INFOSYS SPRINGBOARD INTERNSHIP

ONLINE FOOD ORDERING SYSTEM

PROJECT TITLE: FAST BITE

INTRODUCTION:

This documentation provides a **step-by-step guide** to building an **Online Food Ordering System** using **Django**, a powerful Python web framework. Django simplifies web development by providing essential tools for building secure and maintainable websites.

Each of these apps will handle specific parts of the system, keeping the project organized and modular. Here's how we can structure the system using these three applications:

Application Overview

1. **Restaurant App**: This will handle all the restaurant-related functionality, such as managing the menu items, viewing and processing orders, and managing food categories.
2. **Customer App**: This will handle everything related to customer interaction, such as viewing the menu, placing orders, tracking their orders, and user authentication.
3. **Delivery App**: This app will handle the delivery process, such as managing delivery agents, tracking order delivery status, and updating delivery details.

In this system, we will cover several essential features:

1. **User Authentication (sign up, log in, log out)**:
    - **User authentication** allows users to create an account, log in securely, and log out of the system. Authentication is essential for protecting sensitive data, ensuring that only authorized users can access certain parts of the application (like placing orders or viewing order history).

    - For this, we will use **django-allauth**, a popular Django package that provides a ready-made solution for handling user authentication, including features like account creation, email verification, password reset, and social authentication (e.g., login with Google or Facebook).
2. **Menu Management (adding, displaying, and managing food items)**:
    - The **menu management** feature will allow administrators to add, update, delete, and view food items available for purchase in the restaurant. This includes storing details like the food item's name, description, price, and an image.
    - We will use Django's **model** system to create a `MenuItem` model that represents each food item. The administrator will interact with the Django admin panel to manage these menu items. For displaying the items to users, we will create views and templates.
3. **Order Management (placing orders, calculating total price)**:
    - The **order management** feature allows users to place orders from the menu and track the total price based on the selected items.
    - When a user selects items and proceeds to checkout, the system will calculate the total price of the order. This will include functionality to manage the list of ordered items and store the order details (such as the customer's name and the items they ordered).
    - Administrators will be able to view the placed orders, including details like customer information, ordered items, and the total amount.
4. **Integration with django-allauth for Handling User Authentication**:
    - **django-allauth** will be used for user authentication. It is a third-party Django app that simplifies the user login, registration, and authentication flow.
    - It allows us to implement features like:
        - **Signup**: New users can create an account.
        - **Login**: Users can log into the system securely.
        - **Logout**: Users can log out of the system.
        - **Password management**: Users can reset their passwords via email.
        - **Email verification**: Ensures that the user has provided a valid email address.

    - django-allauth is easy to integrate with Django projects and provides flexible settings, such as enabling social authentication (login via Google or Facebook).
5. **Image Handling for Food Items Using Pillow**:
    - **Pillow** is a Python Imaging Library that adds image handling capabilities to Django, allowing us to manage image uploads (such as photos of food items).
    - In the **Online Food Ordering System**, Pillow will be used to handle image uploads for each food item in the menu. This will allow food items to have images associated with them, providing a visually appealing interface for users to browse the menu.
    - We'll create a model field in the `MenuItem` model to store image files and configure Django to save these files in the appropriate location (typically inside a `media` directory).

Prerequisites

Before proceeding with the setup, ensure that you have the following:

- Python 3.x installed on your system
- Django (installation covered below)
- Basic knowledge of Python, Django, and web development concepts

Step-by-Step Setup

1. Create a New Django Project

Start by creating a new Django project:

```
cd path/to/your/directory
django-admin startproject food_ordering_system
cd food_ordering_system
```

2. Set Up a Virtual Environment

It’s a good practice to use a virtual environment to isolate project dependencies:

```
python -m venv .venv
```

3. Activate the Virtual Environment

Activate the virtual environment. The commands are different for Windows and Mac/Linux:

On **Windows**:

```
.venv\Scripts\activate
```

4. Install Required Packages

Now install the necessary packages for the Django project:

```
pip install django
pip install pillow
pip install django-crispy-forms
pip install crispy-bootstrap4   # For Bootstrap 4
pip install crispy-bootstrap5   # For Bootstrap 5
pip install django-allauth      # For user authentication
```

*. Create a New Django Project

```
Create the project as usual:

bash
Copy code
django-admin startproject food_ordering_system
cd food_ordering_system
```

* Create Three Django Apps:

```
Create three apps: restaurant, customer, and delivery:

bash
Copy code
```

```
python manage.py startapp restaurant
python manage.py startapp customer
python manage.py startapp delivery
```

This creates a new directory `orders` within your project.

