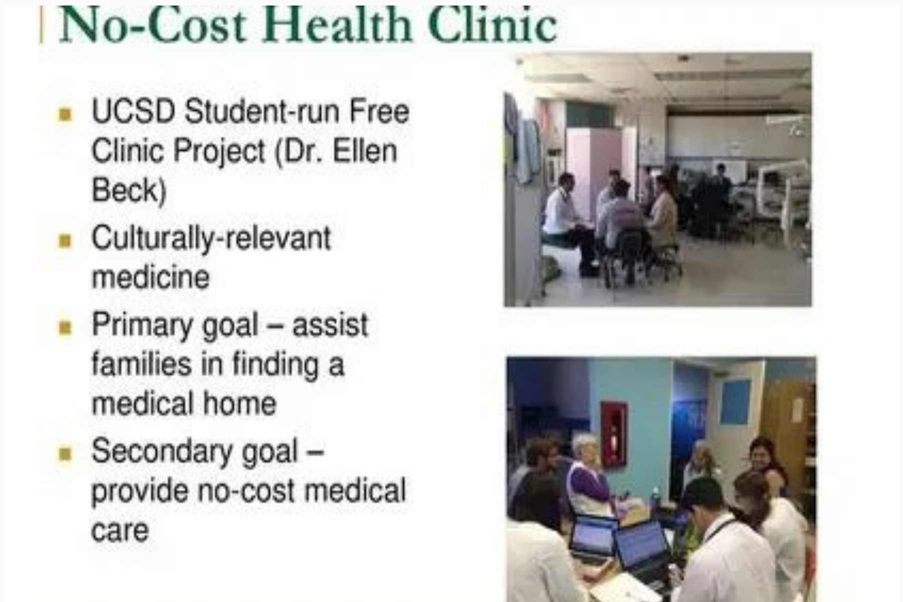
Note Taking Software for Primary Care Doctors

t

What a patient says to a doctor about their health and symptoms is vital to proper treatment

Who are the Users of the Note Taking Software?

1. Primary Care Providers in low cost medical clinics looking for a simple effective note taking application for use in examination rooms that can can allow for effective note taking or voice activation of notes regarding patient symptoms and diagnosis.
2. A reference database that is searchable for low cost generic medications that can give a primary care provider a list of medications and notifications of possible drug interactions
3. An ability to store patient notes locally and print examination notes for insertion into the patient's file.
4. Compatibitable application for a a portable electronic device such as a smartphone, ipad or chromebook.
5. Low cost subscription based product that offers level tiers of usability and upgradability.



No-Cost Health Clinic

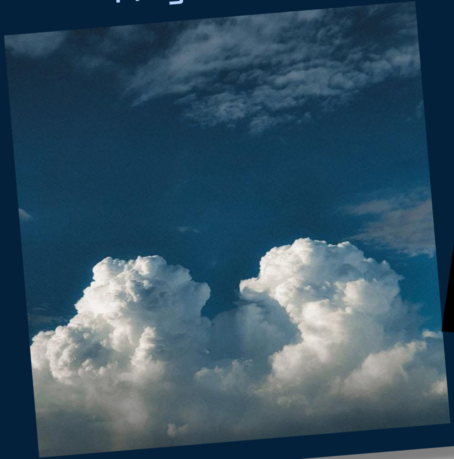
- UCSD Student-run Free Clinic Project (Dr. Ellen Beck)
- Culturally-relevant medicine
- Primary goal – assist families in finding a medical home
- Secondary goal – provide no-cost medical care

Meet the TEAM

Corey

Team Alola

Programmer



Lawrence

Team Alola

Project Director

Renze

Team Alola

User Experience Coordinator

Milestone 1

🔷 Sprint 0: Project Setup & Foundation

Goal: Have the environment ready and the team aligned.
User Tasks: Corey 1-2, Larry,Renze 3-4

1. As a **developer**, I want to create a GitHub repository so that I can manage version control.https://github.com/CGU-IST-303-Alola/group-project
2. As a **developer**, I want to set up a Python/Flask project so that I can build the web application foundation.
3. As a **team member**, I want a backlog in a project board (e.g., Jira, GitHub Projects,) so that I can track user stories.https://ist303-alora.atlassian.net/jira/software/projects/TAP/boards/1
4. As a **team member** I want to identify and download a generic medication database for use in the database integration.https://go.drugbank.com/

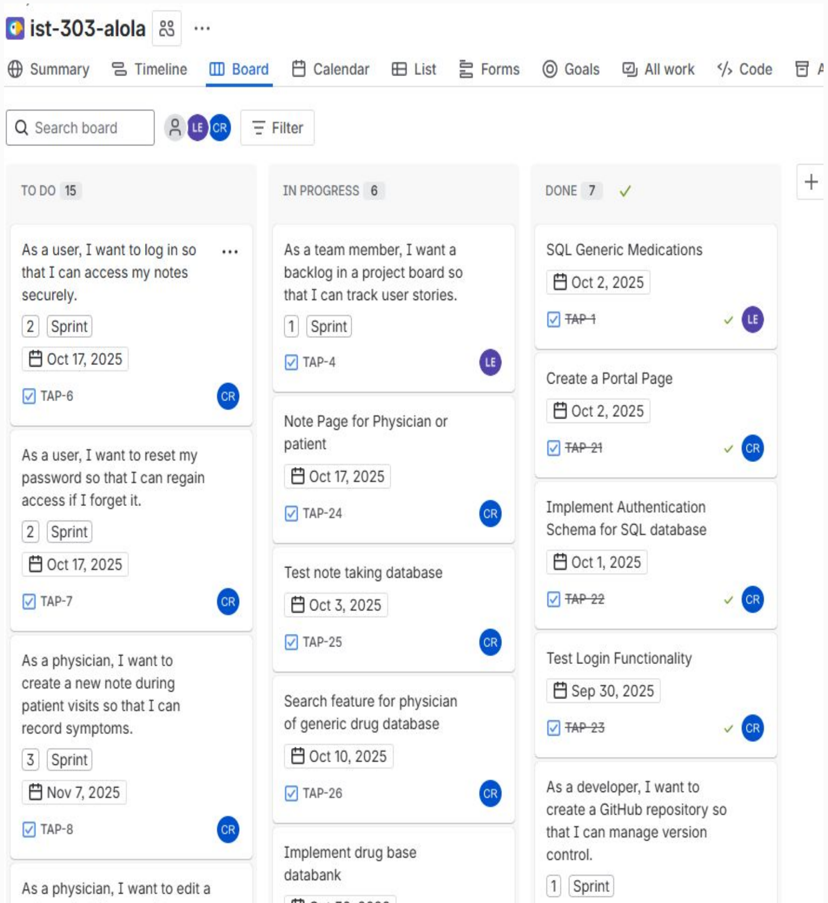
🔹 Sprint 1: Basic User Authentication

Goal: Users can log in and access the app securely as a Portal Page.

User Tasks: Corey 4-5,Lawrence & Renze 1-3

1. As a **user**, I want to sign up with a username and password so that I can create an account.
2. As a **user**, I want to log in so that I can access my notes securely.
3. As a **user**, I want to reset my password so that I can regain access if I forget it.
4. As a **developer** I want to develop code tests for login functionality.
5. As a **developer** I want to implement authentication schema for SQL.

JIRA Storyboard for Note Taking App



ist-303-alola

Summary | Timeline | Board | Calendar | List | Forms | Goals | All work | Code

Search board | LE CR | Filter

TO DO 15

As a user, I want to log in so that I can access my notes securely.
2 Sprint
Oct 17, 2025
TAP-6 CR

As a user, I want to reset my password so that I can regain access if I forget it.
2 Sprint
Oct 17, 2025
TAP-7 CR

As a physician, I want to create a new note during patient visits so that I can record symptoms.
3 Sprint
Nov 7, 2025
TAP-8 CR

As a physician, I want to edit a

IN PROGRESS 6

As a team member, I want a backlog in a project board so that I can track user stories.
1 Sprint
TAP-4 LE

Note Page for Physician or patient
Oct 17, 2025
TAP-24 CR

Test note taking database
Oct 3, 2025
TAP-25 CR

Search feature for physician of generic drug database
Oct 10, 2025
TAP-26 CR

Implement drug base databank

DONE 7 ✓

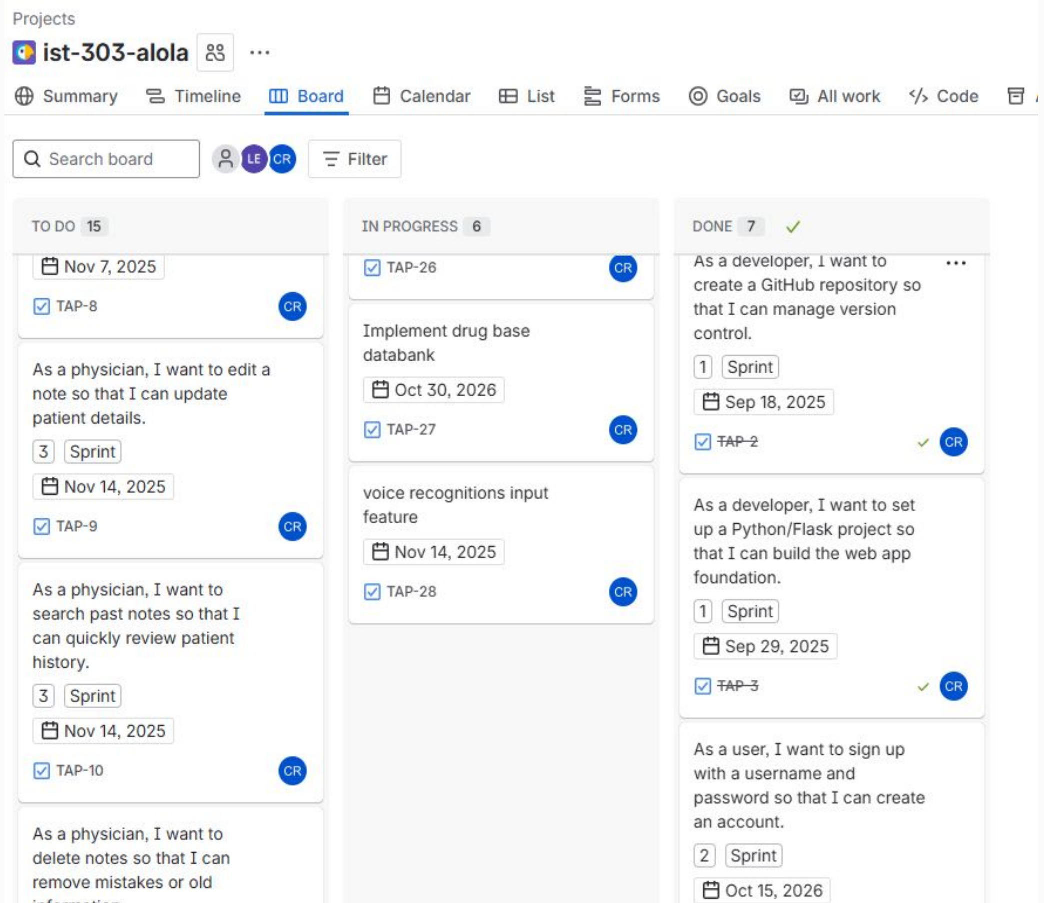
SQL Generic Medications
Oct 2, 2025
~~TAP-1~~ ✓ LE

