Servic_Sync

Online Services Management Application

Ibraheem Halasah

Supervisor: Eng. Muhannad Aljabi

An-Najah National University

Department of Computer Engineering

Software Graduation Project (GP I)

This dissertation is submitted for the degree of

Computer Engineering

Sep 2023

Acknowledgements

To my God!

To all who helped for this project to be possible. To my beloved university. To my department and of its instructors, teaching assistants, and students who helped me in this project.

To my supervisor Eng. Muhannad Aljabi who helped me a in my education journey and in the project performing.

To all who supported and subjected for it; I give you thanks and greetings from my heart. And here I represent you the project and its report.

DISCLAIMER

Abstract

The problem is that some companies and organizations depends on multiple services on the cloud and on the Internet, but that cause a lot of chaos. For example technology companies depends on code hosting services (like GitHub), on task management services (like Notion), and … etc. But the problem is there's a lot of links to manage and share with their clients, interns, and employees.

So this project aims to build a service sharing and management web application. It allows the companies and the organizations to manage their services online in a single place. Also it allows the service providers to offer their services online; so that makes a good chance for: the startups to introduce their new services, and the organizations and user to search about another services that they need.

1. Introduction

1.1 Problem Statement

Servic_Sync is an online system aims to solve many problems like:

1. The biggest problem that this project aims to solve as the chaos that comes from a lot of online services. So it helps to organize them in a single place.
2. Easy to share the services with the stakeholders in a simple and secure way.
3. It offers a marketplace for the services to offer their services to the interested people in the right place.
4. It allows the organizations and user to search about the right place to search about their interested services

1.2 Significance

The main goal of the project is to build an online service that helps the organizations and the users to organize their services in one place. Also to help organizations share their services with their stakeholders in a secure way. In order it enables the service providers to share their services in a marketplace and enables the users and the organizations to search and browse through this marketplace, and find services they are interesting in.

1.3 Objectives and Scope

The purpose of the project is to help companies and organization in organizing and sharing their online services with their stakeholders. And it's also embedded to help the service providers share their services with people may interest. Here's the abilities for this project:-

- Help the organizations to make a workspaces that are a general place to share the services.
- Help the organizations to store their services in a single place.
- Help the organizations to authorize their mangers to the services in different permission classes.
- Help the organizations group their stakeholders and give each one the suitable permissions for him.
- Help the service providers to share their services in a marketplace and deliver them to the interested people in the right place.
- Help the users and the organizations to find and search the suitable services that matches their interst.

1.4 Report Organization

- **Second chapter:** covered the important subjects learned previously, as well as the external courses and the primary constraints, challenges and obstacles encountered while working on the project.

- **Third chapter:** literature review
- **Fourth chapter (Methodology):** talked about the techniques used in building the application, in addition to the features that offered and the technology's used.
- **Fifth and final chapter:** talked about the results, as well as the lessons learned from working on the project and future developments.

2. Constraints, Problems and Earlier Coursework

2.1 Constraints

2.1.1 Time Constraints

The project in the university is considered registered in the suimmer semester; so it must be done in the scope of it. So implementation time of the project mustn't go outside of its time. Therefor this project can't be improved to solve some kinds of a lot of things that are important and related to the scope of the project, like the innovation success and the innovation challenges.

2.1.2 Implementation Constraints

This project is still considered as educational for computer engineering students. So to perform it; the contributor should build it from least complex software, and implement the full backend programming with a database connections, and can't rely on the complete backend solutions like the Firebase or AWS.

2.2 Challenges

2.2.1 Lack of pre-experience

The project have a high experience in the backend, and have almost no experience in the frontend. So that made the experiments of the project very hard and takes a high number of trials to success, and forced him to learn a frontend technology by taking an Angular course.

2.2.2 Coding challenges

The project contributors is simply computer engineering students; so he hasn't a lot of coding problem solving techniques before that project.

2.3 Earlier coursework

2.3.1 Computer Programming(C/C++)

These classes covered the fundamentals of programming and introduced to the most basic concepts of the programming.

2.3.2 Data Structures and Algorithms

This course has provided me how work with the data and organize it in the memory; which a very important process in the programming.

2.3.3 Critical Thinking and Research Skills

This course has provided me how to conduct research and write a report, and it's one of the few non-technical courses that is also lifetime.

2.3.4 Object Oriented Programming

This course has provided me how to deal with the Object Oriented Programming, which is the most common programming paradigm nowadays, and it's used to get and leverage high code reusability.

2.3.5 Database Management Systems

This course has provided me how to deal with database to store data in a way that it's easily accessed. And provided me with data modeling processes and how the data shapes

stored. Also provided me how to query data and apply CRUD operations on the data in the needed way. It also provided me how we should connect with database and work with it in a program that I program with a programming language. It also provided me with the basics of the ORM which is heavily used in almost every framework nowadays.

2.3.6 Web Programming

This course has provided me the basics of web, and how the web works. It also gave me a full overview of the server side codes and database usage in the web servers.

2.3.7 Distributed Operating Systems (DOS)

This course has provided me with the basics of communication protocols and the architectural styles.

2.3.8 Software Engineering

This course has provided a full overview of how software development processes. It also provided me with tools that can be used in programming like Git and GitHub.

3. Literature Review

While building this project most of my thinking was on the time constraints; but near to it we needed to solve as possible of problems as I can and implement features to make the use of the project more powerful. This can be explained as follows:

3.1 The Regular work in organizing and sharing services

In the normal situation organizing services may be not done, or done using a turnaround method, like store them on Notion or Github, or organize them using a file or a folder on the computer, but that will cause some chaos and make the things messy. But at all the methods above, there's no way to track and organize the sharing process, so there's no way to track the utilizers of the services. So then the need to system to organize the storing and the sharing of the online services links needed.

