Cafe project

Maktab 80

Project overview:

Technical Requirements:

- Python language
- Flask: Webserver
- Git: Version control
- PostgreSQL: Database
- HTML, CSS, JS: Front-end
- Bootstrap CSS framework

Description:

Create a Cafe website project with two major functionalities:

A. **Landing page**:

a Public page of the cafe website that contains some sub-pages for customers view:

1. **Home:** Designed to attract customers. Must have keypoint information about the cafe and have good Graphic design
2. **About:** This section presents full information about the cafe, website, staff, ...
3. **Contact us:** contains Geographical Information, Address, Phone number, and other contact information.
   Also, have a form to interact with users from the website.
   (optional) Add a map view on the contact page.
4. **Menu**: List of products and foods to be served by the Cafe.
   The Menu section must contain:
   - **Products** along with their **prices**
   - **Order** button, to order.
5. **Orders list:** Show a list of customer orders along with their status.
   e.g: Waiting, Cooking, Served, ...

B. **Cashier Panel**:

This panel is designed for **Cashier**s, to manage customer orders.
the Cashier Panel must include features like:

1. **Dashboard:** provides shortcuts and important information for the cashier. Like: Received messages, Recent Orders, ...

2. **Orders:** List of orders besides their status and action buttons.
   You can change the status of orders by doing actions on them like **Sending** them to the kitchen, **Serving** them, and ...

   You should have several sub-pages fitting every status of orders.

   - **New orders**: List of new orders from customers.
   - **Cooking orders**: List of orders that are sent to the kitchen.
   - **Served orders**: List of recent orders that are served to the customer.
   - **Deleted Orders**: List of rejected orders by the cashier.
   - **Paid Orders**: List of paid orders.
   - **Archive:** a complete history of customer orders.

   *** Use bootstrap modals to see details of orders**
   *** Each row of orders must have action buttons to change the status of the order.**

3. **Tables:** Map of the cafe tables along with their current orders.
   The cashier can see the order list of each table by clicking on them and doing some actions on them. They can also export **receipts** for each table from this page.

4. **Menu Items:** Manage Items (foods) that are served by the cafe.
   The cashier can Add, Edit or Delete every product from this page.
   - **Add Items:** This section allows the cashier to add new items (foods) to the menu by their details like Name, Price, Category, Description, Discount*(Optional)*, ...
   - **Edit Items:** Change products prices, set discounts for each product, add descriptions for them, and ...
   - **Delete Items:** Delete Items from the menu.
5. **Receipts:** List of exported receipts from the system.
   The cashier should have access to see, report and manage receipts.
   - **Receipt:** receipt details page.
     - **(optional):** implement **@media print**
   - **Paid Receipts:** paid receipts by customers.
   - **Archive:** History of receipts.
   - ...

6. **Charts (Optional):** Statistical charts containing information like:
   - Number of Orders/Hour
   - Number of Orders/Weekdays
   - Order Categories/Hour
   - Order Categories/Weekdays
   - ...

Phase 1#:

In this phase you should analyze your business first, then you can initialize your project by designing structures, basics, and architecture.

Tasks

1. Git Configuration:

First, Create a **private** Git repository on Github.com.

Git name: Maktab78_Cafe_project.

Invite your mentors to collaborate.

2. ERD:

You should store data using Relational Databases management systems into a **PostgreSQL** database.
In the first step, you should analyze your business entities and Design your **ERD.**

Cafe project Entities:

- **Users:**
  - Id
  - First name
  - Last name
  - Phone number
  - Email
  - Password to login with
  - Extra information such as birthday
  - ...
- **Tables:**
  - Id
  - Table number
  - Cafe space position
  - ...
- **MenuItems:**
  - Id
  - Name
  - Price
  - Category
  - Discount (Optional)
  - Serving time period (Optional)
  - Estimated cooking time (Optional)

- **Orders:**
    - Id
    - Table
    - Menu Items (food)
    - Number
    - Status
    - Timestamp
    - ...
- *** Receipts:**
    - Id
    - Orders
    - Total price
    - Final price (with Discount)
    - Timestamp,
    - ...
- (Any other needed models)

Note:

- Analyze your project and extend your ERD fitting the business requirements.
- Do entity extraction if needed: Extracting some attributes from current entities and making them a standalone Entity.
- **Category Tree (Optional):** You can create a hierarchical category database structure, using self-relations in SQL databases. Read about it.

After analyzing the database and designing ERD, implement them into your **PostgreSQL** database.

MAKE SURE YOUR ERD MEETS YOUR PROJECT'S REQUIREMENTS BEFORE STARTING ANYTHING!

3. Database:

Create a database schema based on the analyzes performed in the previous task.
Create and initialize tables and relations between entities into a **PostgreSQL** database.

Notice that constantly changing the database schema may make your system **buggy** easily.

4. OOP Models:

Design and implement your models' classes based on your database entities.
After implementing models, Use **psycopg2** to store and fetch data from the database.
You should provide a **DatabaseManager** to connect and interact with your database. Two options you have:

1. Developing a **DatabaseManager** using Psycopg to map models on database entities and provide four basic **CRUD** (Create, Read, Update, Delete) functions.
2. Utilizing and benefiting a database **ORM**, like **SQLAlchemy**.

5. Front-End:

- Think about all your web pages **User Interface (UI)** design and the sense of **User Experiences (UX)** interacting with your website.
  Seeing some cafe templates like this will help you by getting inspiration.

- the Landing page must be well-designed and customer-attracting.
- The cashier admin page must be panel formed like this.
- All web pages must be **Fully-Responsive.**
- Use Bootstrap modals and other utilities (Tooltip, Toast, Inputs, ...)
- **(Extra)** Design a bilingual (English, Persian) landing page.

6. Flask configuration:

First, create your Virtual Environment into the project directory.
Second, install the latest version of **Flask** using pip, and initialize your flask application.
Finally, Analyze the project architecture, packages, modules, and views you need, according to your business requirements.

Packaging example:

1. core
2. landing
3. cashier
4. orders
5. ...

7. Landing page:

Create your Landing HTML page and develop needed view functions, like:

- **GET /**: Show the Landing page
- **GET /menu**: Show the cafe's menu
- **POST /order/{table_id}**: Create a new order
- **GET /order/{table_id}**: Show the table's orders
- **DELETE /order/{order_id}**: Removing a special order

Use **Jinja2** template engine to pass and handle data into the templates from views.

You may create models, get models, update them, and do other actions through the flask views according to your business.

Note: In this phase, you can develop a simple and minimal front-end design, and have a much focus on view functions instead.

Submission:

- The deadline for submission of each phase of the project is **one week.**
- Git commits' timestamp will be considered as submission time.
- All group members must have **equal responsibilities** in developing the project since each member is graded individually and according to their contribution on Github. Individualism and Passiveness would affect the members' individual grades despite an overall perfectness.

  Be sharp in apportioning responsibilities and dedicate enough time for it at the beginning of each phase.