Create a Portal Page
Oct 2, 2025
~~TAP-21~~ ✓ CR

Implement Authentication Schema for SQL database
Oct 1, 2025
~~TAP-22~~ ✓ CR

Test Login Functionality
Sep 30, 2025
~~TAP-23~~ ✓ CR

As a developer, I want to create a GitHub repository so that I can manage version control.
1 Sprint



Projects

ist-303-alola

Summary | Timeline | Board | Calendar | List | Forms | Goals | All work | Code

Search board | LE CR | Filter

TO DO 15

Nov 7, 2025
TAP-8 CR

As a physician, I want to edit a note so that I can update patient details.
3 Sprint
Nov 14, 2025
TAP-9 CR

As a physician, I want to search past notes so that I can quickly review patient history.
3 Sprint
Nov 14, 2025
TAP-10 CR

As a physician, I want to delete notes so that I can remove mistakes or old

IN PROGRESS 6

TAP-26 CR

Implement drug base databank
Oct 30, 2026
TAP-27 CR

voice recognitions input feature
Nov 14, 2025
TAP-28 CR

DONE 7 ✓

As a developer, I want to create a GitHub repository so that I can manage version control.
1 Sprint
Sep 18, 2025
~~TAP-2~~ ✓ CR

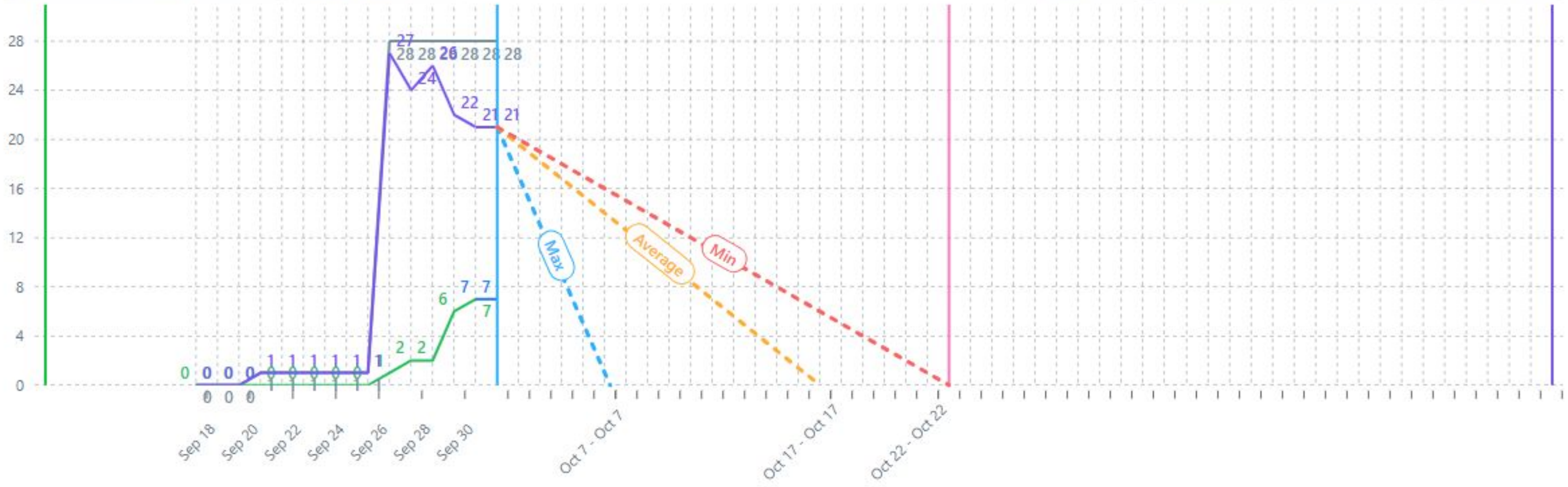
As a developer, I want to set up a Python/Flask project so that I can build the web app foundation.
1 Sprint
Sep 29, 2025
~~TAP-3~~ ✓ CR

As a user, I want to sign up with a username and password so that I can create an account.
2 Sprint
Oct 15, 2026

IST-303-Alola PCP Note Taking App

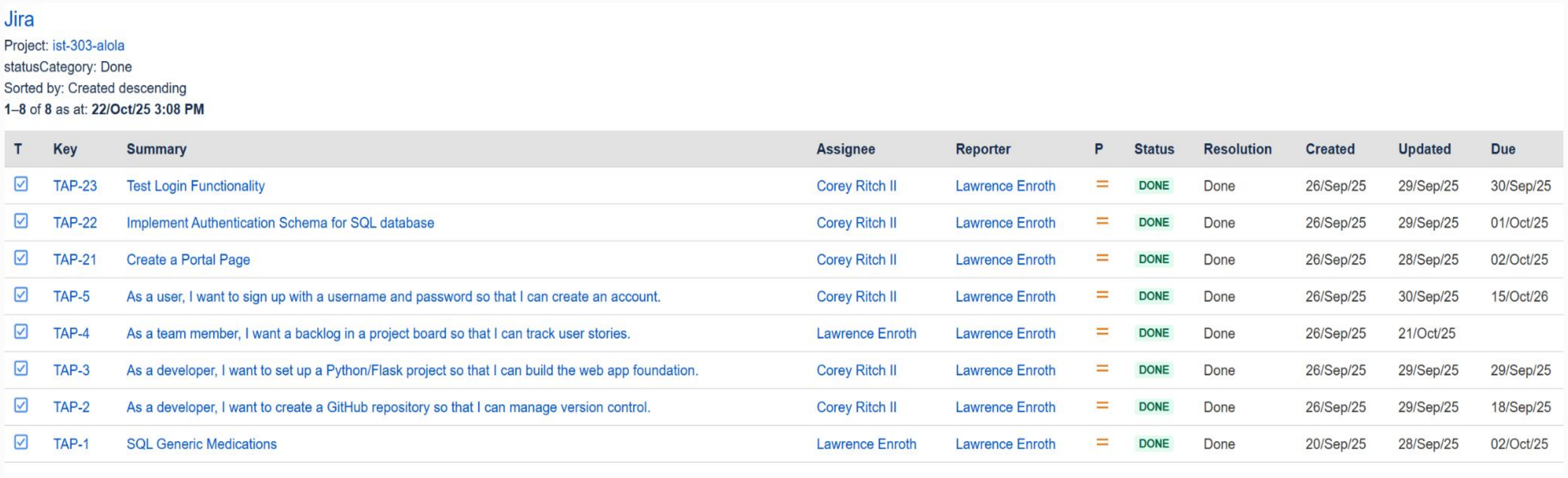
Home

2025-09-18 → 2025-11-20 | Group: Daily | Metrics | Forecast | Scenarios | Targets



Forecast

Label	Type	Velocity	Complete date	Intervals
🟦 Max	Auto	4	10/07/2025	6 days
🟧 Average	Auto	1.4	10/17/2025	16 days
🟥 Min	Auto	1	10/22/2025	21 days



Jira

Project: ist-303-alola

statusCategory: Done

Sorted by: Created descending

1–**8** of **8** as at: **22/Oct/25 3:08 PM**

T	Key	Summary	Assignee	Reporter	P	Status	Resolution	Created	Updated	Due
☑	TAP-23	Test Login Functionality	Corey Ritch II	Lawrence Enroth	=	DONE	Done	26/Sep/25	29/Sep/25	30/Sep/25
☑	TAP-22	Implement Authentication Schema for SQL database	Corey Ritch II	Lawrence Enroth	=	DONE	Done	26/Sep/25	29/Sep/25	01/Oct/25
☑	TAP-21	Create a Portal Page	Corey Ritch II	Lawrence Enroth	=	DONE	Done	26/Sep/25	28/Sep/25	02/Oct/25
☑	TAP-5	As a user, I want to sign up with a username and password so that I can create an account.	Corey Ritch II	Lawrence Enroth	=	DONE	Done	26/Sep/25	30/Sep/25	15/Oct/26
☑	TAP-4	As a team member, I want a backlog in a project board so that I can track user stories.	Lawrence Enroth	Lawrence Enroth	=	DONE	Done	26/Sep/25	21/Oct/25	
☑	TAP-3	As a developer, I want to set up a Python/Flask project so that I can build the web app foundation.	Corey Ritch II	Lawrence Enroth	=	DONE	Done	26/Sep/25	29/Sep/25	29/Sep/25
☑	TAP-2	As a developer, I want to create a GitHub repository so that I can manage version control.	Corey Ritch II	Lawrence Enroth	=	DONE	Done	26/Sep/25	29/Sep/25	18/Sep/25
☑	TAP-1	SQL Generic Medications	Lawrence Enroth	Lawrence Enroth	=	DONE	Done	20/Sep/25	28/Sep/25	02/Oct/25

