# What is Continuous Integration?

Continuous Integration involves many key practices including:

* Frequent Code Integration and Maintain Environment– Instead of bulk pushing code, to push code regularly to ensure the whole team is up to date and allowing the team to see what has been done.
* Automated Build and Testing – Will build the software from the latest code, and run automated tests including unit tests, integration tests, and user acceptance tests, to reduce errors, have quality assurance, eliminating dependencies on key personnel, and a continuous delivery.
* Immediate feedback – To ensure code changes / additions did not have any dependencies, this feedback will allow dependencies to be identified earlier and fix an issue if a test had failed upon a new change.

By maintaining and enforcing these key practices, a delivery of high-quality software can be achieved through the commitment of maintaining frequent code commits, quality code practices, and by fixing problems as soon as they occur. Meaning that issues and errors can be resolved straight away, preventing a pile up and increasing the difficulty of identifying code errors.

# CI Workflow

* **Code Repository**The importance of version control enabled us to locate if there are errors, to identify when and who tasks are done by, enable rollbacks to stable versions of the software, and allow branching when working on different features of the software.
* **Triggering the CI Pipeline**By allowing commits to be pushed to feature branches, if an error occurs on one branch’s specific implementation it will not affect another, reducing the risk of conflict and allowing parallel development.
* **Automated Tests**  
  Automated tests are done to ensure quality control when adding and changing code to the repository, these tests would allow us to identify key errors and fix it upon them arising. Using maven to run unit and integration tests for a backend to ensure the connection to the database and the data collated is what we are expecting. Using Vitest allows us to make sure that the server is rendering what we expect it to in regards to the frontend, also that it is using the correct names for things, for example making sure a header is correctly written.

# Automated Test Cases

For tests we followed Continuous Integration coding practices, this is where we:

1. Create tests for the expected the outcome of functions,
2. Develop the functions to pass the tests,
3. Refactor the functions.

The following is the tests in order:

