# Stock App

# Testing Document

# Team 5

Paul Negrido

Capstone Project Spring, 2025

The University of West Florida

30 Mar 2025

CIS4592 - Capstone Project

Instructor: Dr. E.L. Fridge

## Table of Contents

# 1 Testing Strategy

Our original testing strategy involves the use of automated jest testing for unit tests, use of testCafe for end-to-end tests, and manual testing. For unit tests, we aim to cover the critical portions of business our app's logic and automate the testing with jests. We utilized jest mock testing to simulate data coming from the API and avoid calling the API multiple times during the test. Jest unit tests are automatically run during pull requests with github actions.

We initially tested the basic functionality of our app regarding a basic stock search functionality outside of any features and user logged in. For our first feature, stock news, we tested the functionality of retrieving relevant news from the API. This included automated unit tests and manual tests. For our second feature, portfolio creation, the testing is currently only manual, but plan to automate more of the tests. Our third feature, detailed stock info, will be the last feature and we plan to automate unit tests for this feature as well.

We tried incorporating testCafe end-to-end testing for the features created thus far. So far we were able to test the basic feature that does not require log-in. This involved navigating through the links in the header and going through a basic stock search.

# 2 Testing Summary Table

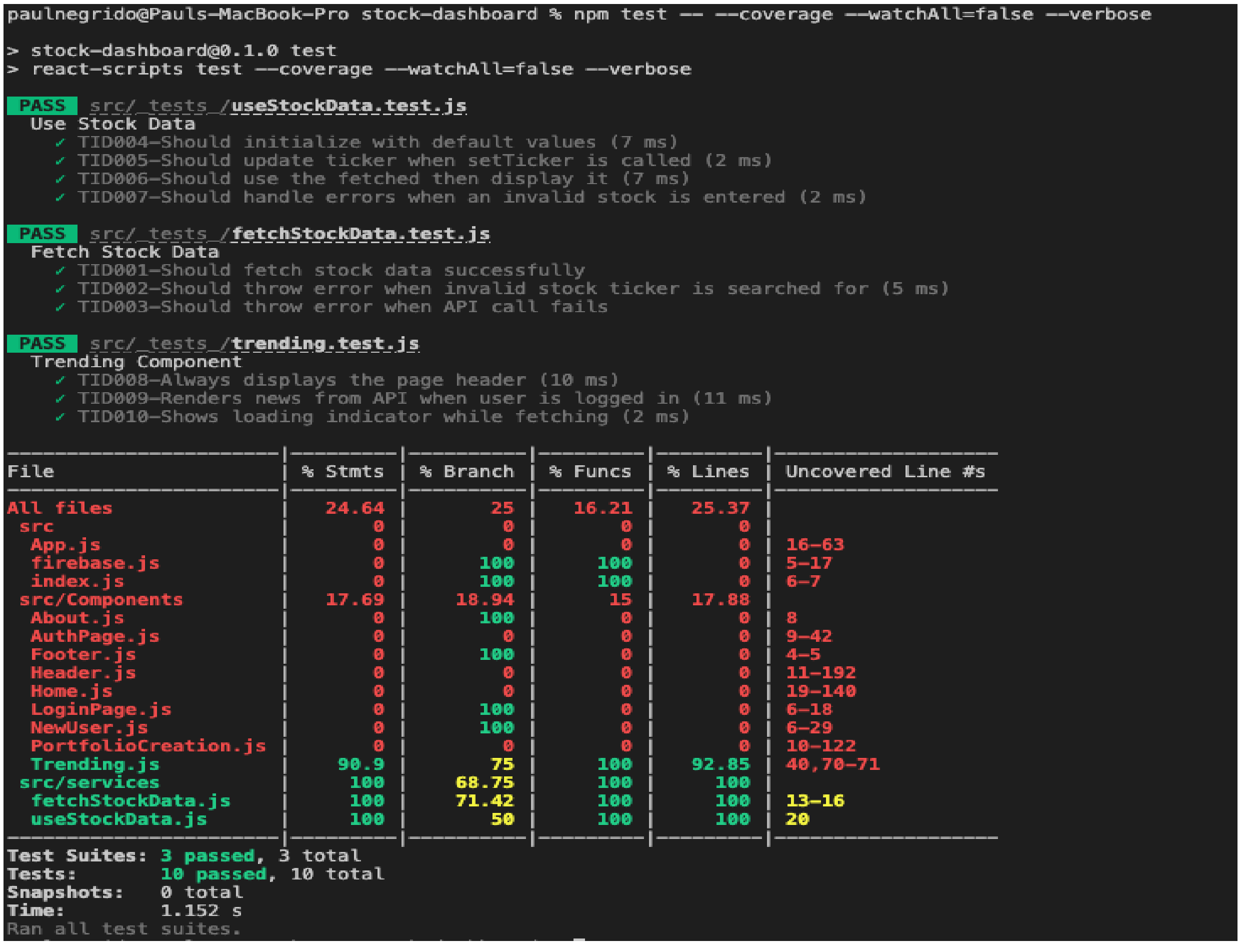
| Requirement Traceability | | Business Requirements | | | |
|---|---|---|---|---|---|
| Test Cases | Test ID | BID001 - News and Notification | BID002 - Portfolio Recommendation | BID003 - Stock Info | BID004 - Create User Stock Portfolio |
| Fetching Stock data (Basic Feature - No Login Required) | | | | | |
| Should fetch stock data successfully | TID001 | | | PASSED (Jest) | |
