Boston University
Electrical & Computer Engineering
EC464 Senior Design Project

Second Prototype Testing Plan

WhereTo

by
Team 5

Team Members:
John Burke - jwburke@bu.edu
Muhammad Ahmad Ghani - mghani@bu.edu
Haochen Sun - tomsunhc@bu.edu
Erick Tomona - erickshi@bu.edu
Marta Velilla - martava@bu.edu

<u>Required Materials</u>

<u>Hardware:</u>

- Personal Computer (MacOS preferable)
  - To host both the database and API, as well as the React Native application
- Personal Smartphone (iOS preferable)
  - To utilize the frontend of the application

<u>Software:</u>

- Python Park API
  - This is hosted on our machine, makes use of ML models as well as algorithmic design and API calls in order to analyze parking information for the user
- React Native UI
  - This is hosted on our machine, React code to display our user interface on mobile
- Expo Go
  - A mobile application designed for testing react native applications on iOS
- SQLite Database
  - This is a cache, to reduce excessive calls to the Google Services APIs

<u>Set-Up</u>

This test requires a computer and a smartphone with Expo Go installed. The Python API as well as the React Native application are both to be hosted on the personal computer. The smartphone is then used to download the React Native application and begin testing the backend. Once configuration is complete, the backend should be accessible from the frontend seamlessly to execute testing. The database should require no additional setup beyond the setup already necessary to host the backend.

<u>Pre-Testing Setup Procedure</u>

ParkAPI:

1. Ensure that relevant dependencies are installed on the file system where we are running the API
2. Ensure that the necessary API keys and URL configurations are present in the config.py file in the root of the application
3. Run the application by navigating to the root directory of the API in terminal and executing: ‘python3 app.py’
   a. This will run the application on every open host possible, so we can access it through our React Native Application

React Native UI:

1. Ensure the smartphone has the Expo Go application installed
2. Once the ParkAPI is being hosted, it will output its URL, ensure this is the same baseURL being used by the axios object within the React Native requests
3. Once the correct URL is present, run the application by executing: ‘npx expo start’

    a. This will output a QR code to the terminal, scan this and open Expo Go to download and run the application

Database:

1. Nothing must be done for the database, it is connected to the ParkAPI. If there is no whereto.db file present within the ParkAPI/database folder, use the queries in ParkAPI/database/whereto.sql to generate the database tables for population

<u>Testing Procedure</u>

1. For each address and radius combination for which we want to examine:
    a. On the InputDisplay screen for the UI, enter a valid address string and radius value
    b. Press the "Find Parking" button on the UI
    c. Wait for the backend to finish completing the request
    d. Examine the UI/map output present after the request is done processing
2. Additionally, we should perform each combination of inputs twice, to ensure that proper caching is being performed; that is, we are not performing duplicate calls to the Google API services if we have analyzed them previously/recently

<u>Measurable Criteria</u>

The measurable criteria for the Python API is as follows:

I. The Python API should be capable of analyzing the streets in a given area, this is validated by a local file in the map/ directory which displays markers on every analyzed coordinate
II. The Python API should be capable of using the pretrained model to make and deliver predictions with a confidence level of possible parking meters in the area as a JSON response. This is also verified partially in the map/ directory, where markers displaying detections are placed
III. The Python API should return an error in the case of an invalid parameter, and should be capable of receiving multiple requests in a row and processing them accurately to the requirements of the application

The measurable criteria for the React Native UI is as follows:

I. The React Native UI should allow for the user to input a radius and address that they desire
II. The React Native UI should be capable of graphically displaying the JSON response in the case of a successful API procedure
III. The React Native UI should be capable of handling and informing the user of an error in the case of an unsuccessful API procedure

The measurable criteria for the SQLite Database is as follows:

I. The SQLite Database should be used instead of a duplicate call to the Google API services and machine learning model if we have examined a coordinate before, this is going to be displayed/shown through API logging