BACKEND

What is Backend ?

The backend (or “server-side”) is the portion of the website you don’t see. It’s responsible for storing and organizing data and ensuring everything on the client-side works. The backend communicates with the frontend, sending and receiving information to be displayed as a web page. It is the portion of software that does not come in direct contact with the users. The parts and characteristics developed by backend designers are indirectly accessed by users through a front-end application. Activities, like writing APIs.

Technologies we used

Python:

Python is a high-level, interpreted, general-purpose programming language. Python was created in 1991 by Guido Van Rossum. The best feature of python is its code readability which makes the language user-friendly and easier to use. Python is also used as a backend language for website and application technologies.

Django:

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. Django is a web framework, where python is used as the backend, and Django is used as communication between the frontend and backend through API.

Django Rest Framework:

Django REST framework is a powerful and flexible toolkit for building Web APIs.It is a wrapper over the default Django Framework, basically used to create APIs of various kinds. There are three stages before creating an API through the REST framework, Converting a Model’s data to JSON/XML format (Serialization), Rendering this data to the view, and Creating a URL for mapping to the viewset.

Docker:

Docker is an open-source containerization platform. It enables developers to package applications into containers—standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment.

Redis:

Redis is an in-memory data structure store, used as a distributed, in-memory key-value database, cache, and message broker, with optional durability. It's also perfect for real-time data processing. Redis supports different kinds of abstract data structures, such as strings, lists, and maps.

Packages we used.

Django Rest Framework:
Django REST framework is a powerful and flexible toolkit for building Web APIs.It is a wrapper over the default Django Framework, basically used to create APIs of various kinds.

Djangorestframework-simplejwt:
JSON Web Token (JWT) is an open standard that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWT is used to create access tokens for an application.

Redis
Redis is an in-memory data structure store, used as a distributed, in-memory key-value database, cache and message broker, with optional durability. Redis supports different kinds of abstract data structures, such as strings, lists, maps

Celery
Celery is one of the most popular background job managers in the Python world. Celery is compatible with several message brokers like RabbitMQ or Redis and can act as both producer and consumer. Celery-Beat is used for periodic tasks.

Dlib
Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems. It is used in both industry and academia in a wide range of domains including robotics, embedded devices, mobile phones, and large high performance computing environments.

Why do we use Django and It's advantages?

Versatile:
Django can be used to build almost any type of website — from content management systems and wikis to social networks and news sites. It can work with any client-side framework and can deliver content.

Secure:
Django provides a secure way to manage user accounts and passwords, avoiding common mistakes like putting session information in cookies where it is vulnerable

Scalable
Django uses a component-based "shared-nothing" architecture (each part of the architecture is independent of the others, and can hence be replaced or changed if needed)Having a clear separation between the different parts means that it can scale for increased traffic by adding hardware at any level: caching servers, database servers, or application servers. Some of the busiest sites have successfully scaled Django to meet their demands (e.g. Instagram and Disqus, to name just two).

Maintainable
Django code is written using design principles and patterns that encourage the creation of maintainable and reusable code. In particular, it makes use of the Don't Repeat Yourself (DRY) principle so there is no unnecessary duplication, reducing the amount of code. Django also promotes the grouping of related functionality into reusable "applications" and, at a lower level, groups related code into modules