PAGE OF CONTENTS

INTRODUCTION

The hospital management system (HMS) is integrated software that handles different directions of clinic workflows. It manages the smooth healthcare performance along with administrative, medical, legal, and financial control. That is a cornerstone for the successful operation of the healthcare facility.

This application is the backend part of a hospital management system. You can use this backend application if you have any frontend graphical user interface of a hospital management system. This HMS application is completely designed in Python for specifically PostgreSQL Database.

Here different algorithms have been implanted from the developer's point of view. It is suitable for a mid-level hospital or clinic. If you use this application in the production part of a higher-level hospital, you may face problems because of the lack of functions. Then you have to create functions of your own. Note that many functions don't return any value, they just print the value in the terminal. But you probably need to return values instead of printing them. For that purpose, you have to change the source code.

Working strategy plays a vital role to keep this application running smoothly. Here Admin is the main player of this entire database, he has a handsome set of functions and can perform them to keep the track of every users' activity.

Every user is distinct by their unique 3-digit usernames. Usernames are the primary keys of each table. All of the operations are powered by the username.

There are lots of functionalists that can be applied in this application and lots of code blocks can be improved. I will work on this application in my free time and implement a few other basic functions and error handling. I have also planned to create a Frontend GUI and connect this backend to make it a full-stack project.

If you want to use this application in your project, first contact me.

FUNCTIONS

This Hospital Management System has eight modules and all of the modules have several useful functions.

- **constant.py**: Contains several constants that are used in other modules.

- **DB_config.py**: Configuration for connecting the database server.
    - *admin_config():* Used for extra tasks outside from the hms database server, but in the main database system.
    - *config():* Used for connecting the 'hms' database server.

- **main.py**: This is the main board of Hospital Management System. It is connected to all the modules. All of the backend work will be done from here. main.py can be used for glue code when connecting a frontend GUI to this backend program.

- **login.py**: When a user wants to perform a task in this program, he has to first be logged in or signed up in the system. This module supports this exact functionality.
    - *login():* First a user have to login, if he is a new user, then he can sign up in the system by the signup() function.
    - *signup():* Automatically called from the login() function for new user.
    - *DB_pass():* Fetch the password from the database for the user who tries to login.

- **admin.py**
    - *start_program():* This function will be run at the very first moment while creating a hospital management system. It drops the previous hospital management system database, and also doctor, patient and employee table (if exists). And then start a new database and creates those tables again. It also set up a default doctor named "Hospital" (hpt). This operation is vital for the entire application.

- *see_my_employee():* Admin can see all the employees who are working for the 'hospital' apart from their regular basis work using this function.
- *add_employee():* Admin can add an employee for hospital's extra work by this function.
- *remove_employee():* Admin can also remove their existing employees.
- *update_db():* Only admin can update different fields from every tables of this db. If user make mistakes while signup to the hospital management system. He can request the admin to update this record from the db.
- *add_notifications():* Admin can sent notifications to doctors, patients and employees using this function.
- *total_earning():* Admin can see the total earning of the hospital using this function. It calculates the total income by performing a complex calculation. It sums up total earnings from the patients and the employees and subtracts hospital fixed cost (constant.py) and also subtracts the cost for the employees.
- *remove_doctor_parmanently():* Admin can remove a doctor from the hospital management system's database permanently.
- *remove_patient_parmanently():* Admin can remove a patient from the hospital management system's database permanently.
- *remove_employee_parmanently():* Admin can remove an employee from the hospital management system's database permanently.
- *create_database():* This function is called inside the start_program() function. Admin can also call it externally. It creates the 'hms' database server.
- *delete_database():* Admin can call this function if he wants to delete the 'hms' database server. Using this function is not recommended.
- *show_all_doctor():* Admin can see all doctors info of the hospital.
- *show_all_patient():* Admin can see all patients info of the hospital.
- *show_all_employee():* Admin can see all employees info of the hospital.
- *delete_doctor_table():* This function is called inside the start_program() function. Admin can also call it externally. It deletes the doctor table from the 'hms' database server.

- *delete_patient_table():* This function is called inside the start_program() function. Admin can also call it externally. It deletes the patient table from the 'hms' database server.
- *delete_employee_table():* This function is called inside the start_program() function. Admin can also call it externally. It deletes the employee table from the 'hms' database server.
- *create_doctor_table():* This function is called inside the start_program() function. Admin can also call it externally. It creates the doctor table in the 'hms' database server.
- *create_patient_table():* This function is called inside the start_program() function. Admin can also call it externally. It creates the patient table in the 'hms' database server.
- *create_employee_table():* This function is called inside the start_program() function. Admin can also call it externally. It creates the doctor table in the 'hms' database server.