3.2 Servic_Sync System

But when utilizing service_sync system, then the things become totally different and organized. Each services can be stored on this online service; so then the storing and tracking process becomes highly organized. Also there's an organized way for the sharing process. There's workspaces that created to add managers and groups, and groups may contain many group admin and many group members, so then we have different permission classes, and multiple levels of the privileges, so then the sharing process is organized and have multiple classes of permissions, and then the way of the sharing is organized, and the service utilizers can be easily tracked.

4. Methodology

This chapter contains detailed information about the techniques and methods used to develop the project, from choosing the idea and starting the basics, through build the front end and back end control.

4.1 Choosing the idea

In the beginning, I've tried to found creative different idea, so I tried to make something in the field of networking, but I failed to do that, due to the lack of resources for this field, and to very bad support and experience in my community. So instead I decided to make a project that has some different idea, but with has traditional implementations, and enhance it as possible.
One time when I started an internship at a company, they started to communicate with me and the other interns via email and Discord to deliver us with the information of the training. They delivered us with the information of the online courses that will be used during the internship. They also delivered the dates and calendar application for time management. Also they delivered some other platforms communications. They also delivered the information of the task management timelines and cards on Notion. Also they delivered an online application to book the desks in the office.
What I noticed in this process is that there's a lot of links of online services in delivered to us, but they were delivered in a lot of emails and in different communication places; so this thing is too messy. Also I thought about them (the company and the mentors), from the look to how they organize the services from the buying and distributing the links and the information with stakeholders; everything is messy and cause a lot of chaos; So here's the idea came to me.
From that I noticed that I can build a one place to organize the storage of the information of these services in a central place, and make this place easy for the sharing and the storing process; and this place has multiple permission levels, making the sharing process organized to give each one what he requires.
Then I thought also that I can develop this place to give it the capability to share the services from the service providers on it. And in that they can deliver announcements about their services in the right place.
After I've chosen the idea I discussed it with Dr. Emad Natesheh and with my supervisor, and they told me that it's a good idea, the core of the idea is excellent, and the idea is some different (there's almost no similar earlier work that matches it). So then I moved to do some simple online market research.

4.2 Simple Online Market Research

(References for this section are listed at the end of this report)

After getting the idea, it has been decided to do some online market research to check the demand of the idea and try to enhance it as possible, and the results of the research are listed as the following:

The evolution of the usage of online services after Covid-19 pandemic:

Prior to Covid-19 pandemic which started in 2020, online services were commonly considered as negative untrusted entities by the people in the whole world. However in 2020 due to prevailing health-related circumstances which caused by the Covid-19 virus (Corona pandemic), indivuals were compelled to use online services, so then the usage of these services were increased hundres of times.
After that people recognized that these services are incredibly powerful and can deliver unique things.
So from the listed above, we notice that the usage of the online services was inceremented in unpredicted way and the need for a system to organize them is realy a demand.

The demand of the innovation in the current world:

In the current world and with this high technologies evolution, the demand of almost 86% of the organizations, companies, and institutes is on innovations and make something different to leave their unique fingerprints. Then the whole demand, entrepreneurial culture, next-generation techonology is awarded to the innovators. Also then the innovation is made as the main digital era demand.

The problems faced by the innovators:

In the world whole innovators faces some common problem, which are summarized in: lack of collaboration, advanced experience, material resources, learning resoures, and budgets.

But after the these studies, and with considering the time factor the project is consntraited with: the project decided to be just to help in the problem of the non organized online services, with addition to help the service providers in marketing and advertising their services in the right place to deliver them to the intersted customers. And the problems of the innovators left to be as a future work.

4.3 Features and Roles Selection

The main features and also the roles has been selected after a lot of discussion with multiple colleagues and with the project supervisor and here's the details of them:

4.3.1 Features

The main feature of this system is the workspaces: which allows to deliver links, other information about the online services, announcements, and discussions from the organizations to the right stakeholders. So to achieve the right communication; there were a lot of privileges and roles that aimed to control the permissions.

The workspace has:

- An organization which owns it
- Multiple groups that contains multiple group admins, and group members
- Multiple managers, each of them is able to authorized to work with many groups
- General announcements place; which are announcements that are visible to everybody that has any place in the workspace
- Special announcements place; which are only for the organization admin, and the managers
- Multiple discussion place; inside the groups and outside them

- Multiple places to store and share sensitive information (like usernames, passwords, and API's keys) in secure way (by using encryption and decryption), because these information can't been stored in the database as clear text

Other main feature of the system is that enables the organization to store its online services information, and ability to authorize some managers of the workspaces to view it.

Each service is stored in an object with its information, a secure way to store sensitive information as the mentioned above about the workspaces' secure information, ability to add comments on the service, and also it has a way to store "cards" about it, if there were more than one program of subscription to the same service; each card also can hold comments about it, and way to store sensitive information in a secure way.

Like the organization, the user also is allowed to track his own services and also have "cards" to store more than one version about the same service. The user also have the same features of the organization, but not the manger authorization. So he also have places for comments, and a way also to store the sensitive information.

The other feature of the system is that the service provider can show the services that he provides, with their links, and programs.

It has thought to give them some way to track the subscriptions to their programs locally, but that looks illegal, because they are online service providers, so they should have a way to track their subscribers.

But here's some considered problem: sharing these services with their links may be used in a malicious way, by injecting malicious links, phishing programs, adwares, and things like that in the advertisements, and harm the users. So to solve this problem: it has been decided to scan each link in a web sandbox (which is the most considered technique to check the security of the links before using them) before sharing it with the users.

4.3.2 Roles

This system has a complex permission ways, so increase the number of roles can have very effective impact on the system, because that isolates the data and the permissions for each of them. On the opposite side that will cause a lot of complexity. So to be in between it decided to have intermediate number of roles, and raise the permission handling on the code level across the different API's.

