allow a more user-friendly experience in searching for doctors in their network.

- Supports Multiple User Roles
    - The core functionality of our application surrounds supporting the patient. Patients are able to create an account/log in to manage all of their healthcare needs. In addition to the patient role, we added a system administrator role. This role allows the system admin to view all patients and doctors and perform necessary updates to the overall database. A system administrator will log in in the same way as a patient, however, due to the is_admin flag on their account, they will be directed to an admin specific menu. The login credentials of the 1 system administrator can be found within the README.

Conclusion

Our project journey was a blend of technical challenges and team collaboration, leading to an incredibly positive learning experience. The development of this healthcare application not only bolstered our technical expertise in Python and MySQL, but also provided us with valuable insights into project management and considerations on how to handle healthcare data.

Although our project presents a simplified product, the potential for this patient-centered healthcare application in real-world scenarios is important, particularly in enhancing patient-doctor interactions and streamlining healthcare management processes.

As a team, we are excited about the potential of our project, as we had initially come into the project with many grand ideas for its functionality and impact – the biggest limitation being the time constraints. However, this project showcases our teamwork, technical skills coming from a non-computer science background, and our commitment to creating impactful solutions and ideas.

README:

HealthEase Patient Portal

Introduction

This program is a healthcare management app, designed to provide patients with a unified platform to access and manage their health-related information and services.

File Structure

- SQL:
    - self-contained database dump: "HealthEaseDBDump-FINAL.sql
    - table creation: "HealthEaseDBCreate.sql "
    - test data insertion: "HealthEaseDB-DataCreation.sql "
    - stored procedures/triggers/functions: "HealthEaseDB-ProgrammingObjects.sql "
- Python: "health_ease_CLI.py" (our script for the CLI and running the program)

- README.md
- Final Report: "QuinnNSaslowKQuachL_final_report.pdf"

Model

The model behind the project is a database called "health_ease" built in MySQLWorkbench. The database represents a patient portal, where patients can view and interact with their healthcare team. The database also supports a second user role of system administrator. Patients have access to their doctors, their appointment history, prescriptions and billing, and system administrators can manage patients in the database (their accounts and credentials), doctors (add/delete doctors from the system), prescriptions, etc.

Command Line Interface

The controller and textual view of our project were built in Python. The python script can be found at "health_ease_CLI.py". Once the script is run, the user will be prompted through the program via the CLI, and can choose to interact with the program according to the main menu and command-line prompts.

Technical Specifications

To successfully import the database, the instructor will need to have MySQL (version 8.0 or later, download from MySQL's official download page here: https://dev.mysql.com/downloads/mysql/), and Python (version 3.7 or later, download from Python's official site here: https://www.python.org/downloads/).

To import the database, you will need the database dump, which can be found in the uploads as "HealthEaseDBDump-FINAL.sql". In MySQLWorkbench, establish a connection, navigate to the toolbar at the top of the screen. Go to Server → Data Import → select import From Self-Contained File → choose "HealthEaseDBDump-FINAL.sql" from the drop-down menu → and at the bottom select Dump Structure and Data.

Though not necessary if the data is imported and dumped from a self-contained file, the following files are also available to see the database (table) creation ("HealthEaseDBCreate.sql"), the test-data insertion ("HealthEaseDB-DataCreation.sql"), and the stored procedures/triggers/functions ("HealthEaseDB-ProgrammingObjects.sql"). Once the database has been imported, the instructor can interact with our program via the terminal using our command line interface.

To use and interact with the command line interface (CLI) of our application, the instructor will need to download Python and open up a Python environment. Please install the following packages to ensure successful execution of the following imports in the Python script:

- install ```pymysql``` - establish a connection to a MySQL server and interact with database
- install ```getpass``` - library for secure password input
- install ```datetime``` - library for manipulating dates and times
- install ```re``` - library for regular expressions operations and string matching
- install ```matplotlib.pyplot``` - package for data visualization
- install ```seaborn``` - package for data visualization

Should any of these not already be installed and used by the instructor, ```pip install <package_name>``` can be run in the terminal so that the imports at the top of the Python script run successfully.

To execute our script, please navigate in the terminal to the directory where the files are stored, and run: ```python3 health_ease_CLI.py```. This will launch our program and prompt the user to establish a connection with the MySQL server. Note: with the Python version suggested above, "python3" must be used. If an earlier version of Python is already installed by the instructor, the correct command to launch our application may be ```python health_ease_CLI.py```.

Upon launching the script, the instructor will be prompted to enter their SQL server credentials to establish a connection with the database. One a connection to the MySQL server is established successfully, the homepage of our patient portal HealthEase will be displayed and the user can get started.

For testing operations that require a pre-existing prescription or bill, the instructor can log-in to the patient portal of John Doe or Jane Smith:

John Doe

- username: john.doe@email.com
- password: password123$

Jane Smith

- username: jane.smith@email.com
- password: howdy123$

With either of these users, the instructor can view/refill prescriptions, view/pay bills and create/view/delete reviews that the user left for doctors. The instructor can also create an account and schedule/modify appointments, find doctors, and use the full application functionality (except prescription and bill administration, as these would have required an appointment to have already happened).

For testing the system administrator role of the application, the instructor can log-in to the portal with the following information:

Portal Admin

- username: portaladmin@healthease.com
- password: strongpass!

A different menu will appear which will expose the functionality of the program from the system administrator's side, where you can view/delete patients, update patient accounts, and view/add doctors.

Interacting with the Application

Once all necessary Python libraries are installed, the CLI will guide the instructor through our database application. Depending on whether the instructor chooses to log-in to the portal as a patient or administrator, the menu that appears will be different. At each step of the program, the instructor will be guided and prompted with clear instructions for what to do/what information to enter. If information

is entered incorrectly or an operation is not supported by the specific menu item selected, there are clear and descriptive error messages printed to the terminal so that the user can either try again, or select a different action from the menu.

Exiting the Program

At any time, the user can “exit” and navigate back to the main menu, where either “Q” or “q” can be typed to exit the program, which will terminal upon quitting. terminal upon quitting.