IT-314 Software Engineering

Project: Frontend Unit Testing with Vitest

Group Members and Student IDs

Member Name	Student Id
DHAMECHA AYUSH VIJAYBHAI	202301204
MODI GAUTAM KIRANKUMAR	202301214
SHAH JENIL SMITESHBHAI	202301249
DODIYA VIVEK JODHABHAI	202301250
VADODARIYA JENISH KIRTIKUMAR	202301202
VORA VANSH VAIBHAVKUMAR	202301266
DODIYA CHIRAYU SHANTILAL	202301236
PATEL KRISH PIYUSHKUMAR	202301264

FARHAN ANSARI	202301256
SHETHWALA MOHAMMED NAU ur MAN ZAKIR	202301237
PATEL MUKUNDBHAI RAJESHBHAI	202301234

1. BotSidebar.jsx Unit Testing Report

Unit Testing Details :

```
✓ test/unitTesting/BotSidebar.test.jsx (3 tests) 384ms
  ✓ BotSidebar Component (3)
    ✓ renders sidebar in open state initially 277ms
    ✓ toggles the sidebar open and closed when the toggle button is clicked 58ms
    ✓ shows fallback profile icon and Guest name when userDetails has no profileImage or name 47ms

Test Files  1 passed (1)
     Tests  3 passed (3)
  Start at  23:22:16
  Duration  712ms
```

Test Coverage Summary:

Detailed Test Results:

Test Case ID	Description	Test Steps / Action	Expected Result	Status

01	Sidebar opens by default	Render the <BotSidebar /> component. Verify the toggle button image, section heading, text content, profile image, and user name.	The sidebar root container has the class open. The toggle button icon is close_icon.png. The heading "What You Can Ask" is visible. The profile image is displayed, and the Username is "Gautam".	**PASS**
02	Sidebar toggles open → close → open	Render the <BotSidebar /> component. Click the toggle button once. Click the toggle button again.	The first click closes the sidebar, and the toggle icon changes to open_icon.png. The second click opens the sidebar, and the toggle icon changes back to close_icon.png. The sidebar content is visible again.	**PASS**
03	Fallback profile icon + Guest name when userDetails missing	Mock the userDetails prop to be { name: null, profileImage: null }. Render the <BotSidebar /> component.	The profile image displays the profileicon.png fallback image. The displayed user name is "Guest".	**PASS**

2. ChatWindow.jsx Unit Testing Report

Unit Testing Details :

```
✓ test/unitTesting/ChatWindow.test.jsx (7 tests) 411ms
  ✓ ChatWindow Component (6)
    ✓ renders welcome message 146ms
    ✓ shows 'Guest!' in the welcome message when user has no name 23ms
    ✓ pressing Enter sends message, starts chat, and shows typing bubble 28ms
    ✓ sends message, shows user message, removes typing bubble, and displays bot reply on success 71ms
    ✓ shows error message, removes typing bubble, and stops loading when axios request fails 68ms
    ✓ does not send a message if the input is empty 33ms
  ✓ useEffect for scrolling (1)
    ✓ scrolls to the bottom when a new message is added 40ms

 Test Files  1 passed (1)
      Tests  7 passed (7)
   Start at  00:06:18
   Duration  5.42s
```

Test Coverage Summary:

Detailed Test Results:

Test Case ID	Description	Test Steps / Action	Expected Result	Status

01	Welcome message renders correctly	Render **<ChatWindow />**	"Hello," and "How can I help you Today?" are visible. Username = **"Gautam!"**. Messages area is hidden. Input field is empty. Welcome view is visible.	**PASS**
02	Shows Guest when user has no name	Mock **userDetails = { name: null }**. Render component.	Username displays **"Guest"**. Welcome view is visible. Messages area is hidden. Input field is empty.	**PASS**
03	Pressing Enter sends message	Type "Hello bot" → press Enter	Input field clears. **chatStart = true**. Welcome view is hidden. User message shows in the chat. Typing bubble shows.	**PASS**
04	Successful message send (axios success path)	Mock **axios.success** → reply "Hello from bot!". Type "Hi bot" → click Send	Input field clears. User message shows in the chat. Typing bubble appears then is removed. Bot reply appears (markdown parsed). Loading stops.	**PASS**
05	Error on send (axios error path)	Mock **axios.reject** "Network error". Type "Hello" → click Send	Input field clears. User message shows in the chat. Typing bubble appears then is removed. Bot message shows "Network error". Loading stops. Welcome view is hidden.	**PASS**
06	Empty input does not send message	Click Send with input empty	**axios** NOT called.No typing bubble appears. Welcome view is visible. Messages area is hidden. Input field stays empty. Loading stays off. Pressing Enter also does nothing.	**PASS**

07	Scrolls to bottom when messages change	Simulate sending a message, followed by the bot's reply, using fake timers and a mocked scrollIntoView function to observe scrolling behavior.	**scrollIntoView** called once after user message. Called again after bot message. Called with **{ behavior: "smooth" }**.	**PASS**

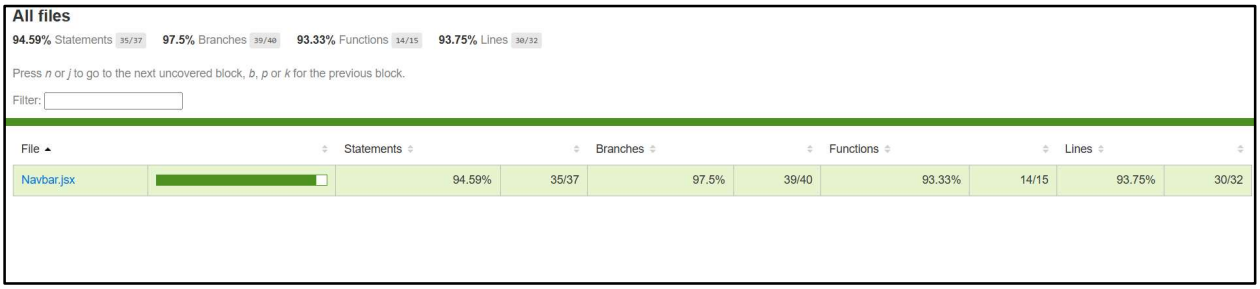
3.Navbar.jsx component Testing Report

Unit Testing Details :

```
Not implemented: navigation to another Document
 ✓ test/unitTesting/Navbar.test.jsx (13 tests) 799ms
   ✓ Navbar Component (13)
     ✓ renders the correct navbar logo based on darkMode prop 86ms
     ✓ shows "Log In" button ONLY when pageType is "/" and sets sessionStorage on clic
k  409ms
     ✓ renders homepage navigation buttons only when pageType is '/' & dashboard navig
ation buttons only when pageType is not 'Dashboard' 36ms
     ✓ shows profile button, uses correct image, and opens popup when profileData exis
ts 87ms
     ✓ opens and closes the mobile menu when hamburger icon is clicked 32ms
     ✓ navigates to My Profile when clicked inside profile popup 18ms
     ✓ route links inside profile popup trigger correct navigation paths 23ms
     ✓ uses fallback profile image when userDetails.profileImage is null & profile ima
ge visible 10ms
     ✓ profile image is HIDDEN for home page '/' 8ms
     ✓ closes the mobile menu when window is resized above 1100px 16ms
     ✓ calls logout API and navigates to home on successful logout 33ms
     ✓ does not navigate when logout API fails (error handled) 18ms
     ✓ mobile login link sets sessionStorage values on homepage 19ms

 Test Files  1 passed (1)
      Tests  13 passed (13)
```

Test Coverage Summary:



Detailed Test Results:

Test Case ID	Description	Test Steps / Action	Expected Result	Status
01	Renders correct logo based on dark mode	The Navbar is first rendered with darkMode = false and then rerendered with darkMode = true, observing which logo file is displayed each time.	The correct light or dark logo appears based on the prop, and remains wrapped inside its anchor tag.	**PASS**
02	Log In button appears only on home page and writes sessionStorage	The Navbar runs with pageType="/" , the Log In button is clicked, and stored session values are checked.	The Log In button appears only on "/", and clicking it stores isLogin=true & forgotpassword=false.	**PASS**
03	Homepage buttons vs dashboard buttons appear correctly	Navbar is rendered first on / and then on a dashboard page while checking which navigation links appear in each mode.	Home-page buttons (Features, How it Works?, FAQs) appear only on /, while dashboard items appear only for dashboard pages.	**PASS**
04	Profile button appears, uses correct image, and opens/closes popup	With valid profileData, the profile icon is clicked to open the popup, and the overlay is clicked to close it.	The profile button appears with the fallback image if needed, the popup opens on click, and closes on overlay click.	**PASS**

05	Mobile menu toggles open/closed with hamburger button	The hamburger icon is clicked to open the menu, closed via overlay, opened again, and then closed again	The mobile menu appears, hides, and toggles correctly while showing proper navigation options.	**PASS**
06	Navigates to My Profile from popup	The profile popup is opened and the “My Profile” entry is clicked.	Navigation is triggered to /my-profile.	**PASS**
07	All profile menu routes navigate correctly	Inside the open profile popup, each item (“My Profile”, “Manage”, “Help & Support”) is clicked.	Navigation is triggered to /my-profile, /data-privacy, and /help-support.	**PASS**
08	Uses fallback profile image when userDetails.profileImage is null	The context is mocked to provide no profile image, and Navbar is rendered on dashboard.	The profile icon falls back to profileicon.svg and remains visible.	**PASS**
09	Profile icon is hidden on home page	Navbar is rendered with pageType="/" but with valid profileData.	The profile icon element renders but CSS visibility is hidden.	**PASS**
10	Mobile menu closes automatically when resizing above 1100px	The menu is opened manually, then window.innerWidth is increased, and a resize event is dispatched.	The mobile menu closes in response to window resizing.	**PASS**

11	Successful logout triggers axios request and navigation to home	With axios mocked to succeed, “Log Out” inside the popup is clicked.	Logout API is called and the user is navigated to /.	**PASS**
12	Logout failure prevents redirect	With axios mocked to fail, “Log Out” is clicked.	The API call runs but no navigation occurs because logout failed.	**PASS**
13	Mobile menu “Log In” sets sessionStorage correctly	On the home page, the mobile menu is opened and the Log In list item is clicked.	sessionStorage is updated: isLogin=true, forgotpassword=false.	**PASS**

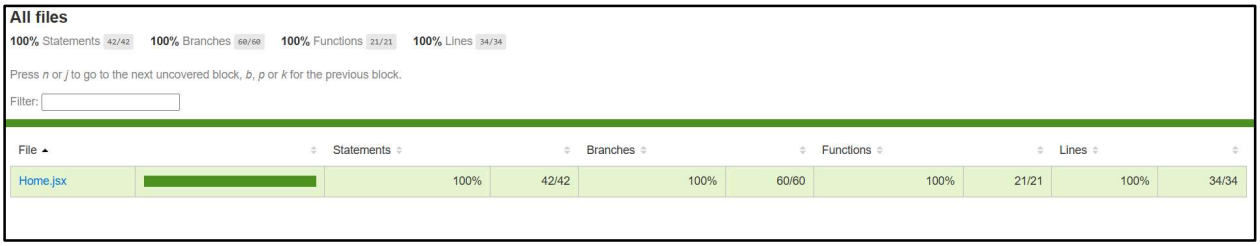
4.Home.jsx component Testing Report

Unit Testing Details :

```
✓ test/unitTesting/Home.test.jsx (8 tests) 736ms
  ✓ Home Component (8)
    ✓ renders Home with Navbar and Footer 72ms
    ✓ redirects to dashboard when token is valid 96ms
    ✓ does not redirect when token is invalid 21ms
    ✓ Get Started button sets sessionStorage correctly 32ms
    ✓ Sign Up Now button sets sessionStorage as expected 24ms
    ✓ all FAQ items toggle open and closed  390ms
    ✓ all feature cards expand individually when clicked 76ms
    ✓ does not expand feature card on mobile view 22ms

 Test Files  1 passed (1)
      Tests  8 passed (8)
   Start at  22:21:07
   Duration  3.04s (transform 230ms, setup 207ms, collect 680ms, tests 736ms, environment 993ms, prepare 104ms)
```

Test Coverage Summary:



Detailed Test Results:

Test Case ID	Description	Test Steps / Action	Expected Result	Status
01	Renders Home with Navbar and Footer	The Home component is rendered normally, allowing its layout to load fully.	The mock Navbar and Footer components are both shown in the DOM, confirming that the page initializes correctly.	**PASS**

02	Redirects to dashboard when token is valid	The axios token check is mocked to return a successful response, and the Home component is mounted to trigger its useEffect.	Navigation is triggered to /dashboard, confirming the redirect for authenticated users.	**PASS**
03	Does not redirect when token is invalid	The axios token check is mocked to reject with an error, and the Home component is rendered.	No navigation occurs, keeping the user on the Home page if authentication fails.	**PASS**
04	Get Started button updates sessionStorage	The “Get Started” button in the hero section is clicked.	sessionStorage is updated so isLogin becomes "true" and forgotpassword becomes "false".	**PASS**
05	Sign Up Now button updates sessionStorage	The Sign Up Now button at the bottom of the page is clicked.	sessionStorage is updated with isLogin="false" and forgotpassword="false" confirming correct behavior for signup.	**PASS**
06	FAQ items toggle open and closed	Each FAQ arrow is clicked to show its answer, then clicked again to hide it.	Every answer block becomes visible after the first click and hides again after the second click, proving proper toggle functionality.	**PASS**
07	Feature cards expand individually	Each feature card is clicked one at a time to expand, while the other cards are observed for their hidden state.	Only the clicked card gains the expanded style, and all others switch to hidden; clicking “See less” collapses the expanded card.	**PASS**

