Oluwatobi Sobola

201316888

Flask Web App Design Document

Introduction

This web application aims to produce a food company e-commerce website. The web application utilises a database and implements feature which allows admin to add and edit products to a database and the product model field contains; a name, a price, discount, the category, the brand, the stock, allergies, a description, images of the product. The admin can add and edit brands and categories.

The database has two database model one for the admin and other for customer. Both admin and customer database model contain; username, email and password. The customer database model should also contain address of the customer. The customer should be able to add and edit their order to a database.

Navigation bar and search bar was implemented to make navigation through the pages easier.

The web application will be using Bootstrap, jQuery and Javascript.

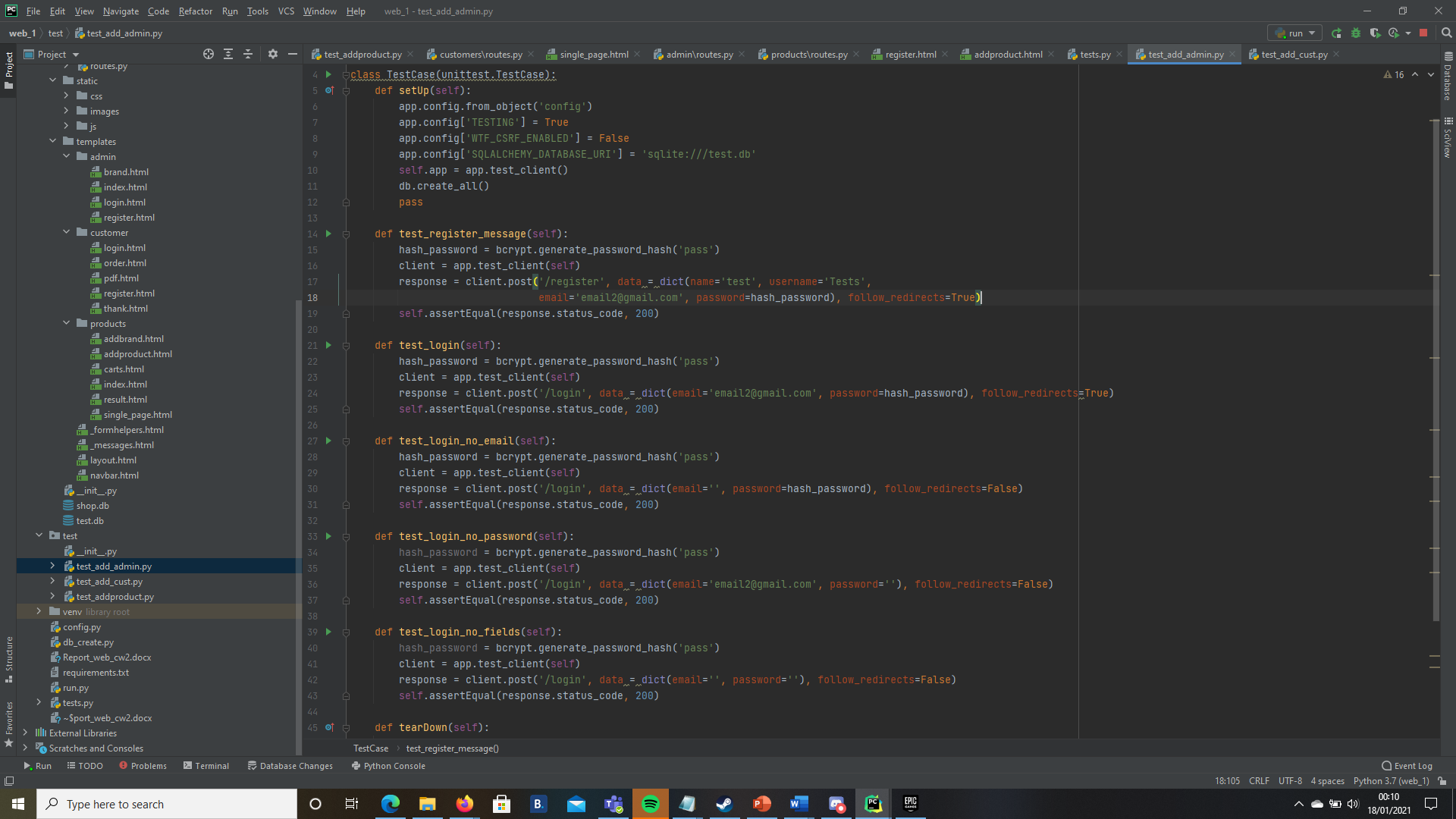
Testing

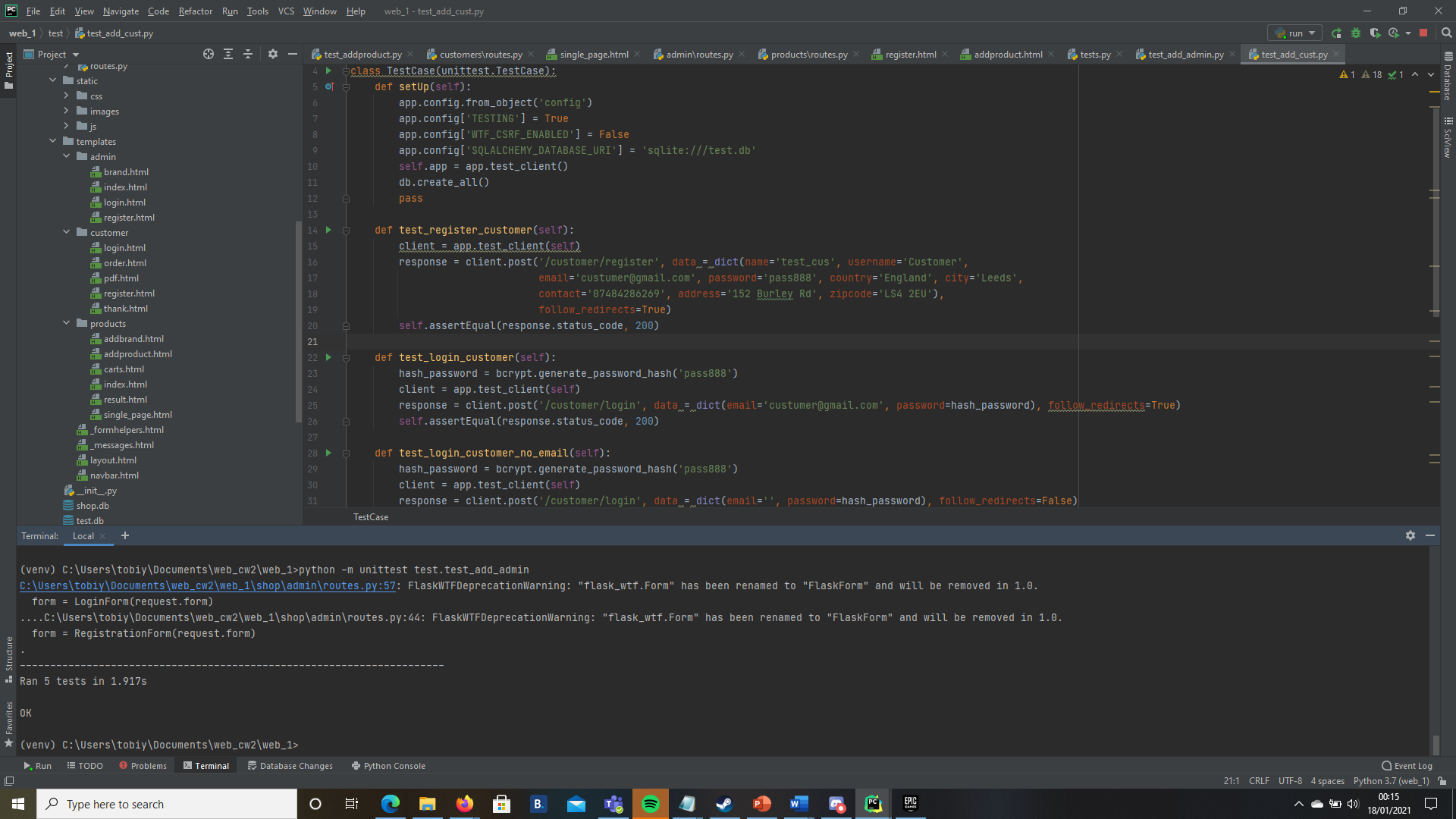
Task list application specification

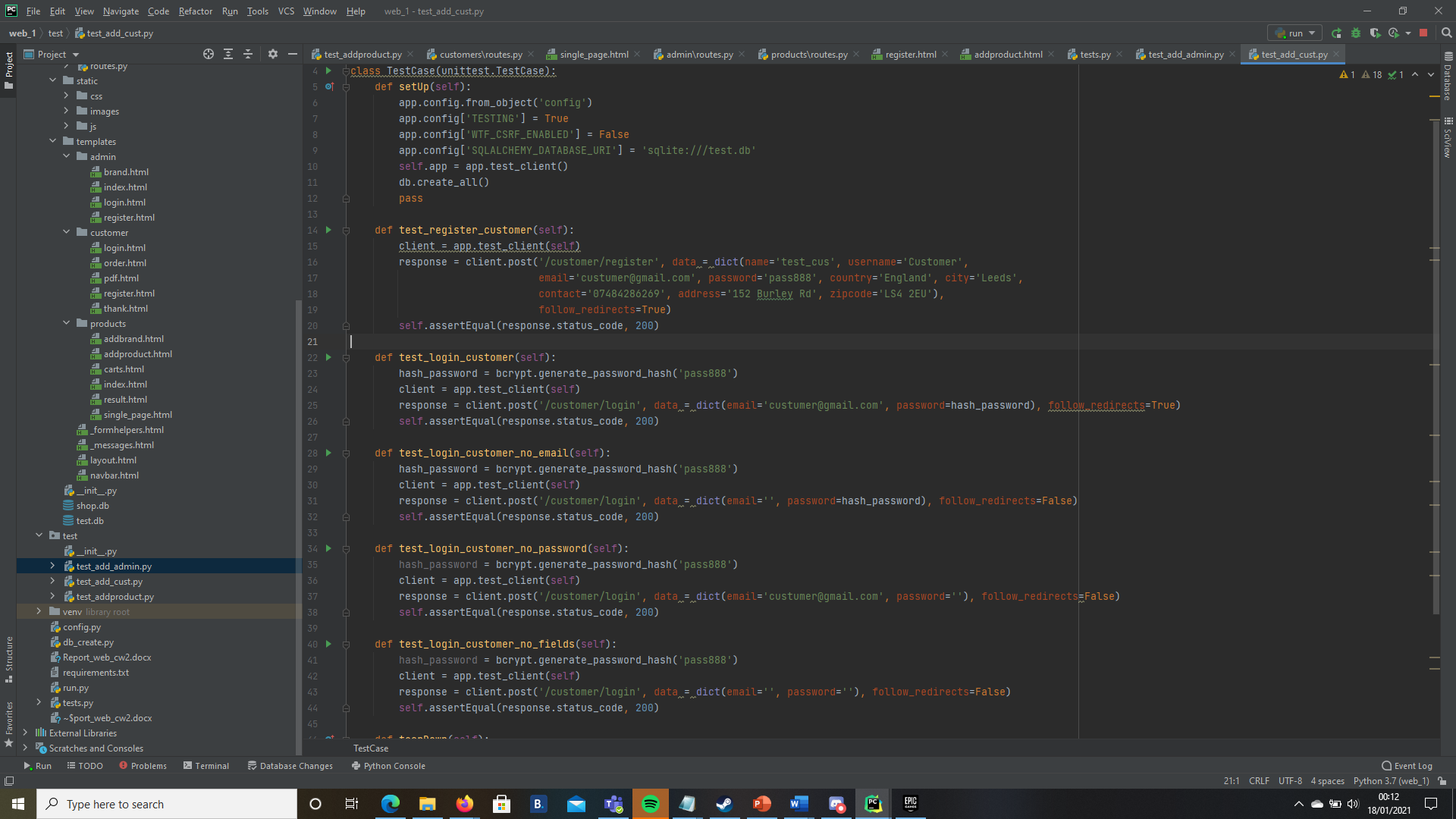
The Task list application should facilitate the following functionality.