🔷 Team Member & User Stories

Team Member Stories

Use **Jira Board** to track backlog and user stories

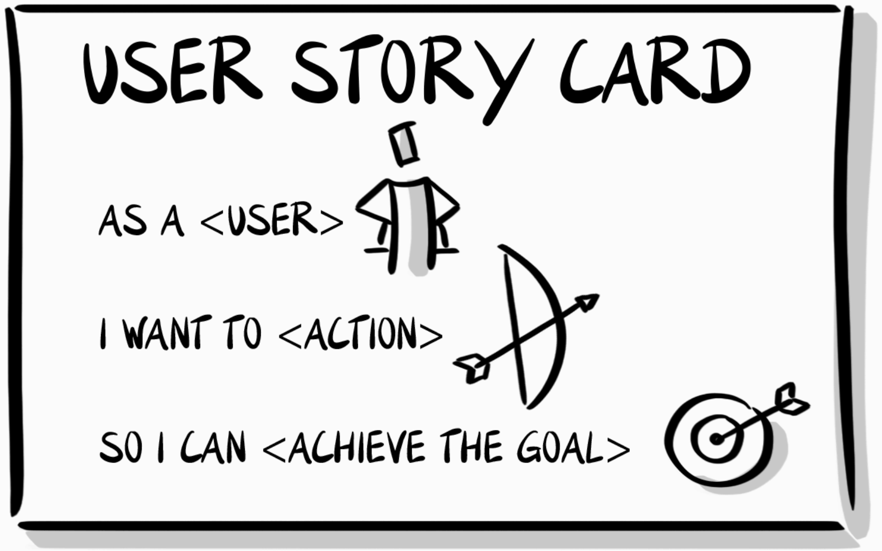
Download **DrugBank** database for medication integration

User Stories

Log in with username & password

Log in to access notes securely

Alerts when successfully log in or have an incorrect username/password



Steps and Project Structure

```
group-project/
├── app/
│   ├── __init__.py
│   ├── init_db.py
│   ├── database.py
│   ├── routes.py
│   ├── schema.sql
│   ├── data/
│   │   └── database.db
│   ├── static/
│   │   └── css/
│   │       └── login.css
│   └── templates/
│       ├── base.html
│       ├── login.html
│       ├── home_patient.html
│       ├── home_physician.html
│       └── home_admin.html
├── tests/
│   ├── conftest.py
│   ├── test_protected_routes.py
│   └── test_login.py
├── run.py
├── guide.md
├── requirements.txt
└── .gitignore
```

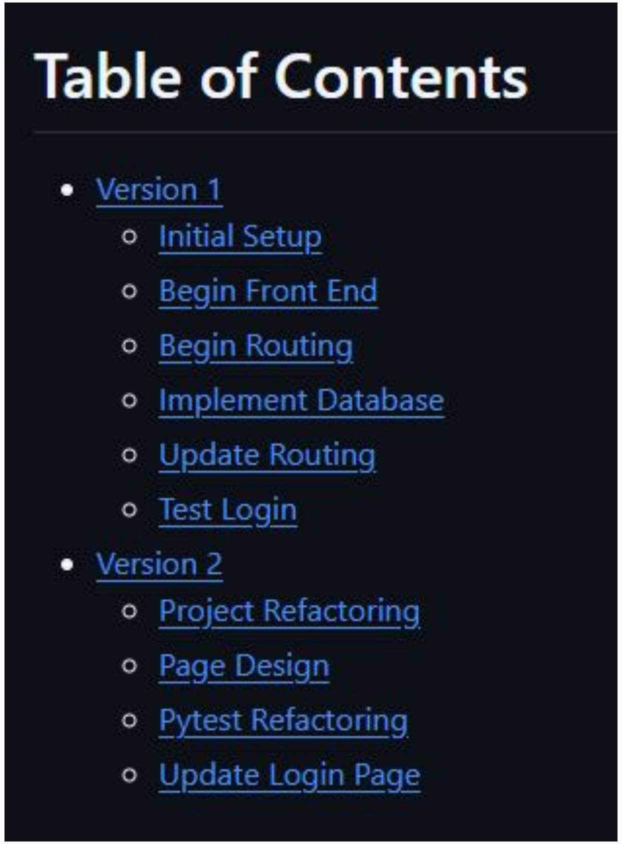
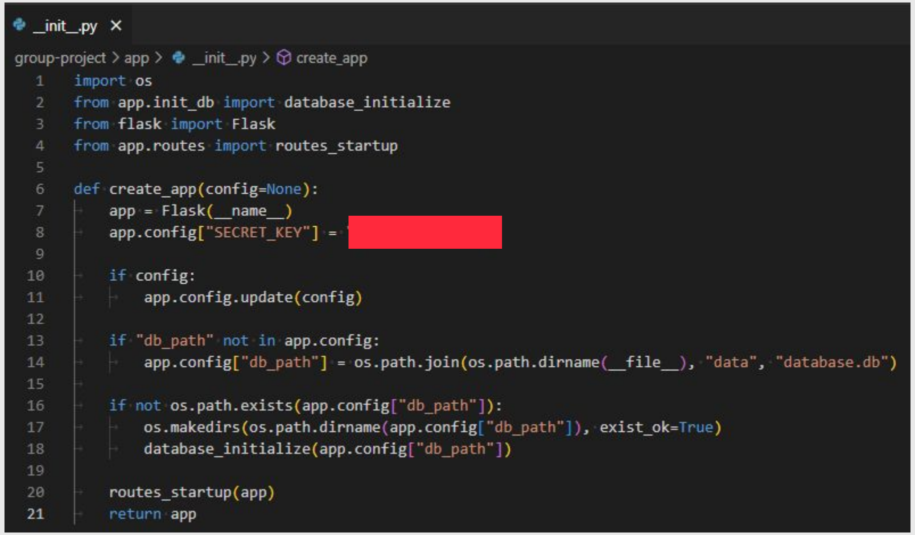


Table of Contents

Flask App

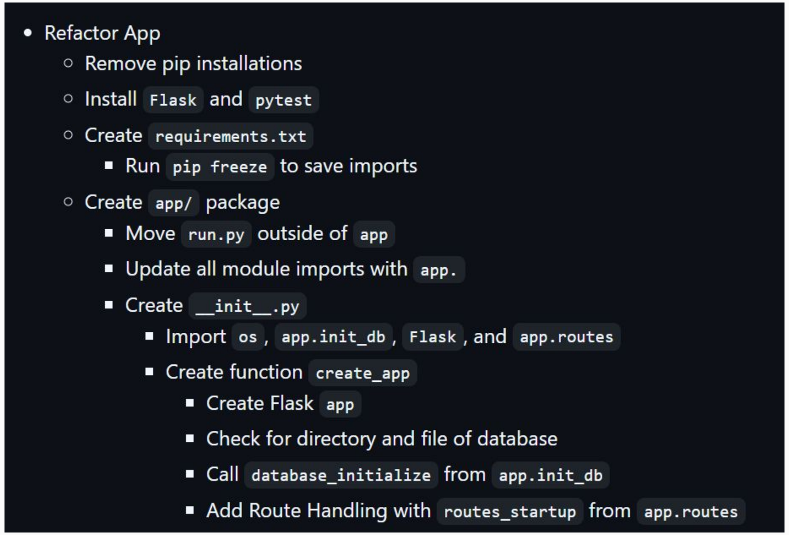


init.py ×

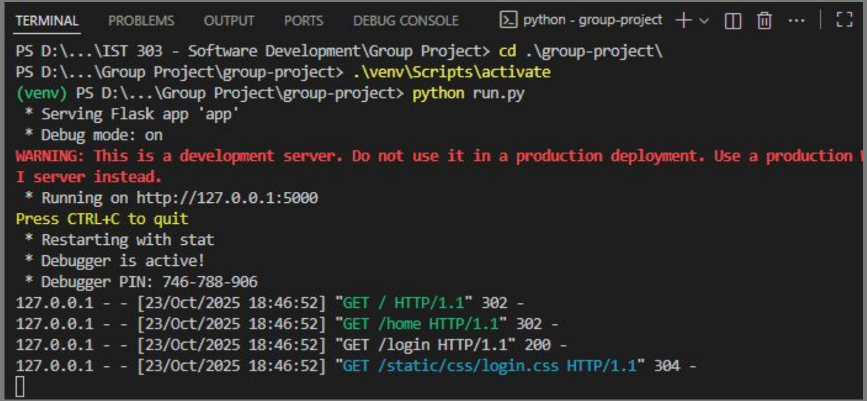
group-project > app > _init_.py > create_app

```python
import os
from app.init_db import database_initialize
from flask import Flask
from app.routes import routes_startup

def create_app(config=None):
    app = Flask(__name__)
    app.config["SECRET_KEY"] = '[redacted]

    if config:
        app.config.update(config)

    if "db_path" not in app.config:
        app.config["db_path"] = os.path.join(os.path.dirname(__file__), "data", "database.db")

    if not os.path.exists(app.config["db_path"]):
        os.makedirs(os.path.dirname(app.config["db_path"]), exist_ok=True)
        database_initialize(app.config["db_path"])

    routes_startup(app)
    return app
```

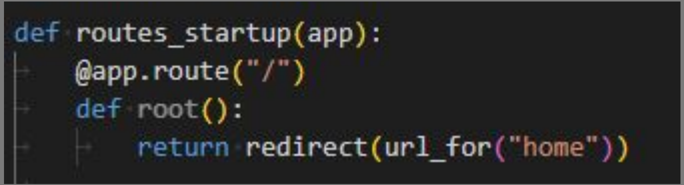
Don't take our secret key
:D



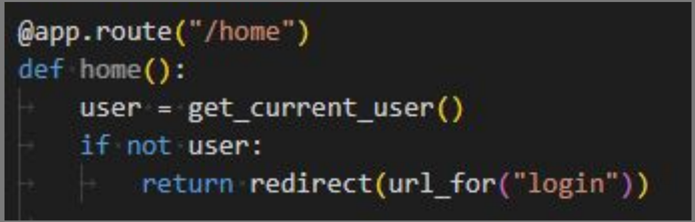
- Refactor App
  - Remove pip installations
  - Install `Flask` and `pytest`
  - Create `requirements.txt`
    - Run `pip freeze` to save imports
  - Create `app/` package
    - Move `run.py` outside of `app`
    - Update all module imports with `app.`
    - Create `__init__.py`
      - Import `os`, `app.init_db`, `Flask`, and `app.routes`
      - Create function `create_app`
        - Create Flask `app`
        - Check for directory and file of database
        - Call `database_initialize` from `app.init_db`
        - Add Route Handling with `routes_startup` from `app.routes`