## Test for feature Search Bar

### Tests creation

A screenshot of a computer program

Description automatically generated

A screen shot of a computer code

Description automatically generated





A screen shot of a computer code

Description automatically generated

A computer screen with text on it

Description automatically generated

### getAllProducts implemented

A computer screen shot of a black screen

Description automatically generated

### searchForItem implemented

A computer screen shot of a computer screen

Description automatically generated

## Test for featureCart

### Tests Created

A screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

### getCartProducts implemented:

A computer screen with text

Description automatically generated

### addItemToCart implemented

A screenshot of a computer

Description automatically generated

### removeProductFromCart implemented

A black screen with white text

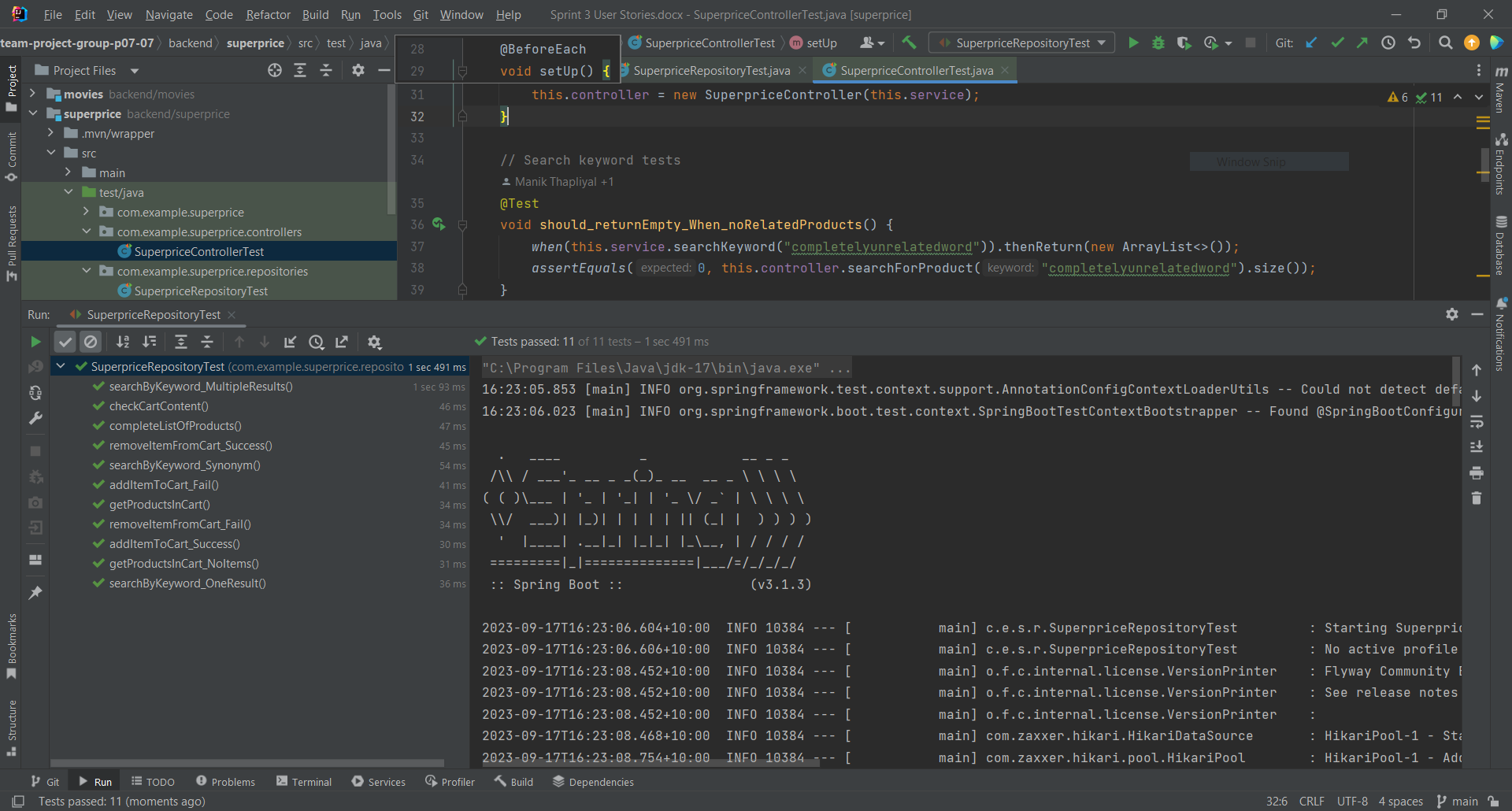
Description automatically generated

# Test Execution and Results

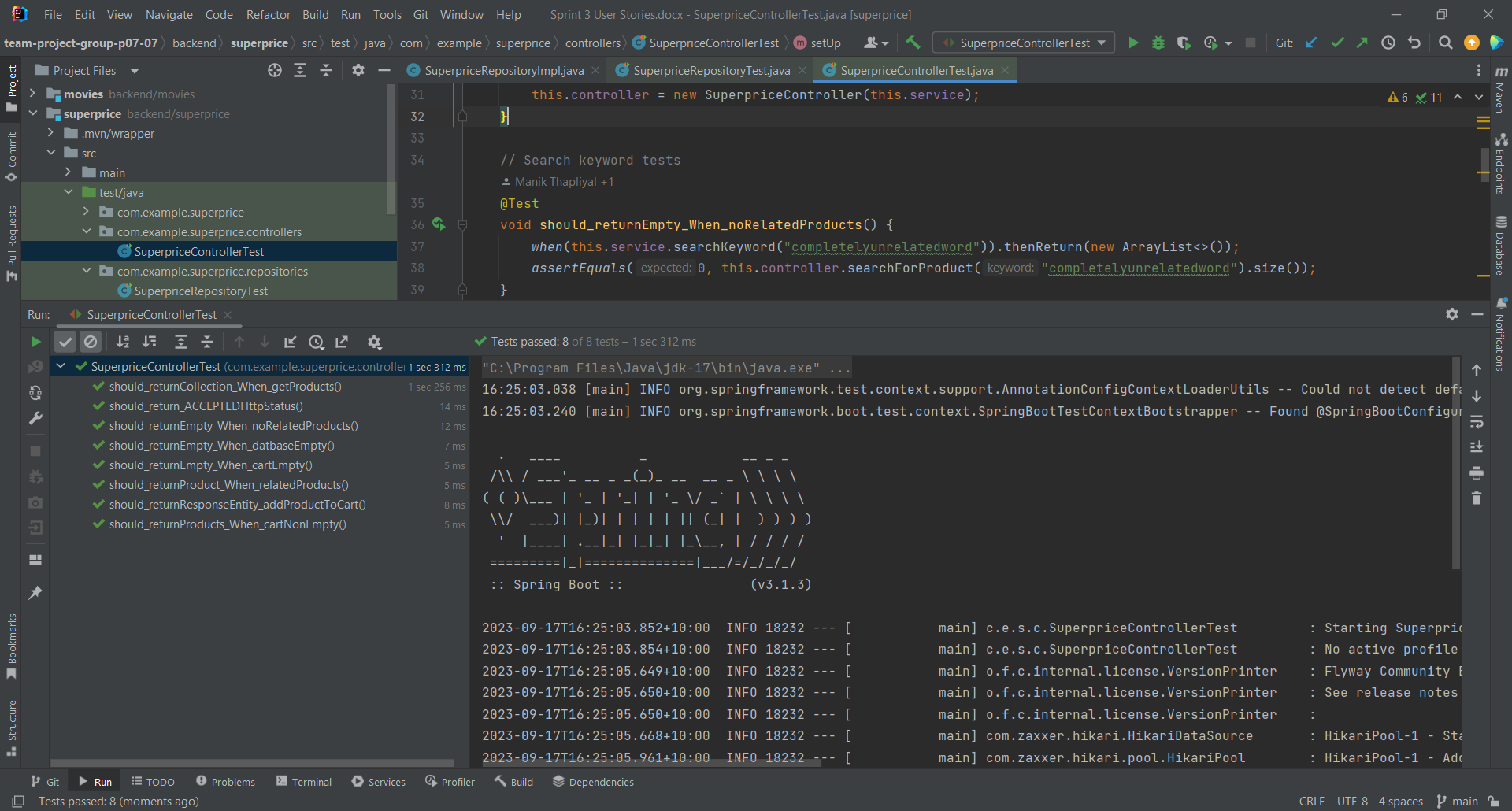
