

**MBARARA UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**FACULTY OF COMPUTING AND INFORMATICS**

**BACHELOR OF COMPUTER SCIENCE**

**COURSE: WEB APPLICATION DEVELOPMENT**

**REG NO: 2021/BCS/073/PS**

**NAME: YUHA IAN BAHATI**

**Introduction**

The project is a web application built using the Django framework, aiming to replicate the main features of Instagram platform. This report demonstrates the implementation of various features and concepts required to create a functional Instagram clone.

**Project Structure**

The project is includes the following apps;

* Userauths: This handles user registration, authentication, and profiles.
* Posts: This manages user posts that is images, favorites-images, tags, likes, followers and captions.
* Comment: This enables users to comment on posts and other comments
* Notifications: This enables to inform the user about new likes, messages, posts and comments about the user posts
* DM: **T**his facilitates private messaging between users.

**Setting up the Django Project:**

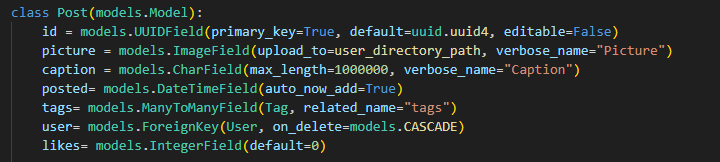
This project was created through the command-line tools.

**Creating Models:**

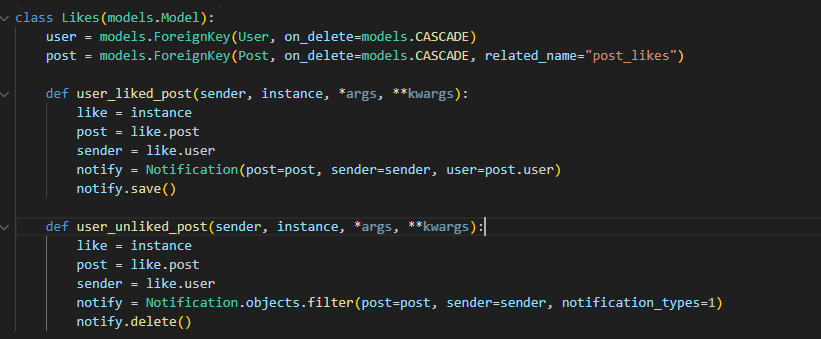
Each app has different models, these models have also fields with their represented data. Relationships are established using Foreign Keys, and ManyToManyField.

Posts app has models like Tag, Likes, Follow, Stream, Post.

Post model has fields like; id , picture, caption, posted, tags, user, likes.



Likes model has fields like user and post.

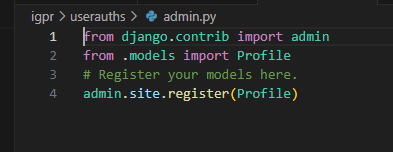


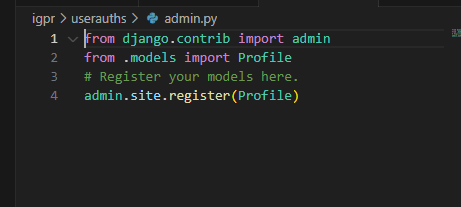
Follow has fields like follower and following,

Stream has fields like following, user, post, date..

