

Project Sprint #5

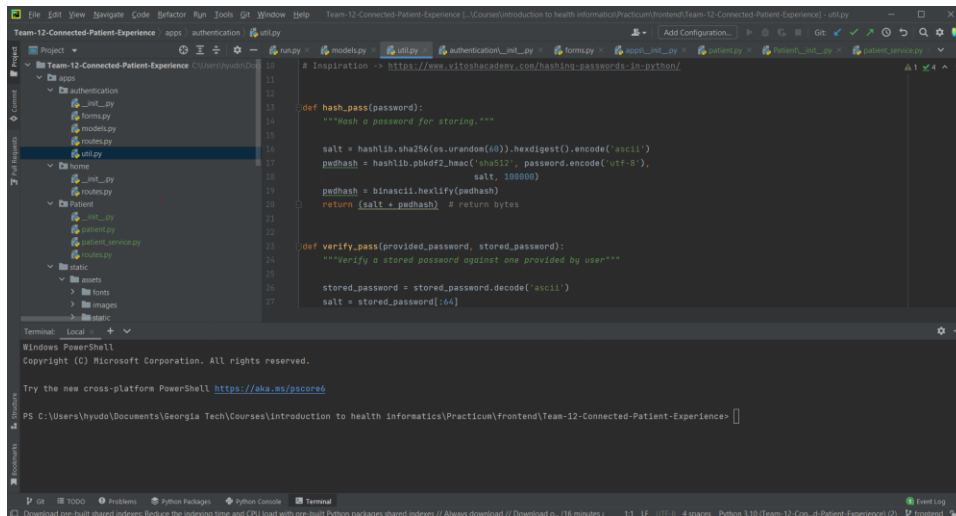
Aijing Gao, Shiyi Qin, Baiyan Ren, Yi Wang, Haojie Yu

agao48@gatech.edu

TASKS ACCOMPLISHED

We have updated the structure of github repository by adding one additional branch, frontend_backend_integrated. The codes in this branch are used for the set-up of local development environments.

Haojie Yu has integrated backend architecture with front-end and improved authentication functionality with sh256 hashing function. A patient service is also implemented on the backend. We have figured out how to connect front end to the backend through REST api.

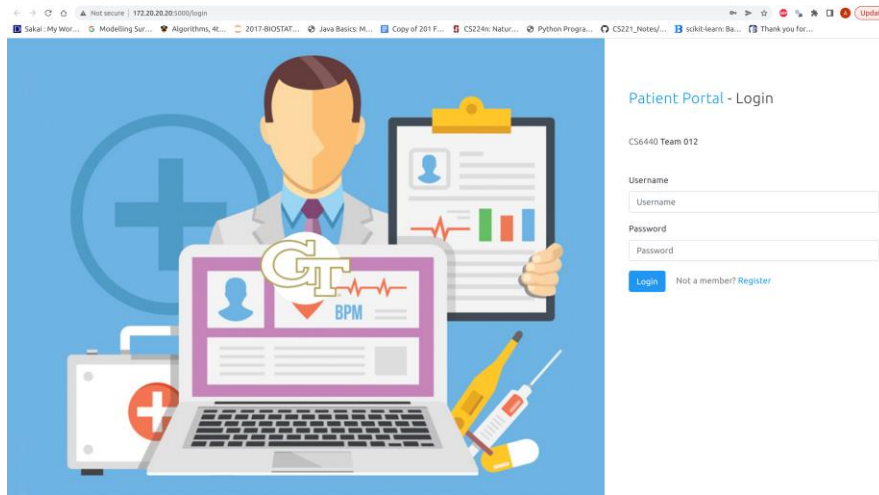


```
def hash_pass(password):
    """Hash a password for storing."""
    salt = hashlib.sha256(os.urandom(30)).hexdigest().encode('ascii')
    pwhash = hashlib.pbkdf2_hmac('sha12', password.encode('utf-8'),
                                salt, 100000)
    pwhash = binascii.hexlify(pwhash)
    return (salt + pwhash) # return bytes

def verify_pass(provided_password, stored_password):
    """Verify a stored password against one provided by user"""
    stored_password = stored_password.decode('ascii')
    salt = stored_password[:64]
```

Baiyan cleaned the table in the database. In addition, Baiyan created a table in the MySQL database: appointment. This table contains patient ID, time of appointment, and description of appointment. We will show future appointments in the calendar.

Aijing Gao designed the login page of the web portal and updated the web app framework to include the minimum required pages/tabs. She also added a User table to keep records of username, password, and email address that are used for the authentication of the web app.



Shiyi has investigated the structure of the Github repository and tried to integrate the test results html into the template. Below is an illustration of the test results visualization, which has replaced the original data table in the template.



Yi investigated integrating the database to the template, setting up the mysql into platform. Yi also tried to create the data models (medication, observation, careplan, appointment 44) in the platform.

CHALLENGES

Although it is feasible to visualize the test result bars using the html document generated from Plotly, the size of the generated html is relatively large since it contains the Plotly.js package. Besides, we also need to adapt the backend code to incorporate the Python code that automatically generates the test results html; this involves routing use of data to the SQL database. An alternative approach is to modify the current Github repository to accommodate the test results data, but we have not figured out a method to show both the normal range and

the test result values using the currently available charts in the template. Also, most of the team members are not familiar with REST API for data handling. We will spend some time learning how to create REST API services.

PLANS AHEAD

We will first try building a dashboard using a flask framework template (<https://github.com/app-generator/flask-administrator>). A large amount of development will be worked on next week: 1) Data and table will be cleaned up to include relevant data 2) Continue to implement multiple endpoints to query or post data by using flask and python 3) Display data on the front-end by calling endpoints using html, javascript and css. 4) Try to figure out tables mapping

Updated timeline

