Goal

Create and host a web application (front-end and back-end) that can dynamically interrogate and visualise data from the provided data tables (CSV files) – a suggested layout and functionality for the web app is available in the provided wireframe document (PDF).

Front-end Requirements

Please build the front end of the application in React – beyond this though feel free to use whichever libraries you like to achieve the desired functionality.

Back-end Requirements

Back-end can be built in whichever language you prefer (Python would be ideal though) – ideally this would be constructed as a serverless API that interacts with the provided data tables via some database/storage technology.

Dataset Considerations

Datasets provided:

- State HH Income Counts
    - Contains household counts broken down by State, Household Composition and Weekly Household Income Bracket (288 rows by 5 columns)
- SA4 HH Weekly Income Counts
    - Contains household counts broken down by SA4 Region, Household Composition and Weekly Household Income Bracket (3136 rows by 5 columns)

The two tables can be joined on 'State ID' (associating each SA4 Region to a State).

Please store these two datasets as separate tables in whichever database/storage technology you prefer.

Hosting Considerations

Please host the web application and make it available via a publicly accessible URL.
You can use any cloud hosting service you prefer, free tier hosting is totally fine and app performance resulting from free tier limitations is not a problem.

Deliverables

Please store all source code in a version-controlled repository that you can make publicly accessible for review (e.g. Github) – please provide the link to this repo as part of your submission.

A link to the publicly hosted web app and the source code repo should be provided back to Stockland by 5pm Sydney time <DD/MM/YYYY>.

What we are most interested in

- API design
    - Intuitive and clear API routes, well integrated with database/storage layer
- Dynamic interaction with data
    - Searching/changing filters on page should responsively poll underlying data
- Dynamic field display on page, depending on the type of object selected (state vs SA4)