Active Suspension Testing Rig Architecture

Nick Thomas | Domin

Deliverables:

1. Design a Streaming Service
2. Implement part of the system

Overview

Task

Design a streaming service for testing active suspension on a development road vehicle. This system will be operated by test engineers and stream data off the vehicle for use by Domin engineers

Key Requirements

- The system should consider 4 suspension units, 1 suspension controller and the vehicle
- The maximum data rate of each of these devices is 1000 Hz and the maximum number of channels is 32 per device. Data types can be floats, integers or strings
- Your system should be capable of storing data offline in the case that internet connectivity is not available (up to 1 hr) and once it is, it should stream the data to the cloud
- Local data should also be accessible to the in-vehicle UI via and API for the last 100 seconds
- An API should be made available that allows engineers to query specific channels, specific time windows or a combination of both
- Data analysis/processing jobs that detect events in the data streams (such as large bumps in the road, the vehicle doing a left turn at 30 kph). Identified events should be stored for later use and should be available from querying

Key Specification and Assumptions

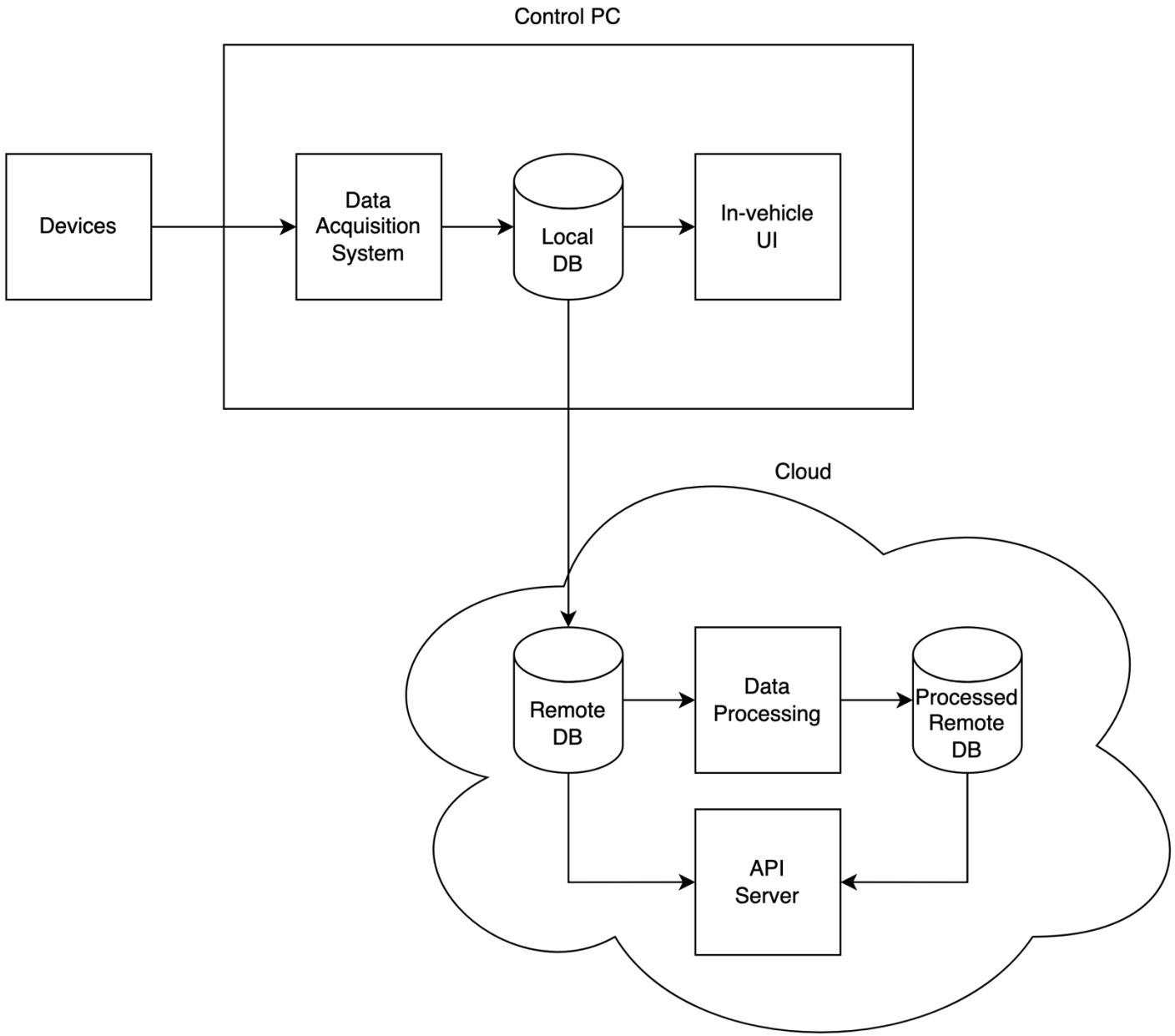
Specifications

1. For each device:
   - Maximum data rate of 1000 Hz
   - Maximum number of channels is 32
   - Handle data type of floats, integers and strings
2. Store data offline for 1 hour and upload if connectivity is restored
3. Local data accessible by vehicle UI and API for last 100 seconds
4. Data can be queried from the cloud via API for specific channels, specific time windows or combination of both
5. Data analysis can be run on data stored in the cloud

