Ecommerce Project

This is a simple ecommerce project using Flask, SQLAlchemy, and Stripe.

**Directory Structure**

ecommerce\_project

├── add\_products.py

├── app

│ ├── config.py

│ ├── forms.py

│ ├── \_\_init\_\_.py

│ ├── models.py

│ ├── static

│ │ └── img

│ ├── templates

│ │ ├── 2fa.html

│ │ ├── base.html

│ │ ├── cart.html

│ │ ├── checkout.html

│ │ ├── enable\_2fa.html

│ │ ├── index.html

│ │ ├── login.html

│ │ ├── product.html

│ │ ├── products.html

│ │ ├── register.html

│ │ ├── success.html

│ │ └── test\_checkout.html

│ ├── utils.py

│ └── views.py

├── requirements.txt

└── run.py

**Installation**

1. Install the dependencies:

*pip install -r requirements.txt*

1. Run the following commands in this order to initialize everything from the root directory i.e. */ecommerce\_project*

*flask db init*

*flask db migrate*

*flask db upgrade*

*flask db run*

**Files Overview**

**The application is organized as follows:**

* **- `static` directory**: Contains static files, such as images
* **- `templates` directory**: Contains the HTML templates for each page
  + **2fa.html:** This template displays the two-factor authentication (2FA) page, where the user enters their one-time password (OTP) after logging in. The form includes a CSRF token for security and an input field for the OTP, as well as a submit button.
  + **base.html**: This template serves as the foundation for all other templates in the application. It includes the basic HTML structure, a Bootstrap CSS and JavaScript files for styling, and a navbar. The navbar contains links to the homepage, products, and cart pages, as well as the registration, login, and logout pages depending on the user's authentication status. The main content area is populated by blocks from child templates. Additionally, it displays flashed messages from the application when necessary.
  + **cart.html**: This template displays the user's shopping cart, showing a table of products, their descriptions, prices, and quantities. Users can update the quantity of items in their cart, remove items, and place an order. The form has separate form actions for placing an order, updating the cart, and removing items, each of which triggers the corresponding route in views.py.
  + **enable\_2fa.html**: This template displays a page for enabling two-factor authentication (2FA) for the user. It shows the user's OTP secret and a QR code that can be scanned with an authenticator app. The user scans the QR code, generates an OTP, and verifies it on the 2FA page. There's a form with a submit button that takes the user to the 2FA page for verification.
  + **index.html:** This template displays the homepage of the e-commerce website. It extends the base.html template and contains a jumbotron with a welcome message and a brief description of the site.
  + **login.html:** This template displays the login form for users. It includes fields for entering an email address and a password. The form extends the base.html template and has a title set to "Login". The form contains CSRF protection with a hidden tag, and the email and password fields are rendered with the "form-control" class for styling.
  + **checkout.html**: This template is used for the checkout process, allowing users to enter their credit or debit card information and submit the payment. It includes a form with a Stripe card-element to capture card information securely, and a hidden field to store the total amount to be charged. JavaScript is included to handle Stripe integration, fetching the client secret, confirming the card payment, and handling errors or redirecting to a success page upon successful payment.
  + **product.html**: This template displays a single product's details, including an image, name, description, and price. It extends the base.html template and has a title set to "Product". A form is included to add the product to the cart by submitting a request to the 'views.add\_to\_cart' route with the product's ID.
  + **products.html**: This template displays a list of products, including the product's name, image, and price. It extends the base.html template and has a title set to "Products". Each product includes a "View Product" button that links to the individual product page using the 'views.product' route and the product's ID.
  + **register.html**: This template displays the registration form for new users. It includes fields for entering a username, email address, password, and a confirmation password. The form extends the base.html template and has a title set to "Register". The form contains CSRF protection with a hidden tag, and the fields are rendered with the "form-control" class for styling. A reCAPTCHA field is also included for added security.
  + **success.html**: This template is displayed after a successful purchase has been made. It extends the **base.html** template and contains a message thanking the user for their purchase. The message also informs the user that their order has been received and is being processed, and that they will receive a confirmation email with tracking information shortly. It encourages the user to contact the store if they have any questions or concerns. The content is placed within the **{% block content %}** block.

