EzParking Client

Code Testing Plan and Result

SSE Capstone project - University of Regina

Team Apollo

Winter 2023

Zhuo Chen

200352092

Table Of Contents

Summary

The ‘EzParking Client’ React application is designed to provide a user interface to the public, which allows users to select the destination, obtain parking lot information such as available spots, chose a parking lot from the list of alternatives, gain the details, navigation and time duration from parking space to the destination. This report provides the test plan and the test results. The test plan is formalized to examine each React component in the client application and integration these components to standardize their qualities and satisfy the project requirements. This report consists of five sections: testing strategy, test plan, test cases, test results, and conclusion.

The two main technologies used in the plan are ‘testing-library/react’ and ‘react-dom’ to perform the unit testing. To ensure every component is rendered correctly, there are 15 unit test cases in total for 6 test suites/components. And 17 integration test cases with the test matrix technique.

Testing strategy

To ensure the React components compile as expected, two types of strategies are applied to this test plan: unit testing for each component and integration testing. Unit testing is to assure the components are rendered properly at a lower level of the application while integration testing is to examine all the components as an entity.

The unit test could be assisted by using technologies ‘testing-library/react’ and ‘react-dom’. They will allow the developers to compare the expected result with actual outcomes rendered by the components. Therefore, the unit test strategy is implemented and executed while the source code has been modified.

The integration test is a higher-level test, which includes multiple components as an entity. In order to test a more complex circumstance, a test case matrix is used to track the test cases and their results. Additionally, the compiler also plays the role of part of the integration test, since it will verify and compile each web page of the client application.

Test plan

Six components in the client application could be categorized into two types: static and dynamic elements. The announcement, footer, and header are static components that should be rendered without any parameters passed from their parent elements. They are tested from two perspectives: if they are rendered correctly, and if they are rendered properly within HTML Tags. On contrary, destinationHeader, dropdown, and parkingListCard are the components that contained passed parameters and dynamically update their contents. Hence, in addition to these two static test plans, one

extra circumstance of passing parameters should be tested as well. All the unit test results information is provided by the testing library.

Integration test requires multiple components, features, and technologies to participate in the process. The test plan is divided into three portions/pages: The destination page contained an announcement, footer, header, and dropdown with a RESTful API. The parking lot list page is integrated by footer, header, destinationHeader, and parkingListCard components with another RESTful API. The third one is the details page that contains the footer, header, destinationHeader with the same API applied in the Parking lot list page. These three pages integrate all the components of this project and each page is tested regarding the correctness of rendering information and their features.

Test Cases

For the static components, the unit test contains two typical test cases:

```
it("renders without crashing",()=>{
    const div=document.createElement("div");
    ReactDOM.render(<testedComponent/>, div)
})
```

, which renders the components within the div tag without any crashing occurring.

```
it("renders announcement correctly", ()=>{
    render(<testedComponent/>)
})
```

The code above is another test case that ensures the component renders correctly.

The parameters passed test cases are:

```
function wrapper(element) {
    return(
      <Provider store={store}>{element}</Provider>
    )
```

```
}
it("if the prop is passed, render the destination", ()=>{
    render(<DestinationHeader destination='test building'/>)
    expect(screen.getByText(/to reach test
building/i)).toBeInTheDocument();
})
```
The destinationHeader test case.

```
it("if the prop is passed, render the name and id", ()=>{
    render(wrapper(<DropdownCard name={'test building'}
        id={'2'}/>))
    expect(screen.getByRole('menuitem', {
        name: /test building/i
      })).toBeInTheDocument();
})
```
The dropdown test case.

```
it("if the prop is passed, render the avaSpot", ()=>{
    render(wrapper(<ParkingListCard parkingLotName={'test 1'}
avaSpot={31}/>))
    expect(screen.getByText(/31/i)).toBeInTheDocument();
})
it("if the prop is passed, render the parkingLotName", ()=>{
    render(wrapper(<ParkingListCard parkingLotName={'test 2'}
avaSpot={31}/>))
    expect(screen.getByText(/2/i)).toBeInTheDocument();
})
```
The parkingListCard test cases.

