Planning Document

Dataset and Task

The dataset consists of multiple txt files, each representing different aspects of the TransLink public transport system in South East Queensland. The key tables include:

- **calendar_dates.txt**: Contains special service dates and any exceptions to the regular schedule, identified by service_id, date, and exception_type.
- **calendar.txt**: Defines the operational schedule of services, with columns representing the days of the week (monday to sunday) and the period of service (start_date, end_date).
- **stops.txt**: Lists all bus stops, including details such as stop_id, stop_name, stop_lat, and stop_lon, which provide the stop's identifier and its geographical location.
- **routes.txt**: Contains information on different bus routes, such as route_id, route_short_name, and route_long_name, providing a descriptive overview of each route.
- **trips.txt**: Specifies the trips made by the vehicles, linked to the route_id and service_id, and includes details like trip_id and trip_headsign.
- **stop_times.txt**: Provides the schedule times for stops, detailing arrival_time, departure_time, and the sequence of stops (stop_sequence) for each trip.

In addition to the static data files, there are three JSON files that represent dynamic data fetched from a live API, providing real-time updates on public transport:

- **trip_updates.json**: Contains real-time information on trip updates, including delays, changes in schedules, and other trip-related information.
- **vehicle_position.json**: Provides live data on the current position of vehicles, including their latitude, longitude, and other movement-related data.
- **alerts.json**: Includes real-time alerts about the service, such as disruptions, cancellations, or important notices affecting the transport services.

These JSON files are used to enhance the static data with real-time updates, enabling the application to provide accurate and timely information to users.

Task: Create a terminal-based Node.js application that parses TransLink data and provides a summary of all bus route stops in South East Queensland. The application must be built using functional programming principles and include features like data filtering, fetching, and caching.

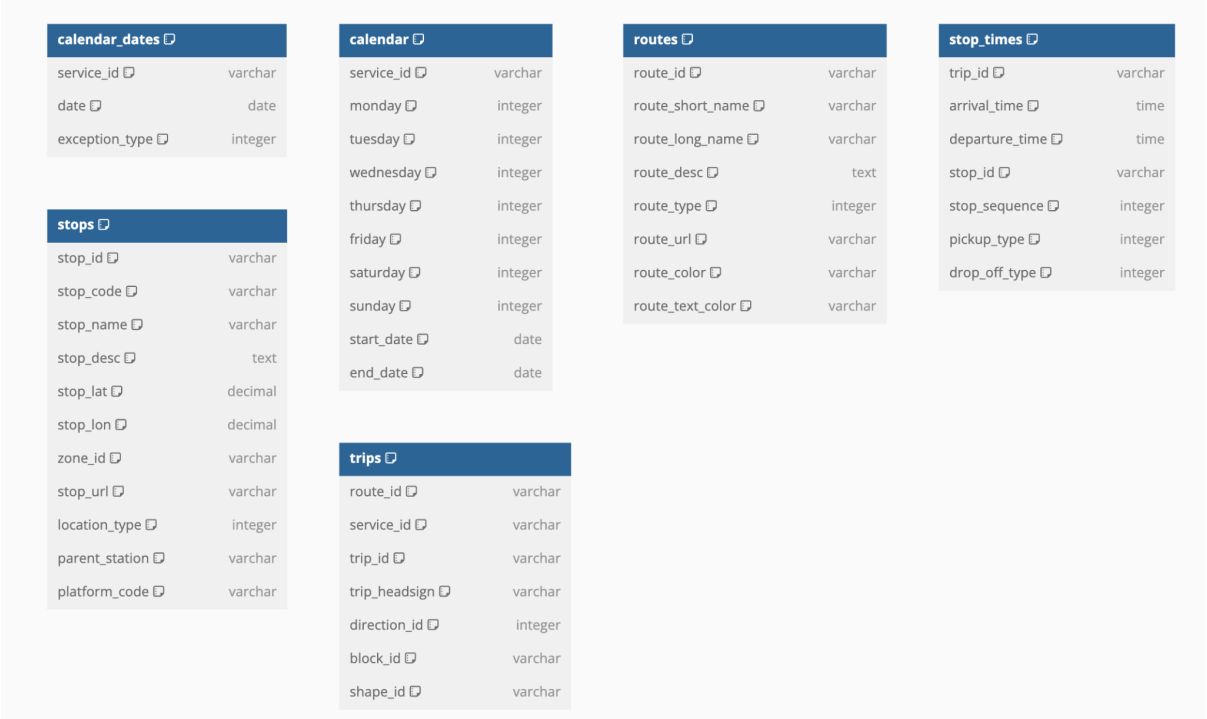

Figure 1. Static-Data Table

Important Fields and Joins

- **route_short_name**: Key field for checking the existence of a specific route in the routes.txt dataset.
- **scheduled_arrival_time**: Field used to filter buses arriving within a specific time frame.
- **dateInput**: Field used to validate the correct date format.
- **timeInput**: Field used to validate the correct time format.
- **currentTime**: Reference time used to compare against scheduled_arrival_time for filtering upcoming buses.

Steps to Perform the Task

1. **Load Route Data**: Load the route data from the routes.txt file.
    - **Action**: Use loadCSV to load the dataset containing routes.
2. **Validate Route Number**: Check if the provided route number exists in the loaded route data.
    - **Action**: Use some method to search for the route_short_name in the loaded data.
3. **Validate Date and Time Formats**: Ensure that the input date and time follow the correct formats (YYYY-MM-DD for date, HH:MM for time).
    - **Action**: Use regular expressions to match the date and time formats.
4. **Run Route Planner**: Execute the route planner function to retrieve the scheduled bus arrivals.
    - **Action**: Use runRoutePlanner to obtain bus arrival data.
5. **Filter Buses by Timeframe**: Filter the buses that are scheduled to arrive within 10 minutes of the current time.
    - **Action**: Compare scheduled_arrival_time from the bus data with currentTime to filter relevant records.
6. **Check Filtered Data**: Ensure that at least one bus meets the arrival timeframe criteria.
    - **Action**: Use an assertion to check that the length of the filtered data is greater than zero.

Reference
ChatGPT was use to convert the static data file to the DBML format.