TCAT Tracker Code Listing

- The TCAT Tracker project adopts a Single Page Application (SPA) structure [1].
- The codebase is divided into two main components: Backend and Frontend.
- The Backend implements the core business logic, such as computing routes for a given trip ID, managing notifications (e.g., email alerts), and serving the static assets. It is built using Python [2] and Flask [3].
- The Frontend handles the user interface and visualization logic, including rendering graphs and implementing interactive features like pop-ups (e.g., the "Notify Me!" button). It is built using React.js [4].
- The entire source code is packaged as a .zip file and attached along with this report

Folder Structure

Directory Name	**Description**
`Backend`	Contains the Python code for the server, managing API end-points and back-end logic.
`Frontend`	Contains the JavaScript code for the website, handling user interface and interactions.
`tcat-ny-us`	Static files containing the meta data of various stops, routes & etc.
`Test-scripts`	Scripts to verify connectivity to the GTFS-API.
`website`	Project Website (using ECE 5725 Template).

File-Level Listing

2.1 Backend

`All files specified below are located within the backend directory`

File Name	**Description**
`__init__.py`	A special file that marks a directory as a Python package. It initializes the package namespace, can contain initialization code that runs when the package is imported, and controls how the package's modules are exposed.
`app.py`	Entry point for the application. Launches the server, provides API endpoints for bus tracking and notifications, and serves the front-end.

`app_config.py`	Singleton class that reads the `Dataset.zip` file to compute routes for a given trip ID. The computed path is used by the front-end to list stops.
`dataset.zip`	Metadata related to stops, buses, and trips (e.g., stop names, route numbers, schedules). Available for public use.
`notification_manager.py`	Manages notifications (e.g., emails).
`Output.log`	Server logs for debugging.
`requirement.txt`	List of Python packages used in the server code.
`routes.csv`	Makeshift database file. Each entry corresponds to a notification, managing routing information.

2.2 Frontend

`All files specified below are located within the frontend directory`

File Name	**Description**
`build`	JavaScript bundle generated by React.js, used by the server to launch the website.
`public`	Static assets (e.g., logos, images) used by the front-end.
`src/App.css`	Local style sheet for the main application.
`src/App.js`	Main entry point for the website.
`src/NotificationButton.jsx`	Front-end code for the "Notify Me!" button and its corresponding popup.
`src/StopsGraph.jsx`	Front-end code for rendering the main graph.
`src/StopsViewer.jsx`	Primary entry point for the TCAT Tracker website. Houses the route selector, "Notify Me!" button, and the StopsViewer graph.
`src/StopsViewer.css`	Local style sheet for the StopsViewer component.
`src/index.css`	Global style sheet for the entire application.
`src/stops.js`	List of stop names, used for the typeahead component in the Notify Me button.
`node_modules`	Directory containing all third-party dependencies installed via npm.
`package.json`	List of dependencies used in the front-end.

Project Root

`All files specified below are located within the root directory`

File Name	**Description**
`README.md`	Project documentation including setup instructions, architecture overview, and deployment guidelines.
`.gitignore`	Specifies which files and directories should be ignored by Git version control (e.g., node_modules, build artifacts, environment files).

`cmd.sh`	A bash script that configures and launches a Chromium browser on a Raspberry Pi with PiTFT display. It sets up the framebuffer, checks for necessary components, and starts the browser in kiosk mode with specific dimensions and scaling.
`start_server.sh`	A bash script that prepares and starts the Flask server. It kills any existing process on port 5000, launches the Flask application in the background with logging, saves the process ID, and opens port 5000 using UFW. Requires SUDO access
`.env`	A configuration file that stores environment-specific variables as key-value pairs. It is used to manage sensitive information of API keys for Sendgrid[5] service.

References

[1] Single Page Application (SPA), `https://developer.mozilla.org/en-US/docs/Glossary/SPA`

[2] Python Programming Language, `https://www.python.org`

[3] Flask Web Framework, `https://flask.palletsprojects.com`

[4] React JavaScript Library, `https://react.dev`

[5] SendGrid, `https://sendgrid.com/en-us`