## Backend

Like the Automated Test Cases, we first created comprehensive test cases then developed the features until they pass the test.

### SuperpriceRepositoryTest



### SuperpriceControllerTest



## Frontend

Due to our negotiations with the product owner it was decided to only do a few frontend tests if we had time due to the various circumstances that affected our group being various people being sick and someone dropping the class. As such we focused on the basic tests that made sure our different pages were running correctly in regards to the CI. However we did extensive manual testing to ensure that everything runs correctly.

Home Page:

For the home page we tested the different buttons being the shop now button, the previous and next product buttons as well as testing to make sure the header of the page both existed and was correct.

The tests: A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

The results:  
A black background with white text

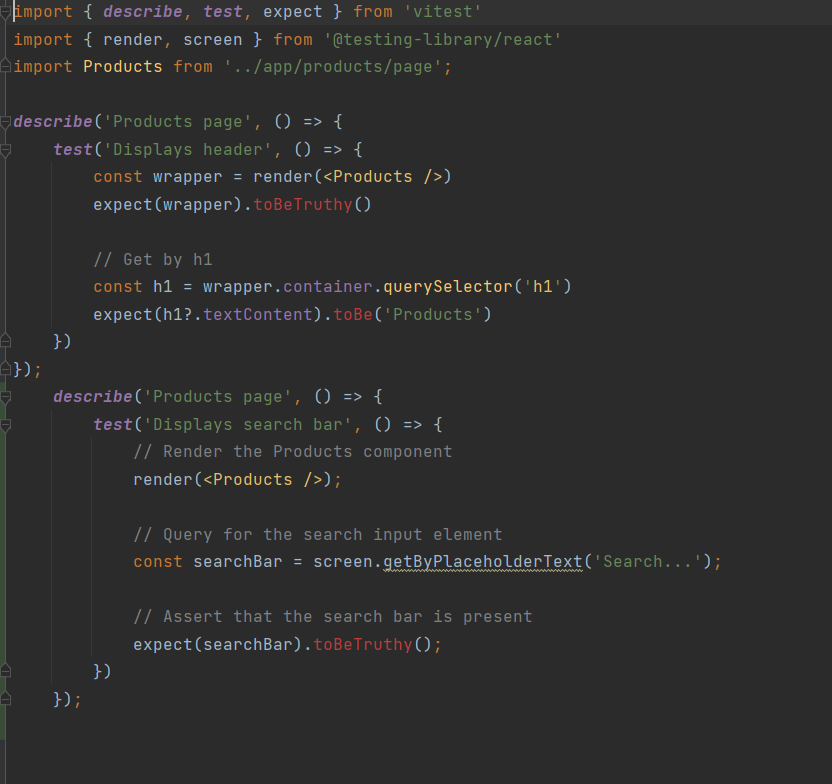
Description automatically generated

All four tests passed successfully.

