# hplus  
  
**N.B.**

**To login through “Admin login” in the website, here are the login credentials:**

**Admin Username- abir**

**Admin Password- abir123**

**When admin login is success, he can enter hospital code and hospital names along with email id of respective Hospital Manager. After he has added successfully, an email will go to the Hospital user/Manager regarding the login credentials through which he can enter into his Hospital Dashboard and add his Hospital details and number of beds available. (\*\*Point to be careful about: the Email will go the respective hospital user. If he can’t find email in the inbox section, he should search in the SPAM folder of his email\*\*) Hospital code will also be inside the email. If the Hospital user types wrong hospital code, a warning message will appear. After adding hospital details reload the page again(‘CTRL+R’ recommended) and he will be able to see his hospital details printed on the right hand side . Then he can’t make entries again otherwise warning message will say “Data is Already Present. You can Update It”. There will appear edit and delete icons on the right hand side from where they can update or delete their Hospital entry as per their choice. Now the Hospital User can safely logout from the session.**

**Now the patient or user can login through user login option to book a slot, see his details or available beds. New user can sign up and safely login. When user books a slot, he can’t book slot again otherwise warning message will appear. He can fill in his details in the form that will appear and book slots according to a specific hospital and the number of beds available there(which will appear to be printed on right hand side of the same slotbooking page). He can now click patient details to see if he has correctly filled the data or not or see his report.**

**\*\*\*[There are two videos in my github link(**[**https://github.com/ALphastrEAM5/hplus.git**](https://github.com/ALphastrEAM5/hplus.git)

**) and also in my drive link. One video is there where I gave the presentation and in the other video, I discussed about how to run the website. To know more, see the video I shared titled ‘How to Run’]\*\*\***

**Dependencies:**

The following Python modules are required to run this Flask web app:

Flask: a micro web framework for Python

Flask-Login: provides user session management

Flask-WTF: provides integration with WTForms for form validation

Flask-MySQLdb: provides integration with MySQL database

Flask-Mail: provides email functionality for sending notifications

Werkzeug: provides password hashing and verification

smtplib: provides SMTP email functionality for sending emails

Make sure to install these dependencies using pip or any other package manager before running the Flask web app.

**Resources:**

Google Cloud MySQL (host=localhost/instance-1, user= root, password=[None], port: 8080)

N2D AMD Instance and Confidential Computing

Compute Engine VM instance(boot disk image: ubuntu-pro-fips-2004-focal-v20230302)

Cloud-storage Bucket(dbms-bucket)

**Manage links:**

Github Repository:

https://github.com/ALphastrEAM5/hplus.git

Website: 127.0.0.1:8080

**N.B:** Before running the website on localhost of your PC import “covid.sql” file from the github repo to your local Xampp server

**Code Overview:**

before\_request() function: This function is a Flask decorator that is executed before every request to the web app. It is used to set the g.hospital\_user global variable, which stores the currently logged-in hospital user. The g object is a global context that can be used to store data that is specific to the current request.

hospital\_login\_required() function: This function is a custom Flask decorator that is used to restrict access to certain routes to only logged-in hospital users. It checks if the g.hospital\_user global variable is set, which indicates that a hospital user is logged in. If not, it redirects the user to the hospital login page. This decorator is used to protect routes that require hospital user authentication.

home() function: This function is a Flask route that renders the home page of the web app. It returns the rendered "index.html" template.

signup() function: This function is a Flask route that handles user registration. It accepts both POST and GET requests. If a POST request is received, it validates the user registration form data, creates a new user in the database, and sends a confirmation email. If a GET request is received, it renders the "usersignup.html" template.

login() function: This function is a Flask route that handles user login. It accepts both POST and GET requests. If a POST request is received, it validates the user login form data, logs in the user using Flask-Login's login\_user() function, and redirects to the home page. If a GET request is received, it renders the "userlogin.html" template.

hospitallogin() function: This function is a Flask route that handles hospital user login. It accepts both POST and GET requests. If a POST request is received, it validates the hospital user login form data, sets the g.hospital\_user global variable, and redirects to the "addhospitalinfo" route. If a GET request is received, it renders the "hospitallogin.html" template.

admin() function: This function is a Flask route that handles admin login. It accepts both POST and GET requests. If a POST request is received, it validates the admin login form data, sets the "user" key in the session, and redirects to the "addHosUser.html" template. If a GET request is received, it renders the "admin.html" template.

logout() function: This function is a Flask route that handles user logout. It requires the user to be logged in, as it is decorated with Flask-Login's login\_required decorator. It logs out the user using Flask-Login's logout\_user() function and redirects to the login page.

addhospitalinfo() function: This function is a Flask route that handles adding hospital information by logged-in hospital users. It requires hospital user authentication, as it is decorated with the custom hospital\_login\_required decorator. It accepts both POST and GET requests. If a POST request is received, it validates the hospital information form data, creates a new entry in the database, and redirects to the home page. If a GET request is received, it renders the "addhospitalinfo.html" template.

viewhospitalinfo() function: This function is a Flask route that handles viewing hospital information. It requires hospital user authentication, as it is decorated with the custom hospital\_login\_required decorator. It retrieves the hospital information from the database and renders the "viewhospitalinfo.html" template with the retrieved data.

edithospitalinfo() function: This function is a Flask route that handles editing hospital information by logged-in hospital users. It requires hospital user authentication, as it is decorated with the custom hospital\_login\_required decorator. It accepts both POST and GET requests. If a POST request is received, it validates the edited hospital information form data, updates the entry in the database, and redirects to the home page. If a GET request is received, it retrieves the current hospital information from the database and renders the "edithospitalinfo.html" template with the retrieved data.

deletehospitalinfo() function: This function is a Flask route that handles deleting hospital information by logged-in hospital users. It requires hospital user authentication, as it is decorated with the custom hospital\_login\_required decorator. It accepts both POST and GET requests. If a POST request is received, it retrieves the hospital information from the database and deletes the entry, then redirects to the home page. If a GET request is received, it renders the "deletehospitalinfo.html" template with the retrieved hospital information.

send\_email() function: This function is a helper function that is used to send emails using Flask-Mail. It takes the recipient email address, subject, and body as input, and sends an email using the configured email settings in the Flask application.

run() function: This function is the entry point of the Flask application. It configures the Flask application with the required settings, such as secret key, database connection, email settings, and error handlers. It registers all the Flask routes and starts the Flask development server to listen for incoming HTTP requests.