Modern Application Development I

Project Statement

Household Services Application

It is a multi-user app (requires one admin and other service professionals/ customers) which acts as platform for providing comprehensive home servicing and solutions.

Frameworks to be used

These are the mandatory frameworks on which the project has to be built.

- Flask for application code
- Jinja2 templates + Bootstrap for HTML generation and styling
- SQLite for data storage

Note: All demos should be possible on your local machine.

Roles

The platform will have **three** roles;

1. **Admin - root access** - it is a superuser of the app and requires no registration.
    - Admin login redirects to the admin dashboard
    - Admin will monitor all the users (customers/service professionals)
    - Admin will create a new service with a base price
    - Admin will approve a service professional after verification of profile docs
    - Admin will block customer/service professionals based on fraudulent activity/poor reviews
    - Other operations*
2. **Service Professional** - An individual that provides the service
    - Login/Register
    - Service professionals will accept/reject a request
    - Each professional may have;
        - ID
        - Name
        - Date created
        - Description
        - service_type
        - Experience
        - etc.
    - One professional is good at one of the services only
    - He/she can accept/reject an assigned service request
    - Professional profiles are visible based on customer reviews
    - The professional will exit the location after the service is closed by the customer
3. **Customer** - an individual who has to book a service request
    - Login/Register
    - View/Search the service by the name/location pin code
    - Open/close a service request
    - He/she can post reviews/remarks on the closed service
    - Others

Terminologies

Service - It refers to the type of service that the customer is looking for e.g. AC servicing, plumbing etc.

Each service may have;
1. ID
2. Name
3. Price
4. Time required
5. Description etc.

Service Request - A customer creates a service request providing the type of service the customers is looking for, when is it required etc.
A service request may contain the following attributes:
1. id - primary key
2. service_id(foreign key-services table)
3. customer_id(foreign key-customer table)
4. professional_id(foreign key-professional table)
5. date_of_request
6. date_of_completion
7. service_status(requested/assigned/closed)
8. remarks (if any) etc.

Note: the above fields are not exhaustive, students can add more fields a/c to their requirements

Application Wireframe

A-Z Household Services

Note: The wireframe is provided only to get the flow of the application and what should appear when a specific user navigates from one page to another. It is NOT mandatory to exactly replicate the views given in the wireframe. Students can work on their front-end ideas.

Core Functionalities

1 Admin login and user login

- A login/register form with fields like username, password etc. for customer, service professional and admin login
- You can create separate forms for each type of user
- You can either use a proper login framework, or just use a simple HTML form with username and password (we are not concerned with how secure the login or the app is)
- The app must have a suitable model to store and differentiate all the types of user of the app.

2. Admin Dashboard - for the Admin

- Admin login redirects to admin dashboard
- Admin will manage all the users (customers/service professional)
- Admin will approve a service professional after verification of profile docs
- Admin will block customer/service professional based on fraudulent activity/poor reviews

3. Service Management - for the Admin

- Create a new service with a base price.
- Update an existing service - e.g. name, price, time_required and/or other fields
- Delete an existing service

4. Service Request - for the customers

- Create a new service request based on the services available
- Edit an existing service request - e.g. date_of_request, completion status, remarks etc
- Close an existing service request.

5. Search for available services

- The customers should be able to search for available services based on their location, name, pin code etc.
- The admin should be able to search for a professional to block/unblock/review them.

6. Take action on a particular service request - for the service professional

- Ability to view all the service requests from all the customers
- Ability to accept/reject a particular service request
- Ability to close the service request once completed*

Recommended Functionalities

- API resources created to interact with the users, service requests and/or services. (Please note: you can choose which API resources to create from the given ones. It is NOT mandatory to create API resources for CRUD of all the components)
- APIs can either be created by returning JSON from a controller or using flask extension like flask_restful
- External APIs/libraries for creating charts, e.g. ChartJS
- Implementing frontend validation on all the form fields using HTML5 form validation or JavaScript
- Implementing backend validation within the controllers of your app.

Optional Functionalities

- Provide styling and aesthetics to your application by creating a beautiful and responsive frontend using simple CSS or Bootstrap
- Incorporate a proper login system to prevent unauthorized access to the app using flask extensions like flask_login, flask_security etc.
- Implement a dummy payment portal (just a view taking payment details from sponsors for an ad request)
- Any additional feature you feel is appropriate for the application

Evaluation

- Student have to create and submit a project report (not more than 2 pages) on the portal along with the actual project submission
- The report must include the following things;
    - Student details
    - Project details, including the question statement and how you approached the problem statement
    - Frameworks and libraries used
    - ER diagram of your database, including all the tables and their relations
    - API resource endpoints (if any)
    - Drive link of the presentation video
- The project report must be included as a PDF **inside** the root submission folder and NOT along with it.

```
Project Root Folder
│
└── Code
    ├── Folder 1
    │   ├── file3.py
    │   └── file4.py
    ├── Folder 2
    │   ├── Folder 3
    │   │   ├── file7.py
    │   │   └── file8.py
    │   ├── file5.py
    │   └── file6.py
    ├── file1.py
    ├── file2.py
    └── Project Report.pdf
```

- All code to be submitted on portal in a single zip file (zipping instructions are given in project document - Project Doc T22024
- Students have to create a brief (5-10 minute) video explaining how you approached the problem, what you have implemented, and any extra features
- The video must be uploaded on the student drive with **access to anyone with link** and the link must be included in the report
    - This will be viewed during or before the viva, so should be a clear explanation of your work
- Viva: after the video explanation, you are required to give a demo of your work, and answer any questions that the examiner asks
    - This includes making changes as requested and running the code for a live demo
    - Other questions that may be unrelated to the project itself but are relevant for the course

Instructions

- This is a live document and will be updated with more details (wireframe)
- We will freeze the problem statement on or before 19th Sept 2024, beyond which any modifications to the statement will be communicated via proper announcements.
- The project has to be submitted as a single zip file.