08	Feature cards do not expand on mobile view	The browser window width is set artificially to a mobile resolution before clicking a feature card.	The clicked card remains unexpanded, confirming that the expansion feature is disabled on mobile screens.	**PASS**

5.WelcomeInvestor.jsx component Testing Report

Unit testing:

```
✓ test/unitTesting/WelcomeInvestor.test.jsx (11 tests) 388ms
  ✓ WelcomeInvestor — 100% Branch Coverage (11)
    ✓ renders dashboard on successful valuation + user fetch 84ms
    ✓ renders zero values when valuation returns 'No summary found' 23ms
    ✓ shows session expired on 401 57ms
    ✓ handles network failure 33ms
    ✓ renders trending stock items 27ms
    ✓ navigates to stock details on click 47ms
    ✓ shows trending error on backend failure 21ms
    ✓ does not navigate when clicking random UI element 17ms
    ✓ shows HIGH risk category 20ms
    ✓ shows MODERATE risk category 26ms
    ✓ shows LOW risk category 31ms

 Test Files  1 passed (1)
      Tests  11 passed (11)
   Start at  16:01:58
   Duration  1.02s
```

Coverage:

Detailed Test Result::

Test Case ID	Description of Test Case	Test Steps	Expected Outcome (Result)	Status

01	Validation of successful dashboard and comprehensive valuation summary rendering.	Await the complete asynchronous loading of data assets.	The dashboard shall render accurately, exhibiting user-specific details and a fully processed valuation summary.	PASS
02	Verification of system response when valuation service returns a 'No summary found' status.	Ensure the backend environment is configured to issue a 'No summary found' response for the valuation API call.	The valuation component shall render nominal or default numerical values, accompanied by a pertinent message indicating the absence of summary data.	PASS
03	Assessment of session expiration management and user redirection.	. Attempt to initiate a dashboard operation (e.g., data refresh).	A "Session Expired" notification shall be displayed, and the user shall be automatically routed to the application's login interface.	PASS
04	Validation of network communication failure handling during dashboard data retrieval.	Sever the network connection immediately subsequent to initiating the data fetching sequence.	A suitable network error or failure message shall be presented on the dashboard, preventing the display of incomplete or corrupt data.	PASS

05	Verification of accurate rendering of trending stock market instruments.	Locate the designated 'Trending Stocks' module.	The 'Trending Stocks' module shall successfully display a populated list of stock instruments, inclusive of requisite financial metrics.	PASS
06	Confirmation of correct navigation to stock detail pages upon interaction with a trending item.	Within the dashboard interface, identify and select a specific trending stock instrument.	The system shall successfully transition the user to the dedicated detail page corresponding to the selected stock instrument.	PASS
07	Evaluation of error message presentation when trending stock data retrieval fails.	Ensure the backend is configured to return a failure or error status for the trending stocks API.	A fitting "trending error" communication shall be displayed prominently within the 'Trending Stocks' module.	PASS
08	Validation that no unauthorized navigation occurs upon clicking non-interactive user interface elements.	Within the dashboard, interact (click) with static textual content, background areas, or non-functional UI components (excluding links or buttons).	No unexpected navigation or functional operation shall be triggered. The user shall be retained on the current dashboard view.	PASS

09	Verification of the correct presentation of the HIGH risk classification within the valuation summary.	. Configure the valuation data to yield a 'HIGH' risk categorization result.	The valuation summary shall distinctly display the "HIGH risk category" label or indicator.	PASS
10	Verification of the correct presentation of the MODERATE risk classification within the valuation summary.	Configure the valuation data to yield a 'MODERATE' risk categorization result.	The valuation summary shall distinctly display the "MODERATE risk category" label or indicator.	PASS
11	Verification of the correct presentation of the LOW risk classification within the valuation summary.	. Configure the valuation data to yield a 'LOW' risk categorization result.	The valuation summary shall distinctly display the "LOW risk category" label or indicator.	PASS

6.PortFolioChart.jsx Testing Report

Unit Testing Details :

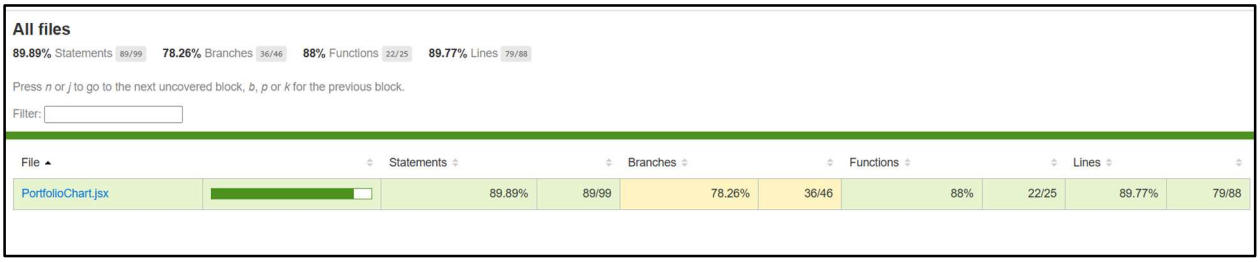
```
    at fetchPortfolioData (C:/Users/vivek/OneDrive/Desktop/Main-branch/InsightStox---Portfolio-analyzer-tracker-and
-console/frontend/src/components/PortfolioChart/PortfolioChart.jsx:171:31)

 ✓ test/unitTesting/PortfolioChart.test.jsx (14 tests) 270ms
   ✓ PortfolioChart Component (14)
     ✓ renders loading state initially 81ms
     ✓ renders error message on API failure 16ms
     ✓ renders session expired on 401 6ms
     ✓ fetches data and renders chart 9ms
     ✓ switches to 6M range 19ms
     ✓ switches to 1Y range 14ms
     ✓ uses cached data on rerender 18ms
     ✓ updates config on window resize 13ms
     ✓ tooltip parser branch covered 8ms
     ✓ covers Chart.js scales & grid callback logic 20ms
     ✓ 6M range produces empty labels except first of month 19ms
     ✓ 1Y range generates month labels on month start 13ms
     ✓ handles missing daily data gracefully 19ms
     ✓ cache path regenerates labels and values correctly 11ms

  Test Files  1 passed (1)
       Tests  14 passed (14)
    Start at  09:52:03
    Duration  2.80s (transform 281ms, setup 323ms, collect 605ms, tests 270ms, environment 1.10s, prepare 68ms)

 % Coverage report from v8
```

Coverage:



Detailed Test Result::

Test Case ID	Description	Test Steps	Expected Result	Status

01	Verification of initial loading state rendering for the component.	Initiate the display of the component or application page incorporating the chart.	The component shall initially display a distinct loading indicator or state prior to the completion of data retrieval.	PASS
02	Assessment of error message presentation upon data API failure.	Configure the operating environment to simulate an erroneous response from the data application programming interface (API). Load the component.	An appropriate and informative error message shall be rendered, superseding the loading state, and preventing the visualization of an empty chart.	PASS
03	Validation of component behavior and display upon detection of an expired user session.	Configure the back-end system to return a "session expired" status code during the data fetching process. Initiate the display of the component.	A formal "Session Expired" notification shall be presented to the user, potentially including options for redirection or re-authentication.	PASS
04	Confirmation of successful data acquisition and subsequent chart rendering accuracy.	Load the component utilizing a correctly configured and valid data source.	The component shall successfully retrieve the requisite data and accurately render the corresponding visual chart representation.	PASS
05	Verification of chart data and scale adjustments following selection of the 6-Month (6M) range.	Access the chart's time range selection interface and select the "6M" option.	The chart shall dynamically adjust its displayed data set and time scale to encompass the preceding six months of activity.	PASS

06	Verification of chart data and scale adjustments following selection of the 1-Year (1Y) range.	Access the chart's time range selection interface and select the "1Y" option.	The chart shall dynamically adjust its displayed data set and time scale to encompass the preceding one year of activity.	PASS
07	Validation of internal data cache utilization upon component re-render operations.	Load the component, permit data fetching completion, and subsequently trigger a component re-render event (e.g., a parent state update unrelated to data dependency).	The component shall execute a rapid re-render utilizing the existing cached data, thereby circumventing the initiation of a new API request.	PASS
08	Assessment of chart configuration responsiveness to changes in browser window dimensions.	Initiate the display of the chart and then substantially modify the size of the browser window (e.g., transitioning from a wide desktop to a narrow mobile resolution).	The chart's configuration parameters (e.g., aspect ratio, label density) shall dynamically update to optimally fit the newly available screen space.	PASS
09	Verification of the complete execution path for the tooltip parser logic.	Interact with various data points on the chart to ensure the display of the associated tooltip.	The tooltip shall present correctly formatted and accurate data, confirming the successful and complete execution of the tooltip parser function.	PASS
10	Validation of callback function execution for Chart.js scales and grid line rendering.	Conduct an inspection of the rendered chart's axes and grid lines.	The scales and grid lines shall be rendered in strict accordance with the defined callback logic (e.g., custom tick value formatting, conditional line display).	PASS

11	Verification of X-axis label generation for the 6M range, displaying only month-start labels.	Select the "6M" time range and visually inspect the X-axis labels.	The X-axis shall present labels exclusively for the first day of each month within the 6-month period, suppressing intermediate daily labels.	PASS
12	Verification of X-axis label generation for the 1Y range, displaying labels precisely at month start.	Select the "1Y" time range and visually inspect the X-axis labels.	The X-axis shall display distinct labels positioned accurately at the initiation of each month within the 1-year period.	PASS
13	Assessment of the component's robustness in handling absent data points within a daily time series.	Configure the data source to return a series intentionally lacking one or more data points for specific days. Load the component.	The chart shall render without error, utilizing techniques such as interpolation or omission for the missing data points, avoiding visual corruption or application failure.	PASS
14	Validation of the cache mechanism's ability to accurately regenerate chart labels and values.	Load the chart, allow data fetching to complete, and then simulate an event that forces the chart to regenerate its rendering based on its internal cache (e.g., a range switch where data is pre-fetched).	The regenerated chart shall display accurate time-series labels and corresponding data values, thereby confirming the integrity and efficacy of the cache.	PASS

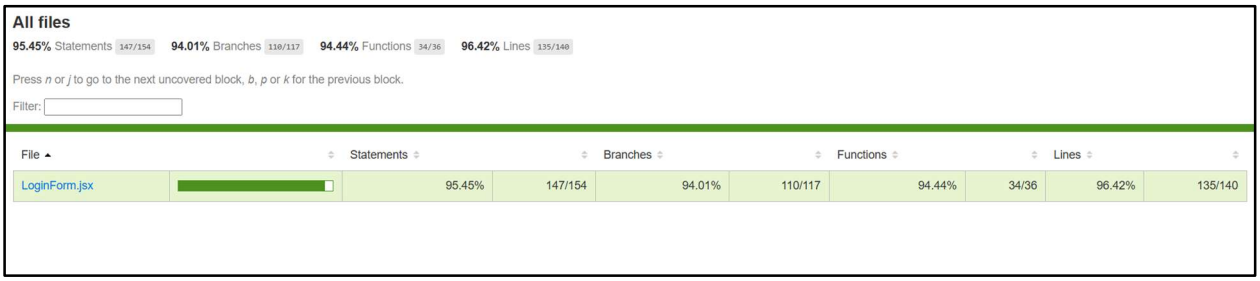
7.LoginForm.jsx component Testing Report

Unit Testing Details :

```
✓ test/unitTesting/LoginForm.test.jsx (14 tests) 1220ms
  ✓ LoginForm Component (14)
    ✓ renders LoginForm with initial login UI correctly 210ms
    ✓ toggles Forgot Password mode and updates UI + sessionStorage 112ms
    ✓ login button enables only when email is valid and password is non-empty 101ms
    ✓ sends OTP successfully and updates UI (Send OTP → Verify OTP) 101ms
    ✓ Google login button triggers login flow 64ms
    ✓ Google login backend failure displays backend error message 79ms
    ✓ Google login onError sets error message 70ms
    ✓ Google login loading stops when window regains focus 56ms
    ✓ Google login throw triggers catch block (popup failure) 62ms
    ✓ handleLogin success redirects to Dashboard and sets userLoggedIn 53ms
    ✓ handleSendOtpForForgotPassword handles backend error (catch block) 61ms
    ✓ OTP verification success sets flags and reset password works successfully 102ms
    ✓ OTP verification failure hits catch block and shows error message 97ms
    ✓ clicking 'Sign up' calls toggleForm, resets parent state and clears titleError 49ms

 Test Files  1 passed (1)
      Tests  14 passed (14)
   Start at  21:58:20
   Duration  1.66s
```

