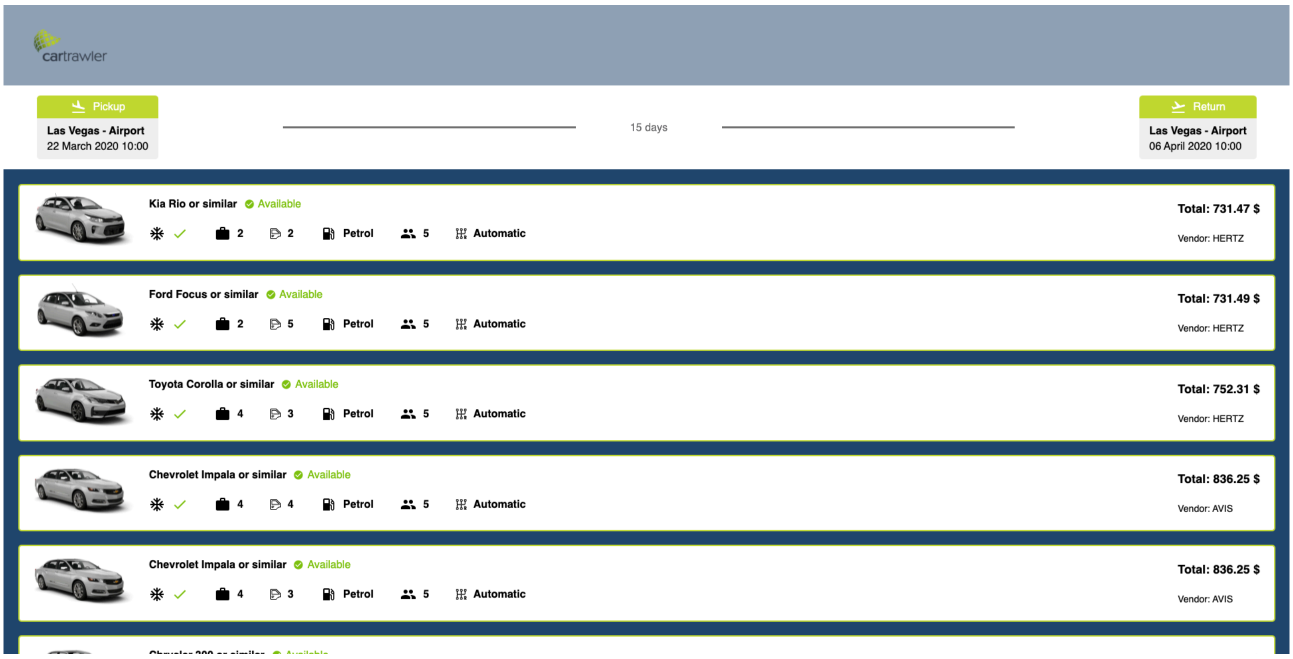
To run application, you need to execute
- npm install
- npm start

After running npm start, it will run website on `localhost:3000` where you will be able to verify final effect of website. I prepared 3 views which are adjusted for 3 different breakpoint as displayed on screenshots below. To verify it in chrome you can press F12 and then you can resize website to verify how changes are looking on live website

Full width



Las Vegas - Airport
22 March 2020 10:00

15 days

Las Vegas - Airport
06 April 2020 10:00

Kia Rio or similar ✓ Available
✓ 2 2 Petrol 5 Automatic
Total: 731.47 $
Vendor: HERTZ

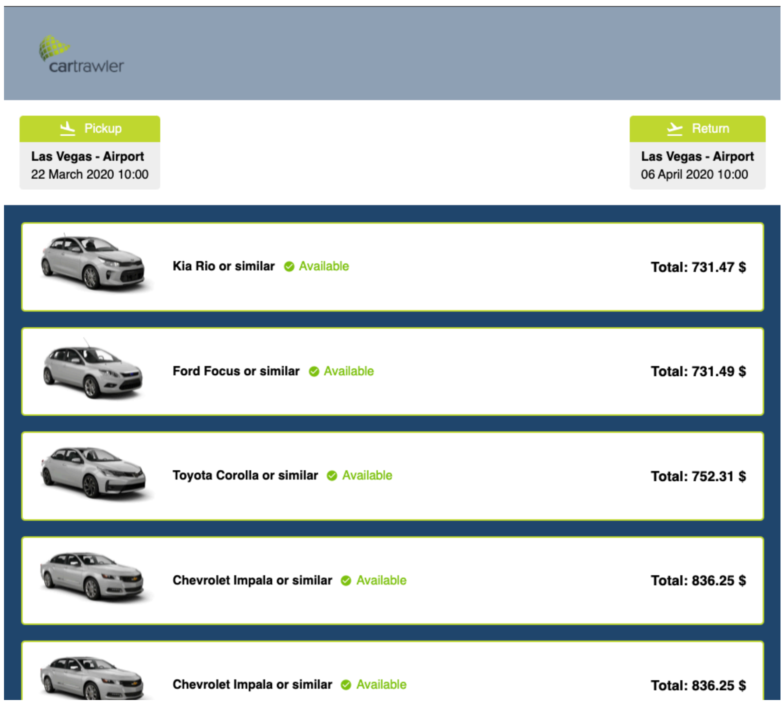
Ford Focus or similar ✓ Available
✓ 2 5 Petrol 5 Automatic
Total: 731.49 $
Vendor: HERTZ

Toyota Corolla or similar ✓ Available
✓ 4 3 Petrol 5 Automatic
Total: 752.31 $
Vendor: HERTZ