**Custom Model Admin Classes**

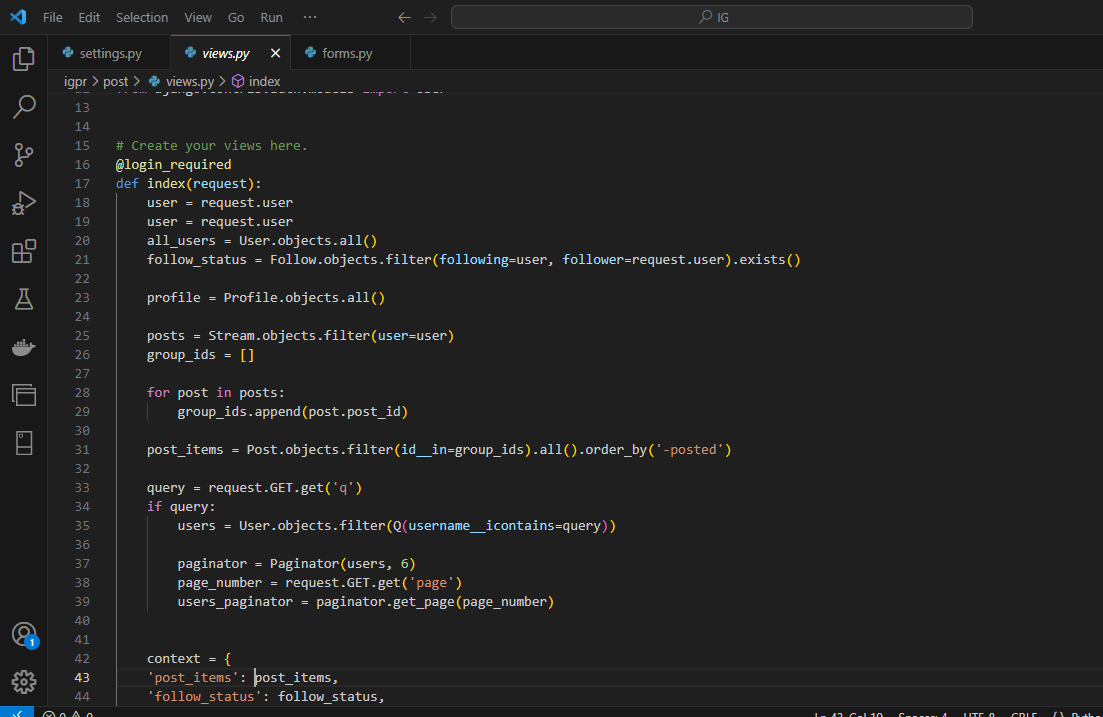
Custom admin classes are created to enhance the admin interface. For instance, the post-admin and userauths class





**Working with Views**

Each app defines its views. These handle user requests, for example post/views.py

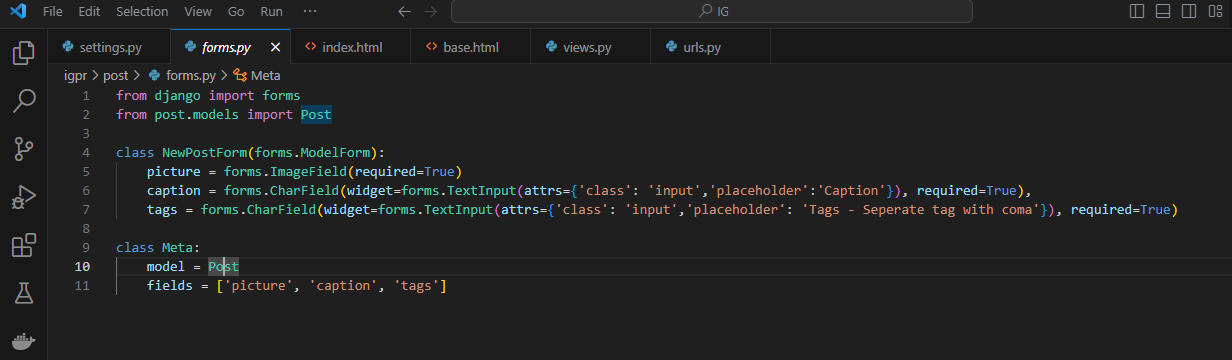


**Working with templates**

Templates are used to render the HTML pages. Template inheritance is employed to maintain a consistent look across pages.