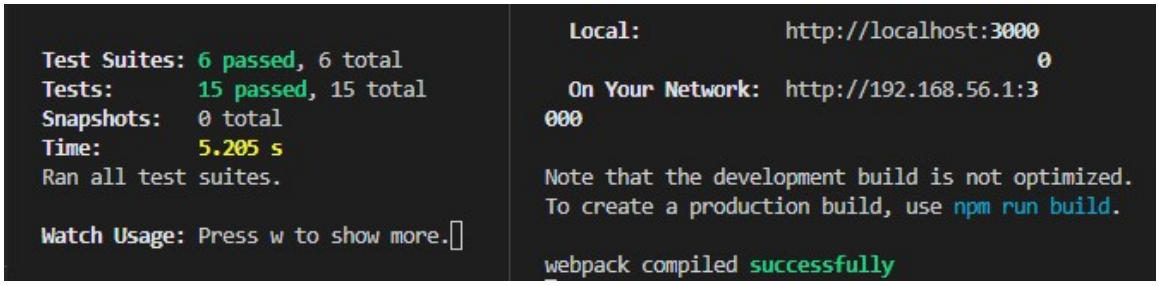
For the integration test cases:

Page	Test Case	Expected outputs	Test result
Destination	Click dropdown button	The dropdown button is clickable, and the destination buildings displayed	
	The building's information is fetched from the database	Using API to check the information, since the data is dynamic, https://ezparking114514.com:9195/getAllDestination.	
	Click on the 'Recommend	The 'Recommend Now' is disabled	

	Now' button		
	Select 'Gym' as the destination	Redirected to 'Parking lot list page'	
	All the components are rendered properly	No error message is displayed on the 'Destination' page and its console log.	
Parking lot list	The destination name is displayed on the top of the card list	'To Reach Gym'	
	Five parking lots are listed with their name and available spots	The detailed information is need to be checked using API, 'https://ezparking114514.com:9195/getPath'	
	Scroll the parking list	The list is slideable	
	Click on the 'Recommend Now' button	Redirected to 'Destination page'	
	All the components are rendered properly	No error message is displayed on the 'Parking lot list' page and its console log.	
	Click the first option card in the list	Redirected to 'Detailed information page'	
Detailed information	The destination name is displayed on the top of the card list	'To Reach Gym'	
	The parking lot name is displayed on the top of the information card	'Parking at Lot7'	
	In the card, three pieces of information are provided.	'Direction', 'walking time', 'running time'	
	Click on the 'direction' icon	Jump to the Google Map application with the navigation route from the user's current location to the parking lot location.	
	Click on the 'Recommend Now' button	Redirected to 'Destination page'	
	All the components are rendered properly	No error message is displayed on the 'Detailed information' page and its console log.	

Test Results

For the unit test: all 15 test cases are passed and the application is rendered successfully.



For the integration test: all 17 test cases are passed and the application is rendered successfully.

Page	Test Case	Expected outputs	Test result
Destination	Click dropdown button	The dropdown button is clickable, and the destination buildings displayed	Passed
	The building's information is fetched from the database	Using API to check the information, since the data is dynamic, https://ezparking114514.com:9195/getAllDestination.	Passed
	Click on the 'Recommend Now' button	The 'Recommend Now' is disabled	Passed
	Select 'Gym' as the destination	Redirected to 'Parking lot list page'	Passed
	All the components are rendered properly	No error message is displayed on the 'Destination' page and its console log.	Passed
Parking lot list	The destination name is displayed on the top of the card list	'To Reach Gym'	Passed
	Five parking lots are listed with their name and available spots	The detailed information is need to be checked using API, 'https://ezparking114514.com:9195/getPath'	Passed
	Scroll the parking	The list is slideable	Passed

	list		
	Click on the 'Recommend Now' button	Redirected to 'Destination page'	Passed
	All the components are rendered properly	No error message is displayed on the 'Parking lot list' page and its console log.	Passed
	Click the first option card in the list	Redirected to 'Detailed information page'	Passed
Detailed information	The destination name is displayed on the top of the card list	'To Reach Gym'	Passed
	The parking lot name is displayed on the top of the information card	'Parking at Lot7'	Passed
	In the card, three pieces of information are provided.	'Direction', 'walking time', 'running time'	Passed
	Click on the 'direction' icon	Jump to the Google Map application with the navigation route from the user's current location to the parking lot location.	Passed
	Click on the 'Recommend Now' button	Redirected to 'Destination page'	Passed
	All the components are rendered properly	No error message is displayed on the 'Detailed information' page and its console log.	Passed

Conclusion

With all the 15 unit test cases and 17 integration tests passed, the current version of the client application is ready to be deployed to the UAT environment. However, with more features and functionalities developed in the future, more test cases will be required. Therefore, I have created a test case folder for each component regarding the SOLID design principle.