Here's the roles of the system:

- Organization: has the ability to create workspaces, with all permissions of them. Also it has the ability to store information of their services in the system, and give the

authorization for multi of their workspaces' managers to work with them
- Service Provider: has the ability to advertise about their services on the system
- User: has the ability to any part of a workspace (manager, group local admin, or group member) and ability to store and manage his services
- Administrator: the global system admin

4.4 Technologies Selection

The selection of the project technologies was depending on three factors: the project nature itself, the experience of the project creator, and the time constraints. Here's an explained view of the technologies utilized in this project:

- System design and architectural style: because this application needed to have multiple kinds of users, and it also has some kind of marketing; so it's better to be on the web to be online for everybody on the Internet, with no need to install specific mobile or desktop application to use it. However it has been decided to use the REST architectural style and make the make the communication between the front-end and the back-end using the http protocol. The idea behind that is to decouple the front-end code from the backend code; so we can easily in the future extend the system with another frontend application, desktop application, or mobile application, with no need to rewrite or modify the backend.
- Backend: it has decided to use Django, because the project creator has some pre-experience of using it. It has been used with its rest framework, which is a framework that integrates the REST communication inside Django applications
- Frontend: it has been decided to use Angular, because it's a complete frontend framework. It implicitly have the routing and SPA (Single Page Application) functionality. It also has code base organization for everything in the client code, like the state management and the http communication code things
- Database: in this there's a lot of things (like the user's information and the permissions) that needs the data consistency, which is the feature of the relational databases. However, there's another things that needs data flexibility (like the advertisements, comments, and discussions) which available in the documents and JSON based databases like MongoDB. So it has been decided to use the PostgreSQL, which is a relational database, so it's offers the data consistency, but it also has the feature of the JSON data fields, so it also combines the data flexibility feature from the non-relational databases

4.5 Coding Bases Preparation

To write the code of this project, it has two separated code bases, one for the frontend and the other for the backend, and the other for the frontend; each of them has a Git repository and hosted on a private Github repository. But before starting working on them, the database has to be installed and configured properly. Here's the illustration of working with the two code bases, and the database preparation.

4.5.1 Database Installation and Preparation

To start the work on backend we need to prepare the database by install it, create a database, create a user, give the user privileges to work on this database, and prepare a visualization tool for the debugging, and development purposes. Here are these steps in detail:

Install the PostgreSQL database system:

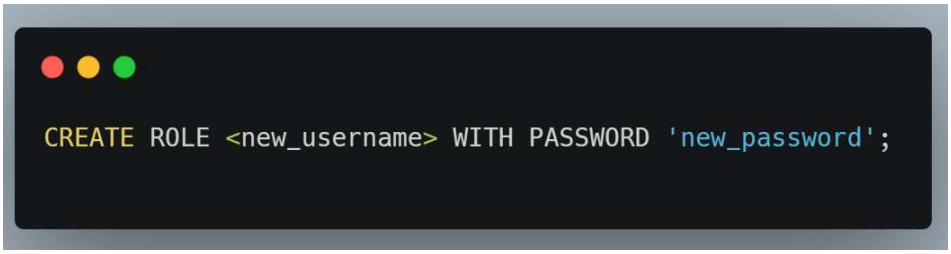
The system has been downloaded from the official website (link is available in the references section at the end of this report). After that it has been installed and the password for the super user has been added with no problems.

Create a user, a database, and give the user privileges on it:

Before create the user, it has be to go to the power shell or the windows CMD as administrator and log in to the PostgreSQL by run the following command:

Code Snap 4.1: command to login to PostgreSQL as the Super User

After the above command we need to enter the password that has been put while installing the PostgreSQL, and then it has been logged in successfully, then we need to create the user by the following SQL command:



Code Snap 4.2: SQL command to create a user on the PostgreSQL

Then it's needed to create a database for scheming the data, it's created by run the following SQL command:

Code Snap 4.3: SQL command to create a database on the PostgreSQL

And after that it's needed to give the user created all privileges to work on the created database by run the following SQL command:

Code Snap 4.4: SQL command to grant all privileges to a user in the PostgreSQL

And all of the above has been executed and succeeded and now it's the time to install a database visualization tool for data viewing and debugging purposes.

Install PostgreSQL database visualization tool:

The official and most recommended tool for this purpose is the PgAdmin. So it has been downloaded and from the official download website (link is in the references at the end of the report) and installed successfully.

But the problem is that it hasn't been able to connect to the local database, and a lot of trial failed (this problem is discusses in the problems in section 5.1).

So as an alternative there were 2 tools for the same purpose and has been utilized: one of them is a VS code extension, and the other is the command prompt, it has been logged in to the user on the database, and utilize the CRUD SQL commands for the debugging purposes.

4.5.2 Back-end Code Base Preparation

Before start coding the backend using Django, first it's needed to verify that Python 3 is available on the machine.

After that the coding process can be started, but that's is a bad practice. The better practice is to create a Python virtual environment for the project.

