Coding Challenge: by Abhinav Manoj Menon

The web app is split into two folders.

1. Webapp – (Containing the Front End of the application built on React JS)
2. Server – (Containing the Back End of the application built on Flask)

Note: Run the Server before running the Webapp. The documentation on how to start the applications is clearly described in the Readme file under the respective folders.

API DESIGN AND STRUCTURE

1. The Backend **Flask** application is built using the Domain Driven Design Principles.
2. The API is split into different files controllers, models, services and best practices of REST apis is followed
3. **SQLite** Database is used to store **County Data**, **Zip codes** and **Population** as server-side DB.
4. Table **countypopulation** is created in the DB.
5. **SQL Alchemy** is used as the ORM to query the database.
6. The test framework **Pytest** is used to test the end points of the API.

ENDPOINTS DESCRIPTIONS

This backend application has two endpoints

1. **/create_phrase**: This endpoint is used to convert name into Pig Latin form along with population and county name and return as a JSON.
2. **/getzipcodes:** This is used to get all zipcodes from the table and return them as a JSON response.

FRONT-END DESIGN AND STRUCTURE

1. The Front-end application is built on **React JS**.
2. Various services used are kept in separate files.
3. **Material-Ui** is used to create the user interface.
4. The application is built keeping in mind the best practices of React JS.
5. Testing is done using **Jest** to create Mock Fetch Apis.
6. Data validations are done before submitting the input.
7. The required messages are displayed on screen if the input from the user is not in the desired format.
8. The output is displayed on a dialog box.

EXTRA FEATURE IMPLEMENTED(AutoFill)

The extra feature that is implemented is an Autofill feature to help user fill in the Zip code. Instead of using a usual Textbox to gather input we have used a drop down to choose from which is dynamically generated. The suggestions are generated once three numbers of the zip code are typed in.

DATA SOURCE

The data for the server-side database is created using the data obtained from the below website:

https://simplemaps.com/data/us-zips