Modern Application Development-I Project Report

Project title : HouseKart

Author

Name : Harsh Narendra Patil | Roll No. : 21f1002234 | Email : 21f1002234@ds.study.iitm.ac.in
Student at IIT Madras's BS in Data Science and Application, currently in Diploma level.
Jalgaon, Maharashtra

Project Description

- HouseKart is an app that acts as a platform for both customers, looking for any household services and for professionals willing to provide services.
- Apart from customers and professionals, there is a role for admin who monitor each and every activity on HouseKart and deny any user if found fraudulent.
- Admins create service categories, professionals offer services under them, and customers request services, which professionals can accept or reject.

Video link of project : MAD-I Video.mp4

Modules Used

1. **Flask**: The core web framework for building your application. It helps handle routing, HTTP requests, and responses.
2. **render_template**: Renders HTML templates, allowing dynamic content to be inserted into web pages.
3. **url_for**: Generates URLs for Flask routes dynamically.
4. **flash**: Displays messages to users, typically used for feedback (like success or error messages).
5. **redirect**: Redirects users to a different route (URL) after an action, like form submission.
6. **request**: Handles incoming request data (like form submissions and query parameters).
7. **session**: Stores user-specific data during a session (e.g., logged-in user information).
8. **Response**: Customizes the HTTP response sent back to the client.
9. **send_from_directory**: Serves files from a specific directory, useful for downloading files or serving static content.
10. **abort**: Raises an HTTP error (e.g., 404 or 403) when something goes wrong in the request handling.
11. **SQLAlchemy**: An Object-Relational Mapping (ORM) library for interacting with the database using Python classes and objects.
12. **func**: Part of SQLAlchemy used for executing database functions like COUNT, AVG, etc.
13. **and_**: A SQLAlchemy helper for combining multiple conditions in queries (e.g., "AND" logic in SQL).
14. **datetime**: Used for handling dates and times, useful for time-related queries or displaying current date/time.
15. **FlaskForm**: Base class for defining web forms in Flask using WTForms.
16. **FileField**: Field type in WTForms for file uploads.
17. **FileAllowed**: Validator to restrict file uploads to certain types (e.g., only allowing images).
18. **StringField**, **PasswordField**, **SelectField**, **SubmitField**, **EmailField**, **TextAreaField**, **IntegerField**: Various field types used in WTForms for different types of input (text, password, selection, email, etc.).
19. **validators**: Collection of built-in validation functions (e.g., for checking if a value is required, valid email, within a number range, etc.).

20. **Flask-Login**: Provides user session management, handling login, logout, and protecting routes that require authentication.
21. **login_user**, **logout_user**, **login_required**, **current_user**: Functions to log users in and out, check the logged-in user, and protect routes.
22. **LoginManager**: Manages the login process and session, ensuring users are properly authenticated.
23. **UserMixin**: A helper class to make it easier to manage user authentication (provides default implementations of methods
24. **os**: Provides a way to interact with the operating system, like managing file paths or creating directories.

DB Schema Design

The database has seven tables: Customer, Professional, User, Service, ServiceRequest, RejectedServiceRequest, and ServiceReview.

- User: Stores login details and is linked to Customer and Professional when they are created.
- Customer and Professional: Hold details about users and professionals, which are used in ServiceRequest to log service requests.
- ServiceRequest: Tracks requests made by customers to professionals.
- ServiceReview: Logs reviews for services, linked to data from ServiceRequest.
- RejectedServiceRequest: Records requests declined by professionals.
- Relationships are maintained using primary keys (PK) and foreign keys (FK).

ER Diagram of Database