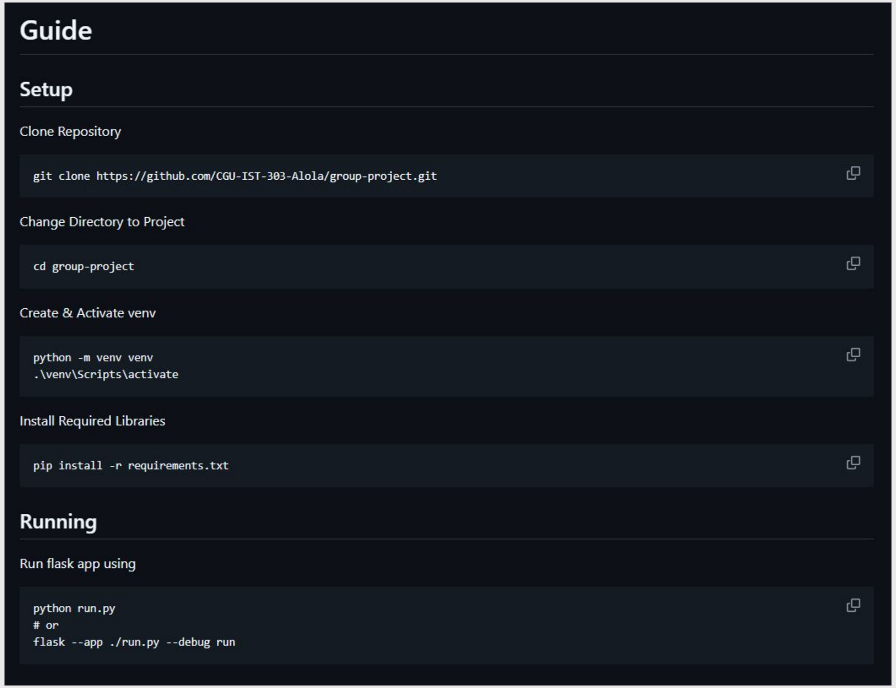
```
PS D:\...\IST 303 - Software Development\Group Project> cd .\group-project\
PS D:\...\Group Project\group-project> .\venv\Scripts\activate
(venv) PS D:\...\Group Project\group-project> python run.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production 
I server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 746-788-906
127.0.0.1 - - [23/Oct/2025 18:46:52] "GET / HTTP/1.1" 302 -
127.0.0.1 - - [23/Oct/2025 18:46:52] "GET /home HTTP/1.1" 302 -
127.0.0.1 - - [23/Oct/2025 18:46:52] "GET /login HTTP/1.1" 200 -
127.0.0.1 - - [23/Oct/2025 18:46:52] "GET /static/css/login.css HTTP/1.1" 304 -
```



```python
def routes_startup(app):
    @app.route("/")
    def root():
        return redirect(url_for("home"))
```



```python
@app.route("/home")
def home():
    user = get_current_user()
    if not user:
        return redirect(url_for("login"))
```



Guide

Setup

Clone Repository

```
git clone https://github.com/CGU-IST-303-Alola/group-project.git
```

Change Directory to Project

```
cd group-project
```

Create & Activate venv

```
python -m venv venv
.\venv\Scripts\activate
```

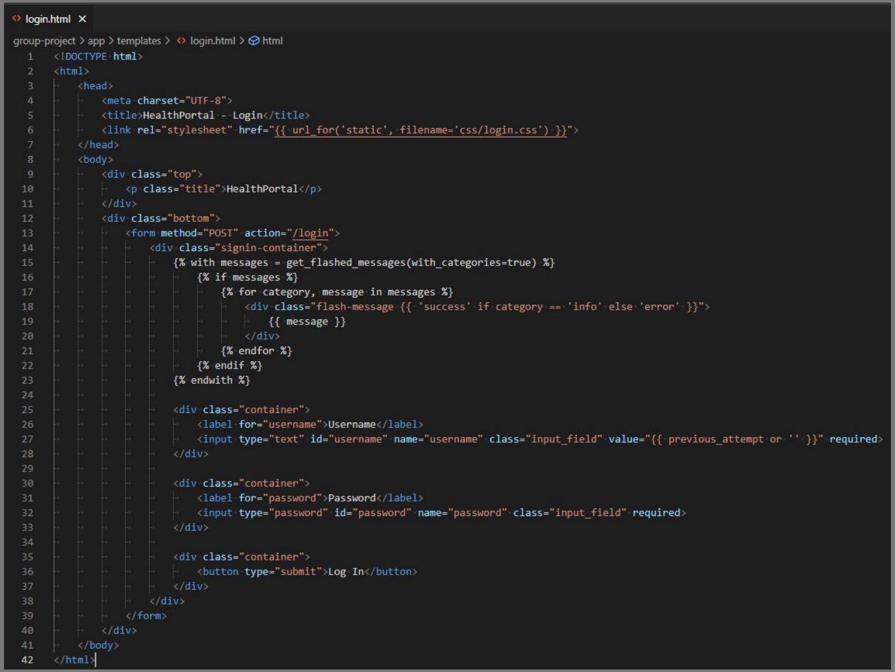
Install Required Libraries

```
pip install -r requirements.txt
```

Running

Run flask app using

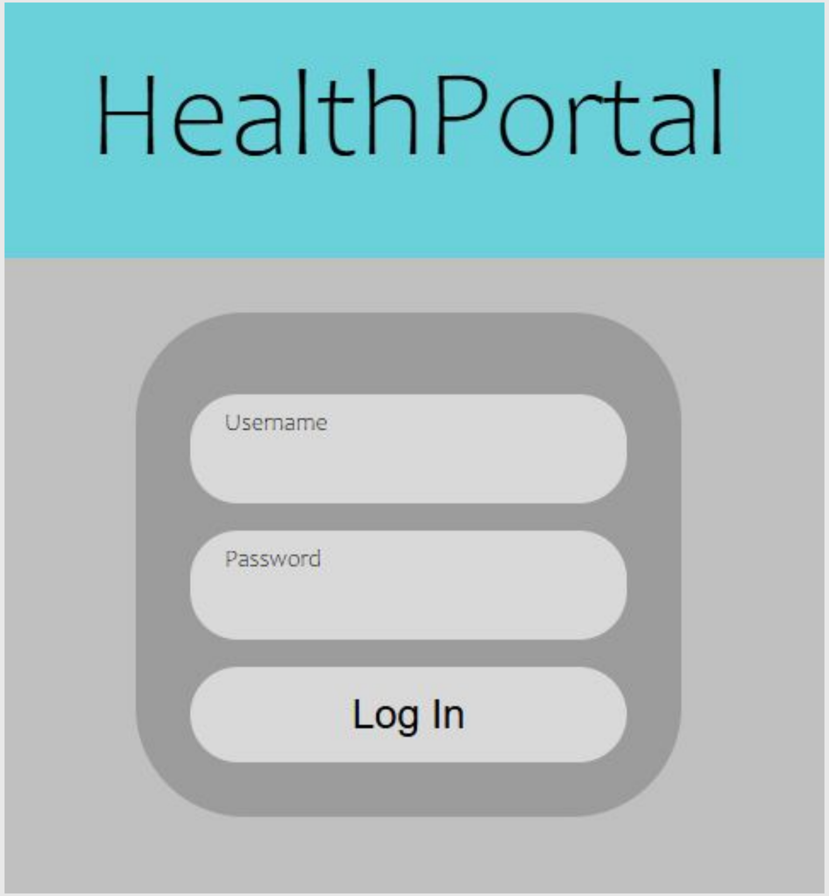
```
python run.py
# or
flask --app ./run.py --debug run
```

