BT-SRI-20-24006

Project Documentation

CAFETERIA MANAGEMENT SYSTEM

FastCafe@DA

Introduction

Fast Cafe is a canteen automation website project developed for DAIICT (Dhirubhai Ambani Institute of Information and Communication Technology) college using ReactJS for the frontend and Django for the backend. This web application aims to streamline the food ordering process in the college canteen, enhancing efficiency and user experience for students and staff.

Key Features:

User Authentication and Profiles:

- Secure login functionality for students, faculty, and staff using college credentials
- User profile management
- Role-based access control (student, faculty, staff, admin)

Digital Menu:

- Dynamic, customizable menu display
- Real-time updates of available items

Online Ordering (Student and Staff user):

- Intuitive item selection and customization
- Shopping cart functionality

- Order review and modification
- Order feedback facility

Order Tracking:

- Real-time order status updates
- Push notifications through emails for order progress

Feedback and Ratings:

- Post-order rating system
- Detailed feedback submission
- Display of aggregated ratings for menu items

Canteen Dashboard:

- Comprehensive order management
- Menu creation and modification
- Feedback Management Setup

Supervisor Dashboard:

- Canteen Management
- Canteen's cumulative feedback management

Technical Architecture:

Frontend:

- ReactJS for building the user interface

Backend:

- Django REST framework for API development
- Django SQLite for data storage

Authentication:

- JWT (JSON web tokens) for secure authentication

User roles and Permissions:

Students:

- Browse menu
- Place and track orders
- Provide feedback
- View order history

Canteen Staff:

- View and process incoming orders
- Update menu items
- Manage order statuses
- View feedbacks

Canteen Supervisor:

- Canteen User Management
- View Cumulative Feedback of all canteens

Key Workflows:

Placing an Order:

- User logs in
- Browses digital menu
- Selects items and customizes as needed
- Reviews order in cart
- Proceeds to checkout
- Selects payment method
- Confirms order
- Receives order confirmation and tracking details

Managing Order (Canteen Staff):

- Staff logs in to dashboard
- Views incoming orders queue
- Updates order status as it progresses

- View feedback if provided by user and ratings for the order

Future Enhancements:

- Proper integration of payment gateway with razorpay Merchant Kit
- Mobile app development for iOS and Android

Initializing the project on your computer:

Backend:

The project is a Django web application named CanteenAutomation. Here are the steps to run this project in VS Code:

1. **Install VS Code**: Ensure you have Visual Studio Code installed. You can download it from here.
2. **Install Python**: Make sure Python is installed on your system. You can download it from here.
3. **Install Django**: Open a terminal (Command Prompt, PowerShell, or Terminal on macOS/Linux) and install Django using the following command:

```
pip install django
```

4. **Open the Project in VS Code**:
    - Open VS Code.
    - Go to File > Open Folder and select the extracted project folder.
5. **Create a Virtual Environment**:
    - Open the terminal in VS Code (View > Terminal).
    - Navigate to the project directory if you aren't already there.
    - Create a virtual environment:

```
python -m venv env
```

    - Activate the virtual environment:

- On Windows:

```
.\env\Scripts\activate
```

- On macOS/Linux:

```
source env/bin/activate
```

6. **Install Project Dependencies**:
   - With the virtual environment activated, install any dependencies listed in a requirements.txt file if it exists:

```
pip install -r requirements.txt
```

7. **Database Setup**:
   - Apply the migrations to set up the database:

```
python manage.py makemigrations
python manage.py migrate
```

8. **Run the Development Server**:
   - Start the Django development server:

```
python manage.py runserver
```

9. **Access the Application**:
   - Open your web browser and navigate to http://127.0.0.1:8000/ to see your application running.

If any additional setup or specific dependencies are needed, then they should be installed along the process while debugging using errors and warnings in the terminal.

Frontend:

To run the frontend of the project, follow these steps:

1. **Install Node.js and npm**:
    - Ensure you have Node.js and npm (Node Package Manager) installed on your system. You can download them from Node.js official website.
2. **Open the Project in VS Code**:
    - Open VS Code.
    - Go to File > Open Folder and select the “frontEnd” folder of your project.
3. **Install Dependencies**:
    - Open the terminal in VS Code (View > Terminal).
    - Navigate to the frontend project directory if you aren't already there.
    - Install the necessary dependencies using npm. Run:

```
npm install
```

4. **Run the Development Server**:
    - After the dependencies are installed, you can start the development server. The command can vary depending on the framework or setup.

```
npm start
```

5. **Access the Frontend Application**:
    - Once the development server is running, open your web browser and navigate to the address provided by the terminal output on http://localhost:3000.

Models Summary:

The models.py file defines the database models for a Django-based canteen automation application. Below are the details of each model:

1. Profile

- Represents a user's profile with additional information.
- Fields:
    - user: One-to-one relationship with Django's built-in User model.
    - type: Type of user (Canteen, Customer, Supervisor).
    - name: Name of the user.
    - contact_number: Contact number of the user.