Product page

For the product page we tested that the page renders and the header is displayed successfully as well as testing that the search bar is successfully loaded.

The tests:

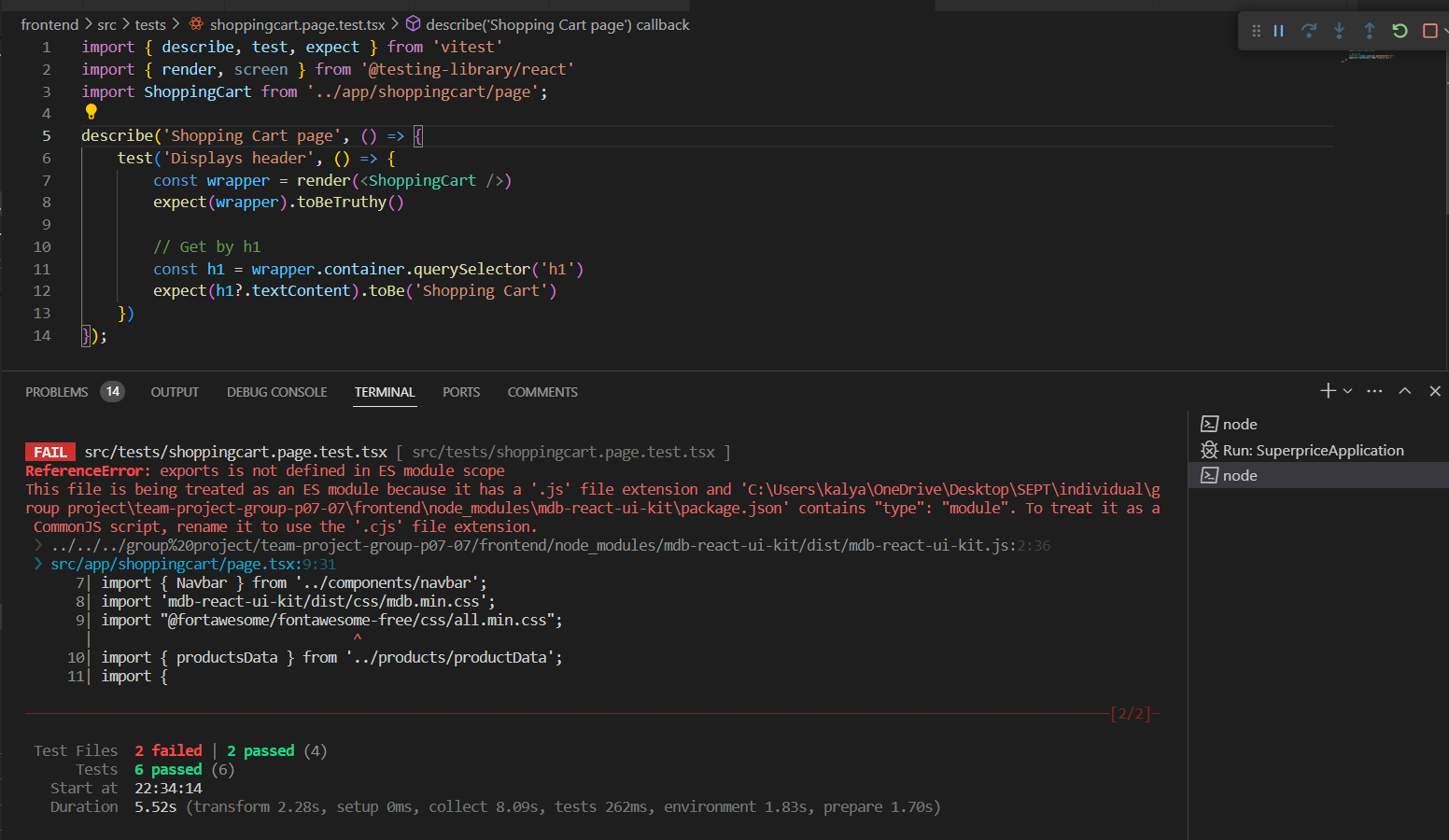


The results:



Both tests passed successfully.

There are some tests that we were not able to run due to the tests being unable to import the necessary libraries. For example, the tests for the shopping cart page.