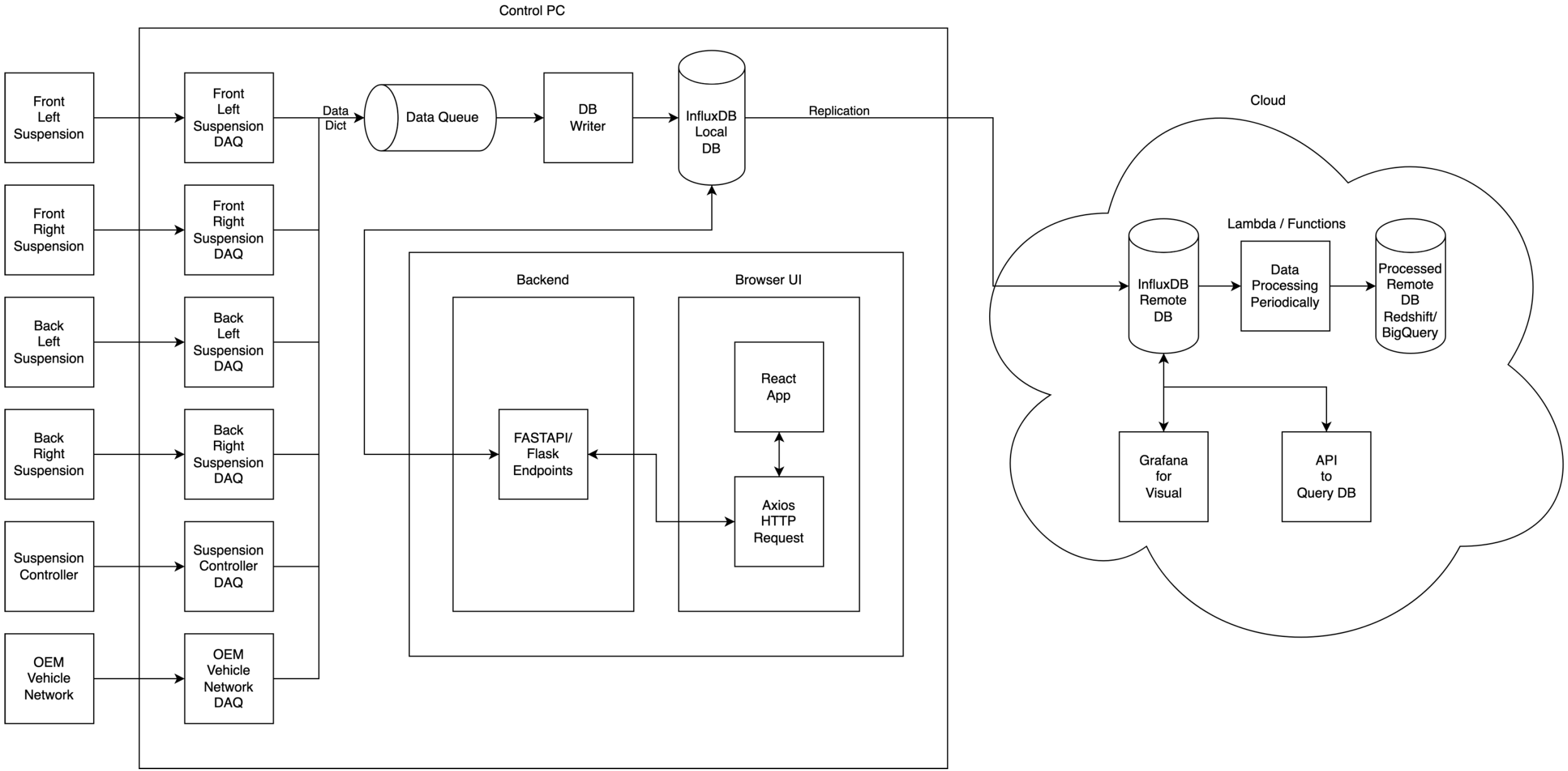
Assumptions

1. Interface with 6 devices at the same time (4 suspension units, 1 suspension controller, 1 OEM vehicle Network
2. Devices are not synced and can have different data rates
3. No need to centralise timing clock, each datapoint has its own timestamp
4. Data will be stored locally for 100 seconds from the