Chevrolet Impala or similar ✓ Available
✓ 4 4 Petrol 5 Automatic
Total: 836.25 $
Vendor: AVIS

Chevrolet Impala or similar ✓ Available
✓ 4 3 Petrol 5 Automatic
Total: 836.25 $
Vendor: AVIS

1019px website width



Las Vegas - Airport
22 March 2020 10:00

Las Vegas - Airport
06 April 2020 10:00

Kia Rio or similar ✓ Available Total: 731.47 $

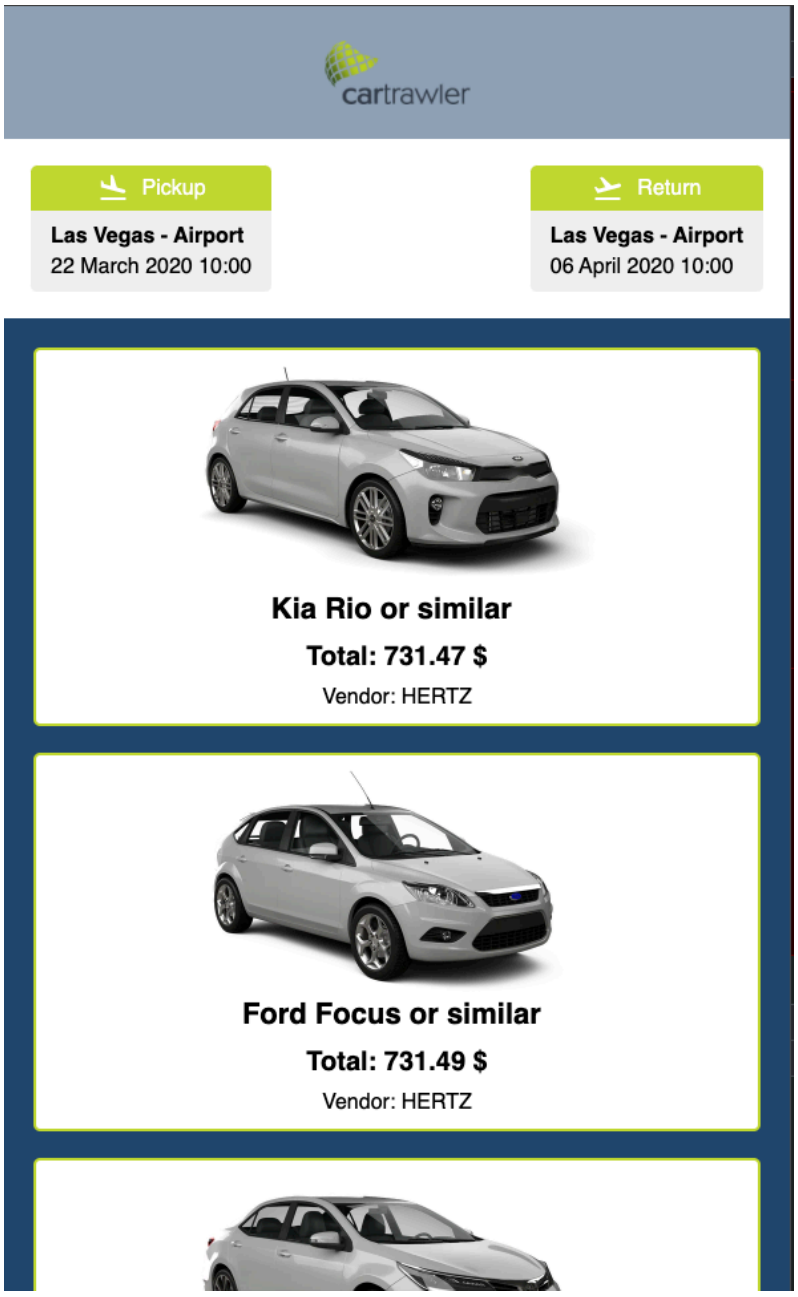
Ford Focus or similar ✓ Available Total: 731.49 $

Toyota Corolla or similar ✓ Available Total: 752.31 $

Chevrolet Impala or similar ✓ Available Total: 836.25 $

Chevrolet Impala or similar ✓ Available Total: 836.25 $

600px website width



cartrawler

Pickup

Las Vegas - Airport
22 March 2020 10:00

Return

Las Vegas - Airport
06 April 2020 10:00

Kia Rio or similar

Total: 731.47 $

Vendor: HERTZ

Ford Focus or similar

Total: 731.49 $

Vendor: HERTZ

I arranged and split components between app specific like list and CarDetailsModal but as well tried to create elements which can be used in different places in the future like loading screen and modal. I set timeout for fetching data from api just to display loading logo but timeout can be removed what will speed up app testing but there is possibility that loading screen won't be visible.

Testing

In terms of testing I would start from creating cosmos library with all possible elements and mock data to display all possible variants of each component where visually I would be able to test if all options works properly when all data passed are properly formatted. It is good just for visual testing where I can test each element in every possible value variant and in different screen sizes and observe how it behaves and if change of one parameter is not affecting not expected components

Another step would be testing components by rendering them in test file and comparing if rendered element is displaying data as expected in test.

Next I would create test which is comparing whole component HTML rendered with expected HTML which is generated at properly working page and then is compared in unit test if rendered object HTML is exactly the same. This option is slightly different comparing to the previous test because previously we compared jus value in specific places but here it will compare whole component generated HTML code.