**Info 409**

**Freelance Application – Django Web site**

**58926 - Guitta Moubarak**

**58688 - Mohammad Fleity**

Summary

[1 Introduction 3](#_Toc163042234)

[2 Class Diagram 3](#_Toc163042235)

[3 Pipenv: 4](#_Toc163042236)

[3.1 Advantages. 4](#_Toc163042237)

[3.2 Commands: 4](#_Toc163042238)

[4 Website Description 4](#_Toc163042239)

[4.1 Models 4](#_Toc163042240)

[4.2 Views 6](#_Toc163042241)

[4.3 Forms 6](#_Toc163042242)

[5 Technical Issues 6](#_Toc163042243)

[5.1 The Issue 6](#_Toc163042244)

[5.2 Solution 6](#_Toc163042245)

[6 Github link 7](#_Toc163042246)

# Introduction

The freelancer application, previously developed using DW and UML, has been selected as the basis for the Django website.

Here's a concise overview of the class diagram:

- Users possess profiles, roles, and genders, and can engage with Service Providers by rating, contacting, providing and feedback. They can also search for services within a specific area.

- Service Providers offer services, each associated with images and information about the service.

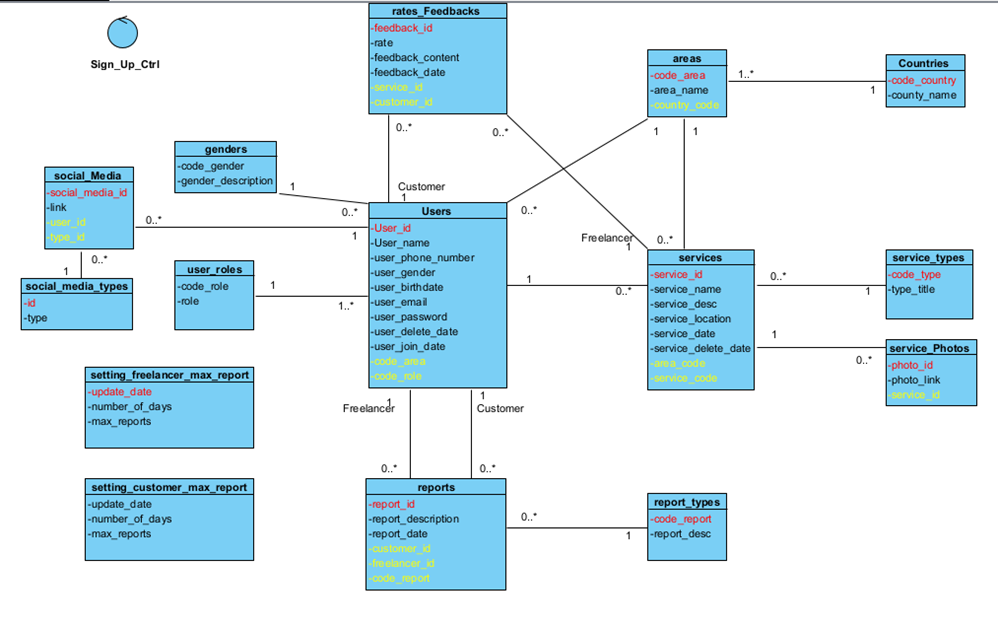
- Services, categorized by area and type, are represented with associated images.

- Services can be edited (change the image, change any information related to the service or it can be deleted).

- User can edit its own profile (only for some fields available).

In our project, we have only three primary tables: Users, rate\_feedback, and services. These tables were divided among team members as follows: Mohammad Fleity handled the rate\_feedback table, Guitta Moubarak managed the services table, and we collaborated on the Users table. Even though we divided the tables among ourselves, we provided assistance to each other throughout the project because we utilized GitHub for collaboration.

# Class Diagram



# Pipenv:

## Advantages.

* Pipenv: Simplifies Python dependency management and virtual environment setup.
* Automatically creates virtual environments and manages dependencies via Pipfile.
* Ensures reproducibility with Pipfile.lock for version locking.
* Streamlines tasks with user-friendly commands for Python developers.

## Commands:

* **pip install pipenv**
* **pipenv shell**
* **pipenv install**
* **python manage.py runserver**

# Website Description

In our Django project, we've developed three key apps:

1. Authentication: This app handles user authentication functionalities such as login, signup, changing passwords, logout, and middleware for request processing and security.
2. Services: This app is responsible for managing services, searching service, and rate\_feedback, allowing users to add, delete, and update services, as well as rate them and provide feedback.
3. User\_Profile: This app is dedicated to editing and viewing user profiles, providing users with the ability to manage their profile information.

## Models

* The Authentication app does not include any custom models since we are utilizing the pre-built User model provided by Django.
* The \_Services app models:

In this model, we have utilized foreign keys to establish relationships between multiple tables, including many-to-many relationships.

Such as the relation between Service table and service\_type table.





* In the user\_profile app, similar to the Authentication app, we also don't have custom models as we rely on the pre-built User model provided by Django.

## Views

Authentication views.py: Brief: In the Authentication app, users log in using their unique email and password. If the provided credentials match an existing user, they're directed to the home page; otherwise, access is denied. During signup, users are required to provide a username, first name, last name, unique email, and a designated role, which are used to create their account. When users wish to change their password, they must provide their previous password along with the new one, ensuring authentication and security before allowing any updates.

\_Services views.py: Each function within the views.py file directs its action to a specific template, with corresponding URLs defined in the urls.py file within the folder.

User\_profile views.py: These views facilitate the viewing and editing of user profiles within the Django framework. They leverage the pre-built user functions provided by Django to access and display user information. Additionally, they allow users to edit both the pre-built fields provided by Django and any new custom-defined fields.

## Forms

In this project, we've leveraged a combination of pre-built forms provided by Django and custom forms tailored to specific requirements. Additionally, we've customized some of the pre-built forms to align them more closely with the project's needs. This hybrid approach allows us to take advantage of Django's built-in form functionality while also accommodating unique form design and validation requirements within the project.

# Technical Issues

## The Issue

Upon logging into the application, users can navigate back to the login page by clicking the browser's back arrow. Furthermore, if an authenticated user attempts to log in again, it can potentially compromise the app's integrity.

## Solution

To address this issue, we've implemented a middleware solution and utilized Django's built-in function `@never\_cache`. This combination helps prevent users from accessing cached pages and data, ensuring a more secure and reliable browsing experience within the application.

# Github link

We utilized GitHub for collaborative development on this project. You can access the project via the following link:

<https://github.com/GuittaMoubarak/Freelancer.git>