The importance of the Python Virtual environment:

- Dependencies Isolation: the main feature got from the Python virtual environment is that it isolated the dependencies of the project from the global machine dependencies and packages, so it prevents the versions and packages clashing with each other and with the packages installed on the system. And that cause safety for the project and the global Python on the machine.
- Reproducibility: the Python virtual environment encapsulates the dependencies of the project in a single place, so that makes the process of migrating the project from the machine to another machines so simple. It can be easily done by:
  1 - Freeze the dependencies of the project on the source machine in a single file (by convention it's called requirements.txt). That can be performed by running the following command:

Code Snap 4.5: command to freeze the Python requirements in a specific file

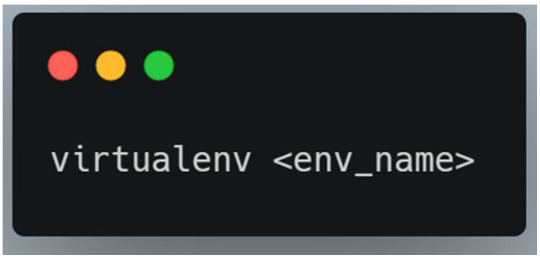
2 – Install the dependencies on the target machine by running the command:

Code Snap 4.6: command to install the Python requirements listed in a file

Creating the Python virtual environment done by the tool called "virtualenv", so first we need to install it globally on the machine by running the command:

Code Snap 4.7: command to install the “virtualenv” package in Python

And after that we navigate to the project folder we create the environment by running the command:



Code Snap 4.8: command to create a Python virtual environment using the “virtualenv” packeage

And after that it’s needed to activate the virtual environment. The activation process differs from an operating system to another, but in that project case; it was on a Windows machine, so it has been activated by running the command:

Code Snap 4.9: command to activate the Python virtual environment in a folder

And all of the mentioned above has been done successfully, and no it’s the time to install the Django framework and create the project for the server.

Installing Django and its rest framework, creating the project and adding the rest framework:

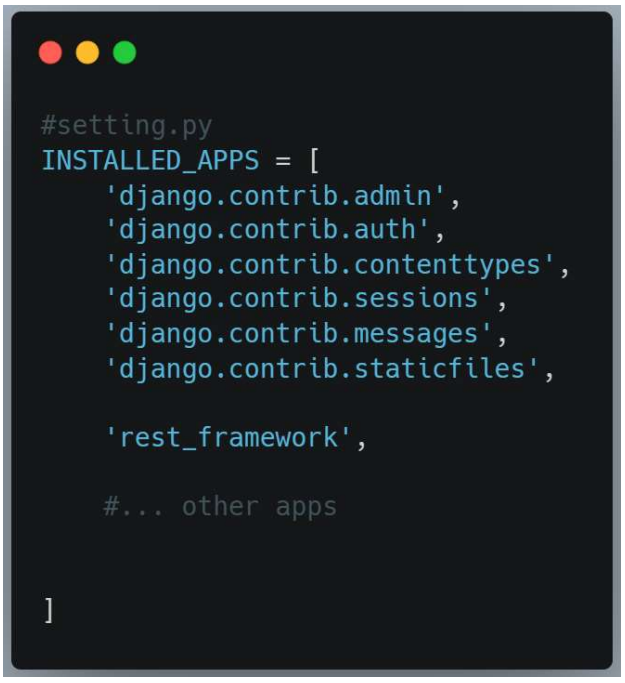
After creating and activating the Python virtual environment, it’s needed to install the Django framework and its rest framework inside the virtual environment. That has been done using the command:

Code Snap 4.10: command to install Django and its rest framework using pip

After that has been executed successfully, it's the time to create the Django project. It has been created using the following command:

Code Snap 4.11: SQL command to create a Django project

After that it's needed to add the rest framework to the project's installed apps (apps in Django discussed fully later in the backend control design). That's performed by update the Installed apps list in the "settings.py" file in the project as the following:



Code Snap 4.12: adding the rest framework to Django installed apps

After that we need to start the Django server to verify the installation success. There's a good light weight development server that pre-installed when installing Django called "runserver". It's also good for debugging purposes, since it refreshes the server on each update in the code files in the server. It can be started by running the following command:

Code Snap 4.13: command to run the "runserver" development server in a Django project

After that we should navigate to the browser and browse to the server, a Django startup page should be appeared to verify the success of the server work. And that has been verified successfully. Now is the time to integrate the server with the PostgreSQL database.

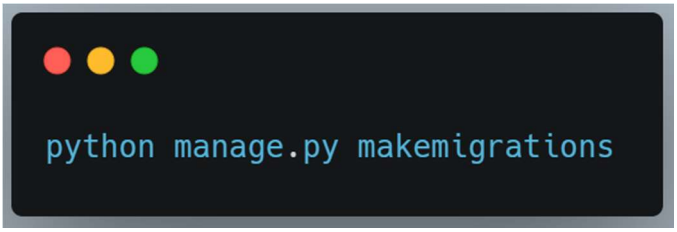
Connecting the backend with the database:

In this Django application, Django built in ORMs were used for data modeling, and they connect with the default databases settings, that configured globally for the whole project. The global configuration is done by putting the database configuration (the type of the database, the host URL, the username, and the password) in the "settings.py" file for the project. And here's how it has been put:

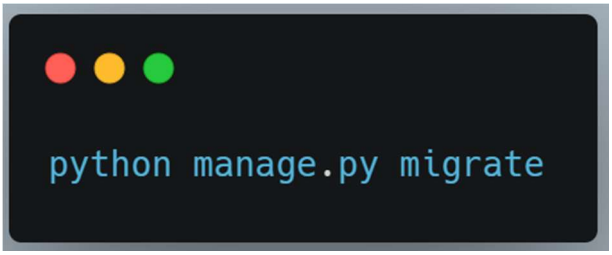
Code Snap 4.14: configure the Django project to connect with PostgreSQL database with specific credentials

Then to verify the connection we need to create the migrations of the data models and migrate them to the database. Also these two steps need to be repeated each time more data models added or altered (discussed in details in the models topic in the next section).

To create migrations, this command should be run:



And to migrate them to the database, this command should be run:



When run them for the first time, there's a small problem of missing dependency were faced, but it has been solved quickly by installing the dependency. (The problem is discussed in a separate section detail in the problems chapter – section 5.2). But after that the commands run successfully.