Test Coverage Summary:



Detailed Test Results:

Test Case ID	Description	Test Steps / Action	Expected Result	Status
01	Initial Login UI Renders Correctly	The LoginForm component is mounted in normal login mode and the interface is observed.	The title “Login to your account”, Email and Password fields, Forgot link, disabled Login button, Google button, and the logo	**PASS**

			all appear correctly.	
02	Toggle Forgot Password Mode	Clicking “Forgot?” switches UI from login mode to forgot-password mode, and clicking “Login” switches back.	The title changes to “Reset your password”, the password field disappears, the Send OTP button appears, sessionStorage updates, and returning to login mode restores the original UI.	**PASS**
03	Login Button Enables When Email & Password Valid	An invalid email is typed, then replaced with a valid email, and finally a password is entered before clicking Login.	Email validation shows and then clears its error, the Login button becomes enabled, clicking it triggers backend error which appears on screen.	**PASS**
04	Send OTP Flow Works (Switch to Verify OTP)	Forgot mode is activated, a valid email is entered, and Send OTP is clicked.	The backend is called with correct OTP API, the OTP field appears, the button label switches to Verify OTP, and the resend timer text becomes visible.	**PASS**
05	Google Login Success Flow	The Google login button is clicked which triggers mocked OAuth success path.	The Google login function fires, backend receives the token, and login succeeds without errors.	**PASS**
06	Google Login Backend Failure Shows Error	Google OAuth succeeds but backend login fails.	The error message from backend is displayed, confirming catch block works.	**PASS**

07	Google Login onError Displays Error Message	Google OAuth intentionally triggers the onError callback.	The component shows “Google login failed”, no backend calls happen, and no navigation occurs.	**PASS**
08	Google Login Loading Stops When Window Regains Focus	Google login is triggered (loading starts), then window focus event is simulated.	The “Processing” loader disappears after timeout, confirming the component resets loading state.	**PASS**
09	Google Login Popup Failure (Throw) Triggers Catch Block	Clicking the Google button throws an exception to simulate popup failure.	Loading stops immediately and the UI does not crash or show Google login errors.	**PASS**
10	Login Success Redirects and Updates Context	Valid email and password are entered, and Login is clicked.	Backend login endpoint is called, setUserLoggedIn is set to true, and navigation occurs to /Dashboard.	**PASS**
11	Send OTP Backend Error Shows Message (Catch Block)	Forgot mode is entered, an email is typed, and Send OTP triggers a backend failure.	Error message appears, OTP field does not show, resend text is hidden, loading stops, and the Send OTP button stays enabled.	**PASS**
12	OTP Verification Success Leads to Password Reset Success	Send OTP succeeds, the OTP is entered and verified, then a new password is submitted.	OTP success reveals new password field, reset password API is called, userLoggedIn becomes true, and navigation moves to /Dashboard.	**PASS**
13	OTP Verification Failure Displays Error & Blocks Progress	OTP send succeeds, but verification fails.	Error message “Invalid OTP” appears, reset password section never appears, and Verify OTP button remains available.	**PASS**

14	Clicking 'Sign Up' Calls toggleForm + Resets Parent State	Login is attempted to generate an error, then the Sign Up link is clicked.	toggleForm and resetFormStates functions are called, and the previous error message is cleared.	**PASS**

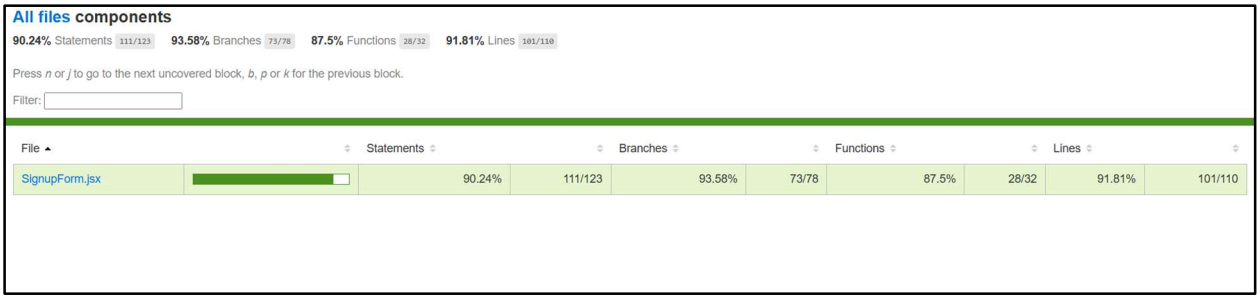
8.Signupform.jsx component Testing Report

Unit Testing Details :

```
 ✓ test/unitTesting/SignupForm.test.jsx (5 tests) 11569ms
   ✓ SignupForm (unit) - OTP generation path (5)
     ✓ fills fields, accepts privacy policy, clicks Sign Up and starts OTP flow  1982
ms
     ✓ even if checkbox checked, wrong inputs must keep SignUp disabled  3964ms
     ✓ OTP page: verify OTP works correctly when user enters OTP  1624ms
     ✓ OTP page: wrong OTP shows error message  1570ms
     ✓ Re-signup with same email triggers OTP error and clicking Login resets all fie
lds  2423ms

 Test Files  1 passed (1)
      Tests  5 passed (5)
   Start at  18:28:19
   Duration  16.38s (transform 461ms, setup 331ms, collect 1.31s, tests 11.57s, envir
onment 2.27s, prepare 104ms)
```

Test Coverage Summary:



Detailed Test Results:

Test Case ID	Description	Test Steps / Actions	Expected Result	Status
01	OTP generation should start only when all inputs are valid and the privacy policy is accepted.	Render the SignupForm, fill all fields with valid inputs, accept the privacy policy checkbox, and click the Sign Up button.	The Sign Up button becomes enabled only after all inputs are valid, the OTP API is called with the registerOtpGeneration endpoint, and the UI switches to the OTP verification screen.	**PASS**
02	Invalid or incomplete input combinations must keep the Sign Up button disabled and the privacy policy popup should open and close correctly.	Tick the privacy checkbox, then enter multiple invalid combinations such as wrong email, invalid name, weak password, or mismatching states, and also open and close the privacy popup using the accept button, close button, and overlay click.	The Sign Up button stays disabled for all invalid input combinations and the privacy popup opens and closes correctly through all three interactions.	**PASS**
03	OTP verification should work correctly when the user enters a valid OTP.	Complete the signup process, navigate to the OTP screen, type a valid OTP, ensure the Verify OTP button becomes active, and click it to submit.	The OTP verification API is called with the correct payload and the user is navigated to the Dashboard screen after successful verification.	**PASS**
04	A wrong OTP should be rejected and the correct error message displayed.	Complete signup, go to the OTP screen, enter an incorrect OTP, and attempt verification.	The API rejects the OTP and the UI displays the message “Invalid OTP” to the user.	**PASS**

05	Returning from the OTP screen to Login should reset all fields, and a second signup with the same email should show a “User already exists” error.	After reaching the OTP screen, click the Login link to trigger reset and toggle actions, verify fields are cleared, fill the form again using the same email, and attempt signup.	The form fields reset successfully, the resetFormStates and toggleForm callbacks are triggered, and the second signup attempt displays the message “User already exists.”	**PASS**

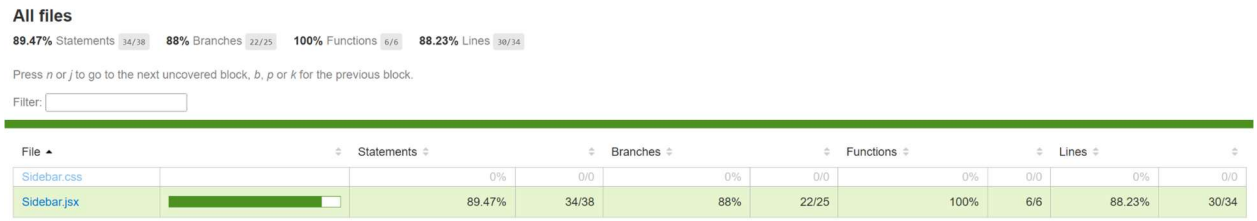
9. Sidebar.jsx component Testing report

Unit Testing results:

```
✓ test/unitTesting/Sidebar.test.jsx (10 tests) 198ms
  ✓ Sidebar Component (10)
    ✓ renders Sidebar with default profile data and highlights My Profile 39ms
    ✓ navigates to Data & Privacy when clicked 12ms
    ✓ calls logout API and navigates home 69ms
    ✓ handles logout error gracefully 6ms
    ✓ renders custom profile image when provided 4ms
    ✓ restores active menu from localStorage 4ms
    ✓ activates Data & Privacy when route is /data-privacy 16ms
    ✓ activates Activity when route is /activity 15ms
    ✓ activates Preferences when route is /preferences 15ms
    ✓ activates Help & Support when route is /help-support 16ms

Test Files  1 passed (1)
     Tests  10 passed (10)
  Start at  10:21:56
  Duration  1.55s (transform 129ms, setup 149ms, collect 308ms, tests 198ms, environment 671ms,
```

Test Coverage Summary:



All files

89.47% Statements 34/38 88% Branches 22/25 100% Functions 6/6 88.23% Lines 30/34

Press n or j to go to the next uncovered block, b, p or k for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
Sidebar.css	0%	0/0	0%	0/0	0%	0/0	0%	0/0
Sidebar.jsx	89.47%	34/38	88%	22/25	100%	6/6	88.23%	30/34

Detailed Test results:

Test Case ID	Description	Test Steps / Action	Expected Result	Status
01	Verify sidebar renders name, email, default profile image, and highlights “My Profile”.	Render Sidebar with basic primaryData.	Profile details are shown and “My Profile” has an Active class.	**PASS**
02	Ensure clicking “Data & Privacy” navigates to correct route.	Render Sidebar and click the “Data & Privacy” button.	navigate("/data-privacy") is called	**PASS**

03	Verify clicking Logout triggers API and redirects to home.	Mock axios success, click “Logout”.	Logout API is called and navigation to / happens.	**PASS**
04	Ensure logout failure doesn’t redirect the user.	Mock axios rejection and click “Logout”.	Logout API is called but navigation does NOT occur.	**PASS**
05	Verify Sidebar uses a custom profile image if provided.	Render Sidebar with profileImage = “custom.jpg”	Sidebar shows "custom.jpg" as image source instead of default.	**PASS**
06	Ensure Sidebar loads previously selected active menu.	Set localStorage.setItem("activeMenu", "Activity"). Render <Sidebar/>. Verify menu highlight.	“Activity” menu item is automatically highlighted on load.	**PASS**
07	Ensure pathname /data-privacy activates correct menu.	Mock location.pathname = "/data-privacy" and render.	“Data & Privacy” menu item has Active class.	**PASS**
08	Ensure pathname /activity activates Activity menu.	Mock pathname as /activity and render.	“Activity” menu item has Active class.	**PASS**
09	Validate selection based on /preferences route.	Mock pathname /preferences and render.	“Preferences” becomes Active.	**PASS**
10	Verify highlight when route is /help-support	Mock pathname /help-support and render.	“Help & Support” is Active.	**PASS**

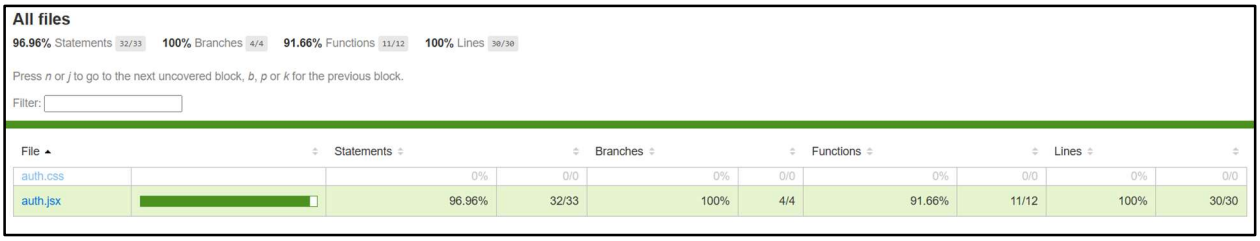
10.Auth.jsx page Testing report

Unit Testing Details:

```
Test Files 0 passed (1)
     Tests 0 passed (0)
  Start at 01:08:45
  Duration 2.00s
✓ test/unitTesting/auth.test.jsx (6 tests) 537ms
  ✓ Auth Component (6)
    ✓ Test 1: renders LoginForm by default 73ms
    ✓ Test 2: renders SignupForm when isLogin = false in sessionStorage 8ms
    ✓ Test 3: toggleForm updates state and sessionStorage, switching LoginForm → SignupForm 21ms
    ✓ Test 4+5: Back button and popstate event both trigger resetFormStates() and clear sessionStorage  326ms
    ✓ Test 6: console.error is called when ensureAuth initial check fails 90ms
    ✓ Test 7: ensureAuth is called repeatedly by setInterval every 5000ms 15ms

Test Files  1 passed (1)
     Tests  6 passed (6)
  Start at  01:08:45
  Duration  1.37s
```