time of ingestion regardless if new data is being ingested or not
5. More powerful edge databases will be available in production to handle data rates
6. Database edge solution not finalised so the architecture has been designed so that it is easier to drop a different one in

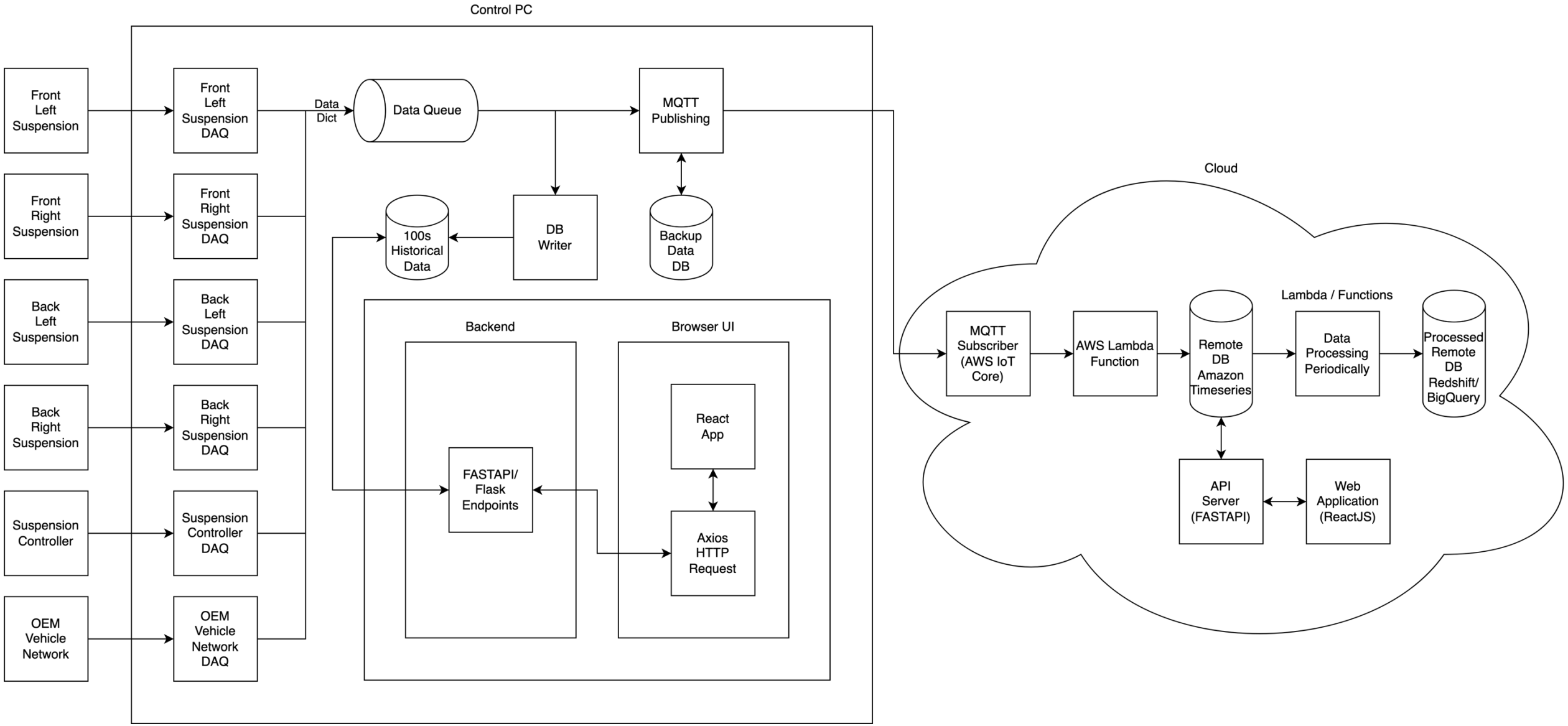
Architecture Diagrams – General Overview