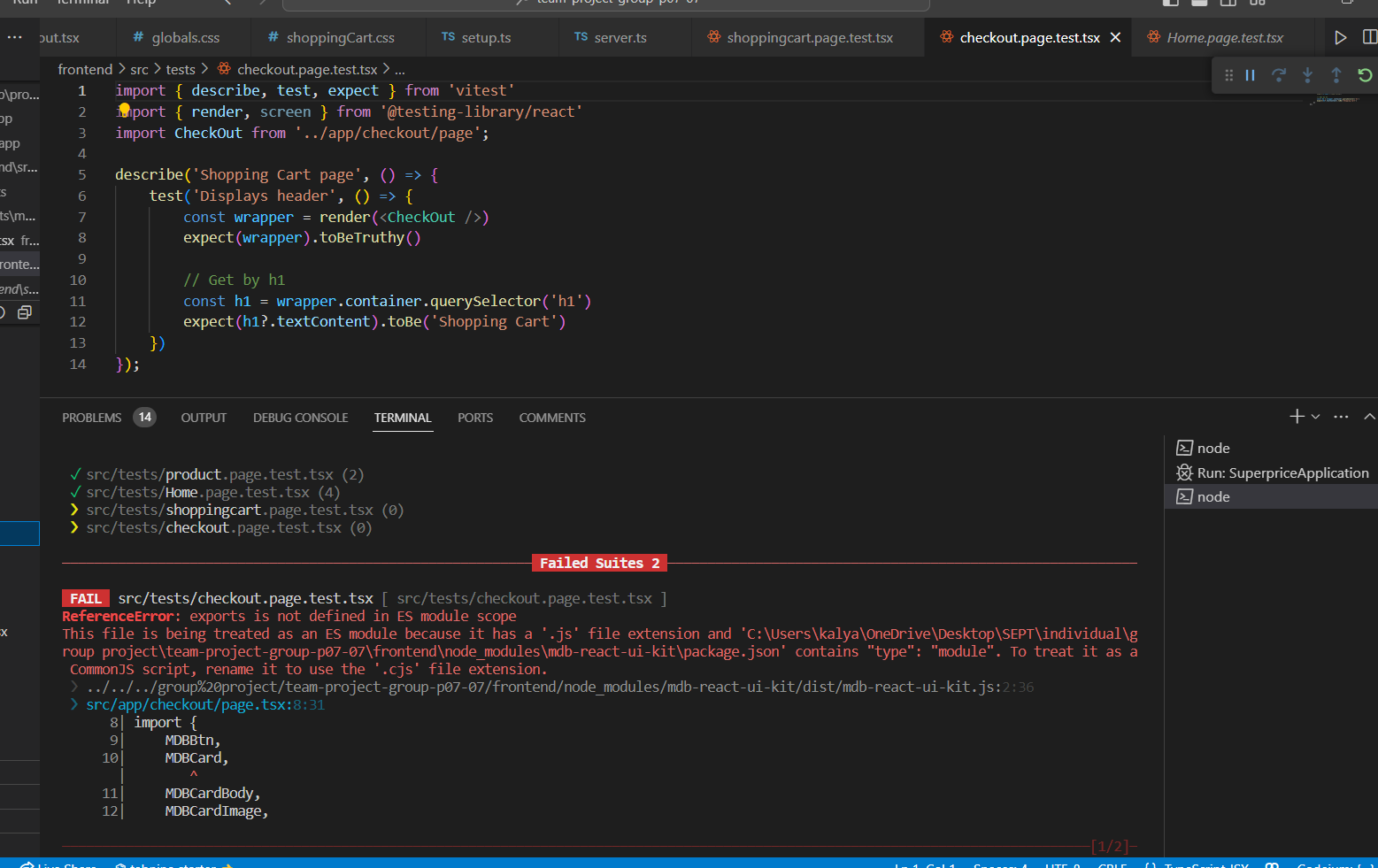


For some reason, it is not importing the necessary libraries that we used in the page. Due to this the shoppingcart page would not even mount. However, this led us to do manual testing, and as you can see when running the website the shopping cart page works well with no form of import error.

A screenshot of a computer

Description automatically generated

The same issue was also encountered when doing the test for the checkout page.



We could not remove the libraries, obviously as our page uses them. This is what led us to do manual testing for the pages that have import issues with the test. Again, everything seems fine when we run the actual website.