Test Coverage Summary:



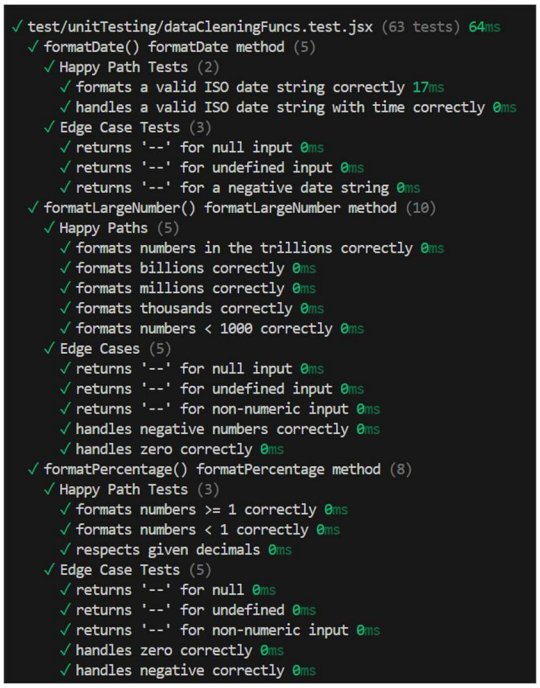
Detailed Test results:

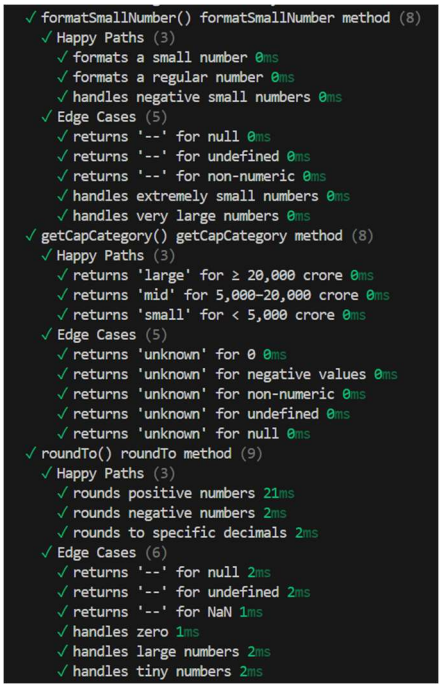
Test Case ID	Description	Test Steps / Actions	Expected Output	Status
01	Render default LoginForm	The test clears sessionStorage, renders the Auth component, and checks whether LoginForm appears while SignupForm does not.	LoginForm is displayed, SignupForm is absent.	**PASS**

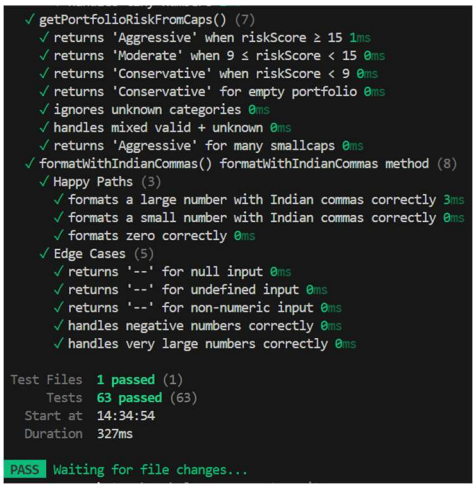
02	Render SignupForm when isLogin = false	The test sets sessionStorage.isLogin to "false", renders Auth, and verifies that SignupForm appears instead of LoginForm.	SignupForm is displayed and LoginForm is not.	**PASS**
03	Toggle Login → Signup and update sessionStorage	After rendering AuthToggle (mocked LoginForm with toggle), the test clicks the mock toggle button and verifies that SignupForm replaces LoginForm and sessionStorage updates to "false".	SignupForm becomes visible and isLogin is updated to "false".	**PASS**
04	Back button & popstate event clears sessionStorage	The test sets sessionStorage values, renders Auth, clicks the Back button, and confirms that both keys are cleared.	isLogin and forgotpassword are removed from sessionStorage.	**PASS**
05	Initial ensureAuth failure logs console.error	After mocking ensureAuth to reject, the test renders Auth and waits to confirm that console.error is called during the initial check.	console.error is triggered once due to failure.	**PASS**
06	Interval repeatedly calls ensureAuth	Using fake timers, the test renders Auth, advances time in 5000ms steps, and checks ensureAuth call count.	ensureAuth is called once on mount and again every 5000ms.	**PASS**

11. dataCleaningFunc.jsx Testing report

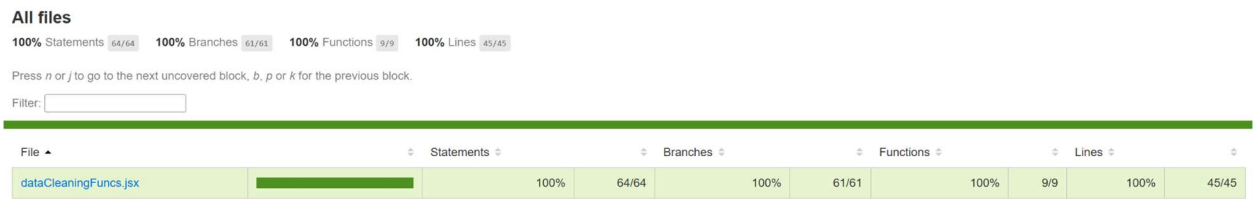
Unit Testing details:







Test Coverage Summary:



- This file mainly contains functions to beautify the data for representation and has two functions which determine the risk profile based on the market capitalization of stocks in the portfolio. Thus, there are a huge number of tiny test cases which have tested all these functions. Above are the screenshots which contain the names of tested cases.

12. Portfolio.jsx Testing report

Unit Testing details :

```
✓ test/unitTesting/Portfolio.test.jsx (14 tests) 182ms
  ✓ Portfolio() Portfolio method (14)
    ✓ Happy Paths (7)
      ✓ renders Portfolio with default summary mode 54ms
      ✓ triggers handleMode('summary') when Summary button is clicked 12ms
      ✓ switches to holdings mode when clicked 13ms
      ✓ switches to fundamentals mode when clicked 10ms
      ✓ assigns 'profit' class when today's and overall gain are positive 16ms
      ✓ assigns 'loss' class when today's and overall gain are negative 16ms
      ✓ activates search panel when Add Stock is clicked 8ms
    ✓ Edge Cases (7)
      ✓ handles axios failure safely (Summary stays visible) 6ms
      ✓ handles empty summary safely 9ms
      ✓ defaults to summary for invalid localStorage mode 6ms
      ✓ handles fundamentals API failure and sets error 9ms
      ✓ handles holdings API failure (success = false) 6ms
      ✓ handles holdings API error via catch block 6ms
      ✓ catches fundamentals API error (catch block) 6ms

Test Files  1 passed (1)
     Tests  14 passed (14)
  Start at  15:28:28
  Duration  1.75s (transform 267ms, setup 157ms, collect 421ms, tests 182ms, environment
```

Test Coverage summary:

Detailed Test Results:

Test Case ID	Description	Test Steps / Actions	Expected Output	Status

01	Renders Portfolio with default summary mode	Render <Portfolio/>	Navbar, Header, Chart, Footer, and PortfolioSummary mock should appear	**PASS**
02	Triggers handleMode('summary') on Summary click	Click **Summary** button	PortfolioSummary mock stays visible (branch covered)	**PASS**
03	Switches to holdings mode	Click **Holdings**	PortfolioHoldings mock is displayed	**PASS**
04	Switches to fundamentals mode	Click **Fundamentals**	PortfolioFundamentals mock is displayed	**PASS**
05	Applies profit class when today's & overall gain positive	Mock API with positive profits → render	today-gl-amount and overall-gl-amount have profit classes	**PASS**
06	Applies loss class when today's & overall gain negative	Mock API with negative profits → render	today-gl-amount and overall-gl-amount have loss classes	**PASS**
07	Activates search panel on Add Stock click	Click **Add Stock** button	setIsSearchActive(true) is called	**PASS**

08	Handles axios failure safely on initial load	Mock axios.get reject	PortfolioSummary mock still renders	**PASS**
09	Handles empty summary safely	Mock summary: [ ]	Empty PortfolioSummary mock renders normally	**PASS**
10	Defaults to summary on invalid localStorage mode	Set localStorage to invalid mode → render	PortfolioSummary mock loads	**PASS**
11	Handles fundamentals API failure (success=false)	Mock fundamentals: success = false	Summary mock still renders (error branch covered)	**PASS**
12	Handles holdings API failure (success=false)	Mock holdings: success = false	Summary mock still renders	**PASS**
13	Handles holdings API error via catch block	Mock holdings API throw error	Summary mock still renders	**PASS**
14	Handles fundamentals API error via catch block	Mock fundamentals API throw error	Summary mock still renders	**PASS**

13.SectorAllocation.jsx Testing report

Unit Testing Details:

```
stderr | test/unitTesting/SectorAllocation.test.jsx > SectorAllocationChart Component – Full Coverage Suite > handl
es missing labels while values exist
Error fetching allocation data: Error: Invalid data format received from backend
    at fetchAllocationData (C:/Users/vivek/OneDrive/Desktop/Main-branch/InsightStox---Portfolio-analyzer-tracker-an
d-console/frontend/src/components/SectorAllocation/SectorAllocation.jsx:105:15)

 ✓ test/unitTesting/SectorAllocation.test.jsx (10 tests) 223ms
   ✓ SectorAllocationChart Component – Full Coverage Suite (10)
     ✓ renders loading state initially 75ms
     ✓ renders chart on successful fetch 22ms
     ✓ handles missing labels or values (invalid backend format) 30ms
     ✓ handles generic backend failure 31ms
     ✓ handles 401 session expired 15ms
     ✓ chart receives correct labels, values, and dynamic colors 9ms
     ✓ tooltip callback formats label correctly 6ms
     ✓ animation options exist and are passed to chart 5ms
     ✓ handles missing values while labels exist 10ms
     ✓ handles missing labels while values exist 16ms

 Test Files  1 passed (1)
      Tests  10 passed (10)
   Start at  10:15:55
   Duration  874ms

 % Coverage report from v8
```

Test Coverage Summary:

Detailed Test results:

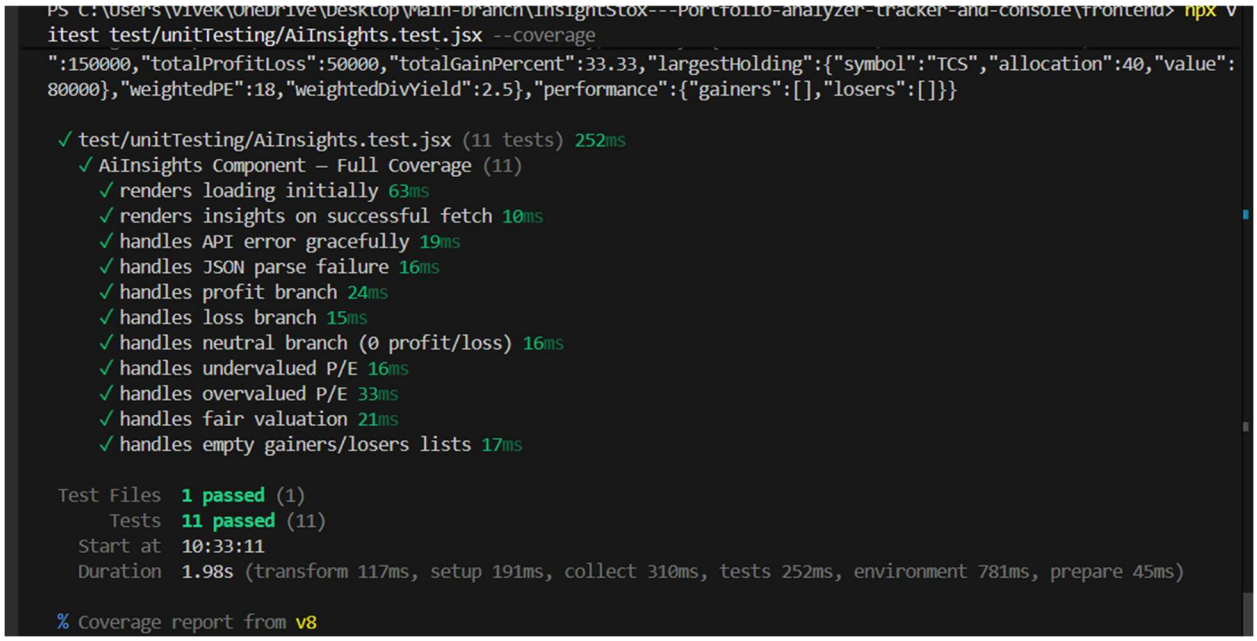
TC ID	Description	Test Steps	Expected Result	Status
01	Verify initial loading state.	Observe initial load.	Loading indicator/skeleton state displayed before data fetch.	**PASS**