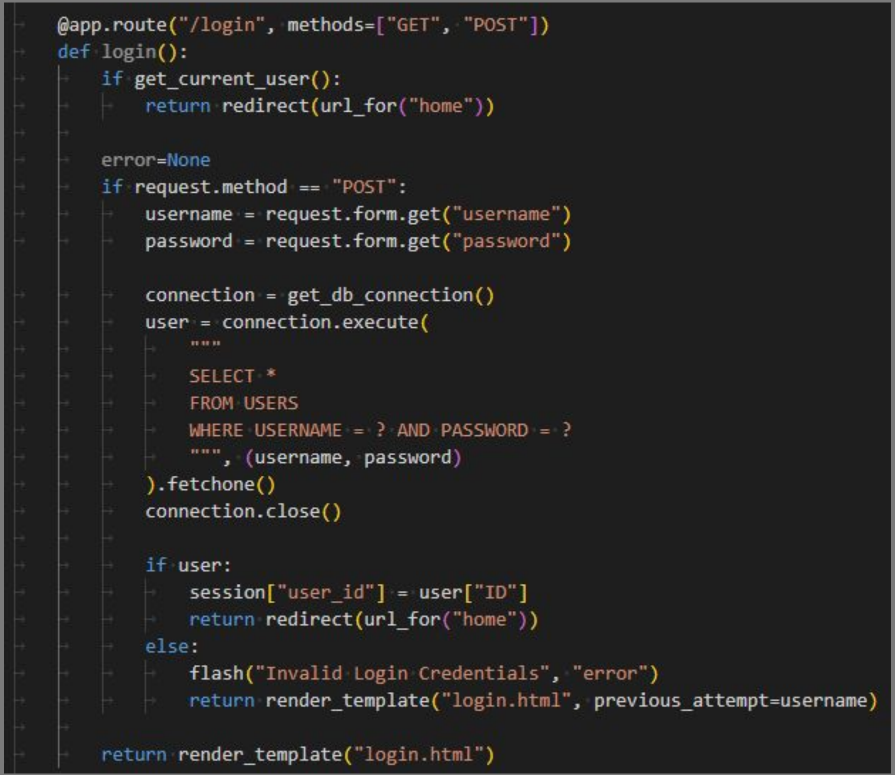


login.html

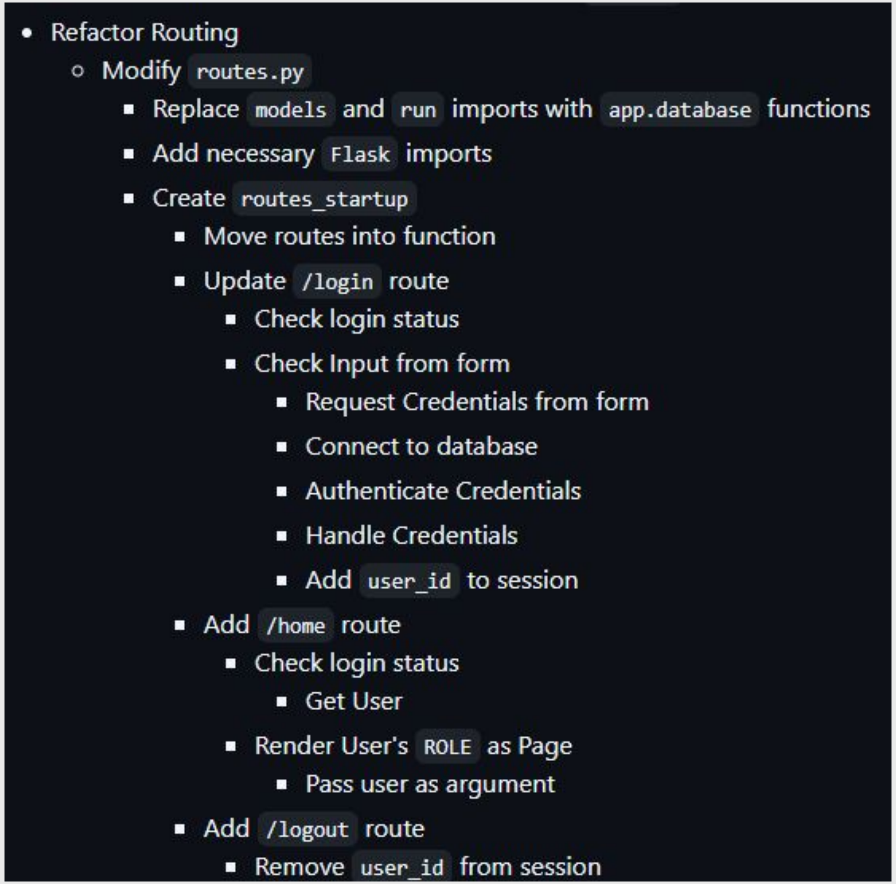
group-project > app > templates > login.html > html

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>HealthPortal - Login</title>
        <link rel="stylesheet" href="{{ url_for('static', filename='css/login.css') }}">
    </head>
    <body>
        <div class="top">
            <p class="title">HealthPortal</p>
        </div>
        <div class="bottom">
            <form method="POST" action="/login">
                <div class="signin-container">
                    {% with messages = get_flashed_messages(with_categories=true) %}
                        {% if messages %}
                            {% for category, message in messages %}
                                <div class="flash-message {{ 'success' if category == 'info' else 'error' }}">
                                    {{ message }}
                                </div>
                            {% endfor %}
                        {% endif %}
                    {% endwith %}

                    <div class="container">
                        <label for="username">Username</label>
                        <input type="text" id="username" name="username" class="input_field" value="{{ previous_attempt or '' }}" required>
                    </div>

                    <div class="container">
                        <label for="password">Password</label>
                        <input type="password" id="password" name="password" class="input_field" required>
                    </div>

                    <div class="container">
                        <button type="submit">Log In</button>
                    </div>
                </div>
            </form>
        </div>
    </body>
</html>
```

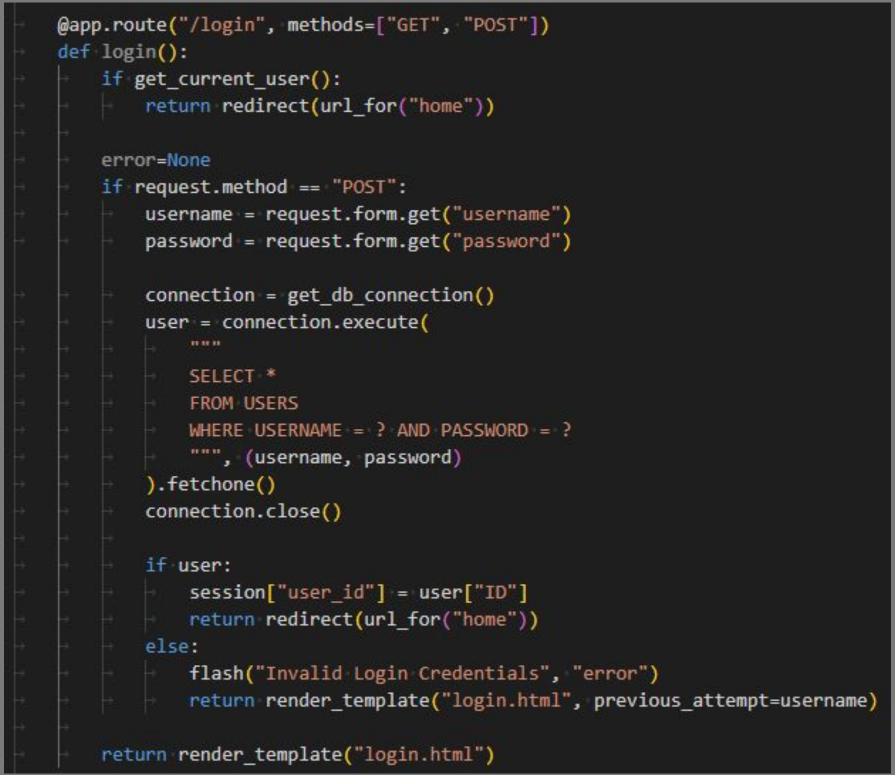




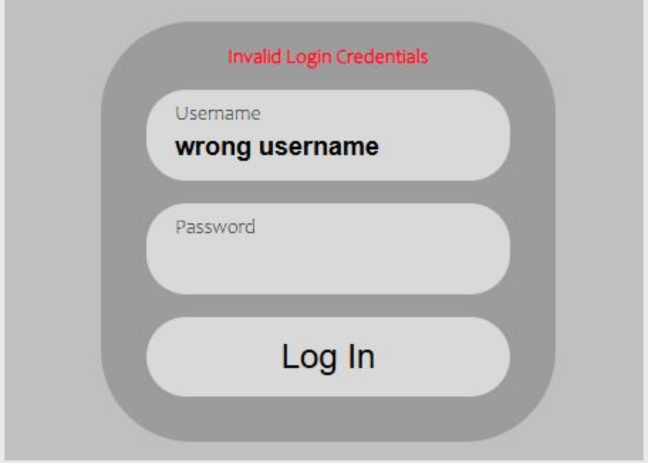
```python
@app.route("/login", methods=["GET", "POST"])
def login():
    if get_current_user():
        return redirect(url_for("home"))

    error=None
    if request.method == "POST":
        username = request.form.get("username")
        password = request.form.get("password")

        connection = get_db_connection()
        user = connection.execute(
            """
            SELECT *
            FROM USERS
            WHERE USERNAME = ? AND PASSWORD = ?
            """, (username, password)
        ).fetchone()
        connection.close()

        if user:
            session["user_id"] = user["ID"]
            return redirect(url_for("home"))
        else:
            flash("Invalid Login Credentials", "error")
            return render_template("login.html", previous_attempt=username)

    return render_template("login.html")
```



- Refactor Routing
  - Modify `routes.py`
    - Replace `models` and `run` imports with `app.database` functions
    - Add necessary `Flask` imports
    - Create `routes_startup`
      - Move routes into function
      - Update `/login` route
        - Check login status
        - Check Input from form
          - Request Credentials from form
          - Connect to database
          - Authenticate Credentials
          - Handle Credentials
          - Add `user_id` to session
      - Add `/home` route
        - Check login status
          - Get User
        - Render User's `ROLE` as Page
          - Pass user as argument
      - Add `/logout` route
        - Remove `user_id` from session



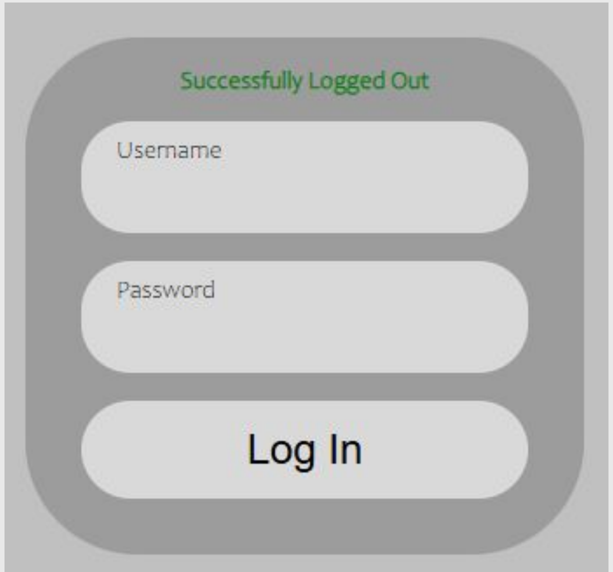
```python
@app.route("/login", methods=["GET", "POST"])
def login():
    if get_current_user():
        return redirect(url_for("home"))

    error=None
    if request.method == "POST":
        username = request.form.get("username")
        password = request.form.get("password")

        connection = get_db_connection()
        user = connection.execute(
            """
            SELECT *
            FROM USERS
            WHERE USERNAME = ? AND PASSWORD = ?
            """, (username, password)
        ).fetchone()
        connection.close()

        if user:
            session["user_id"] = user["ID"]
            return redirect(url_for("home"))
        else:
            flash("Invalid Login Credentials", "error")
            return render_template("login.html", previous_attempt=username)