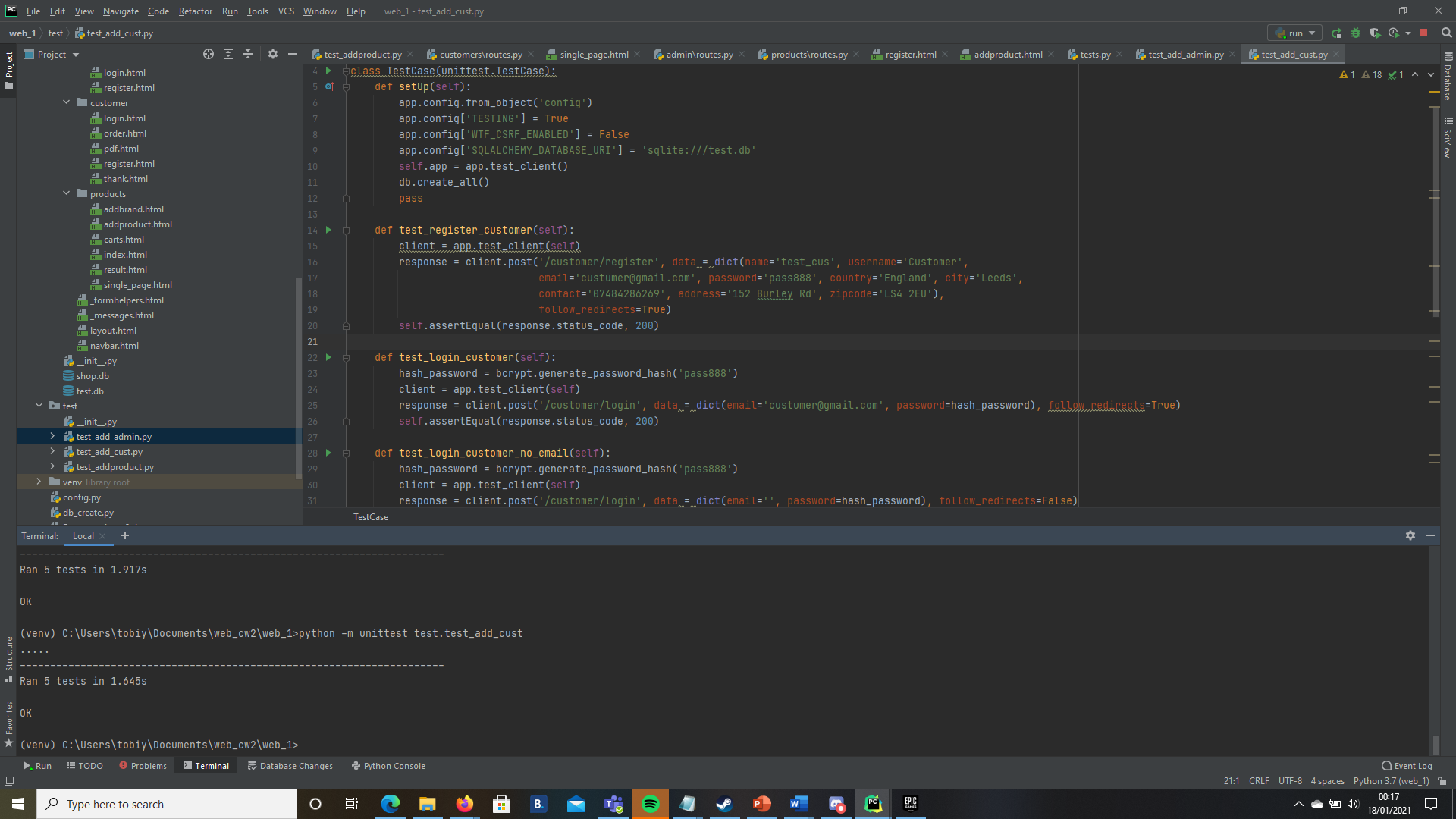
1. To be able to create new admin/ manger accounts
2. To be able to log in as an admin and go to the admin page
3. To be able to create new products through the interface
4. To be able to create new customer accounts
5. To be able to log in as a customer and go to the home page
6. To list all products in home page
7. To able to add products to order list
8. To able view order list

