Flask E-Commerce Application Documentation

Overview

This Flask application is a complete e-commerce solution featuring user authentication, product management, order processing, and payment integration. It also includes email notifications for order updates.

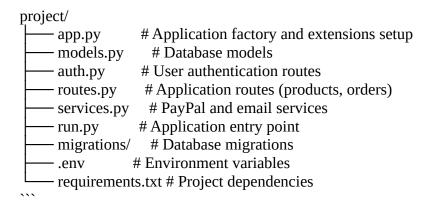
.....

Features

- User Authentication:
- Register and login functionality with JWT-based authentication.
- Password hashing for secure storage.
- Product Management:
- View a list of available products.
- Product stock management.
- Order Processing:
- Place orders and track their status.
- Integration with PayPal for payments.
- Email Notifications:
 - Notify users via email for order confirmations and status updates.

.....

Project Structure



Installation

Prerequisites

- Python 3.8+
- Flask
- MvSQL
- PayPal Developer Account

Steps

1. Clone the Repository:

git clone https://github.com/mahm0udismail/taskAppgain.git cd taskAppgain

2. Set Up a Virtual Environment:

```
python3 -m venv venv
source venv/bin/activate # Linux/Mac
venv\Scripts\activate # Windows
```

3.Install Dependencies:

pip install -r requirements.txt

4. Configure Environment Variables:

Create a `.env` file in the root directory and add the following:

```
JWT_SECRET_KEY=your_jwt_secret_key
DB_URI=
MAIL_USERNAME=
MAIL_PASSWORD=
PAYPAL_CLIENT_ID=
PAYPAL_CLIENT_SECRET=
```

5. Set Up the Database:

flask db init flask db migrate flask db upgrade

6. Run the Application:

flask run

API Endpoints:

Authentication:

```
1. Register
```

```
- URL: `/auth/register`
- Method: `POST`
- Payload:
    json
    {
        "email": "user@example.com",
        "password": "password123"
    }

- Response:
    json
    {
```

```
2. Login
  - URL: `/auth/login`
  - Method: `POST`
  - Payload:
    json
    {
       "email": "user@example.com",
       "password": "password123"
    }
  - Response:
    json
    {
       "access_token": "<JWT_TOKEN>"
    }
}
```

"message": "User registered successfully."

Product Management:

1. List Products

Order Management

1. List Orders

```
- URL: `/orders/orders`- Method: `GET`- Response:
json[
```

"email": "customer@example.com",

```
"id": 1,
"product_name": "Laptop",
"quantity": 2,
"status": "Paid",
"total_price": 2000.0
},
"email": "customer@example.com",
"id": 2,
"product_name": "Laptop",
"quantity": 2,
"status": "Paid",
"total_price": 2000.0
}
]
2. make order:
       - URL: `/orders/orders`
       - Method: `POST`
       -Payload:
  json
       "product_id": 1,
       "quantity": 1,
       "email": "ahmdalbdwy924@gmail.com",
       "payment_details": {
       "amount": 1000
       }
}
 - Response:
  json
"approval_url": "https://www.sandbox.paypal.com/cgi-bin/webscr?cmd=_express-
checkout&token=EC-9T33A",
"message": "Order created successfully. Approve payment to proceed.",
"order_id": 13
}
```

Payment Flow

- After creating the order, the response will include an approval URL (e.g., https://www.sandbox.paypal.com/cgi-bin/webscr?cmd=_expresscheckout&token=EC-9T33A).
 - This URL will redirect the user to PayPal's Express Checkout page where they will approve the payment.
- 2. Post-payment, once the user approves the payment, they will be redirected to a URL you specify (for example, your success page). The URL will contain the paymentId and payerId as query parameters. These can be used to confirm the payment and complete the order.

Example:

http://localhost: 5000/payment/execute?paymentId=PAYIDM5QS26V1A&token=EC-9T33A&PayerID=SAQN9

payment Id: The unique identifier for the payment transaction.

payerId: The unique identifier for the payer (the user who made the payment).

2. execute order:

Security Considerations

- Use HTTPS in production.
- Use jwt to make token
- Store sensitive environment variables securely (`.env` file with restricted access).

- Enable CSRF protection.

Contact

For questions or support, please contact:
- Email: mahmoudismail9114@gmail.com

- **Phone**: 01062452517