Overview

Project Summary

The project is a web application designed as a marketplace where users can list, view, and manage items for sale. It also includes features for user authentication, conversations between users, and a user dashboard for managing personal items.

Technologies Used

- **Django**: A high-level Python web framework for building the application.
- **Tailwind CSS**: For styling and responsive design.
- **SQLite**: Default database used for development.

Features / Functionality

User Authentication

- **Signup**: Users can create a new account using the Signup form.
- **Login**: Users can log into their account.
- **Logout**: Users can log out of their account and are redirected to the login page.
- **Profile Management**: Users can view and manage their own items via the dashboard.

Core Pages

- **Homepage**: Displays featured items and categories, providing a gateway to other parts of the site.
- **Contact Page**: Provides a contact form for users to reach out to the site administrators or support.
- **About Page**: Contains information about the project or organization.

Item Management

- **Item Listing**: Users can view a list of items available for sale, with options to search and filter based on categories and keywords.
- **Item Detail**: Provides a detailed view of an individual item, including images, price, description, and seller information.
- **Item Creation**: Logged-in users can create new items to list for sale, including uploading images and providing details.
- **Item Editing**: Users can edit their own items to update information or images.
- **Item Deletion**: Users can delete their own items from the marketplace.

User Dashboard

- **Dashboard**: Displays a user-specific view showing all items created by the logged-in user, allowing them to manage their listings.

Conversations

- **Inbox**: Shows a list of conversations the logged-in user is involved in.
- **Conversation View**: Displays messages within a specific conversation.
- **New Conversation**: Allows users to start a new conversation with another user regarding an item.

Search and Filters

- **Search Functionality**: Users can search for items using keywords present in item names or descriptions.
- **Category Filtering**: Users can filter items based on selected categories.

Administrative Features

- **Admin Interface**: Provides a backend interface for administrators to manage categories and items.

Static and Media Handling

- **Static Files**: Includes CSS and JavaScript for styling and functionality, managed via Tailwind CSS.
- **Media Files**: Handles user-uploaded images and files, such as item images, stored in the media directory.

User Experience Enhancements

- **Responsive Design**: The application uses Tailwind CSS for a responsive and modern design.
- **Error Handling**: Displays form errors and validation messages to help users correct issues during form submissions.\

Description of Components

puddle/

- **__init__.py**: Initializes the puddle package.
- **asgi.py**: ASGI configuration for asynchronous server gateway interface.
- **settings.py**: Contains all Django settings for the project.
- **urls.py**: The root URL configuration for the project, including URLs from other apps.
- **wsgi.py**: WSGI configuration for web server gateway interface.

core/

- **admin.py**: Admin site configuration for the core app.
- **apps.py**: Configuration for the core app.
- **forms.py**: Contains forms used in the core app, including login and signup forms.
- **models.py**: Defines the data models for the core app (if any).
- **tests.py**: Contains test cases for the core app.

- **urls.py**: URL routing for the core app.
- **views.py**: View functions for the core app.
- **templates/core/**: HTML templates for the core app.

dashboard/

- **admin.py**: Admin site configuration for the dashboard app.
- **apps.py**: Configuration for the dashboard app.
- **views.py**: View functions for the dashboard app.
- **urls.py**: URL routing for the dashboard app.
- **templates/dashboard/**: HTML templates for the dashboard app.

item/

- **admin.py**: Admin site configuration for the item app.
- **apps.py**: Configuration for the item app.
- **forms.py**: Forms used for item creation and editing.
- **models.py**: Defines the Item and Category models.
- **tests.py**: Contains test cases for the item app.
- **urls.py**: URL routing for the item app.
- **views.py**: View functions for the item app.
- **templates/item/**: HTML templates for the item app.

conversation/

- **admin.py**: Admin site configuration for the conversation app.
- **apps.py**: Configuration for the conversation app.
- **forms.py**: Forms used in the conversation app.
- **models.py**: Defines the data models for the conversation app.
- **tests.py**: Contains test cases for the conversation app.
- **urls.py**: URL routing for the conversation app.
- **views.py**: View functions for the conversation app.

media/

- **item_images/**: Directory for storing uploaded item images.

manage.py

- A command-line utility that lets you interact with this Django project.

requirements.txt

- Lists the project's dependencies.

Running

1- Open your terminal or command prompt
2- Create a Virtual Environment "python -m venv venv"
3- After creating the virtual environment, activate it:
"venv\Scripts\activate"
4- Creates and applies database migrations "python manage.py makemigrations"
"python manage.py migrate
5- Start the Development Server "python manage.py runserver"