Architecture Diagrams – Proposed Solution 1



Architecture Diagrams – Proposed Solution 2



Additional Considerations

Data Processing

How would the system handle processing jobs and go through historical data?

- Cloud Lambdas and Cloud functions
- Key events labelled during acquisition
- Uniqueness added to test data at source
- Smart SQL queries to quickly reduce dataset

Data Storage

How is the data stored long term to reduce cost?

- Raw timeseries data would be sectioned into tests and stored in long term blob storage in parquet files
- Lambdas would be created for quick redeployment of raw data for further analysis
- Timeseries database would remain performant for current data

Effective data queries

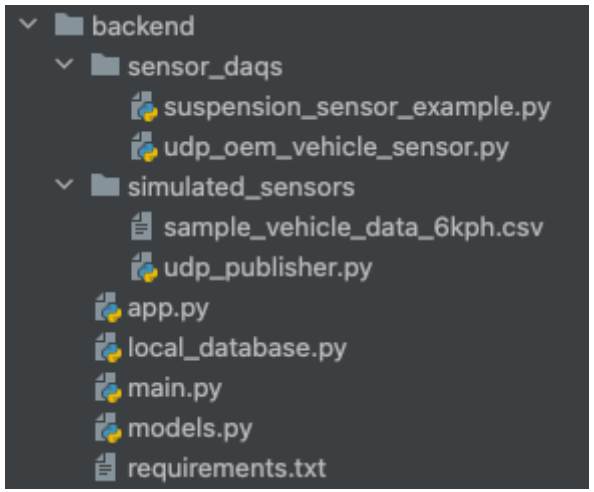
Effectively query a large dataset and serve to a UI?

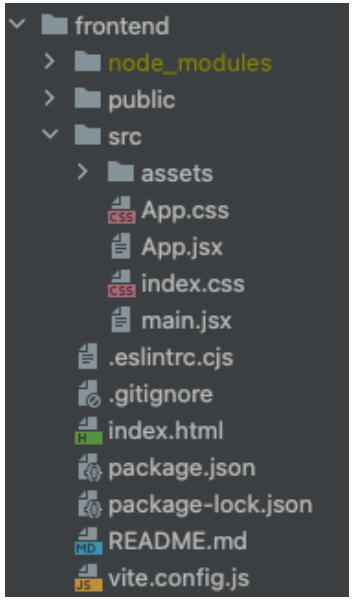
- Down sample the raw data for the required grouping by time
- Query number of datapoints to fill resolution of screen between the time period
- Reduce level of aggregation as time period is reduced due to zooming in on events

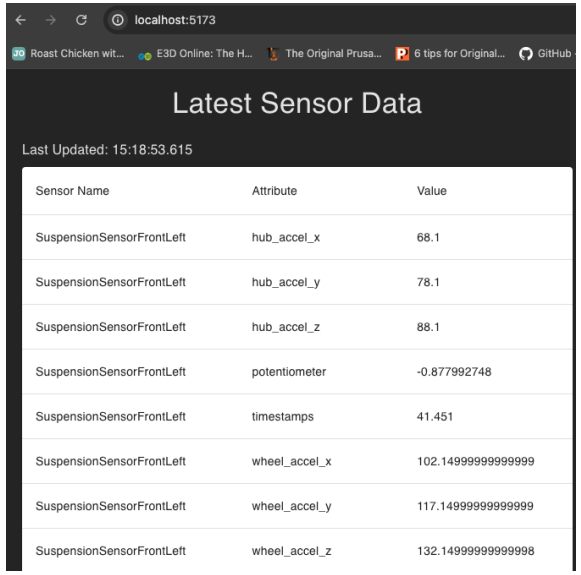
Deliverables:

2

Implement part of the system







Available to view in GitHub, here

Nick Thomas | Domin