2. Canteen

- Represents a canteen.
- Fields:
    - owner: One-to-one relationship with the Profile model.
    - canteen_id: Auto-increment primary key.
    - is_verified: Boolean indicating if the canteen is verified.

3. Customer

- Represents a customer.
- Fields:
    - cust: One-to-one relationship with the Profile model.
    - customer_id: Auto-increment primary key.

4. Supervisor

- Represents a supervisor.
- Fields:
    - supervis: One-to-one relationship with the Profile model.
    - supervisor_id: Auto-increment primary key.

5. Items

- Represents items available in a canteen.
- Fields:
    - canteen: Foreign key to the Canteen model.
    - id: Auto-increment primary key.
    - name: Name of the item.
    - desc: Description of the item.
    - price: Price of the item.
    - available: Boolean indicating if the item is available.

6. Orders

- Represents customer orders.
- Fields:
    - order_cust: Foreign key to the Customer model.
    - order_canteen: Foreign key to the Canteen model.
    - id: Auto-increment primary key.
    - items: Many-to-many relationship with the Items model.
    - total_amount: Total amount of the order.
    - date: Date and time of the order.
    - status: Status of the order (PaymentLeft, Received, InProgress, Delivered, AddedToCart).

7. OrderQuantity

- Represents the quantity of items in an order.
- Fields:
    - order_id: Foreign key to the Orders model.
    - item_id: Foreign key to the Items model.
    - quantity: Quantity of the item in the order.

8. Feedback

- Represents customer feedback for orders.

- Fields:
    - order_id: Foreign key to the Orders model.
    - review: Text review of the order.
    - rating: Integer rating (1 to 5 stars).

9. CustomBlacklistedToken

- Inherits from BlacklistedToken of rest_framework_simplejwt.tokens.
- Used for managing blacklisted tokens (custom behavior can be added here).

10. Points to remember:

- The models use Django's ORM for defining database schema.
- Each model includes string representations (__str__ methods) for readability.
- Relationships between models are defined using Django's relational fields (ForeignKey, OneToOneField, ManyToManyField).

Views Summary:

The Django views.py file contains several classes and functions designed to handle user authentication, account management, and CRUD operations for a canteen ordering system. Below are all of the key components:

1. **Authentication and User Management**:
    - CustomLogoutView: Handles user logout and blacklisting of refresh tokens.
    - UserLogin: Authenticates users and provides JWT tokens upon successful login.
    - activate: Activates a user account based on a token received via email.
    - UserRegistration: Handles user registration, email confirmation, and creation of user profiles based on the type (Canteen, Customer, Supervisor).
2. **Token Management**:
    - RefreshAccessToken: Generates a new access token using a refresh token.
3. **Canteen Item Management**:
    - DisableItems, DeleteItems, GetItems: Enable, disable, delete, and fetch items associated with a canteen.

    - getOrders: Handles creating and retrieving orders for customers and canteens.
4. **Order Management**:
    - getPendingOrders, getaccountdetails, getcustOrders, createorder, ConfirmOrder, GetMenu: Manage various states and actions related to orders including creation, confirmation, fetching pending orders, and viewing menu items.
5. **Feedback Management**:
    - GetFeedback, GetAllFeedback: Handle customer feedback for orders, allowing customers to provide reviews and ratings, and canteen supervisors to retrieve all feedback.
6. **Miscellaneous Views**:
    - index: A simple view that returns "Hello world".
    - DeleteCanteen: Allows a supervisor to delete a canteen.
    - OrderDelivered, seefeedback, CustPendingOrders, CustDeliveredOrders: Handle updating order status to "Delivered", viewing feedback for orders, and retrieving pending and delivered orders for customers.

The views leverage Django REST framework for API functionality, JWT for token-based authentication, and Django's built-in authentication system for user management. The file also includes email functionalities for account activation and order status notifications.

Conclusion:

This project provides a comprehensive system for managing a canteen service, offering functionalities for user registration, authentication, menu management, order processing, and feedback collection. It uses Django for the backend, REST framework for API development, and JWT for secure token-based authentication. Key features include:

- **User Management:** Handles user registration, login, logout, and account activation.
- **Order Management:** Allows customers to create, view, and manage their orders, including adding items to the cart and confirming orders.
- **Canteen Management:** Enables canteen owners to manage their menu items, disable or delete items, and view orders and feedback.
- **Feedback System:** Collects customer feedback on delivered orders and allows canteen owners to view this feedback.

The views are implemented using Django's class-based views, providing a structured and reusable way to handle HTTP requests. JWT authentication ensures secure access to the API endpoints, while serializers streamline data validation and serialization.

Overall, the project is a robust solution for managing canteen operations, ensuring smooth interactions between customers, canteen owners, and supervisors. It leverages the power of Django and REST framework to deliver a scalable and secure application, suitable for deployment in a real-world scenario.