bargain_box

Class	Responsibilities	Collaborators
asgi.py	• Entry point for compatible servers to serve the django application asynchronously • Enables asynchronous support(such as support for running background tasks)	• settings.py • ASGI server
settings.py	• Configures the projects settings (database connections, installed apps, middleware, static files paths • Sets configurations for different environments	• urls.py • wsgi.py • asgi.py • Django settings framework import
urls.py	• Defines the URL routing patterns, which directs requests to specific views and specific apps • Manages and includes URLs from other apps	• settings.py
wsgi.py	• Entry point for compatible web servers for the django application • Configures app environment	• settings.py • WSGI server

home

Class	Responsibilities	Collaborators
HomeConfig (in apps.py)	• Configure settings for home app • Define primary key behavior • Specify the apps name and app-specific settings • Registers the app in Django's framework	• Django app import
Home_page (in views.py)	• Handle HTTP requests for home page • Renders necessary data in the home.html templates	• Home.html • foundation.html
Foundation and home html (in templates folder)	• Template for the app, providing structure to the page like navigation top bar, search bar, ect. • Also includes some scripts (for the search bar functionality)	• Home.html • Foundation.html • Static files (css)

post

Class	Responsibilities	Collaborators
PostConfig (in apps.py)	• Configure settings for post app	• Django app import

	• Define primary key behavior • Specify the apps name and app-specific settings • Registers the app in Django's framework	
Post (in models.py)	• Define structure of post (title, author, content, etc) • Manage post validation	• PostListView • PotDetailView • PostUpdateView • PostCreateView • PostDeleteView
urls.py	• Defines the url routes for viewing listings, viewing details of the post, creating, updating, and deleting posts as well. • Defines the mapping from each url to each view	• PostListView • PotDetailView • PostUpdateView • PostCreateView • PostDeleteView
PostListView(in views.py)	• Displays all the posts in a list, as well as search feature view listings • Renders the view to the correct html template	• Post model • PostDetailView
PostDetailView(in views.py)	• Displays a specific posts details (title, author, content, etc) • Renders the details in the correct view template	• Post model
PostUpdateView(in views.py)	• Handles the editing of an existing post. • Displays the form with current data that is editable • Validates any new data if the user changes something, and saves the updates • Redirects to success page upon success	• Post model
PostCreateView(in views.py)	• Handles the creation of new posts. • Displays empty form for users to fill • Validates the data and saves the data to the database • Redirects to success page upon success	• Post model
PostDeleteView(in views.py)	• Handles deletion of user posts, only made by the logged in user • Takes user to “confirm deletion” page before deletion occurs • Upon deletion, success message displayed	• Post model

user_authentication

Class	**Responsibilities**	**Collaborators**
userAuthenticationConfig (in apps.py)	• Configure settings for userauthentication app • Define primary key behavior • Registers the app in Django's framework	• Django app import

user_account_registration (in views.py)	• Handles the http requests for user registration • Validates the users data, making sure all fields are correct including strong password, valid email, unique username	• User model • UserRegistrationForm • User_account_signout
user_account (in views.py)	• Display users account details • Fetch users details and organize on page according to the profile view template	• User model
User_account_signout (in views.py)	• Handles user sign out functions • Upon success, success message displayed	• Authentication framework • user_account_registration

user_posts

Class	**Responsibilities**	**Collaborators**
userPostsConfig (in apps.py)	• Configure settings for user posts app • Define primary key behavior • Registers the app in Django's framework	• Django app import
urls.py	• Defines the url routes for accessing the posts including listing and detailed views. • Defines the mapping from each url to each view for the user posts app	• view_posts
View_posts (in views.py)	• Displays list of posts • Fetches and presents the data of the post loaded into the correct template for users to view	• Post model • urls.py

user_profile

Class	**Responsibilities**	**Collaborators**
userProfileConfig (in apps.py)	• Configure settings for user_profile app • Define primary key behavior • Specify the apps name and app-specific settings	• Django app import
UpdateUserTableForm (in forms.py)	• Defines the form for updating the users information such as name and email. • Validates the users new data, and saves any updates to the database	• Profile model
UpdateProfileTableForm (in forms.py)	• Provides form for updating profile information such as bio, profile picture, etc • Validates the users new profile data and saves any updates to the database	• Profile model

Profile (in models.py)	• Defines the profiles data structure including bio, profile image, etc. • Relates the profile to the user using a one-to-one relationship	• updateUserTableForm • Signals.py
signals.py	• Handle the creation of a user profile when a new user is created • Handles deletion of user profile resources such as profile picture (the user deletes them)	• Django signals • Profile model
urls.py	• Routes the URLs to profile related views like viewing and editing the profile	• view_profile • edit_profile
view_profile (in views.py)	• Handles the HTTP request to display the profile page of the user • Renders and fetches the users profile data (only for logged in users)	• Profile model
edit_profile (in views.py)	• Provides the form for editing a logged in users profile • Handles form validation to make sure information is correct, and saves the updates.	• Profile model • updateUserTableform • updateProfileTableForm

user_registration

Class	**Responsibilities**	**Collaborators**
User_registration_config (in apps.py)	• Configure settings for user_registration app • Define primary key behavior • Specify the apps name and app-specific settings	• Django app import
register (in views.py)	• Render the user registration form • Process the user registration form • Validate user_data with a successful form • Handles GET requests to show registration page • Handles POST requests to process the data from the form	• UserRegisterForm • User Model (creates user) • Messages (for message feedback (success,fail))
UserRegistrationForm (in forms.py)	• Defines the fields in the form and the order in which they appear in registration (username, email, password) • Validates user input, making sure password is strong, email is valid, username not taken. • Handle data cleaning and transformation before saving to the database	• User Model • Register (View) • Django forms import