    return render_template("login.html")
```



Invalid Login Credentials

Username: wrong username

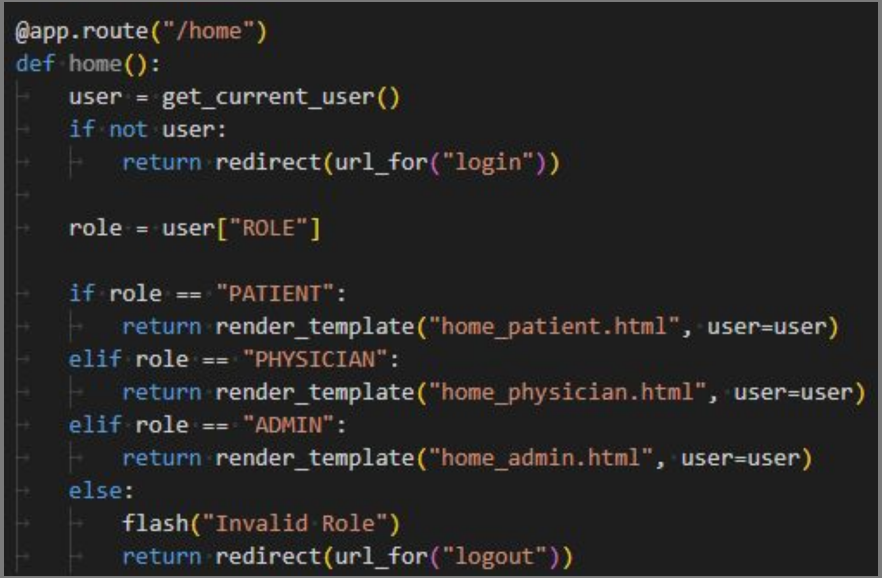
Password:



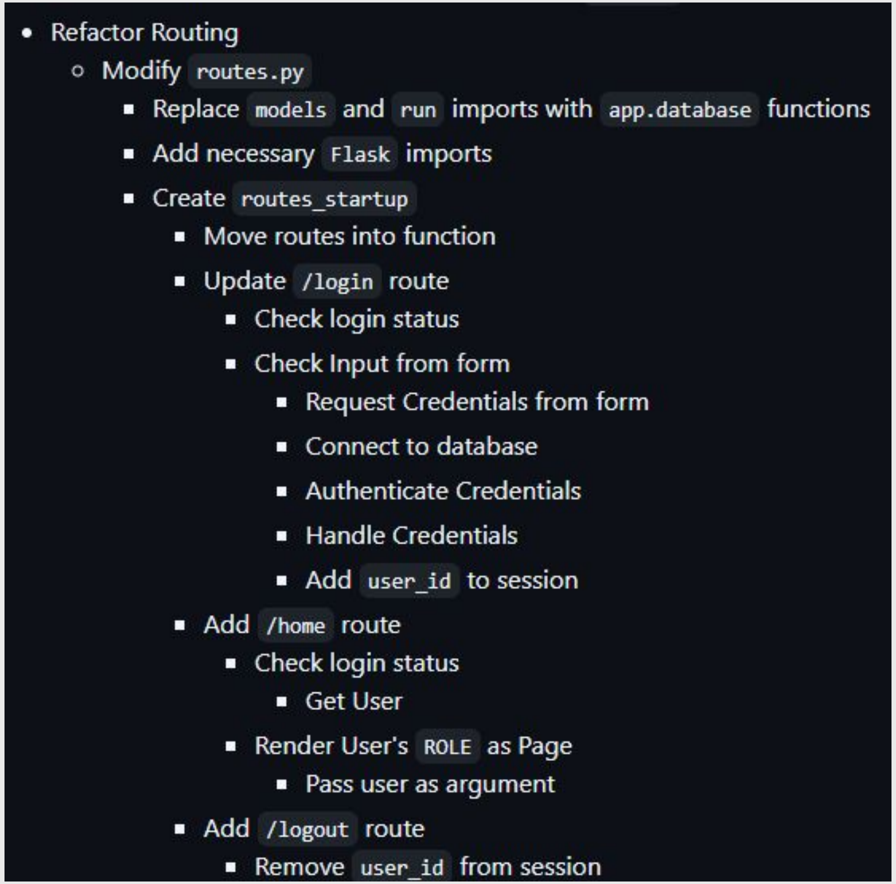
Successfully Logged Out

Username:

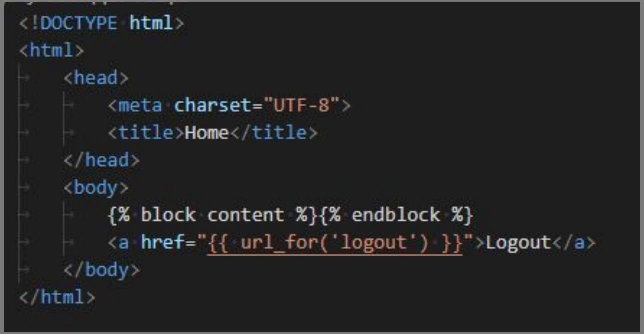
Password:



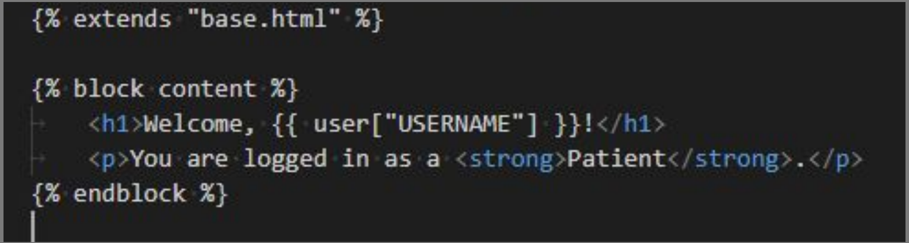
```python
@app.route("/home")
def home():
    user = get_current_user()
    if not user:
        return redirect(url_for("login"))

    role = user["ROLE"]

    if role == "PATIENT":
        return render_template("home_patient.html", user=user)
    elif role == "PHYSICIAN":
        return render_template("home_physician.html", user=user)
    elif role == "ADMIN":
        return render_template("home_admin.html", user=user)
    else:
        flash("Invalid Role")
        return redirect(url_for("logout"))
```



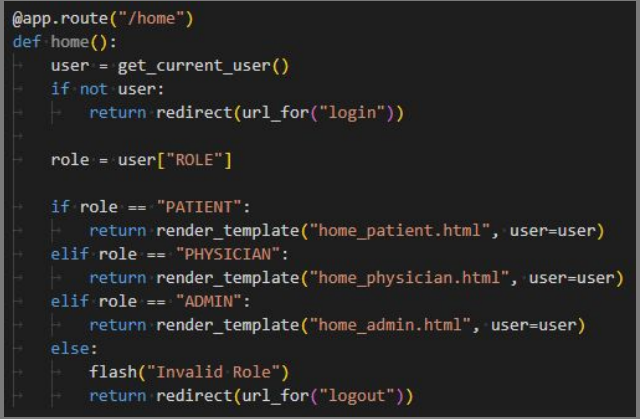
- Refactor Routing
  - Modify `routes.py`
    - Replace `models` and `run` imports with `app.database` functions
    - Add necessary `Flask` imports
    - Create `routes_startup`
      - Move routes into function
      - Update `/login` route
        - Check login status
        - Check Input from form
          - Request Credentials from form
          - Connect to database
          - Authenticate Credentials
          - Handle Credentials
          - Add `user_id` to session
      - Add `/home` route
        - Check login status
          - Get User
        - Render User's `ROLE` as Page
          - Pass user as argument
      - Add `/logout` route
        - Remove `user_id` from session



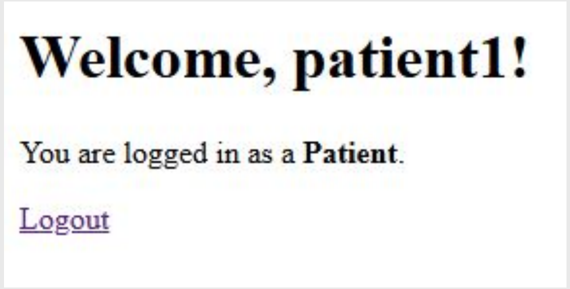
```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Home</title>
    </head>
    <body>
        {% block content %}{% endblock %}
        <a href="{{ url_for('logout') }}">Logout</a>
    </body>
</html>
```



```html
{% extends "base.html" %}

{% block content %}
    <h1>Welcome, {{ user["USERNAME"] }}!</h1>
    <p>You are logged in as a <strong>Patient</strong>.</p>
{% endblock %}
```



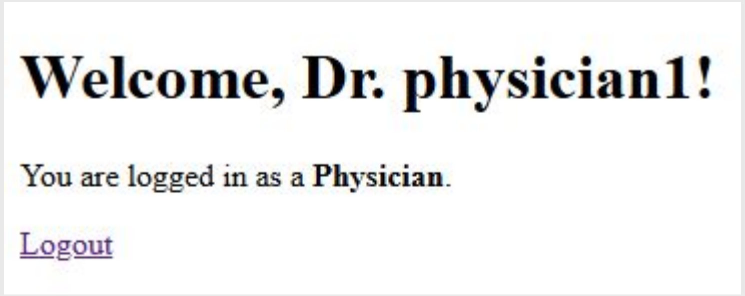
```python
@app.route("/home")
def home():
    user = get_current_user()
    if not user:
        return redirect(url_for("login"))

    role = user["ROLE"]