6. Run Migrations

Run migrations to set up the database schema:

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

7. Configure `settings.py`

Open the `settings.py` file of your project and make the following changes:

- Add `crispy_forms`, `crispy_bootstrap4` or `crispy_bootstrap5`, `allauth` and your app `orders` to the `INSTALLED_APPS` list:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'crispy_forms',
    'crispy_bootstrap4',  # or crispy_bootstrap5
    'allauth',
    'allauth.account',
    'allauth.socialaccount',
    'orders',  # Your app for handling orders and menu
]
```

- Set the `AUTHENTICATION_BACKENDS` for `django-allauth`:

```
AUTHENTICATION_BACKENDS = (

'allauth.account.auth_backends.AuthenticationBackend',
)
```

- Set the `EMAIL_BACKEND` (use the console email backend for development):

```
EMAIL_BACKEND =
'django.core.mail.backends.console.EmailBackend'  # For
development
```

- Set the login redirect URL:

```
LOGIN_REDIRECT_URL = '/'  # Or wherever you want users to
go after logging in
```

8. Configure URLs

In your `urls.py`, include the necessary routes for user authentication and your app's views.

In `food_ordering_system/urls.py`:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
```

```
    path('accounts/', include('allauth.urls')),  #
Allauth authentication routes
    path('', include('orders.urls')),  # Your app's URLs
for menu and orders
]
```

9. Create Models for Menu Items and Orders

In the `orders/models.py` file, create models for menu items and orders:

```
from django.db import models

class MenuItem(models.Model):
    name = models.CharField(max_length=200)
    description = models.TextField()
    price = models.DecimalField(max_digits=6,
decimal_places=2)
    image = models.ImageField(upload_to='menu_items/')

    def __str__(self):
        return self.name

class Order(models.Model):
    customer_name = models.CharField(max_length=200)
    items = models.ManyToManyField(MenuItem)
    total_price = models.DecimalField(max_digits=6,
decimal_places=2)
    date_ordered =
models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f'Order {self.id} - {self.customer_name}'
```

- `MenuItem` model: Stores food items including their name, description, price, and an image.
- `Order` model: Stores customer order information including the customer name, ordered items, total price, and the date of the order.

10. Migrate the Database Again

After creating the models, run the migration commands again to update the database schema:

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

11. Create a Superuser for Admin Panel

To access the Django admin interface and manage orders and menu items, create a superuser:

```
python manage.py createsuperuser
```

Follow the prompts to create the superuser account.

12. Run the Development Server

Now, run the development server:

```
python manage.py runserver
```

Visit the following URLs in your browser:

- Home page: `http://127.0.0.1:8000`
- Admin panel: `http://127.0.0.1:8000/admin/`

Use the superuser account you created to log into the admin panel.

13. Test Authentication and Features

- Access the login and registration pages via `http://127.0.0.1:8000/accounts/login/` and `http://127.0.0.1:8000/accounts/signup/` (or your custom login/signup views).
- You should be able to sign up, log in, and log out, and after logging in, you should be redirected to the home page (as per the `LOGIN_REDIRECT_URL` setting).

14. Upload Menu Items and Place Orders

- Use the Django admin panel to upload images for food items and create orders.
- You can manually add menu items and orders via the admin interface or create views/forms to handle these operations from the front end.

15.Run the Development Server

After completing all configurations, you can run the Django development server:

bash

Copy code

```
python manage.py runserver
```

Now, you should be able to access the system at:

- Restaurant-related pages: `http://127.0.0.1:8000/restaurant/`
- Customer ordering pages: `http://127.0.0.1:8000/customer/`
- Delivery assignment pages: `http://127.0.0.1:8000/delivery/`
- Admin panel for managing content: `http://127.0.0.1:8000/admin/`

Key Dependencies:

- **Django**: The web framework.
- **Pillow**: To handle image uploads (for menu items).

- **django-crispy-forms**: For easy form styling using Bootstrap.
- **django-allauth**: For user authentication (login, registration, and password management).
- **Bootstrap** (optional): For styling your forms (Bootstrap 4 or 5).

Conclusion

You have successfully set up a Django-based **Online Food Ordering System** with authentication, menu item management, and order management features. By following these steps, you've integrated important packages like **django-allauth** for user management, **django-crispy-forms** for form styling, and **Pillow** for image handling.

This system can be extended further by adding features such as payment gateways, order tracking, and customer reviews.