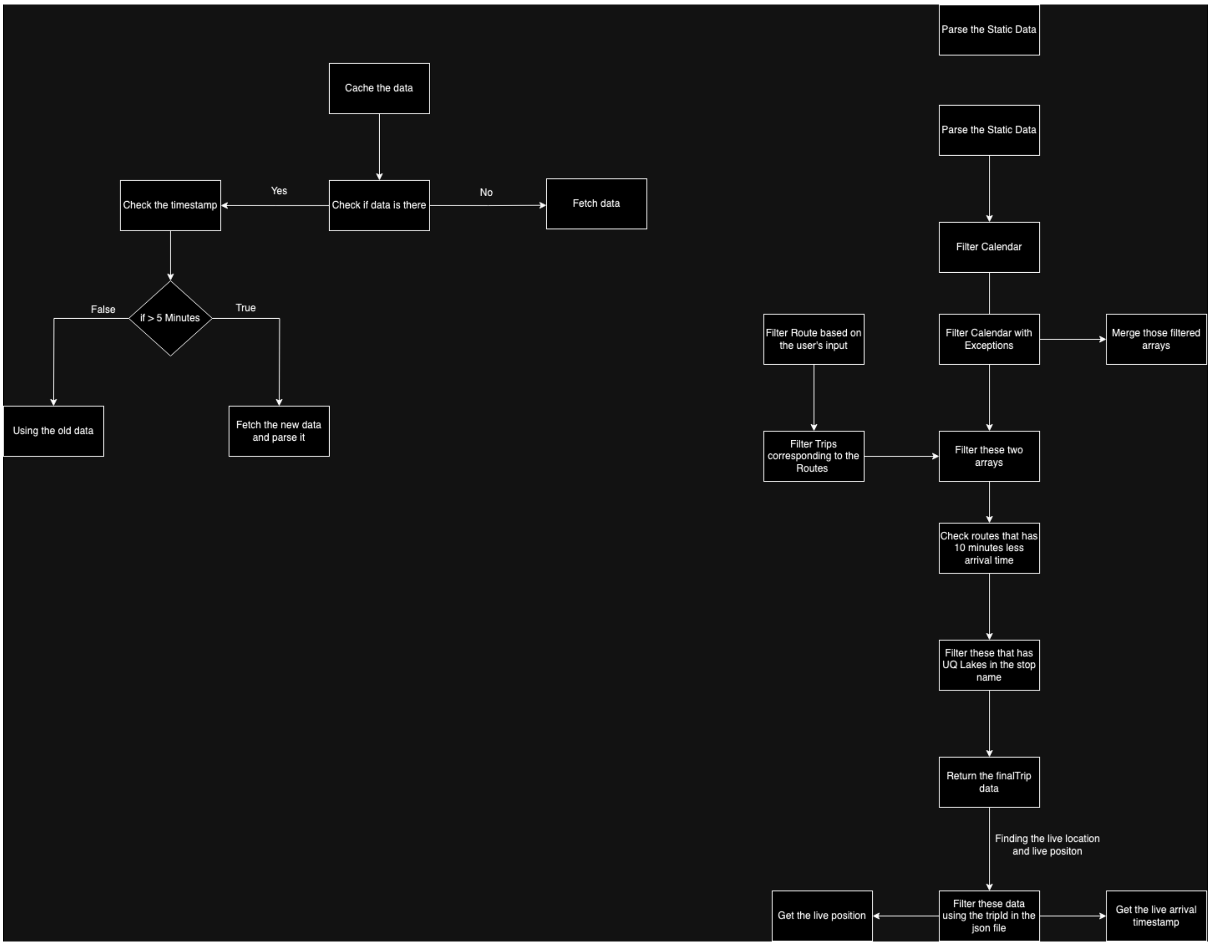
Winaldo Amadea Hestu - s47670622
Javascript Functional Programming