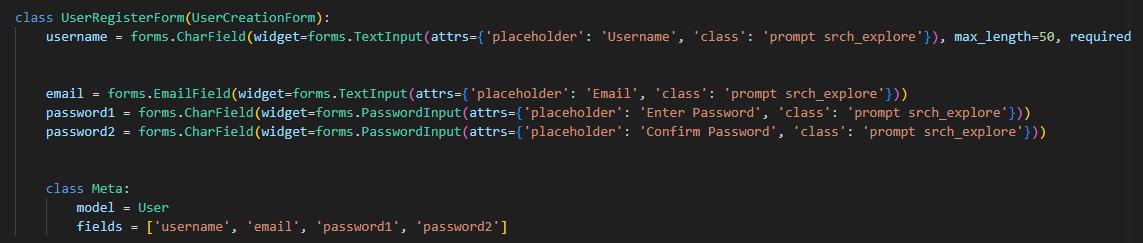
**Working with Forms:**

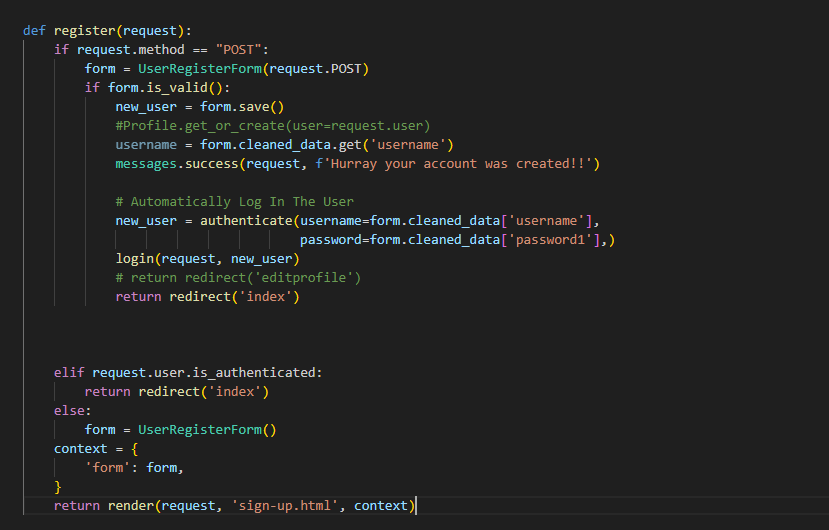
Django forms are used for creating and updating models. For example, the `NewPostForm` allows users to upload images, tags and captions.



**User Registration and Authentication:**

The built-in Django authentication system is used for user registration and login. Custom views are implemented for registration and login along with appropriate forms and templates such as the sign-in and log-in templates. Example of one of the user Registrations and authentication codes used in the project;





**Database Design and Migrations:**

The creation of models helps in implementation and designing of databases. Migrations are used to synchronize the database schema with model changes. The migrations are made through commands such as; python manage.py makemigrations and then python manage.py migrate

**Templating and Template Inheritance:**

Templating is the process of rendering dynamic HTML content by combining static HTML templates with data from the server. It allows you to separate the presentation layer (HTML) from the application logic.

Template Tags: Template tags are special constructs enclosed in `{% %}` that allow you to add logic and control structures to your templates. Filters are used within template tags to format or modify data displayed in templates.

Template inheritance is a powerful feature in Django that allows you to create a base template with a common structure (like headers and footers) and then extend or override specific sections in child templates. This promotes code reusability and maintainability.

* Base Template: This contains the common structure of the pages. This includes elements like headers, footers, navigation menus, and placeholders for dynamic content.
* Child Templates: Create child templates that inherit from the base template using the `{% extends "base.html" %}` tag. Override specific blocks defined in the base template as needed. The child templates in this project include; edit-profile.html, index.html, newpost.html, post-details.html, profile.html, sign-out.html, sign-up.html and tag.html
* Rendering: This helps to combine the content from the base template and the child template with the help of Django, replacing the blocks defined in the child template into the corresponding blocks in the base template.

**Styling with CSS:**

Custom CSS styles aid presentation and layout of the HTML content. CSS selectors are used to target HTML elements for styling. For example, `h1` targets all `<h1>` headings, and `.class-name` targets elements with a specific class.CSS properties determine how an element should be styled, and values define the specific styling.CSS files are considered static assets and Django provides a built-in way to manage them. The CSS files are placed the `static` directory of the app. In the HTML templates, use the `{% static %}` template tag to generate the correct URL for the CSS file.

**Data Validation and Error Handling:**

Forms and models incorporate data validation to ensure the integrity of user-submitted data. Error handling mechanisms are implemented to display relevant error messages to users.



**Deploying the Django App**:

The app is deployed using Configuration files, database settings through the command line prompt.

**Documentation:**

The project includes a detailed README file explaining the purpose of the project, how to set it up locally, and any deployment instructions. It also outlines the features and technologies used.

**Conclusion:**

The Instagram clone project demonstrates a comprehensive understanding of Django concepts and best practices. By successfully implementing various features like user authentication, model relationships, custom admin classes, and more, the project showcases the ability to create a functional and visually appealing web application.