4.5.3 Front-end Code Base Preparation

As discussed above, the Angular framework were used to implement the frontend part of this project. So its CLI tools should be installed on the machine, they can be installed using the NodeJS package manger "NPM" (since they are an NPM package). The command to install them is:

After they got installed successfully, it's needed to create an Angular project using them. It can be created by running the following command:

Then a project create with sample template - it should be deleted later to complete implementing the project. But now the working of the project should be verified. To verify it, the Angular development server should be run using the following command:

And then it should browse the URL that the frontend serves on. This has done and the work verified successfully.

4.5.4 Git VCS and Github Hosting

To track the project changes, and keep the history of the code editing, the “Git” version control system were used separately for each project of the frontend and the backend.

For the frontend, Angular CLI tool creates a “Git” repository automatically when creating the project, but for the backend, it should be created manually by running the command:

After the repository created, we commit each change when progressing each more thing in the code. The whole work of the project were on a single branch; since there’s no collaborators other the project creator.

One of the benefits that we get from this step is that ability to go back to review and retrieve the old codes. In one case while implementing the project this benefit were a “life saver”, this case discussed in detail in the problems chapter – section 5.7.

Also the project repositories were hosted online on the “Github” service. This was performed by the following steps:

- Create two repositories on the “Github” service; one of them for the frontend, and the other for the backend
- Link each local repository to the corresponding remote repository. For each of them, the linking process can be done by running the command:

- When more commits created locally, they should be synced with the remote repository by running the command:

There are multiple benefits that could be leveraged from this technique:

- Ability to share the code with others, like with the project's committee.
- Ability to retrieve the code if the code machine were unavailable, or get corrupted.

4.6 Backend Control Design and Implementation

As discussed in the earlier sections of this chapter, the backend uses the REST architectural style, and it has been implemented using Django framework. Here's in this chapter there is a full detail about the server codes implementation, how the payment transition has been handled, the security techniques have been used, and the backend testing using the Postman.

4.6.1 Server Application Structure and Apps

In Django, the project is divided in multiple smaller pieces called "apps". There are multiple benefits that got from dividing the project into multiple apps like:

- Separate the logic into smaller pieces, which reduces complexity, and simolify the extensibility and the maintainability.
- Reusability: when separating the logic in multiple apps, each app itself is an integrated piece of logic, so it can be packaged, published to a package manager hub (like pypi which uses pip), and then it can be reused in another projects.

The following diagram illustrates the general structure of a Django project:

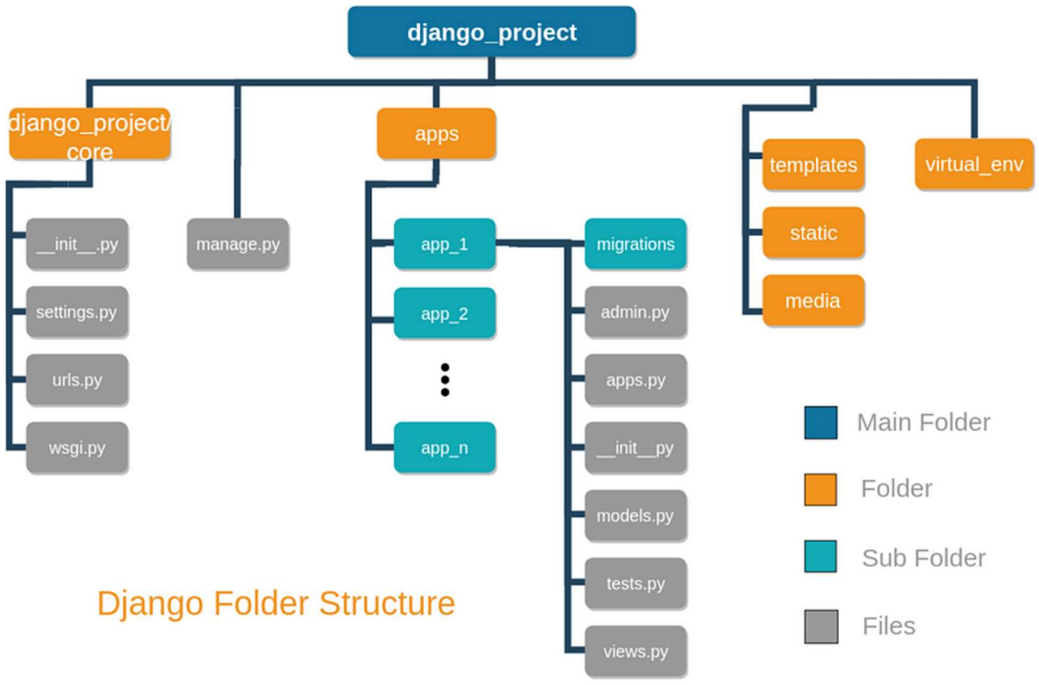


Figure 4.1 Django project folder anatomy

And here's a thorough look about each thing in the project folder, and explain how it used in this project:

- The "manage.py" file: this file used as the interface that we work with the built in CLI commands that pre-defined in Django, so this file doesn't altered in this project, it's just used to run CLI commands on the project. There are multiple code snaps in this chapter that are considered as an examples on the Django CLI commands that run

through the “manage.py” file like code snap 4.999999 and code snap 4.999999.

- The static, the media, and the templates folders: the static and the media are used to serve the static assets, like icons, static logos, videos, and background images. The templates folder is used to serve HTML templates for the pages that should serve from this server. However in this project the Django part is considered as only a restful API, and it’s completely decoupled from the front end server, so then no need for this folders in the Django project folder, and they haven’t been even created.
- The “virtual_env” folder: this folder is to contain the implementation details of the dependencies of the projects that are installed by the package manger, it’s special only for the dependencies of the virtual environment of this Python virtual environment, which has been discussed in the previous section.
- The “settings.py” file in the core folder of the project: It’s one of the most important file of the project. It contains the settings of the project, like the database type, the database credentials, the installed apps in the project, the server settings, the debugging configuration, the secret keys, and much more. In this report there’s multiple examples and code snaps of the usages of this file like code snap 4.99999, and code snap 4.99999.
- The “urls.py” file in the core folder of the project: which contains the global routing configuration of the project. In Django there are two types of routing: global routing, which this file is used for, and the app local routing; which performed in each app on its own. The routing in this project is discussed in detail in the first topic of this section.
- The apps folders: as discussed above; the Django project is divided into multiple parts called apps. Each app has a folder contains the files that are specialized in this app, and that will be discussed now.

