Coding Standards Documentation

Coding Standards for
Domain Pulse

Ctrl Alt Defeat

Contents

1 Coding Standards

1.1 File Structure

Our system (Domain Pulse) has multiple different services and components that are often times independant from one another, as a result of our service oriented architecture. Furthermore, our system makes use of a Angular frontend and Django backend. As a result of these factors, the file structure of our system is designed to with modularity and simplicity as focuses, and we have achieved such. These qualities have been achieved with there being three main folder paths of note within our system: 'backend', 'frontend' and 'documentation' (of which will not be covered as it is not 'Coding'). Each folder is self-explanitory in its purpose and all code relating to said folder is contained within it, with little to no overlap.

1.1.1 Frontend

The frontend folder contains a folder 'src' containing the developed resources. The 'assets' folder conveniantly stores all assests (icons,backgrounds, etc...) needed for the UI development. The 'app' folder contains all developed Angular UI pages, neatly stored in their corresponding component folders (login-page, register-page, etc...)

1.1.2 Backend

Within the backend folder, every seperate service of the system is stored within the relevant folders. This allows for our clearly defined and seperated software oriented architecture. This ensures modularity and flexibility within our system by clearly defining individual services. Each of these services folders will contain one folder storing the needed Django settings and others containing the developed code needed for the proper execution of the service.

1.2 Naming Conventions

1.2.1 Frontend

1.2.2 Backend

Within the backend of the project naming conventions are followed to ensure readable and understandable code. These include a well defined structure for endpoints whereby the endpoint function shall have a descriptive name. The URL shall be the same as the endpoint name. Auxiliary files are created to store the logic of the functions within the backend and these are stored in folders such as 'util' or 'preprocess' to give a clear understanding of their grouping of purpose. The functions within these auxiliary files include functions with the same name as the endpoint (indicating they are what is to be called in the endpoint) and helper functions such as 'remove_whitespace' to create an orderly structure within the main functions. This clear definition of different main and auxiliary functions increases readability and flexibility of our code as it leads to bugs being easier to 'track down'. As well as this variables are ensured to name that clearly describes their purpose within the codebase.

1.3 Formatting

Hellooo