    if role == "PATIENT":
        return render_template("home_patient.html", user=user)
    elif role == "PHYSICIAN":
        return render_template("home_physician.html", user=user)
    elif role == "ADMIN":
        return render_template("home_admin.html", user=user)
    else:
        flash("Invalid Role")
        return redirect(url_for("logout"))
```



Welcome, patient1!

You are logged in as a **Patient**.

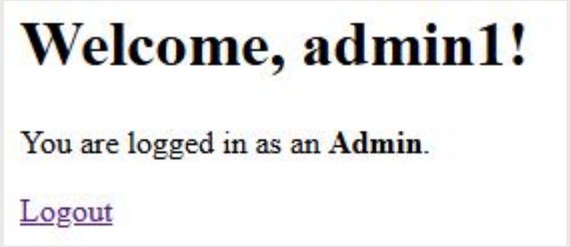
Logout



Welcome, Dr. physician1!

You are logged in as a **Physician**.

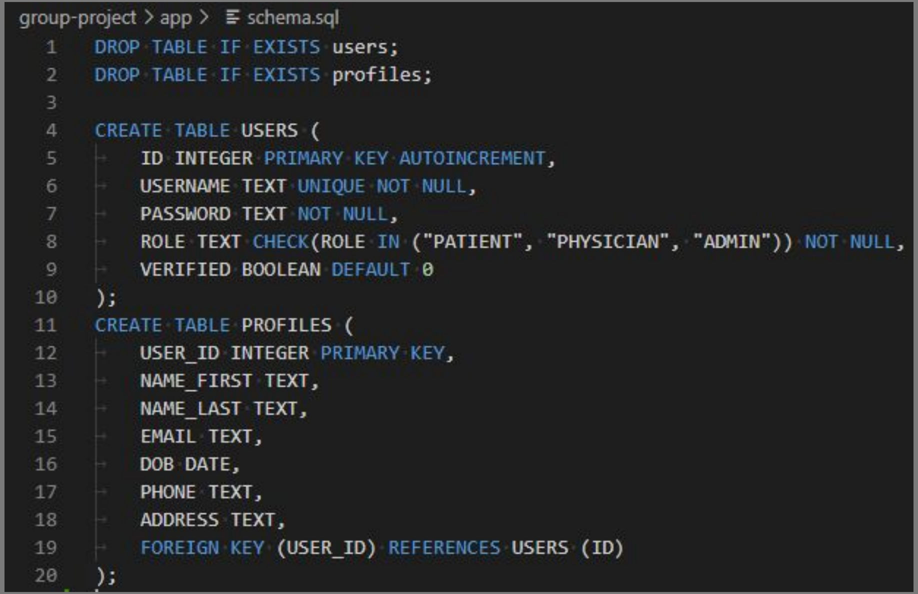
Logout



Welcome, admin1!

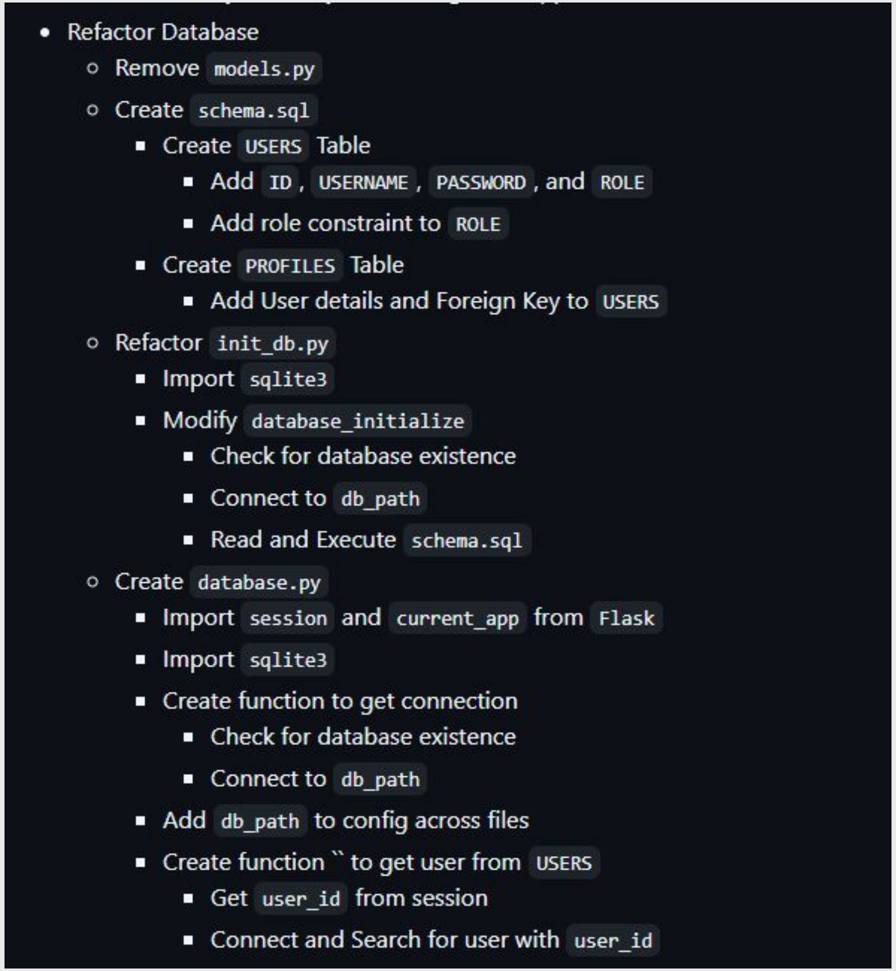
You are logged in as an **Admin**.

Logout

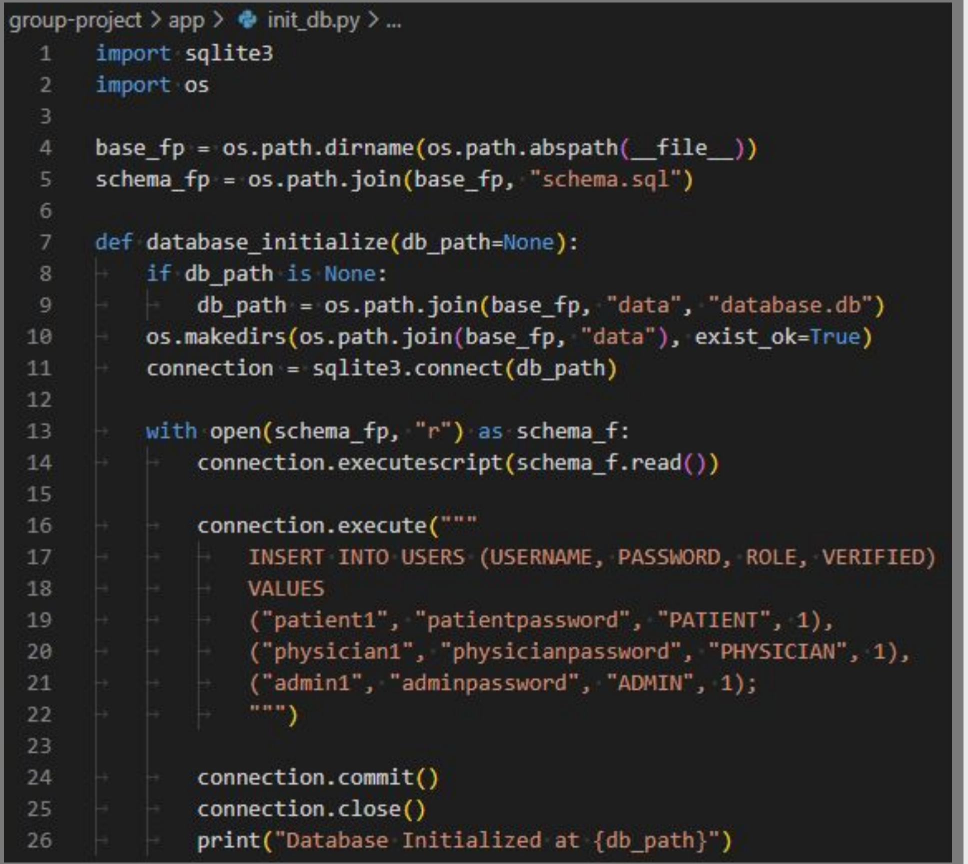


group-project > app > schema.sql

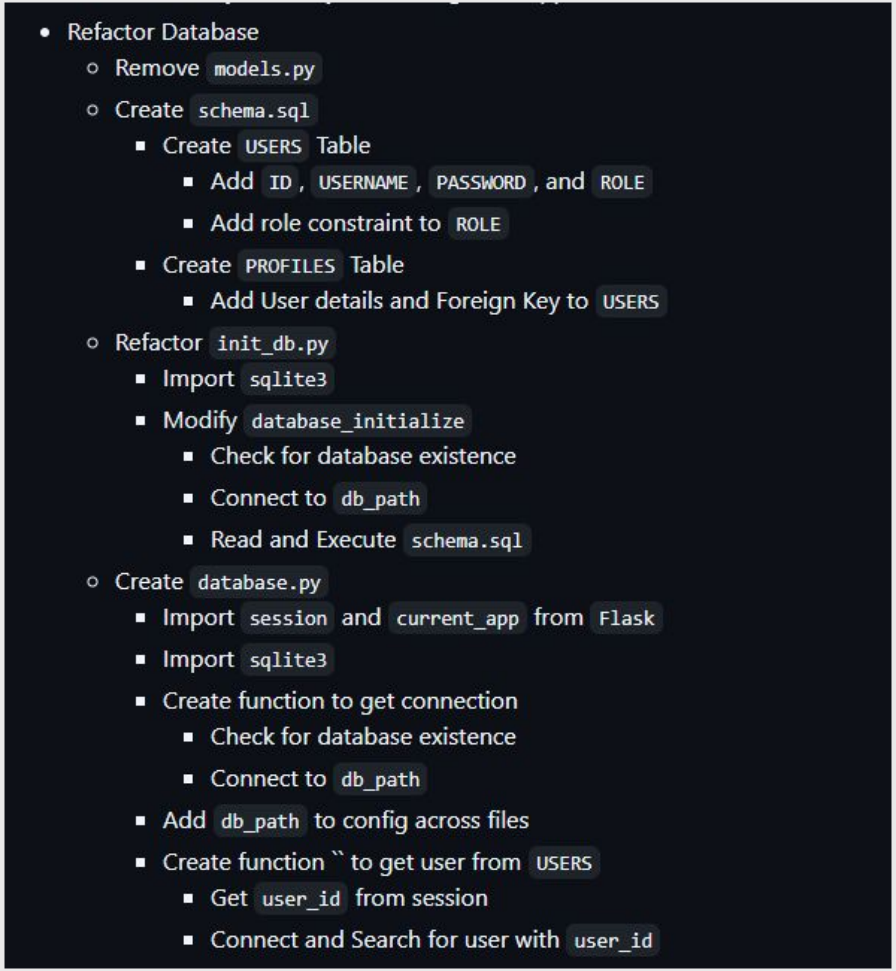
```sql
DROP TABLE IF EXISTS users;
DROP TABLE IF EXISTS profiles;

