Technical Documentation

Group: JARS with an E

Project: Napoelon's Adventures

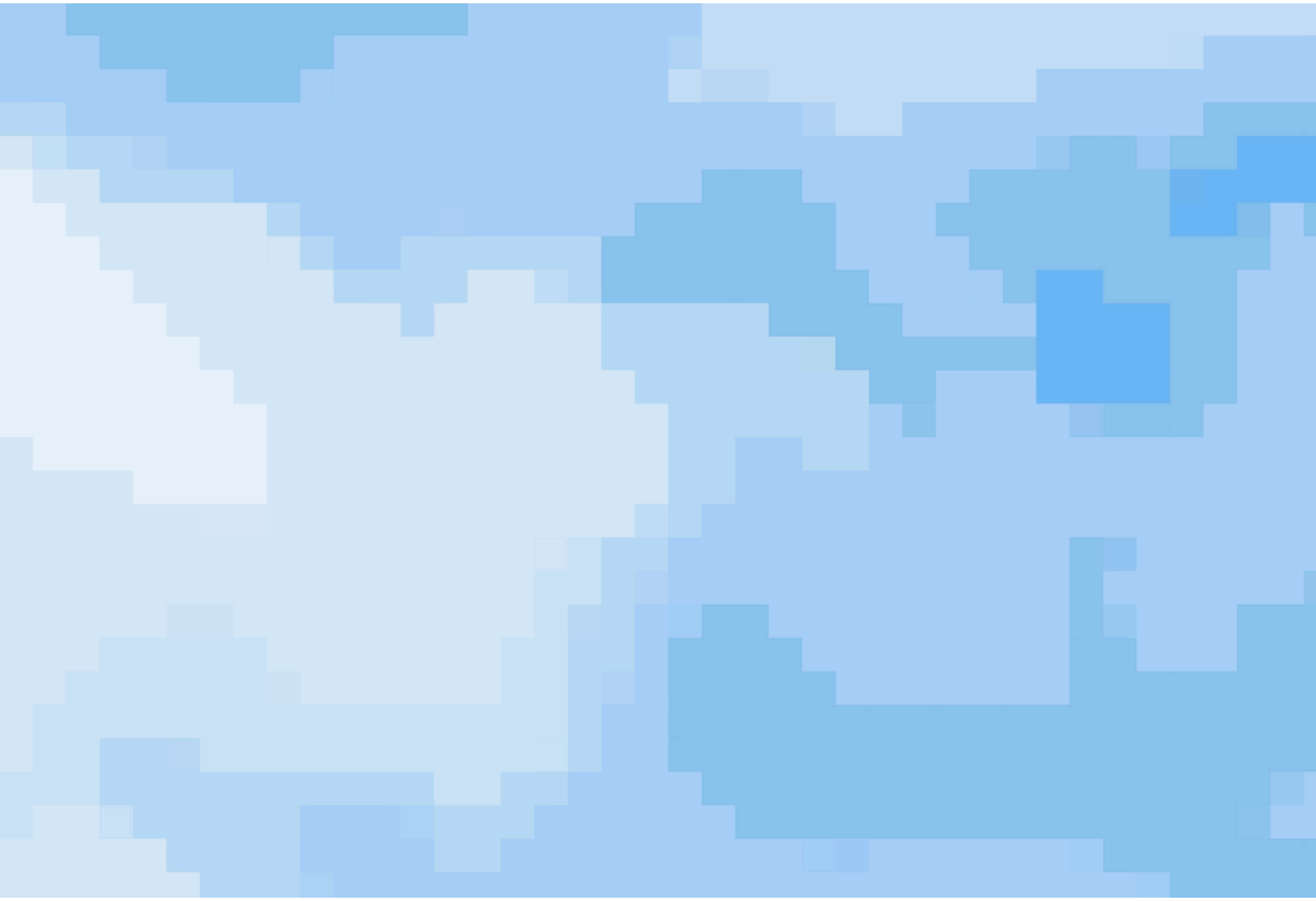

Table of Contents:

Introduction: 1
System Architecture: 1
APIs and Endpoints: 2
Deployment: 3
Design Choices: 3

Introduction:

This project uses the Django framework to host a website which, upon the input of the 6-digit code (1 for each game), allows users access to a series of recycling-themed minigames.

System Architecture:

The main components of the project are as follows:

- Django project:
    - Project folder (napoleons_adventure)
    - Core app
    - Minigames app
    - The database (db.sqlite)
    - Manage.py
- QR code folder(location-qr)

Project folder:
This folder contains the core configurations and settings for the project.

Core app:
This app handles the main webpage (everything on the website other than the games).

Minigames app:
This app handles the minigames: recycle rush, sea sweepers, sort the recycling, sort n serve, and whack-a-waste.

The database:
The database stores the following:

- Location[id, name, access_code, x_percentage, y_percentage, image_name, minigame_id] - This model stores the information necessary to display a location on the map, and which game is connected to that location.
- Minigame[id, name, intro_url_name] - This model stores the information necessary to access the intro page of a page.
- Profile[id, profile_image, user_id] - This model stores the information necessary to display a user's profile.
- Userprogress[id, completed, location_id, user_id] - This model represents whether or not a user has beat an individual minigame.
- Gamescore[id, score, minigame_id, user_id] - This model represents a user's high score in an individual minigame.

This project also uses several models imported from django.contrib.auth.models for matters of account management and authorisation.

Manage.py:
This file is used for managing the Django application, including database migrations, running the server, and other administrative tasks.

QR code folder:
This folder contains the code needed to create a QR code to store the 6 digit codes for the games.

APIs and Endpoints:

The following are the APIs and endpoints used by this project:

- POST register - Registers a new user with a given username, email, and password. The user account is created with the is_active status set to False, and an email confirmation is sent for account activation.
- POST login - Authenticates the user by using a username and password. Upon successful authentication, the user is logged in.
- POST logout - Logs the user out by clearing the session and authentication credentials.
- POST user-profile - Updates the user's profile information such as name, email, and other fields.
- POST delete-account - Deletes the user's account from the database.
- GET game-description - Returns the description and sustainability theme of the game based on the specified location.

- POST save-score - Saves the user's game score to the database. If a score already exists, it is updated if the new score is higher.
- GET leaderboard - Retrieves the top 10 users based on their total score across all games.

The next few are specific to their individual games, but have the same effect (gamename is to be replaced by str, recycle_rush, sea_sweepers, whack-a-waste, and sns):

- GET gamename_intro - Displays the introductory screen for the game.
- GET gamename_game - Starts the game.
- POST gamename_endDisplays the final screen and allows saving the score.

Deployment:

In order to deploy this project, please ensure that you have followed the instructions of requirements.txt:

Local deployment:
Entire the following command into a terminal in the same folder as manage.py:
python manage.py runserver

Online deployment:
Our web application is deployed on an EC2 instance hosted by AWS. This will enable users to interact with our web application. We launched an EC2 instance, installed Nginx and Gunicorn, and configured them to handle our Django application. Nginx accepts incoming HTTP requests, passes them along to Gunicorn to serve the app. We cloned our GitHub repository to our EC2 instance and set up directories for myenv (the virtual environment for project dependencies) and staticfiles (where we store static files such as CSS, JavaScript, and images), where which also Nginx will serves static files from.

Design Choices:

Using PyScript to code the logic of two of the minigames:
Reason - The developers of sort the recycling and recycle rush were more familiar with

Python than JavaScript, and therefore felt it would be more time-efficient to code using PyScript rather than learning JavaScript.