| Should throw error when invalid stock ticker is searched for | TID002 | | | PASSED (Jest) | |
| Should throw error when API call fails | TID003 | | | PASSED (Jest) | |
| | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Use Stock Data (Basic Feature - No Login Required) | | | | | |
| Should initialize with default values | TID004 | | | PASSED (Jest) | |
| Should update ticker when setTicker is called | TID005 | | | PASSED (Jest) | |
| Should use the fetched then display it | TID006 | | | PASSED (Jest) | |
| Should handle errors when an invalid stock is entered | TID007 | | | PASSED (Jest) | |
| | | | | | |
| Trending Component (Feature 1) (Requires Logged In) | | | | | |
| Always Displays the Header | TID008 | PASSED (Jest) | | | |
| Renders news from API when user is logged in | TID009 | PASSED (Jest) | | | |
| News links to external page from list | TID010 | Manual | | | |
| Shows loading indicator while fetching | TID011 | PASSED (Jest) | | | |
| Inbox contains news | TID012 | Manual | | | |
| News in inbox links to external page | TID013 | Manual | | | |
| Delete news button in inbox works | TID014 | Manual | | | |
| | | | | | |
| | | | | | |
| Portfolio Creation (Feature 2) (Requires Logged In) | | | | | |
| Generate portfolio button redirects to portfolio form | TID015 | | Manual | | |
| Test Form Component functionality | TID016 | | Manual | | |
| Test Form submits data | TID017 | | Manual | | |
| After submission a message pops up successful portfolio creation | TID018 | | Manual | | Manual |
| Portfolio link in header shows portfolio popup | TID019 | | Manual | | Manual |
| Portfolio pop up works | TID020 | | Manual | | |
| Portfolio pop up has a list of stocks if not empty | TID021 | | Manual | | |
| Delete stock from portfolio | TID022 | | TBD | | |
| | | | | | |
| Detailed Stock Info Page (Feature 3) (Requires Logged In) | | | | | |
| Navigate to Detailed Stock Info Page | TID023 | | | TBD | |
| Display the stock price | TID024 | | | TBD | |