CREATE TABLE USERS (
    ID INTEGER PRIMARY KEY AUTOINCREMENT,
    USERNAME TEXT UNIQUE NOT NULL,
    PASSWORD TEXT NOT NULL,
    ROLE TEXT CHECK(ROLE IN ("PATIENT", "PHYSICIAN", "ADMIN")) NOT NULL,
    VERIFIED BOOLEAN DEFAULT 0
);
CREATE TABLE PROFILES (
    USER_ID INTEGER PRIMARY KEY,
    NAME_FIRST TEXT,
    NAME_LAST TEXT,
    EMAIL TEXT,
    DOB DATE,
    PHONE TEXT,
    ADDRESS TEXT,
    FOREIGN KEY (USER_ID) REFERENCES USERS (ID)
);
```



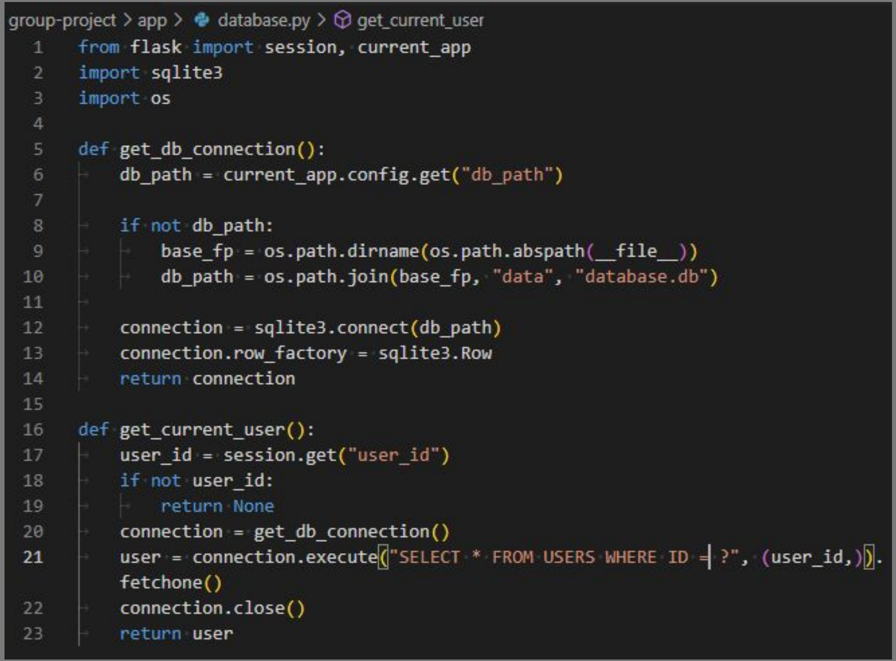
- Refactor Database
  - Remove `models.py`
  - Create `schema.sql`
    - Create `USERS` Table
      - Add `ID`, `USERNAME`, `PASSWORD`, and `ROLE`
      - Add role constraint to `ROLE`
    - Create `PROFILES` Table
      - Add User details and Foreign Key to `USERS`
  - Refactor `init_db.py`
    - Import `sqlite3`
    - Modify `database_initialize`
      - Check for database existence
      - Connect to `db_path`
      - Read and Execute `schema.sql`
  - Create `database.py`
    - Import `session` and `current_app` from `Flask`
    - Import `sqlite3`
    - Create function to get connection
      - Check for database existence
      - Connect to `db_path`
    - Add `db_path` to config across files
    - Create function `` to get user from `USERS`
      - Get `user_id` from session
      - Connect and Search for user with `user_id`


group-project > app > init_db.py > ...

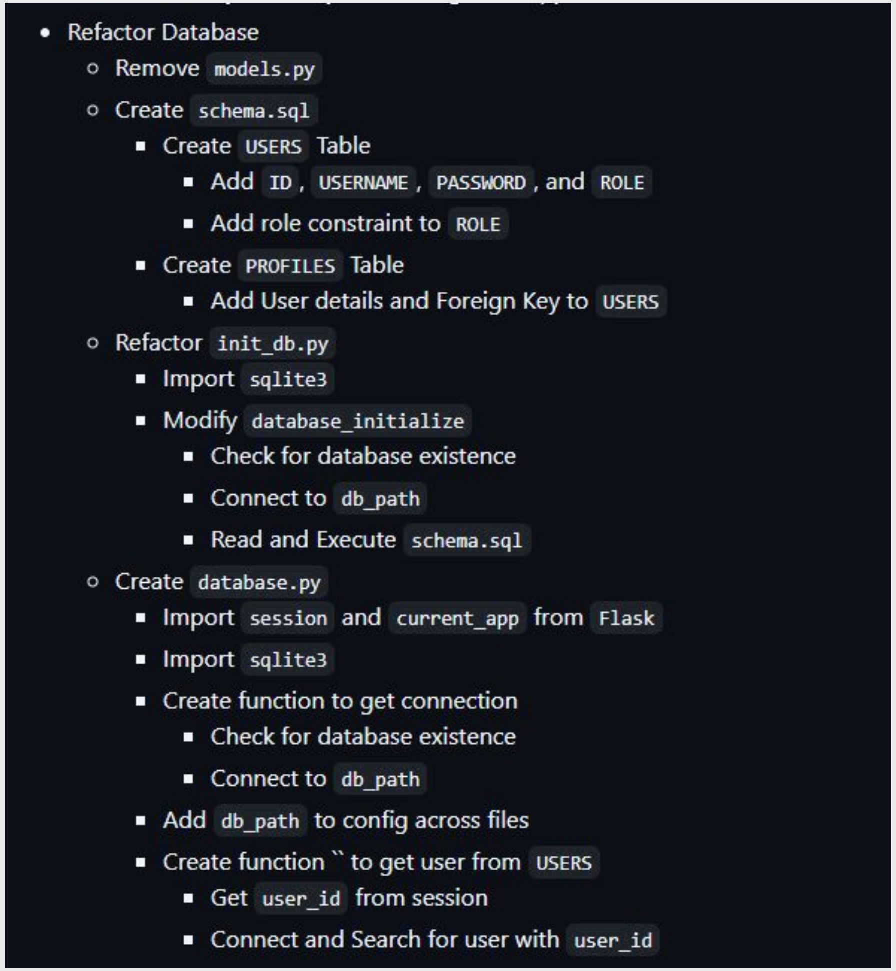
```python
import sqlite3
import os

base_fp = os.path.dirname(os.path.abspath(__file__))
schema_fp = os.path.join(base_fp, "schema.sql")

def database_initialize(db_path=None):
    if db_path is None:
        db_path = os.path.join(base_fp, "data", "database.db")
    os.makedirs(os.path.join(base_fp, "data"), exist_ok=True)
    connection = sqlite3.connect(db_path)

    with open(schema_fp, "r") as schema_f:
        connection.executescript(schema_f.read())

        connection.execute("""
            INSERT INTO USERS (USERNAME, PASSWORD, ROLE, VERIFIED)
            VALUES
            ("patient1", "patientpassword", "PATIENT", 1),
            ("physician1", "physicianpassword", "PHYSICIAN", 1),
            ("admin1", "adminpassword", "ADMIN", 1);
            """)

        connection.commit()
        connection.close()
        print("Database Initialized at {db_path}")
```



- Refactor Database
  - Remove `models.py`
  - Create `schema.sql`
    - Create `USERS` Table
      - Add `ID`, `USERNAME`, `PASSWORD`, and `ROLE`
      - Add role constraint to `ROLE`
    - Create `PROFILES` Table
      - Add User details and Foreign Key to `USERS`
  - Refactor `init_db.py`
    - Import `sqlite3`
    - Modify `database_initialize`
      - Check for database existence
      - Connect to `db_path`
      - Read and Execute `schema.sql`
  - Create `database.py`
    - Import `session` and `current_app` from `Flask`
    - Import `sqlite3`
    - Create function to get connection
      - Check for database existence
      - Connect to `db_path`
    - Add `db_path` to config across files
    - Create function `` to get user from `USERS`
      - Get `user_id` from session
      - Connect and Search for user with `user_id`


group-project > app > database.py > get_current_user

```python
from flask import session, current_app
import sqlite3
import os

def get_db_connection():
    db_path = current_app.config.get("db_path")

    if not db_path:
        base_fp = os.path.dirname(os.path.abspath(__file__))
        db_path = os.path.join(base_fp, "data", "database.db")

    connection = sqlite3.connect(db_path)
    connection.row_factory = sqlite3.Row
    return connection

def get_current_user():
    user_id = session.get("user_id")
    if not user_id:
        return None
    connection = get_db_connection()
    user = connection.execute("SELECT * FROM USERS WHERE ID = ?", (user_id,)).
    fetchone()
    connection.close()
    return user
```


- Refactor Database
  - Remove `models.py`
  - Create `schema.sql`
    - Create `USERS` Table
      - Add `ID`, `USERNAME`, `PASSWORD`, and `ROLE`
      - Add role constraint to `ROLE`
    - Create `PROFILES` Table
      - Add User details and Foreign Key to `USERS`
  - Refactor `init_db.py`
    - Import `sqlite3`
    - Modify `database_initialize`
      - Check for database existence
      - Connect to `db_path`
      - Read and Execute `schema.sql`
  - Create `database.py`
    - Import `session` and `current_app` from `Flask`
    - Import `sqlite3`
    - Create function to get connection
      - Check for database existence
      - Connect to `db_path`
    - Add `db_path` to config across files
    - Create function `` to get user from `USERS`
      - Get `user_id` from session
      - Connect and Search for user with `user_id`

Milestone 2- Next Steps

🔹 Sprint 2: Core Note-Taking Features

Goal: Users can record and manage notes.

User Tasks: Corey 1-5,Lawrence & Renze 1-4

1. As a **physician**, I want to create a new note during patient visits so that I can record symptoms.
2. As a **physician**, I want to edit a note so that I can update patient details if needed.
3. As a **physician**, I want to search past notes so that I can quickly review patient history.
4. As a **physician**, I want to delete notes so that I can remove mistakes or old information.
5. As a **developer**, I want to implement a note taking code test.

🔹 Sprint 3: Drug Reference Database Integration

Goal: Provide treatment suggestions alongside notes.

User Tasks: Corey 1-3

1. As a **physician**, I want to see a list of recommended generic drugs for common symptoms so that I can make informed treatment choices.
2. As a **physician**, I want drug information to auto-suggest when entering symptoms so that I save time.
3. As a **physician**, I want links to dosage guidelines so that I can confirm prescriptions quickly.

🔷 Sprint 4: Advanced Features

Goal: Make the app more powerful and user-friendly.

User Tasks: Corey 1-4

1. As a **physician**, I want to generate a summary of patient visits so that I can share it with patients.
2. As a **physician**, I want to tag notes with conditions (e.g., “allergy,” “diabetes”) so that I can filter them later.
3. As a **physician**, I want to export notes to PDF so that I can attach them to medical records.
4. As a **developer** I add voice activation to the note entries.

🔷 Sprint 5: Deployment & Feedback

Goal: Deliver to real users and gather feedback.

User Tasks: Corey 2 Larry 1 Larry and Corey 3,Renze 2 &3

1. As a **developer**, I want to deploy the app to a cloud service so that it's available to users.
2. As a **physician**, I want to provide feedback on usability so that the team can improve the product.
3. As a **team**, I want a retrospective meeting so that we can reflect on what worked well and what didn't.