02	Confirm chart renders on successful data fetch.	. Ensure valid backend response.	Chart successfully renders visual representation without errors.	**PASS**
03	Handle missing labels/values (invalid format).	1. Configure backend for invalid data structure (missing arrays). 2. Load component.	Graceful error handling (error message or "No Data"); no crash.	**PASS**
04	Evaluate generic backend failure (e.g., 500).	1. Configure backend for generic failure (e.g., HTTP 500). 2. Load component.	Appropriate, informative error message displayed instead of chart.	**PASS**
05	Validate session expiration handling (401).	1. Configure backend for HTTP 401. 2. Load component.	"Session Expired" notification or automatic redirection to login.	**PASS**
06	Verify correct data, labels, and dynamic colors.	1. Load with valid data. 2. Visually inspect data points, labels, and colors.	Chart displays correct values, labels, and applied dynamic colors.	**PASS**
07	Confirm tooltip label formatting.	1. Load component. 2. Hover over data points to trigger tooltip.	Tooltip displays data labels and values formatted per business rules.	**PASS**
08	Validate animation options are applied.	1. Load component. 2. Observe chart rendering and updates.	Chart exhibits expected animation behavior (duration, easing, etc.).	**PASS**

09	Handle missing values but existing labels.	1. Configure data with all labels, but specific values omitted. 2. Load component.	Chart renders without error, showing gaps or using interpolation for missing values.	**PASS**
10	Handle missing labels but existing values.	1. Configure data with values, but corresponding labels omitted. 2. Load component.	Chart attempts to render, using default/generated labels, or displays an error/warning.	**PASS**

14.AiInsights.jsx Testing report

Unit Testing Details:



Test Coverage Summary:

Detailed Test results:

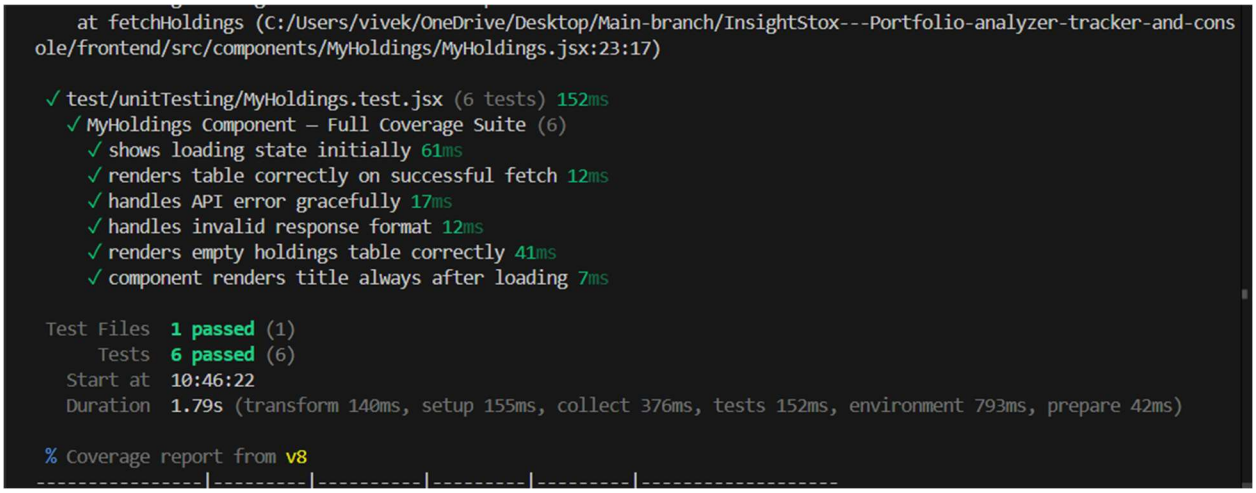
TC ID	Description	Steps	Expected Result	Status
01	Verify initial loading state.	Load component; observe display pre-data.	Loading indicator/skeleton screen displays.	**PASS**

02	Validate insight rendering on success.	Fetch data successfully; load component.	Renders all financial insights without errors.	**PASS**
03	Handle generic API errors (e.g., 500).	Configure API to return error; load component.	A graceful, user-friendly error message displays.	**PASS**
04	Handle unparsable JSON failure.	Configure API for malformed data; load component.	Handles parsing failure gracefully (e.g., error msg, no crash).	**PASS**
05	Validate positive profit/gain display.	Configure data for positive profit; load component.	Prominently displays positive profit with green UI.	**PASS**
06	Validate negative profit/loss display.	Configure data for loss; load component.	Prominently displays loss with red UI.	**PASS**
07	Validate neutral (zero) profit/loss display.	Configure data for zero profit/loss; load component.	Displays neutral value (0) with appropriate neutral UI.	**PASS**
08	Confirm 'undervalued' P/E indicator.	Configure data for 'undervalued' P/E; load component.	Displays "Undervalued" P/E label/indicator.	**PASS**
09	Confirm 'overvalued' P/E indicator.	Configure data for 'overvalued' P/E; load component.	Displays "Overvalued" P/E label/indicator.	**PASS**
10	Confirm 'fair valuation' P/E indicator.	Configure data for 'fair valuation' P/E; load component.	Displays "Fair Valuation" P/E label/indicator.	**PASS**
11	Assess rendering with empty gainers/losers.	Configure API to return empty gainers/losers	Displays "No Data Found" message in relevant	**PASS**

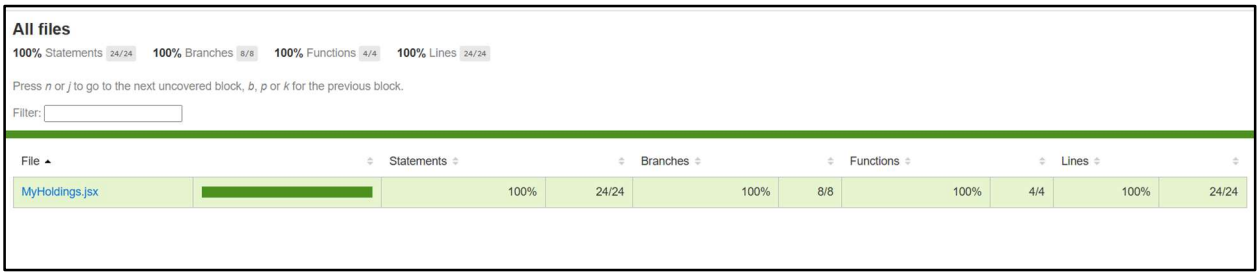
		lists.	modules.	

15.MyHoldings.jsx Testing report

Unit Testing Details:



Test Coverage Summary:



Detailed Test results:

01	Verify initial loading state.	Display component.	Loading indicator must show before data fetch.	**PASS**
02	Confirm correct table rendering on success.	Load with valid data.	Table renders accurately with retrieved data.	**PASS**
03	Assess graceful API error handling.	Return generic API error (e.g., HTTP 500).	Display friendly error message; prevent corrupt	**PASS**

			table.	
04	Validate stability on invalid response format.	Return unparsable data (e.g., malformed JSON).	Handle failure gracefully (e.g., "No Data Found"); no crash.	**PASS**
05	Verify empty holdings rendering.	Return empty holdings array.	Render correctly with "No Holdings Found" message.	**PASS**
06	Validate title always renders post-loading.	Load and await fetch completion (success/fail).	Title must be persistent regardless of data outcome.	**PASS**

16.MarketMovers.jsx Testing report

Unit Testing Details:

```
onsole/frontend/node_modules/@vitest/runner/dist/index.js:1526:3)
    at startTests (file:///C:/Users/vivek/OneDrive/Desktop/Main-branch/InsightStox---Portfolio-analyzer-tracker-and
-console/frontend/node_modules/@vitest/runner/dist/index.js:1559:3)

 ✓ test/unitTesting/MarketMovers.test.jsx (10 tests) 320ms
   ✓ MarketMovers Component — Full Coverage Suite (7)
     ✓ shows loading state initially 40ms
     ✓ renders full dataset correctly 85ms
     ✓ navigates correctly when clicking a stock item 31ms
     ✓ handles empty arrays gracefully 21ms
     ✓ MarketNewsItem anchors open correct link 16ms
     ✓ business group static cards render 27ms
     ✓ timeAgo utility covers minutes / hours / days branch 14ms
   ✓ handles API failure and still stops loading 36ms
   ✓ formats market news when backend returns valid news array 33ms
   ✓ skips news formatting when backend returns non-array news 14ms

 Test Files  1 passed (1)
      Tests  10 passed (10)
   Start at  11:03:22
   Duration  1.89s (transform 153ms, setup 147ms, collect 344ms, tests 320ms, environment 716ms, prepare 39ms)
```

Test Coverage Summary:

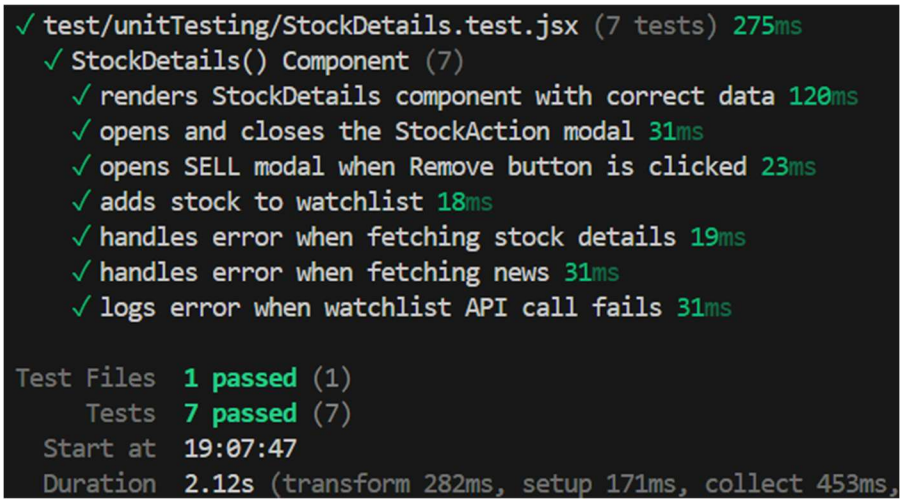
Detailed Test results:

01	Initial loading state check.	Load component; observe display pre-data.	Loading indicator/skeleton displays first.	**PASS**
02	Full dataset rendering validation.	Load component with valid, non-empty data.	Component renders all retrieved data points accurately.	**PASS**

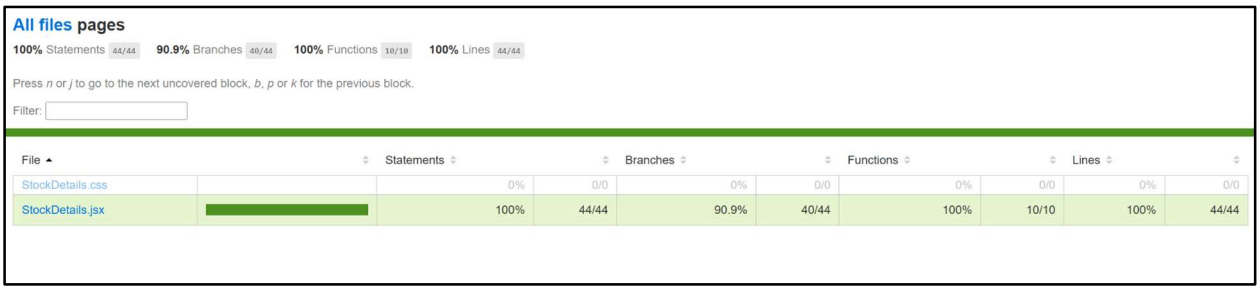
03	Stock item navigation check.	Click a stock/instrument item.	Navigates to the dedicated detail page.	**PASS**
04	Empty array graceful handling.	Configure API to return empty arrays. Load component.	Renders without error, showing empty state/message.	**PASS**
05	MarketNewsItem anchor link validation.	Inspect MarketNewsItem and check link's href.	Link's href matches expected news source URL.	**PASS**
06	Static business card rendering.	Load component; visually check static business area.	All static business group cards render correctly.	**PASS**
07	timeAgo utility branch test (min/hr/day).	Test timeAgo with inputs for minute, hour, and day differences.	Returns correctly formatted strings for minutes, hours, and days.	**PASS**
08	Loading state termination on API failure.	Configure API to fail (e.g., HTTP 500). Load component.	Error message shows; loading indicator stops.	**PASS**
09	Market news formatting with valid array.	Configure API to return valid news array. Load component.	News items display with correct formatting (e.g., date/title).	**PASS**
10	Market news robustness with non-array data.	Configure API to return non-array data (e.g., string/null) for news.	Skips formatting, prevents crash, ideally shows "No Data."	**PASS**

17. StockDetails.jsx Testing report

Unit Testing Details:



Test Coverage summary:



Detailed Test Results:

Test Case ID	Description	Test Steps / Action	Expected Result	Status
01	Renders StockDetails page with correct static components	Render component	Navbar, DashboardHeader, Footer are visible, "AAPL" appears on screen, Text equals +2.00(+1.35%)	**PASS**
02	Opens BUY modal when "Add" button clicked close when	Click "Add", Click "Close"	Modal displays "StockAction Component - BUY" and closes after	**PASS**