These templates work together to create the user interface for the e-commerce application, allowing users to navigate between pages, interact with the site, and perform actions such as registration, login, browsing products, adding products to their cart, and placing orders.

* **run.py**: The entry point of the application. It initializes the Flask app and runs it in debug mode.
* **requirements.txt**: Contains the list of required Python packages for the project.
* **add\_products.py**: A script to add products to the database.
* **app** directory: Contains the main application code.
  + **config.py**: **:** Handles the application's configuration settings.
  + **forms.py**: Defines Flask-WTF forms used in the application.
  + **\_\_init\_\_.py**: Initializes the application and its components
  + **models.py**: Defines the SQLAlchemy database models.
  + **static**: Contains static files, such as images.
  + **templates**: Contains the Jinja2 templates for the application.
  + **utils.py**: Contains utility functions for handling two-factor authentication
  + **views.py**: Contains the route handlers for the application.

**app/\_\_init\_\_.py**

This file initializes the Flask application, SQLAlchemy database, Flask-Login manager, and Flask-Migrate instance. It also registers the views blueprint to associate the application's views with the main app.

**app/config.py**

This file sets up the application's configuration, including environment variables, database settings, and API keys. It uses the `dotenv` package to load environment variables from the `.env` file.

**app/forms.py**

This file defines various forms used throughout the application, including:

* `**RegistrationForm**`: A form for user registration, with fields for the username, email, password, and a ReCAPTCHA field for security.
* `**LoginForm**`: A form for user login, with fields for email, password, and a "Remember Me" checkbox.
* `**TwoFactorForm**`: A form for two-factor authentication, with a field for the OTP (One-Time Password).
* `**UpdateCartItemForm**`: A form for updating the quantity of an item in the cart.
* `**CartForm**`: A form with buttons for updating the cart or placing an order.

**app/models.py**

This file defines the database models used in the application:

* + `**User**`: Represents a registered user, with fields for username, email, password hash, and OTP secret for two-factor authentication. This model also includes methods for managing the user's shopping cart and placing orders.
  + `**Product**`: Represents a product available for purchase, with fields for name, price, description, and image.
  + `**Order**`: Represents an order placed by a user, with fields for user ID, product ID, quantity, and a flag to indicate if the order has been placed.
  + `**CartItem**`: Represents an item in a user's shopping cart, with fields for user ID, order ID, and quantity.
  + `**OrderItem**`: Represents an item in an order, with fields for order ID, product ID, quantity, and price.

**app/utils.py**

This file contains utility functions for handling two-factor authentication, including generating an OTP secret for a user and verifying an OTP for a given user.

**app/views.py**

Below is a summary of each route:

* **@bp.route('/'):** The index route renders the homepage (index.html).
* **@bp.route('/register', methods=['GET', 'POST']):** This route handles user registration. When the registration form is submitted and validated, a new user is created and added to the database.
* **@bp.route('/login', methods=['GET', 'POST']):** This route handles user login. If the email and password provided match a user in the database, the user is logged in. If the user has 2FA enabled, they are redirected to the 2FA page.
* **@bp.route('/logout'):** This route logs out the current user and redirects them to the homepage.
* **@bp.route('/products'):** This route displays all products in the products.html template.
* **@bp.route('/product/<int:product\_id>'**): This route displays a specific product with the given product\_id in the product.html template.
* **@bp.route('/two\_factor\_auth', methods=['GET', 'POST']):** This route handles the 2FA process. If the submitted OTP is valid, the user is redirected to the homepage.
* **@bp.route('/enable\_two\_factor\_auth'):** This route generates a QR code for the user to enable 2FA with an authenticator app.
* **@bp.route('/create-payment-intent', methods=['POST'**]): This route creates a payment intent using the Stripe API based on the provided amount.
* **@bp.route('/checkout'):** This route displays the checkout page, where the user can enter their payment details and pay using Stripe.
* **@bp.route('/add-to-cart/<int:product\_id>', methods=['POST']):** This route adds a product to the user's cart and updates the quantity if the product is already in the cart.
* **@bp.route('/cart', methods=['GET', 'POST']):** This route displays the user's cart and allows them to update the quantity of each item.
* **@bp.route('/remove-from-cart/<int:product\_id>', methods=['POST']):** This route removes a product from the user's cart.
* **@bp.route('/place-order', methods=['POST']):** This route handles order placement. If the order is successfully placed, the user is redirected to the checkout page.
* **@bp.route('/success'):** This route displays the success page if the user has recently placed an order.