- **doctor.py**
  - *see_patients_report():* open patients specific report in the web browser.
  - *see_my_employee():* See the employees who are working for this doctor.
  - *recent_notifications():* Returns the recent notification from the admin.
  - *salary(): Returns the total income from the patients subtracts the employees' salaries.*
  - *show_all_employee():* show all employee according to their work.
  - *add_employee():* A doctor can add an employee for his personal work by this function.
  - *remove_employee():* Doctors can also remove their existing employees.
  - *remove_patient():* A doctor can remove a patient after or before the treatment.
  - *see_my_patient():* A doctor can see all of his patients info.
  - *see_all_requested_patient(): Doctors can see the patients info who are requesting for the appointment of the doctor.*

- *see_all_patient_of_my_speciality():* A doctor can see all of the patients who are facing problem in that particular category where this specific doctor have skills.

- ***employee.py***
    - *isreceptionist():* Checks if an employee is a receptionist or not.
    - *recent_notifications():* Returns the recent notification from the admin.
    - *appoint_doctor():* Only a receptionist can appoint a doctor including a date (format: “dd-mm-yy hh-mm”) for the patient.
    - *salary():* Calculate the total salary earned from doctors and hospital. And then subtracts the percentage of salary given to the hospital.
    - *see_my_doctors():* See the doctors info for whom the employee is working for.

- **patient.py**
    - *recent_notifications():* Returns the recent notification from the admin.
    - *add_report():* Add a new report with url.
    - *cost():* See the doctor’s fee and the hospital charge.
    - *remove_request():* remove the appointment request.
    - *request_doctor():* request a doctor for appointment.
    - *see_all_doctors_for_my_problem():* see the info of all doctors according to the problem of the patient facing.
    - *see_my_doctors_stat():* After the appoint request have been approved, the patient can see his doctor stats using this function.

Work Strategy

In this Hospital Management System, every user has some specific work strategies. They have their own duties and according to their duties they have several functions. If any user does not follow his work strategy the whole application might crash.

Admin

Admin is the main character in his entire application. All the application is designed based on the admin's work strategy. For starting this application successfully, the admin has to log in to the database server first. Note that, while asking the category in the login function, we haven't mentioned the admin category as we don't want other users to log in as admin accidentally. But the admin can input admin as the category. Then he has to input his password. Admin's password is not stored in the database. In my machine, it is stored in the environment variable. Admin can store his password in the environment variable and grab the password in the constant.py file. Or he can directly change the variable in the constant.py module (not recommended). Then he has to call the start_program() function as it sets up the default environment and creates the database server and tables. Admin can add employees, remove employees, see the employees who work extra for the hospital. He can calculate the total earning of the hospital. He can also create a doctor, patient, and employee table and delete them. And the important part is that he can add notifications to doctor, employee, or patient records so that they remain notified about their workflows. If any doctor, employee, or patient request the admin for any update, only the admin can update the database. Admin can also delete and create the database. He can also delete a doctor, employee, or patient permanently from the database.

Doctor

A doctor has to first log in to the application or sign-up y inputting proper information. Note that, the password of the doctor has to be a maximum of 5 characters. And for the specialty section doctor will input certain field names where he is capable like Eye, Teeth, Skin, Brain, Muscle, Bone, etc. He can only mention one specialty. After login into the application, he can add employees, remove employees, and see his employees who are working under him. He can also see his recent notification from the admin. He can see his patients' info and reports. Also, he can remove his patients. He can see his total salary. He can see which patients fall into his specialty and which patient is asking for his appointment.

Patient

A patient has to first login or signup to the application. During signup, he can request a doctor an appointment. But if he is not familiar with the doctors, then he can skip this question. After logging in to the application, he can add reports using URLs, see his recent notifications from the admin. He can see all the doctors who are capable to give him treatment. He can request an appointment. Once his request has been approved by the receptionist, he can see his doctor's stat. He can also remove his request and see his total cost for the treatment.

Employee

An employee first has to log in to the application or he can sign up. For signing up, he needs to choose his salary per work and also his work. An employee's work can be a receptionist, assistant, guard, cleaner, etc. As soon as he signed up on the database server, he will get his first client which is no other than the hospital. He will get only 10% of his salary per work from the hospital for joining the hospital. Hospitals admin can add the employees to working for the hospital. If the admin adds them, then they will get the 100% of their salary. They can also be added by the doctors. An employee can work for the hospital as well as for multiple doctors. He can see his salary too. But note that, he will only get 80% of his total salary, the other 20% will be added to the hospital's salary. This value can be changed from the constant.py file. Only a receptionist can appoint the doctor with his patients.