	“Close” is clicked		hitting close.	
03	Opens SELL modal when "Remove" clicked	Click "Remove"	Modal displays "StockAction Component - SELL"	**PASS**
04	Calls API when adding stock to watchlist	Calls API when adding stock to watchlist	axios.post("/api/v1/dashBoard/addToWatchlist", {symbol:"AAPL"}) is called	**PASS**
05	Logs error when stock details API fails	Make first axios.get reject	console.error("Error fetching stock details:", error) is called	**PASS**
06	Logs error when news API fails	First GET resolves, second rejects	console.error("Error fetching news:", error) is called	**PASS**
07	Logs error when watchlist API fails	Make axios.post reject	console.error("Error in adding the stock to watchlist:", error) is called	**PASS**

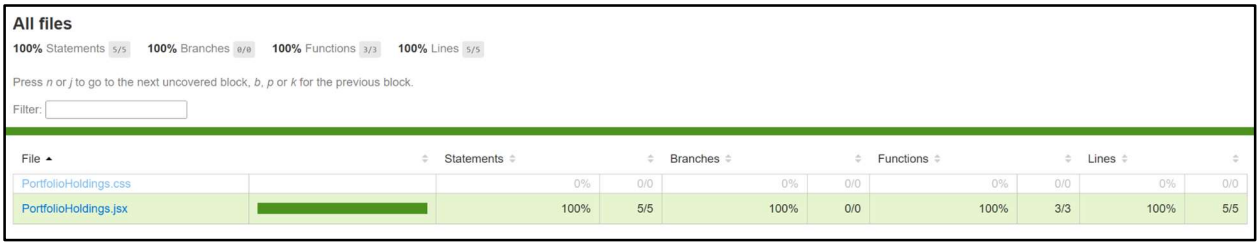
18. PortfolioHoldings.jsx Testing report

Unit Testing Details:

```
✓ test/unitTesting/PortfolioHoldings.test.jsx (5 tests) 248ms
  ✓ PortfolioHoldings Component (5)
    ✓ renders all table headers 47ms
    ✓ renders portfolio rows with correct values 13ms
    ✓ navigates to stockdetails page on clicking stock symbol 9ms
    ✓ renders empty tbody when portfolioHoldings is empty 146ms
    ✓ does not crash when portfolioHoldings is undefined 30ms

Test Files  1 passed (1)
     Tests  5 passed (5)
  Start at  22:24:58
  Duration  1.93s (transform 95ms, setup 199ms, collect 254ms, tests 248ms,
```

Test Coverage Summary:



Detailed Test Results:

Test Case ID	Description	Test Steps / Action	Expected Result	Status
01	Verifies that all table headers render correctly	Render component with mockData and query all header labels	All header texts (Stock, Status, Shares, Last Price, AC/Share, etc.) appear	**PASS**
02	Renders each portfolio row with correct values	Render with mockData and query each cell	All values for AAPL (shares, avg price, total cost, gain %, etc.) appear correctly	**PASS**
03	Clicking on stock symbol triggers navigation	Render and click the "AAPL" element	mockNavigate is called with "/stockdetails/AAPL"	**PASS**
04	Handles empty array	Render with	Only header row exists	**PASS**

	safely	portfolioHoldings={[]}	(row count = 1)	
05	Handles undefined portfolioSummary safely	Render without passing portfolioHoldings	Component does not crash; header row only (row count = 1)	**PASS**

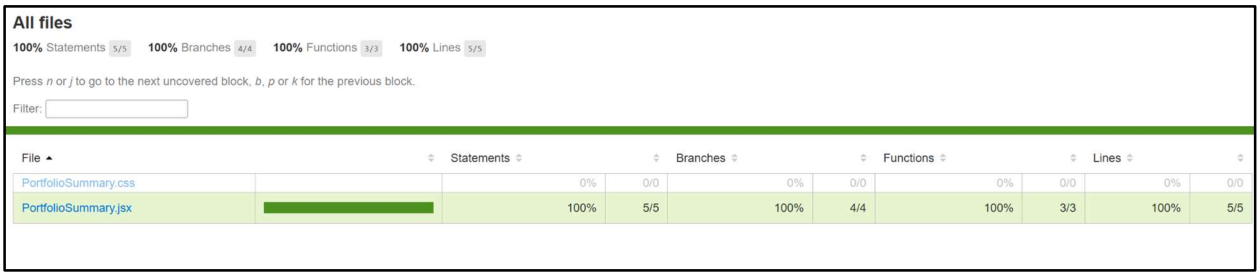
19. PortFolioSummary.jsx Testing report

Unit Testing Details:

```
✓ test/unitTesting/PortfolioSummary.test.jsx (5 tests) 263ms
  ✓ PortfolioSummary Component (5)
    ✓ renders all table headers 52ms
    ✓ renders portfolio rows with correct values 16ms
    ✓ navigates to stock details on clicking symbol 10ms
    ✓ renders only header row when portfolioSummary is empty 148ms
    ✓ does not crash when portfolioSummary is undefined 35ms

Test Files  1 passed (1)
     Tests  5 passed (5)
  Start at  22:35:26
  Duration  1.97s (transform 85ms, setup 198ms, collect 248ms, tests 263ms,
```

Test Coverage Summary:



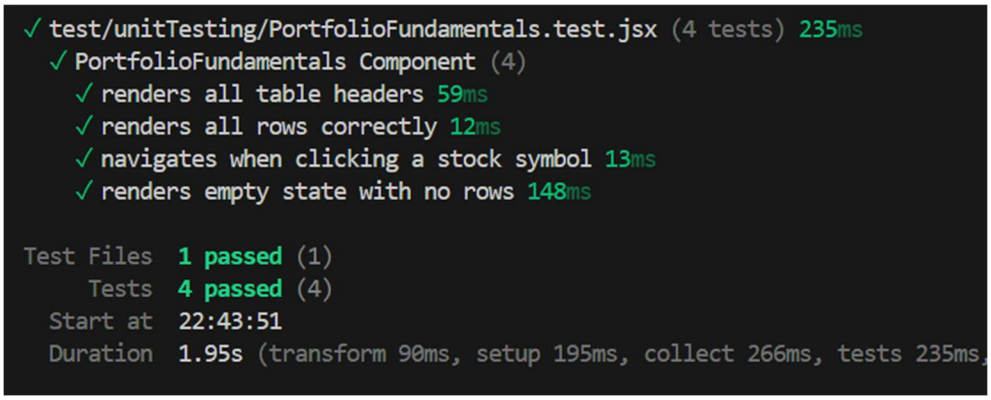
Detailed Test Results:

Test Case ID	Description	Test Steps / Action	Expected Result	Status
01	Verifies that all table headers render correctly	Render component with mockData and query all header labels	All header texts (Stock, Last Price, Change (%), etc.) appear in the document	**PASS**
02	Renders portfolio rows with correct values and positive/negative classes	Render with mockData and query each cell	Values appear correctly; positive values have .positive class, negative values have .negative class; range values have .range class	**PASS**

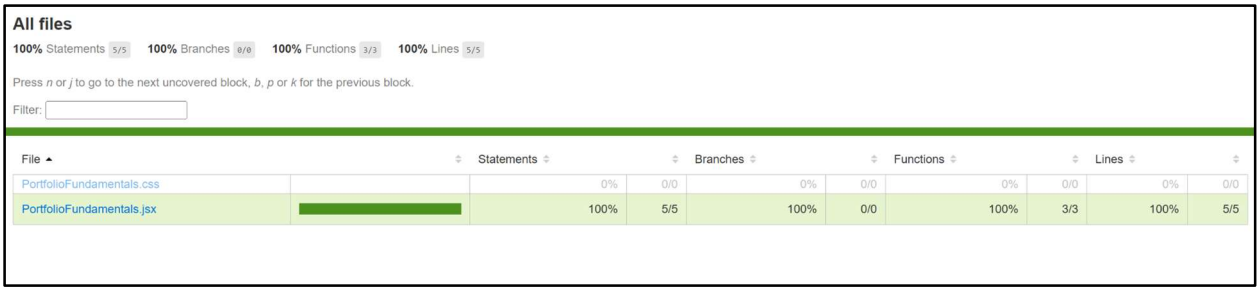
03	Clicking on stock symbol triggers navigation	Render and click the "AAPL" element	mockNavigate is called with "/stockdetails/AAPL"	**PASS**
04	Handles empty array gracefully	Render with portfolioSummary={[]}	Only the header row is rendered (row count = 1)	**PASS**
05	Handles undefined portfolioSummary safely	Render without passing portfolioSummary	No crash; only the header row appears (row count = 1)	**PASS**

20. PortfolioFundamentals.jsx Testing Report

Unit Testing Details:



Test Coverage Summary:



Detailed Test Results:

Test Case ID	Description	Test Steps / Action	Expected Result	Status
01	Ensures all table headers render correctly	Render component with mockData and query all header labels	All headers (Stock, Last Price, Market Cap, EPS Est. Next Yr, etc.) appear in the document	**PASS**
02	Renders all portfolio fundamentals rows with correct values	Render with mockData and query each text cell	AAPL + MSFT values render correctly (prices, market cap, dividend info, etc.)	**PASS**
03	Clicking on stock symbol triggers navigation	Render component and click on "AAPL" and "MSFT"	mockNavigate called with /stockdetails/AAPL and /stockdetails/MSFT	**PASS**

04	Handles empty array gracefully	Render with portfolioFundamentals ={[]}	Only header row is rendered (row count = 1)	**PASS**

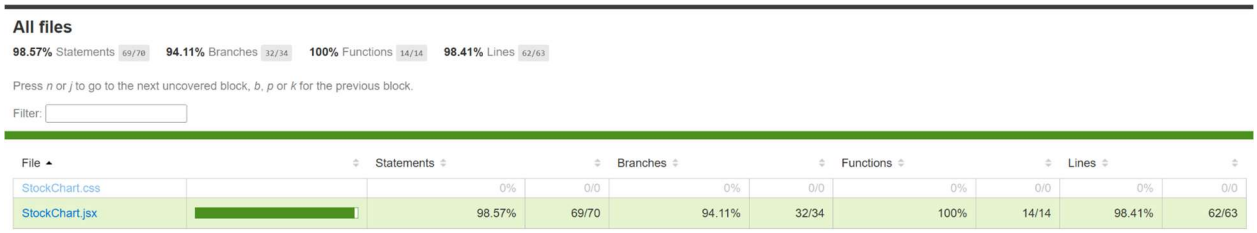
1. StockChart.jsx Testing Report

Unit Testing Details:

```
✓ test/unitTesting/StockChart.test.jsx (8 tests) 221ms
  ✓ StockChart component (8)
    ✓ shows loading while fetching data and then renders the chart 76ms
    ✓ renders error message when axios throws 13ms
    ✓ has range buttons and toggles active class 63ms
    ✓ sliceByRange returns entire arrays if not enough data (default fallback) 15ms
    ✓ unsupported range returns raw ISO labels 17ms
    ✓ tooltip callbacks produce expected title and label strings 12ms
    ✓ sliceByRange returns all points if data is shorter than requested range 18ms
    ✓ y-axis ticks callback formats numbers correctly 4ms

Test Files  1 passed (1)
     Tests  8 passed (8)
  Start at  18:53:07
  Duration  1.76s (transform 101ms, setup 170ms, collect 313ms, tests 221ms, environment 731ms, prepare 48ms)
```

Test Coverage Summary:



Detailed Test results:

Test Case ID	Description	Test Steps / Action	Expected Result	Status
01	Shows loading while fetching and then renders chart	Render component, resolve mocked axios with 10 points	Chart renders with 10 data points, loading message disappears	**PASS**
02	Renders error message when axios fails	Render component, mock axios rejection	Error message "F“iled to fetch stock chart data." ”isplayed, chart not rendered	**PASS**
03	Range buttons toggle	Click 1M, 3M buttons	Active class moves to	**PASS**

	active class and slice data	after rendering 400 points	clicked button, chart shows sliced data (30 for 1M, 90 for 3M)	
04	sliceByRange returns all points if data shorter than requested range	Render 10 points, click 1M button	Chart shows all 10 points, no error	**PASS**
05	Unsupported range returns raw ISO labels	Render component with range="X”Z" ”nd 3-point data	Chart labels remain raw ISO dates, data values unchanged	**PASS**
06	Tooltip callbacks produce correct title/label	Call tooltip callbacks on first data point	Title shows formatted date, label shows "P“ice: value"	**”ASS**
07	y-axis ticks format numbers	Call y-axis tick callback on 1000 and 2000	Returns "1“000" ”nd "2“000"	**”ASS**
08	Initial load renders correct chart with predefined data	Render component with initial data prop of 5 points	Chart renders 5 points correctly without axios request	**PASS**