These route handlers define the functionality of the web application, making it possible for users to perform actions such as registration, login, browsing products, adding products to their cart, and placing orders.

**Top of Form**

**Application Features**

* User registration and login
* Product listing and individual product pages
* Shopping cart functionality
* Checkout with Stripe integration
* Two-factor authentication (2FA) using pyotp

**Adding Products**

To add products to the database, you can choose to update **all\_products** list in **add\_products.py** with the desired product information and run the script:

*python add\_products.py*

**Using the Application**

After following the installation instructions, you can use the application by navigating to **http://localhost:5000** in your browser. The available pages and their functionalities are as follows:

1. **Home (index.html)**: Displays the list of available products.
2. **Product (product.html)**: Displays the details of a selected product, including its name, price, description, and image.
3. **Cart (cart.html)**: Allows users to view their shopping cart, update quantities, and remove items from the cart.
4. **Checkout (checkout.html)**: Provides a form for users to enter their credit card information and complete the payment using Stripe.
5. **Login (login.html)**: Allows users to log in to their accounts.
6. **Register (register.html)**: Allows new users to create an account.
7. **Enable 2FA (enable\_2fa.html)**: Enables two-factor authentication for users after logging in.
8. **2FA (2fa.html)**: Provides a form for users to enter their 2FA token.
9. **Success (success.html)**: Displays a success message after a successful payment.

**Testing the 2FA**

To test the Two-Factor Authentication (2FA) feature of your e-commerce application, follow these steps:

1. Make sure you have installed all the required dependencies and set up the necessary configurations (e.g., database, secret key, email service, and others).
2. Run your application locally or deploy it to a server.
3. Register a new user by navigating to the registration page and filling out the form. Complete the email verification process, if necessary.
4. Log in to your account using the registered email and password.
5. Navigate to the "Enable Two-Factor Authentication" page. If it's not directly accessible from the navigation bar, you can manually enter the URL in your browser.
6. On the "Enable Two-Factor Authentication" page, you will see your OTP secret and a QR code. Install an authenticator app on your mobile device if you haven't already (e.g., Google Authenticator, Authy, or Microsoft Authenticator).
7. Open the authenticator app and add a new account by scanning the QR code displayed on the "Enable Two-Factor Authentication" page or by manually entering the OTP secret.
8. After adding the account, the authenticator app will start generating One-Time Passwords (OTPs) for it.
9. Click the "Verify OTP" button on the "Enable Two-Factor Authentication" page. You will be redirected to the "Two-Factor Authentication" page.
10. Enter the OTP currently displayed in your authenticator app into the input field on the "Two-Factor Authentication" page and submit the form.
11. If the OTP is correct, your account's 2FA will be enabled, and you should see a confirmation message. From now on, whenever you log in, you will be asked to provide an OTP generated by your authenticator app.
12. To test the 2FA during login, log out of your account and then try logging in again. After entering your email and password, you should be prompted to enter an OTP from your authenticator app.
13. Enter the current OTP from your authenticator app and submit the form. If the OTP is correct, you will be successfully logged in.

By following these steps, you can test the Two-Factor Authentication feature of your e-commerce application to ensure that it works as expected.

***Top of Form***