App Anatomy and App’s Folder Structure:

The app in Django is a piece of logic that is specialized in manage the whole logic that related of a single thing, and by convention it’s named by the name of the thing that it manages. The App in Django follows the MTV style, which equivalent to the MVC architectural style in most of the other frameworks. So logic in the App is divided into main three parts, which are models, templates, and views, and there’s some helper things like App’s local routes, signals and handlers, decorators. Here is an overview of each of them, and after that there will be a complete sub-topic on each of them:

- Models: which are used to model the skeleton of the data. Each model is a represented as a class. They are implemented by the Django ORM. In Django ORM; each model is represented as a class that inherits the base model class in “dajngo.db” module. And each model in the Django ORM comes with built-in repository object (follows the repository pattern) which is used to perform queries in code.
- Templates: which are used for the HTML templates that will hold the data that served from the server. However, because in this project the Django part is only a restful API that used to serve the data, and not responsible for the representation; the templates haven’t been used. Instead of them, there’s a library to transform the data from models to JSON data, and it comes with the Django Rest Framework. This library called the Serializers, and it will be discussed in detail in a special topic in this section.
- Views: which are equivalent to controllers in the other backend frameworks. And they are responsible to handle the client requests, perform some logic that corresponds to

the request, and return him the appropriate response.

- App's local routes: which are legible to redirect the incoming requests to the appropriate views, depending on the URL that is addressed in the request.
- Decorators: which are used to add logic to the function or class in Python. In this project they have been built for the permission classes and the authorization. That will be discussed in detail in the Security section in this chapter.
- Signals and Handlers: they are used in Django to implement a decoupled communication mechanism between the apps. They are same of the concept of events in other languages, like C# and NodeJS.

As mentioned above, there will be a detailed topic on each of the utilized parts of the apps, but here there's two things to discuss before: the process of creating an app, and the utilized apps and their roles in this project.

Creating an App Process, and adding it to the project:

To create an app in Django, the Django CLI commands will be utilized. There's a command in Django CLI to create an App, and its boilerplate files. The command is:

```
python manage.py startapp <app_name>
```

After the App folder is created, its configuration should be registered in the installed apps list in the "settings.py" file as the following:

```
#settings.py
# Application definition

INSTALLED_APPS = [

    # Django built-in Apps
    # third party Apps

    '<app_name>.apps.<AppNameConfig>',


    # .... other Apps

]
```

The Apps in this project and the role of each of them:

Here's the list of the installed Apps in the "settings.py" file:-

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    'corsheaders',

    'rest_framework',

    'auth_app.apps.AuthAppConfig',
    'custom_users.apps.CustomUsersConfig',
    'custom_workspace_logging.apps.CustomWorkspaceLoggingConfig',
    'organizations.apps.OrganizationsConfig',
    'payment_app.apps.PaymentAppConfig',
    'provided_services.apps.ProvidedServicesConfig',
    'sevice_providers.apps.SeviceProvidersConfig',
    'sic_programs.apps.SicProgramsConfig',
    'users_notifications.apps.UsersNotificationsConfig',
    'users_services.apps.UsersServicesConfig',
    'workspaces.apps.WorkspacesConfig',
    'workspaces_groups.apps.WorkspacesGroupsConfig',
    'workspaces_managers.apps.WorkspacesManagersConfig',


]
```

And here's some explanation of the functionality of each of them:-

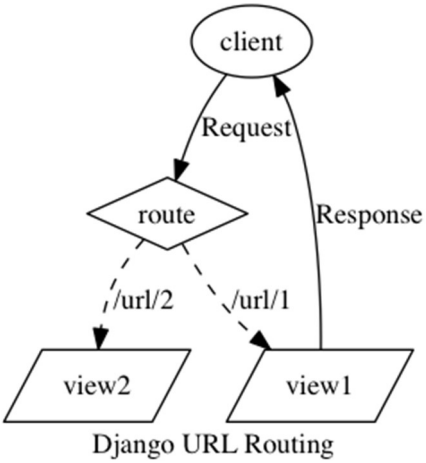
- The first 6 applications are Django built-in applications that helps in the development process.
- The "rest_framework" App: is an app for the official framework to work with the REST communication in Django. It has been discussed in detail in the previous section of this chapter.
- The "corsheaders" App: is an App to allow the requests with foreign CORS headers to enter the server. It will be discussed in detail in the CORS problem section in the next chapter.
- The "auth_app": is an App built to manage the authentication functionalities in the app. It has the whole capabilities for login, signup, and for mail validation. It has been built on the JWT authentication. More details about it in the security topic of this section.
- The "custom_user" App: is an app to manage the data and the functionality of the profile and permissions of the normal user role. It has been called "custom_user" to distinguish it from the Django built-in "User" model, which hasn't been used in this project.
- The "custom_workspace_logging" App: is a logging service that used to log the critical actions performed on the workspaces, in order to have ability to review them later. More details about it in the security topic in this section.
- The "organizations" App: is an app to manager the data, the permissions and the profile functionality of the Organization role in the project.

- The “payment_app”: is the app that handles the functionality of redirection after the payment. The payment process when it’s completed, it will redirect to a specific view in the Django server. This view is available in this app. It’s legible to extract the payment information from the request and sends a signal back to the interested app to handle the payment process. More details in the payment topic of this section.
- The “provided_services” App: is an app to enable the service providers to show their services with details in the marketplace. It manages the whole functionality of the CRUD operations on them with the suitable permission classes.
- The “sevice_providers” App: is an app to manage the profile functionality of the Service Provider role in the system.
- The “sic_programs” App: is an app to manage the subscription functionalities of the Service providers and the Organizations roles. Both of the roles should buy a subscription to the system in order to be legible to use its services. There were some default values numbers, and prices. Also the payment to subscribe is discussed in the payment topic of this section.
- The “users_notifications” App: is an app to manage to functionality of notifying the user of his interested things, like invitations to enter some workspaces. Note that notifications aren’t real time. This app communicates with the other apps by the signals.
- The “users_services” App: is an app to manage the functionality of the storing and managing the services of the user. It holds the whole CRUD functionality of the users’ services and their cards of the details. It also allow to store the secure information in an encrypted way, and it also controls the permission classes to access the services.
- The “workspaces” App: is an app to manage the workspaces functionality with the whole related CRUD statements to them – but not the specific things that related to the groups and to the managers, and with controlling the authorization with the suitable permission classes.
- The “workspaces_groups” App: is an app to manage the groups inside the workspaces; which are the working with the group local admins and the group members, and the processes of the sharing and the announcing in the groups
- The “workspaces_managers” App: is an app to manage the authorization and the data of the mangers in the workspaces.

Then, as mentioned above, there will be a sub-topic on each of the components of the Apps.

4.6.1.1 Routes

As mentioned above, the routes are used to forward the requests to the suitable views (which are the functions in Django to handle the requests, perform some processing, and returns responses back to the client – like the controllers in other backend frameworks) depending on the URL in the request. Here’s a diagram gives more clear view about the process:



Django URL Routing

There two types of routes in Django: global routing, and App's local routing. Here are them in detail:

- Global routing: and they are places in the "urls.py" file in the core folder of the project. They are usually used to match the route depending on the first part of it to the suitable app, or to a View if the route is just for a simple operation. Here's the code of the routes in the project as all:

```python
#urls.py
from django.contrib import admin
from django.urls import path, include

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

    path('auth/', include('auth_app.urls')),
    path('c_user_notifs/', include('users_notifications.urls')),
    path('c_user_servic/', include('users_services.urls')),
    path('orga/', include('organizations.urls')),
    path('pay/', include('payment_app.urls')),
    path('provided_service/', include('provided_services.urls')),
    path('sub_to_sic_prog/', include('sic_programs.urls')),
    path('workspaces/', include('workspaces.urls')),
]
```

Code Snap 4.99999: global routing configuration in the project

As shown in the in the Code Snap 4.9999, each works depending on the first part of the URL in the request. It forwards it to the router in the target App. It can be noticed that the routing configuration as all is listed as a python list, referenced by a variable called "urlpatterns"; which is a special syntax in Django. It can be noticed also that each element in this list contains a call to the path function, which a part of the "django.urls" package. The function first argument is a string that the URL should match, and the second is the View that needed to handle the request. But in case that it's needed to forward the request to another App's router, then the "include" function should called, and passing it the router of the target App.

- Apps' local routing: In each app there's also a "urls.py" file. Its structure is similar to the global "urls.py" file, but instead of matching the first part of the URL in the request to the appropriate App router, it matches the other parts of the URL to the appropriate View.

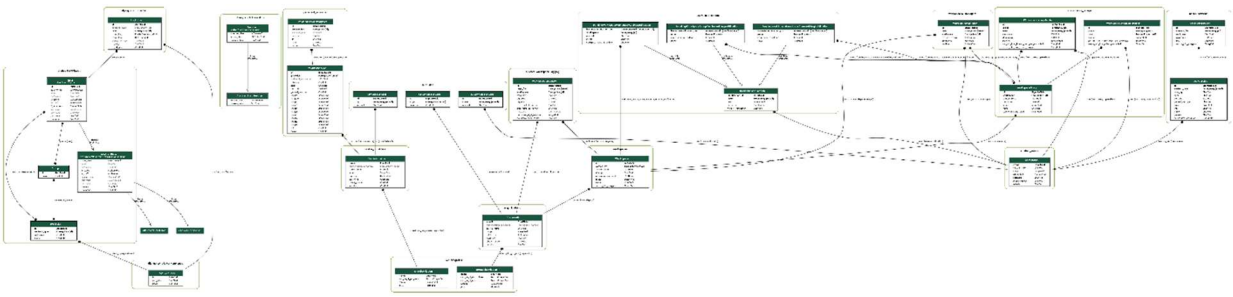
Note that Django by default if the URL in the request isn't match any route, it returns by default the "HTTP_404_NOT_FOUND_ERROR"; so no need to handle these cases in code.

4.6.1.2 Models

Models in Django define the skeleton structure (the entities) of the Data stored in the database. Each entity is represented using a class that inherits from the model base class. Here in this topic: first discuss the ERD diagram of the data in the project, and then discuss sample codes that cover all models techniques used in this project.

ERD diagram of the data in the project:

To represent a diagram defines the whole data of the project, the common ERD (Entity Relationship Diagram) has been used, here's the image of the diagram, knowing that it's auto-generated by a Django extension that designed for graphing techniques, check more about it in the references. (Note that due to its big size, it can't be fitted as all in a single place, so here's a picture for it, and it can be downloaded or viewed online to look at it in bigger size from the following link: https://imgur.com/a/gblu0fb)



Note that all models are encapsulated into their App.

Coding the models, and querying them:

As mentioned above, each model is represented in Django with a class that inherits from the base "models.Model" class, which is a class declared in the "django.db" package.

All of these classed should be strictly declared in the "models.py" file in each App. This point is very important to be recognized by Django when the migrations is created. In each App we encapsulate the necessary models to the certain point of logic, which is the App is specialized in. Inside this class, the fields of the class should be declared.

Here’s some sample codes of the class declarations, with the illustration of the codes. To cover all pieces of code that used in work with models, codes of 4 models that contains samples from everything used were been selected; the “Workspace” model, the “WorkspaceManager” model, the “WorkspaceGroup” model, the “BaseUserNotification” model, and one other of the user’s notifications models.

First, take a look to the “Workspace” data model:

```python
#workspaces\models.py
import os
from django.db import models

from utils.image_validators import validate_image_size_default_5mb
from custom_users.models import CustomUser
from organizations.models import Oragnization
from cryptography.fernet import Fernet
from servic_innovat_sync import settings

ws_sImgsPath = os.path.join('images', 'workspaces')


class WorkSpace(models.Model):
    owned_by = models.ForeignKey(Oragnization, on_delete=models.CASCADE)
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=511)
    descripe = models.TextField(max_length=1000)
    announcments = models.JSONField(default=list)
    general_announcments = models.JSONField(default=list)
    secure_info_encrypted = models.BinaryField()
    image = models.ImageField(upload_to=ws_sImgsPath, null=True, validators=[
                              validate_image_size_default_5mb])
    managers = models.ManyToManyField(
        CustomUser, through='workspaces_managers.WorkspaceManager')

    @property
    def secure_info(self):
        key = settings.ENCRYP_SECRET
        cipher_suite = Fernet(key)
        binary_data = self.secure_info_encrypted.tobytes()

        secure_info_decrypted = cipher_suite.decrypt(
            binary_data).decode('utf-8')
        return secure_info_decrypted

    @secure_info.setter
    def secure_info(self, value):
        key = settings.ENCRYP_SECRET
        cipher_suite = Fernet(key)
        self.secure_info_encrypted = cipher_suite.encrypt(
            value.encode('utf-8'))
```

Let’s discuss the basics of the codes used in this class:

-

4.6.1.3 Views

4.6.1.4 Serializers
4.6.1.5 Signals and Handlers
The y……..

4.6.2 Utilities
The ……

4.6.3 Payment
The …….

4.6.4 Security
4.6.4.1 Authentication, Authorization, and Permission Classes
4.6.4.2 Email Safety
4.6.4.3 Email Security
4.6.4.4 Links Security
4.6.4.5 Sensitive Information Secure Storage
4.6.4.6 Logging Service

4.6.5 Testing
To test the correctness for the functionality of the backend, the …….

4.7 System Administration

The. …….

4.7.1 Django Administration
The …….

4.7.2 Django REPL
The ……

4.8 Frontend Control Design and Implementation

The …….

4.8.1 Sth….

The …….

4.8.2 Sth…..

The ……..

4.8.2.1 Sth ……

5. Results And Problems Discussion

5.1 PgAdmin refuse connection to the database

First,…….

5.2 Missing Dependency to Connect to PostgreSQL from Django

First, …….

5.3 Missing Dependency to Store Images

When ……

5.4 Failure in Migrations After Updating Data Models

When ……

5.5 PayPal Braintree payment doesn’t work

At first, the to

5.6 Http request from the Angular application to the Dajngo server blocked by CORS

At first, the to

5.7 Huge number of codes files deleted by accident

At first, the to

5.8 Fernet Problems

At first, the to

5.9 Styles in the material components

At first, the to

5.10 Asynchronous server operations gives errors

5.11 Signals' Handlers doesn't work

At first, the to

5.12 Final Result

The end result is a …….

6. Conclusion

6.1 Summary

created …….

6.2 Improvements

This ……..

6.3 Future Work

As for ……

6.4 Outcome

have …….

Bibliography

Resources for the online simple market research in section 4.2:

1 - The Ultimate List of Marketing Statistics for 2022 (Link: https://www.hubspot.com/marketing-statistics)

2 - Digital strategy in a time of crisis (Link: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/digital-strategy-in-a-time-of-crisis)
3 - Gartner Top Strategic Technology Trends for 2021 (Link: https://www.gartner.com/smarterwithgartner/gartner-top-strategic-technology-trends-for-2021)
4 - Build for the Future with Innovation Success (Link: https://www.linkedin.com/comm/pulse/build-future-innovation-success-boston-consulting-group?lipi=urn%3Ali%3Apage%3Aemail_email_series_follow_newsletter_01%3B2B%2FEJOnMR3eXSKCIeXVUuA%3D%3D&midToken=AQFBF9FPhe_RjA&midSig=0nZsH-kQNpWaQ1&trk=eml-email_series_follow_newsletter_01-newsletter_content_preview-0-readmore_button_&trkEmail=eml-email_series_follow_newsletter_01-newsletter_content_preview-0-readmore_button_-null-if2sm9~lkl7jk4r~7r-null-null&eid=if2sm9-lkl7jk4r-7r)

Resources utilized while implementation:

1 – ChatGPT Language optimization model (Link: https://chat.openai.com)

2 – PostgreSQL official download (Link: https://www.postgresql.org/download/)

3 – PgAdmin official download (Link: https://www.pgadmin.org/download/)

4 – Generate ERD Diagram for Django Project (Link: https://www.youtube.com/watch?v=yvf_J225iM8)

Appendix A

Attachment A

Proposed Disclaimer Statement Format The report is a document written by the student(s). It should reflect expertise in research methodology and technical writing skills. The supervisor's job is to guide the student so that she/he can achieve the objectives in an efficient way while gaining the skills sought. While maintaining credit the disclaimer statement is simply a statement protecting the Department and the University from any legal liability claims associated with the use of the results and the methods presented. Its format is as follows: DISCLAIMER This report was written by Ibraheem Halasah at the Computer Engineering Department, Faculty of Engineering, An-Najah National University. It has not been altered or corrected, other than editorial corrections, as a result of the assessment and it may contain language as well as content errors.