22. MyProfile.jsx Testing Report

Unit Testing Details:

```
✓ test/unitTesting/MyProfile.test.jsx (20 tests) 400ms
  ✓ MyProfile component (20)
    ✓ renders user info correctly 75ms
    ✓ toggles name editing and validates input 62ms
    ✓ toggles password editing and validates input 61ms
    ✓ handles cancel password editing 28ms
    ✓ calls handleCancelInfo and closes name edit mode 19ms
    ✓ closes edit mode on Escape key without crashing 10ms
    ✓ toggles showPassword states 24ms
    ✓ triggers file input click when 'Change Photo' button is clicked 10ms
    ✓ calls handleSaveName when name is valid and Save changes is clicked 16ms
    ✓ calls handleSavePass when passwords are valid and Save changes is clicked 25ms
    ✓ calls handlePicChange when a file is selected in the hidden input 7ms
    ✓ handles OTP modal Cancel button click and resets state 3ms
    ✓ calls resendOtp handler when Resend button is clicked 4ms
    ✓ calls verifyOtpAndReset handler when Continue button is clicked 2ms
    ✓ calls handleInvExp for all Investment Experience options 7ms
    ✓ calls handleRiskProf for all Risk Profile options 7ms
    ✓ calls handleFinGoals for all Financial Goals options 9ms
    ✓ calls handleInvHorizon for all Investment Horizon options 8ms
    ✓ displays the linked accounts section when registrationMethod is 'google' 13ms
    ✓ does NOT display the linked accounts section when registrationMethod is 'normal' 7ms

Test Files  1 passed (1)
     Tests  20 passed (20)
  Start at  01:22:02
  Duration  2.03s (transform 238ms, setup 154ms, collect 466ms, tests 400ms, environment 697ms, prepare 40ms)
```

Test Coverage Summary:

Detailed Test Results:

Test Case ID	Description	Test Steps / Action	Expected Result	Status
01	Renders fieldname and value correctly	Render component with primaryData	Name, email, and UI elements appear correctly	**PASS**
02	toggles name editing and validates input	Click edit name → change input	Name edit mode activates and validation	**PASS**

			runs	
03	Toggles password editing and validates input	Click edit password → change inputs	Password edit mode activates and validation runs	**PASS**
04	handles cancel password editing	Click cancel button in password section	Password edit mode closes without saving	**PASS**
05	calls handleCancelInfo and closes name edit mode	Click cancel in name edit section	Name edit UI closes and handler is called	**PASS**
06	closes edit mode on Escape key without crashing	Press Escape key	Edit mode closes safely without errors	**PASS**
07	toggles showPassword states	Click show/hide icons	Password fields switch between text/password	**PASS**
08	triggers file input click when 'Change Photo' is clicked	Click Change Photo button	Hidden file input is triggered for image selection	**PASS**
09	calls handleSaveName when name is valid and Save is clicked	Fill valid name → click Save	handleSaveName executes successfully	**PASS**
10	calls handleSavePass when passwords are valid and Save is clicked	Enter valid passwords → click Save	handleSavePass executes successfully	**PASS**
11	calls handlePicChange when a file is selected	Select a mock image file	handlePicChange runs with selected file	**PASS**
12	handles OTP modal Cancel button click and resets state	Click Cancel in OTP modal	Modal closes and OTP state resets	**PASS**
13	calls resendOtp handler when Resend button is clicked	Click Resend button	resendOtp handler is triggered	**PASS**
14	calls verifyOtpAndReset when Continue button is clicked	Enter OTP → click Continue	verifyOtpAndReset is called	**PASS**
15	calls handleInvExp for all Investment Experience	Select any Exexperienception	handler is called for an option	**PASS**

	options			
16	Calls handleRiskProf for all Risk Profile options	Select any Risk option	handler is called for an option	**PASS**
17	calls handleInvHorizon for all Investment Horizon options	Select any Horizon option	handler runs for an option	**PASS**
18	calls handleFinGoals for all Financial Goals options	Select any Goal option	handler runs for an options	**PASS**
19	displays linked accounts section when registrationMethod is ‘google’	Render with registrationMethod=google	Linked accounts section appears	**PASS**
20	does NOT display linked accounts when registrationMethod is ‘normal’	Render with registrationMethod=normal	Linked accounts section stays hidden	**PASS**

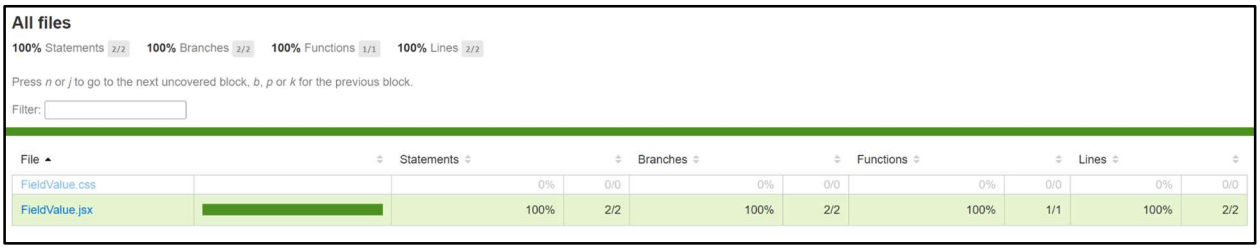
23. FieldValue.jsx Testing Report

Unit Testing Details:

```
✓ test/unitTesting/FieldValue.test.jsx (4 tests) 40ms
  ✓ FieldValue Component (4)
    ✓ renders fieldname and value correctly 30ms
    ✓ applies custom classname when provided 3ms
    ✓ does not add 'undefined' classname when no classname is provided 2ms
    ✓ renders with empty value 3ms

Test Files  1 passed (1)
     Tests  4 passed (4)
  Start at  01:27:21
  Duration  1.36s (transform 67ms, setup 151ms, collect 187ms, tests 40ms, environment
```

Test Coverage Summary:



Detailed Test Results:

Test Case ID	Description	Test Steps / Action	Expected Result	Status
01	Renders fieldname and value correctly	Render <FieldValue fieldname="P” Ratio" ”alue="2”.3" ”>	"P“ Ratio" ”nd "2“.3" ”ppear in the document	**PASS**
02	Applies custom classname when provided	Render with classname="e”tra-class" ”nd query wrapper element	Wrapper has classes "f“eld-value" ”nd "e“tra-class"	**”ASS**
03	Does NOT add unwanted "u“defined" ”r "n“ll" ”lass when classname not provided	Render without classname and get wrapper element	Wrapper contains "f“eld-value" ”nly; does NOT contain "u“defined" ”r "n“ll"	**”ASS**
04	Renders gracefully with	Render <FieldValue	Fieldname displays;	**PASS**

	an empty value	fieldname="E”S" ”alue=""”> and query value span	value span exists but is empty (no crash, no missing element)	

24. StockAction.jsx Testing Report

Unit Testing Details:

```
✓ test/unitTesting/StockAction.test.jsx (17 tests) 406ms
  ✓ StockAction Component (17)
    ✓ renders correctly for BUY action with positive change 75ms
    ✓ renders correctly for SELL action with negative change 14ms
    ✓ displays zero change correctly 6ms
    ✓ calls handler with "SELL" when "Rmv" is clicked in BUY mode 10ms
    ✓ calls handler with "BUY" when "Add" is clicked in SELL mode 5ms
    ✓ resets quantity on toggleModel call 11ms
    ✓ updates quantity state when input changes 5ms
    ✓ increments quantity when up arrow button is clicked 20ms
    ✓ decrements quantity when down arrow button is clicked, min 0 14ms
    ✓ prevents non-numeric input keys ("e", "E", "+", "-") 7ms
    ✓ calls onClose when Escape key is pressed 4ms
    ✓ shows alert and prevents submission if quantity is empty 12ms
    ✓ shows alert and prevents submission if quantity is not an integer 11ms
    ✓ successfully submits a BUY transaction and calls onClose 18ms
    ✓ successfully submits a SELL transaction and calls onClose 16ms
    ✓ handles server validation error response (err.response.data.message) 80ms
    ✓ handles generic network/axios error response (err.message) 94ms

Test Files  1 passed (1)
     Tests  17 passed (17)
  Start at  02:30:25
  Duration  2.00s (transform 123ms, setup 160ms, collect 347ms, tests 406ms, environment 734ms,
```

Test Coverage Summary:

Detailed Test Results:

Test Case ID	Description	Test Steps / Action	Expected Result	Status
01	Renders correctly for BUY action with positive change	Render component with action="B”Y" ”nd positive price movement	UI shows BUY state, positive color, correct price details	**PASS**

02	Renders correctly for SELL action with negative change	Render component with action="S”LL" ”nd negative price movement	UI shows SELL state, negative color, correct price details	**PASS**
03	Displays zero change correctly	Render with 0% price movement	Shows neutral styling without positive/negative colors	**PASS**
04	Calls handler with "S“LL" ”hen Rmv clicked in BUY mode	Click "R“v" ”utton	handler called with "S“LL"	**”ASS**
05	Calls handler with "B“Y" ”hen Add clicked in SELL mode	Click "A“d" ”utton	handler called with "B“Y"	**”ASS**
06	Resets quantity on toggleModel call	Call toggleModel() programmatically	Quantity resets to default (usually 0 or empty)	**PASS**
07	Updates quantity state when input changes	Enter a value into quantity input	State updates and total amount recalculates	**PASS**
08	Increments quantity when up arrow clicked	Click increment/up button	Quantity increases by 1	**PASS**
09	Decrements quantity when down arrow clicked without going below 0	Click decrement/down button repeatedly	Quantity decreases but never becomes negative	**PASS**
10	Prevents non-numeric input keys	Type ‘e’, ‘E’, ‘+’, ‘-’ in number input	Input rejects these characters	**PASS**
11	Calls onClose when Escape key pressed	Press Escape key while modal is open	onClose callback fires	**PASS**
12	Shows alert and prevents submission if quantity empty	Click Confirm with empty quantity	Alert shown and submit API not called	**PASS**
13	Shows alert and prevents submission if	Enter decimal quantity and click Confirm	Alert shown and API not called	**PASS**

	quantity not integer			
14	Successfully submits BUY transaction	Fill valid data and click Confirm in BUY mode	API called, success alert shown, modal closes	**PASS**
15	Successfully submits SELL transaction	Fill valid data and click Confirm in SELL mode	API called, success alert shown, modal closes	**PASS**
16	Handles server validation error from err.response.data. message	Mock API failure with server message; click Confirm	Shows server message alert and error logged	**PASS**
17	Handles generic network/axios error	Mock network failure (err.message); click Confirm	Shows generic error alert and logs error	**PASS**

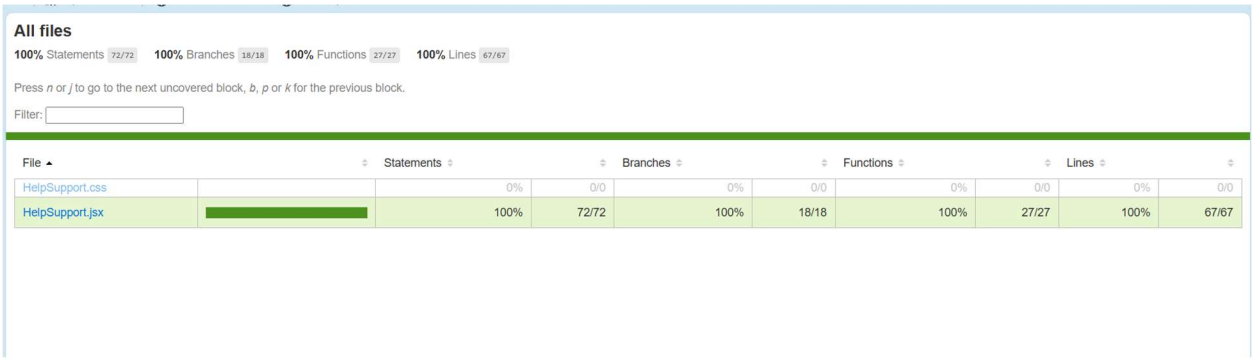
25. HelpSupport.jsx Testing Report

Unit Testing Report:

```
✓ test/unitTesting/HelpSupport.test.jsx (18 tests) 614ms
  ✓ HelpSupport Page Coverage Tests (18)
    ✓ 1. Renders complete page structure (Navbar, Sidebar, Footer, Sections) 64ms
    ✓ 2. Opens modal, renders specific content, and handles internal navigation 36ms
    ✓ 3. Closes modal when clicking the overlay 20ms
    ✓ 4. Does NOT close modal when clicking inside the content card (stopPropagation) 16ms
    ✓ 5. Closes modal via Escape key (useEffect listener) 18ms
    ✓ 6. Contact Form: Validation prevents empty submission 11ms
    ✓ 7. Contact Form: Successful submission clears input 20ms
    ✓ 8. Contact Form: API returns 200 but logical failure (success: false) 78ms
    ✓ 9. Contact Form: Handles Network Error (Catch Block) 78ms
    ✓ 10. Feedback Form: Validation prevents empty submission 9ms
    ✓ 11. Feedback Form: Successful submission 21ms
    ✓ 12. Feedback Form: API returns 200 but logical failure (success: false) 79ms
    ✓ 13. Feedback Form: Handles Network Error 76ms
    ✓ 14. Renders Troubleshooting specific hidden button correctly 43ms
    ✓ 15. HelpTopicModal returns null for invalid topic 2ms
    ✓ 16. AI Suggestions modal button navigates to /ai-insight 14ms
    ✓ 17. Build Portfolio modal button navigates to /portfolio 14ms
    ✓ 18. Non-Escape key does not close modal (covers else branch) 13ms

 Test Files  1 passed (1)
      Tests  18 passed (18)
   Start at  18:09:52
   Duration  1.92s (transform 199ms, setup 111ms, collect 394ms, tests 614ms, environment 475ms, prepare 42
ms)
```