|  |  |  |  |
| --- | --- | --- | --- |
| Specification | Method | Description | Expected |
| 1 | Create a new admin account with all populated fields |  | A new admin account is created |
| 1 | Create a new admin account without an email |  | No new admin account should be created |
| 1 | Create a new admin account without password |  | No new admin account should be created |
| 1 | Create a new admin account without any fields |  | No new admin account should be created |
| 2 | Login with admin’s email and password |  | A login page should be redirected to admin page |
| 2 | Login with wrong admin’s email |  | The login page should not be redirected to admin page |
| 2 | Login with wrong admin’s password |  | The login page should not be redirected to admin page |
| 3 | Create a new product with all populated fields | Create a new brand and category | A new product is created |
| 3 | Create a new product without a brand | Create a new category | No new product should be created |
| 3 | Create a new product without a category | Create a new brand | No new product should be created |
| 3 | Create a new product without any fields |  | No new product should be created |
| 4 | Create a new customer account with all populated fields |  | A new customer account is created |
| 4 | Create a new customer account without an email |  | No new customer account should be created |
| 4 | Create a new customer account without password |  | No new customer account should be created |
| 4 | Create a new customer account without any fields |  | No new customer account should be created |
| 5 | Login with customer’s email and password |  | A login page should be redirected to customer page |
| 5 | Login with wrong customer’s email |  | The login page should not be redirected to customer page |
| 5 | Login with wrong customer’s password |  | The login page should not be redirected to customer page |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |









Analysis

Time

Evaluation

The web development app made to be straightforward and easy to use and to be accessible to a diverse set of people. The SQLite was used to have a light framework and organised the data types in the correct files. SQLite helped the project to implement a feature such as add, delete, and update.

Another feature implemented was the validation feature. To ensure that each date field type was filled and formatted correctly and not empty. This feature stops invalid forms submitted, which are is a potential entry of attack.

To ensure people with disabilities able to use the web application. One of the solutions to accessibility options is to provide the website is straightforward and easy to understand English, with simple fonts and basic colouring. This help people who have reading issues (such as dyslexia) and people who find it hard to process information (such as autism). Accessibility with people with colour -blindness, the web application is not colour dependent. Webpages are keyboard navigable, which will help those who cannot use mouse and keyboard to use the web application.

The layout is easy and straightforward to ensure the interface is not cluttered with button and visual items and is simple to navigate through the web application. Each page has a title to help the user instantly know what page is about, from the big title top left of the page. Each page has navigation bar to improve interactions with users, and to make it easier for the user to move around the website pages.

The home page shows products at the bottom of the page. There is a cart function that allows users to add, edit and delete their order as the order being displayed in cart. A search bar was implemented in home page navigation bar to allow customers to find a product there are looking for quickly.

The admin page shows products at the bottom of the page. There is a function that allows admin to add, edit and delete products as the products being displayed.

Bootstrap, jQuery and Javasript has been incorporated to make a responsive web application.

To help keep track of which user logged in, sessions was used. Sessions also helps personalise the user experience.

Security

To prevent Cross-Site Request Forgery (CSRF) attacks, the web application used WTForm’s secret key. This secret key used to send and decrypt a CSRF token which is sent to the server each time a form rendered. If that token not returned, then an error (422) occurs.

Flask-Login was also used to help prevent users’ sessions from being stolen.

Flask-Login module gives access control. It provides user session management for Flask: logging in, logging out, and remembering session. The module stores the user ID, restricts views to logged in users, protects cookies.

Another feature implemented was the validation feature. To ensure that each date field type was filled and formatted correctly and not empty. This feature stops invalid forms submitted, which are is a potential entry of attack.

Brcypt helps migrate CSRF attacks. By using brcypt to hash passwords, ensures data is encrypted

Used:

flask-login

brcypt

session

login\_user