| | | | | | |
|---|---|---|---|---|---|
| Display PE ratio, and additional stock attributes | TID025 | | | TBD | |
| Add to portfolio button | TID026 | | | TBD | TBD |
| | | | | | |
| TestCafe End-to-End Tests (Non Logged-In users Only) | | | | | |
| Navigate to About Page | TID027 | | | | |
| Navigate to Home | TID028 | | | | |
| Quick Stock Ticker Search Valid | TID029 | | | | |
| Invalid search | TID030 | | | | |
| Navigate to Login | TID031 | | | | |

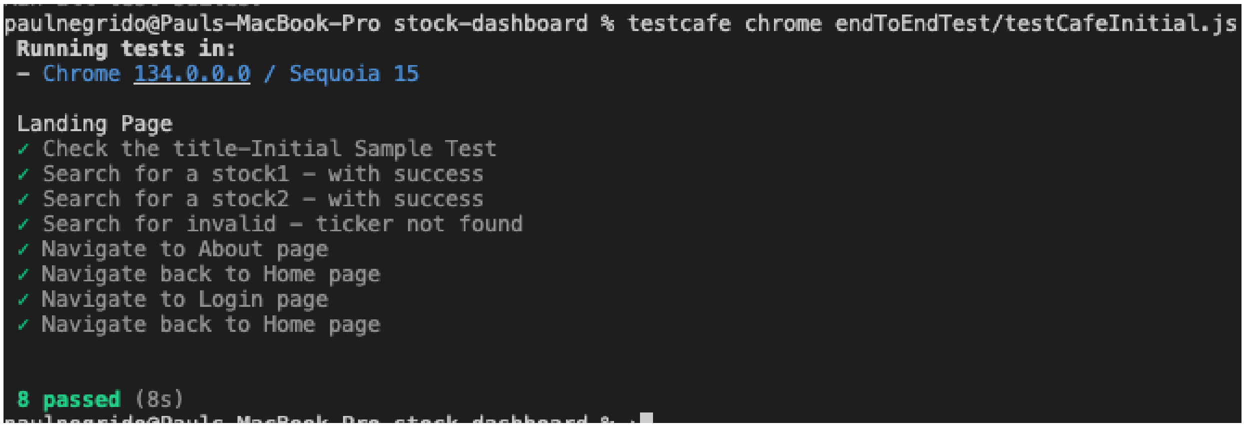
## 3 Testing Coverage Automated Tests

Below is a screenshot of our coverage of automated tests using jest testing using the command “npm test -- --coverage --watchAll=false --verbose”.



```
paulnegrido@Pauls-MacBook-Pro stock-dashboard % npm test -- --coverage --watchAll=false --verbose

> stock-dashboard@0.1.0 test
> react-scripts test --coverage --watchAll=false --verbose

 PASS  src/__tests__/useStockData.test.js
  Use Stock Data
    ✓ TID004-Should initialize with default values (7 ms)
    ✓ TID005-Should update ticker when setTicker is called (2 ms)
    ✓ TID006-Should use the fetched then display it (7 ms)
    ✓ TID007-Should handle errors when an invalid stock is entered (2 ms)

 PASS  src/__tests__/fetchStockData.test.js
  Fetch Stock Data
    ✓ TID001-Should fetch stock data successfully
    ✓ TID002-Should throw error when invalid stock ticker is searched for (5 ms)
    ✓ TID003-Should throw error when API call fails

 PASS  src/__tests__/trending.test.js
  Trending Component
    ✓ TID008-Always displays the page header (10 ms)
    ✓ TID009-Renders news from API when user is logged in (11 ms)
    ✓ TID010-Shows loading indicator while fetching (2 ms)

-----------------------|---------|----------|---------|---------|-------------------
File                   | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-----------------------|---------|----------|---------|---------|-------------------
All files              |   24.64 |       25 |   16.21 |   25.37 |
 src                   |       0 |        0 |       0 |       0 |
  App.js               |       0 |        0 |       0 |       0 | 16-63
  firebase.js          |       0 |      100 |     100 |       0 | 5-17
  index.js             |       0 |      100 |     100 |       0 | 6-7
 src/Components        |   17.69 |    18.94 |      15 |   17.88 |
  About.js             |       0 |      100 |       0 |       0 | 8
  AuthPage.js          |       0 |        0 |       0 |       0 | 9-42
  Footer.js            |       0 |      100 |       0 |       0 | 4-5
  Header.js            |       0 |        0 |       0 |       0 | 11-192
  Home.js              |       0 |        0 |       0 |       0 | 19-140
  LoginPage.js         |       0 |      100 |       0 |       0 | 6-18
  NewUser.js           |       0 |      100 |       0 |       0 | 6-29
  PortfolioCreation.js |       0 |        0 |       0 |       0 | 10-122
  Trending.js          |    90.9 |       75 |     100 |   92.85 | 40,70-71
 src/services          |     100 |    68.75 |     100 |     100 |
  fetchStockData.js    |     100 |    71.42 |     100 |     100 | 13-16
  useStockData.js      |     100 |       50 |     100 |     100 | 20
-----------------------|---------|----------|---------|---------|-------------------
Test Suites: 3 passed, 3 total
Tests:       10 passed, 10 total
Snapshots:   0 total
Time:        1.152 s
Ran all test suites.
```

Below is a screenshot of our end-to-end test results using TestCafe:



```
paulnegrido@Pauls-MacBook-Pro stock-dashboard % testcafe chrome endToEndTest/testCafeInitial.js
 Running tests in:
 - Chrome 134.0.0.0 / Sequoia 15

 Landing Page
 ✓ Check the title-Initial Sample Test
 ✓ Search for a stock1 - with success
 ✓ Search for a stock2 - with success
 ✓ Search for invalid - ticker not found
 ✓ Navigate to About page
 ✓ Navigate back to Home page
 ✓ Navigate to Login page
 ✓ Navigate back to Home page


 8 passed (8s)
paulnegrido@Pauls-MacBook-Pro stock-dashboard % 
```

## 3 Testing Issues and Next Steps

An issue was encountered trying to log-in with Google auth using TestCafe end-to-end test. This could be a conflict with auth security and automated log-ins. Currently, we plan to leave this at the basic feature for now and search for any solutions and implement as needed.

Another issue encountered was that parts of the features were harder to test. One solution to this is to refactor the code to create modular functions for the feature and make testing easier.