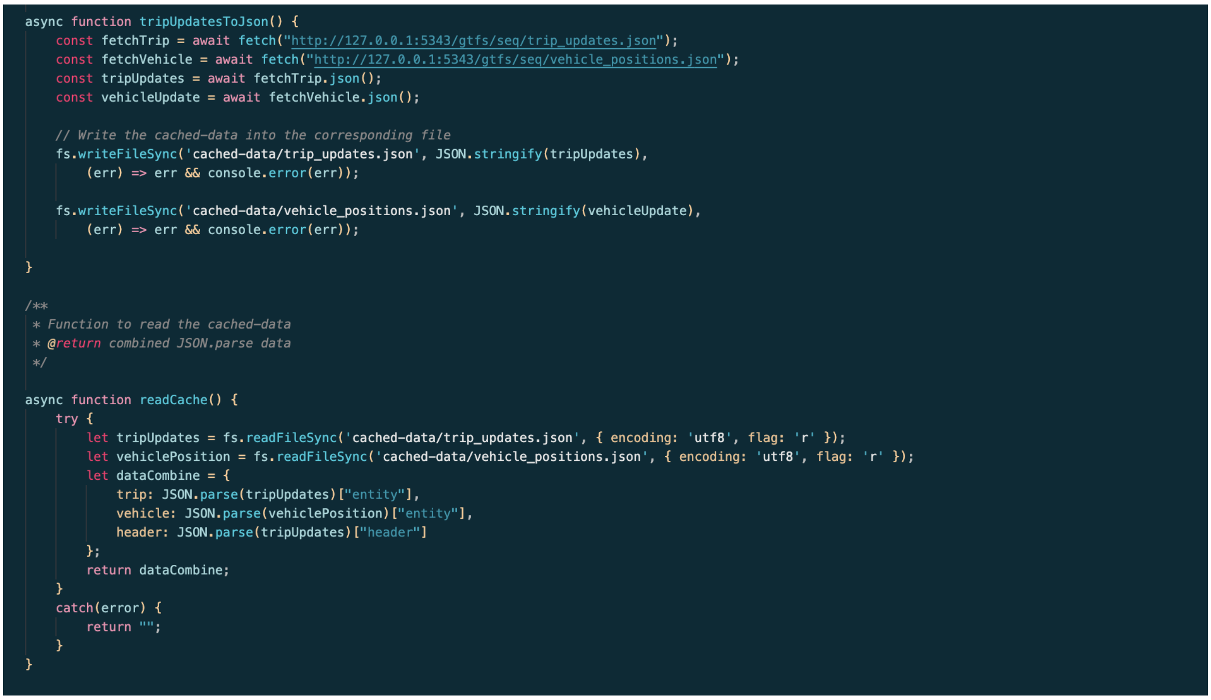
On doing this project, to be honest I tried a couple ways first before finally using these algorithms. I was trying to find the routes first and filter it and find the trips, but I found difficulty on doing the **calendar_dates.** I found it a bit hard to combine it with the **calendar**, after that I tried to do it again from the first and I tried to filter those exception types first and combine the serviceId with the one in the calendar.


Cache the data
Check the timestamp
Yes
Check if data is there
No
Fetch data
False
if > 5 Minutes
True
Using the old data
Fetch the new data and parse it
Parse the Static Data
Parse the Static Data
Filter Calendar
Filter Route based on the user's input
Filter Calendar with Exceptions
Merge those filtered arrays
Filter Trips corresponding to the Routes
Filter these two arrays
Check routes that has 10 minutes less arrival time
Filter these that has UQ Lakes in the stop name
Return the finalTrip data
Finding the live location and live positon
Get the live position
Filter these data using the tripId in the json file
Get the live arrival timestamp

After getting the combined calendars with filtered exception_type and the dates, I continue finding the routes and trips that have less than 10 minutes arrival. The next step I check whether it has UQ Lakes in the stop name by checking the stopId. By doing these, I believe I already got the finalTrip of the static data. After getting the static data, I continue finding the live position and live arrival by using the fetch data and checking the tripId. It is basically easier since I just need to check the tripId and get the live position and timestamp. The problem was I needed to convert the timestamp to Australian Timezone time in HH:MM:SS. But I managed to solve it afterwards.

Fetching the Data

To fetch the data, I try to read the data and check the timestamp whether the data is 5 minutes old or not. If it is already 5 minutes old, we fetch the data again. If there is no data, it will automatically fetch the data.



```
async function tripUpdatesToJson() {
    const fetchTrip = await fetch("http://127.0.0.1:5343/gtfs/seq/trip_updates.json");
    const fetchVehicle = await fetch("http://127.0.0.1:5343/gtfs/seq/vehicle_positions.json");
    const tripUpdates = await fetchTrip.json();
    const vehicleUpdate = await fetchVehicle.json();

    // Write the cached-data into the corresponding file
    fs.writeFileSync('cached-data/trip_updates.json', JSON.stringify(tripUpdates),
        (err) => err && console.error(err));

    fs.writeFileSync('cached-data/vehicle_positions.json', JSON.stringify(vehicleUpdate),
        (err) => err && console.error(err));

}

/**
 * Function to read the cached-data
 * @return combined JSON.parse data
 */

async function readCache() {
    try {
        let tripUpdates = fs.readFileSync('cached-data/trip_updates.json', { encoding: 'utf8', flag: 'r' });
        let vehiclePosition = fs.readFileSync('cached-data/vehicle_positions.json', { encoding: 'utf8', flag: 'r' });
        let dataCombine = {
            trip: JSON.parse(tripUpdates)["entity"],
            vehicle: JSON.parse(vehiclePosition)["entity"],
            header: JSON.parse(tripUpdates)["header"]
        };
        return dataCombine;
    }
    catch(error) {
        return "";
    }
}
```