Test Coverage Summary:



Detailed Test results:

Test Case ID	Description	Test Steps / Action	Expected Result	Status
01	Page loads with all main sections	Open the Help & Support page	Navbar, Sidebar, Footer and all section titles are visible	**PASS**
02	Modal opens and navigation works	Click **Getting Started Guide** and click **Explore My Dashboard**	Modal opens and user is redirected to /dashboard	**PASS**
03	Modal closes when clicking outside	Open a modal and click on background overlay	Modal should close	**PASS**
04	Modal stays open when clicking inside	Open modal and click inside the content box	Modal should remain open	**PASS**
05	Modal closes with Escape key	Open modal and press **Escape** key	Modal should close	**PASS**
06	Prevent empty contact form submission	Click **Submit Ticket** without typing	Alert shows “Please enter a message” and API is not called	**PASS**

07	Contact form submits successfully	Enter message and click **Submit Ticket**	API is called, success alert appears and textbox clears	**PASS**
08	Contact form handles backend failure	Submit form but backend returns success:false	Error alert appears	**PASS**
09	Contact form handles network error	Submit form and simulate network failure	Error alert shows about network issue	**PASS**
10	Prevent empty feedback submission	Click **Send feedback** without typing	Alert shows “Please enter a feedback”	**PASS**
11	Feedback submits successfully	Enter feedback and click **Send feedback**	Success alert appears	**PASS**
12	Feedback logical failure handled	Submit feedback with backend failure	Error alert appears	**PASS**
13	Feedback network error handled	Simulate network error while sending feedback	Proper error alert appears	**PASS**

14	Hidden troubleshooting button shows	Click **Troubleshooting common issues**	Hidden button appears (but stays hidden visually)	**PASS**
15	Invalid help topic handled	Pass an invalid topic name to HelpTopicModal	Modal does not render (returns null / empty output)	**PASS**
16	AI Suggestions modal navigation	Click **Understanding AI Suggestions** Click **View My AI Insights** button	User is redirected to /ai-Insight	**PASS**
17	Build Portfolio modal navigation	Click **How to Build Portfolio** Click **Go to Portfolio** button	User is redirected to /portfolio	**PASS**
18	Modal does not close for non-Escape key	Open **Getting Started Guide** modal Press **Enter** key	Modal remains open	**PASS**

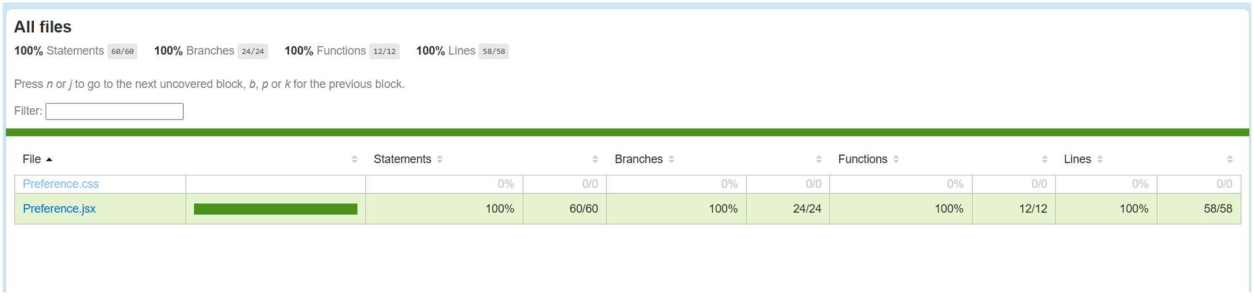
26. Preference.jsx Testing Report

Unit Testing Report:

```
✓ test/unitTesting/Preference.test.jsx (12 tests) 325ms
  ✓ Preference Page – Optimized Coverage Tests (12)
    ✓ 1. Renders full layout 54ms
    ✓ 2. Covers successful GET fetch 9ms
    ✓ 3. Covers GET failure (catch block) 26ms
    ✓ 4. Covers Theme PATCH success branch 31ms
    ✓ 5. Covers Layout PATCH success branch 16ms
    ✓ 6. Covers PATCH catch block 15ms
    ✓ 7. Covers GET success false branch 32ms
    ✓ 8. Covers ensureAuth interval catch block 10ms
    ✓ 9. Covers layout update failed (success: false branch) 21ms
    ✓ 10. Covers theme update failed (success: false branch) 16ms
    ✓ 11. Covers default fallback values for theme and layout 7ms
    ✓ 12. Covers initial ensureAuth failure (catch block) 86ms

 Test Files  1 passed (1)
      Tests  12 passed (12)
   Start at  18:06:43
   Duration  1.61s (transform 165ms, setup 110ms, collect 380ms, tests 325ms, environment 485ms, prepare 57
ms)
```

Test Coverage Summary:



Detailed Test results:

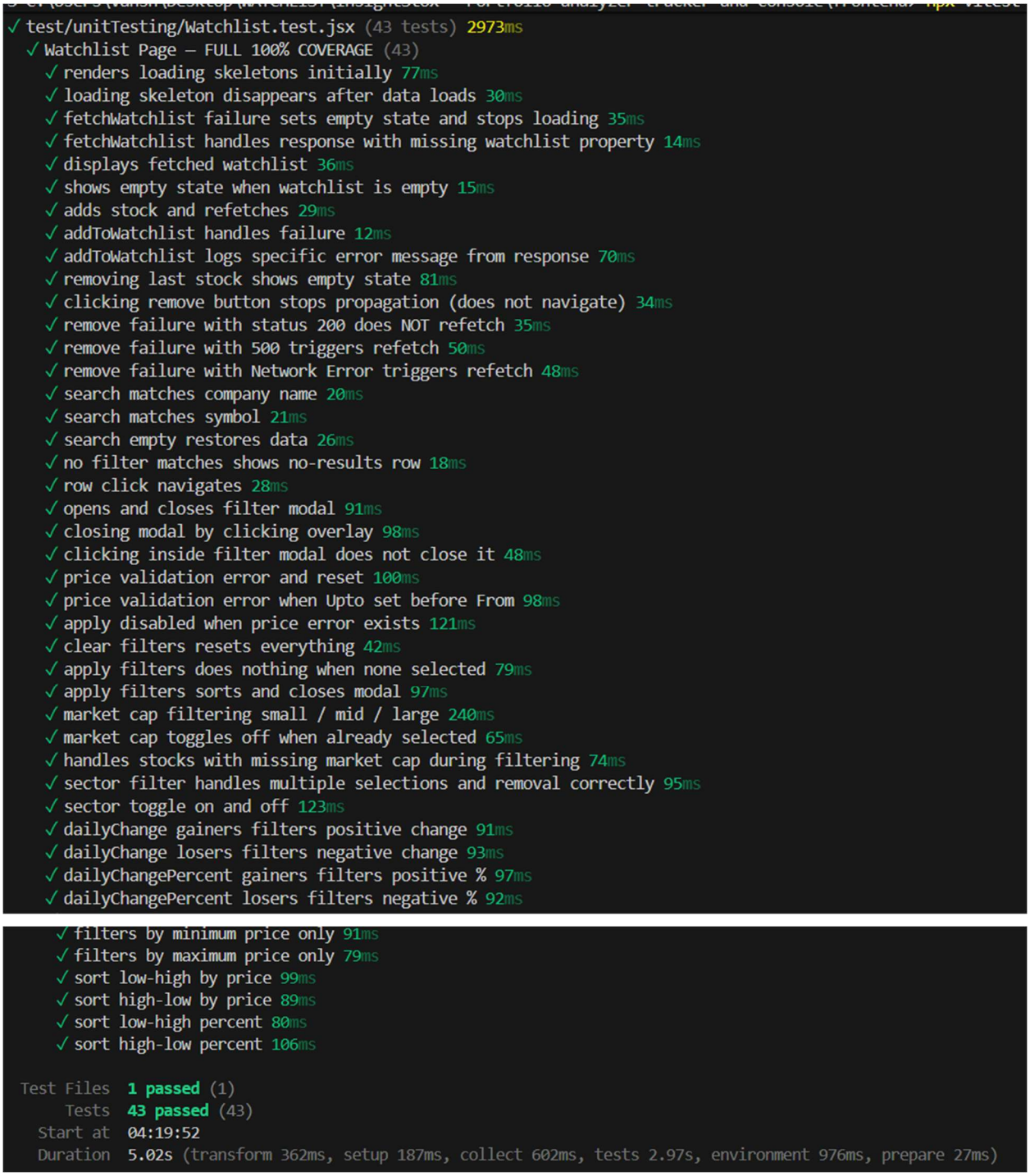
Test Case ID	Description	Test Steps / Action	Expected Result	Status

01	Page loads correctly	Open the Preferences page	“Preferences & Personalisation” heading is visible	**PASS**
02	Preferences are fetched from backend	Load the page normally	API GET request is called to fetch saved preferences	**PASS**
03	Shows error when API fails to load data	Simulate network error while loading page	Alert message shows “Failed to load preferences...”	**PASS**
04	Theme change updates successfully	Change theme from Dark to Light	API PATCH is called and success alert is shown	**PASS**
05	Layout change updates successfully	Change dashboard layout option	API PATCH is called and success alert is shown	**PASS**
06	Shows error when saving preferences fails	Simulate PATCH API failure	Alert shows “Something went wrong...”	**PASS**

07	Handles backend success=false response	Simulate API returning success:false	Alert shows “Could not fetch your preferences.”	**PASS**
08	Handles session validation failure during background auth check	Simulate ensureAuth() failing inside setInterval	Error is logged using console.error()	**PASS**
09	Layout update fails when backend returns success:false	Change dashboard layout and mock API to return Success: False	Alert shows **“Failed to update layout.”**	**PASS**
10	Theme update fails when backend returns success:false	Change theme and mock API to return Success: False	Alert shows **“Failed to update theme.”**	**PASS**
11	Handles null preference values with default fallback	Mock API response with theme: Null and Dashboard: Null	Default values are applied without crashing	**PASS**
12	Initial authentication check fails on page load	**Steps / Action:** Force ensureAuth() to reject on component mount	Error logged as **“ensureAuth initial check failed”**	**PASS**

26. WatchList.jsx Testing report

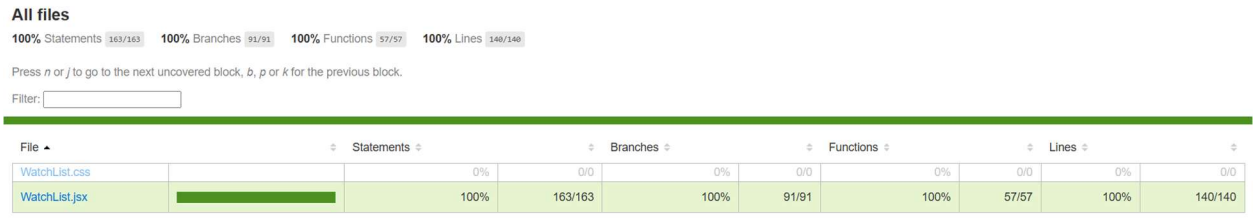
Unit Testing Report:



```
✓ test/unitTesting/Watchlist.test.jsx (43 tests) 2973ms
  ✓ Watchlist Page — FULL 100% COVERAGE (43)
    ✓ renders loading skeletons initially 77ms
    ✓ loading skeleton disappears after data loads 30ms
    ✓ fetchWatchlist failure sets empty state and stops loading 35ms
    ✓ fetchWatchlist handles response with missing watchlist property 14ms
    ✓ displays fetched watchlist 36ms
    ✓ shows empty state when watchlist is empty 15ms
    ✓ adds stock and refetches 29ms
    ✓ addToWatchlist handles failure 12ms
    ✓ addToWatchlist logs specific error message from response 70ms
    ✓ removing last stock shows empty state 81ms
    ✓ clicking remove button stops propagation (does not navigate) 34ms
    ✓ remove failure with status 200 does NOT refetch 35ms
    ✓ remove failure with 500 triggers refetch 50ms
    ✓ remove failure with Network Error triggers refetch 48ms
    ✓ search matches company name 20ms
    ✓ search matches symbol 21ms
    ✓ search empty restores data 26ms
    ✓ no filter matches shows no-results row 18ms
    ✓ row click navigates 28ms
    ✓ opens and closes filter modal 91ms
    ✓ closing modal by clicking overlay 98ms
    ✓ clicking inside filter modal does not close it 48ms
    ✓ price validation error and reset 100ms
    ✓ price validation error when Upto set before From 98ms
    ✓ apply disabled when price error exists 121ms
    ✓ clear filters resets everything 42ms
    ✓ apply filters does nothing when none selected 79ms
    ✓ apply filters sorts and closes modal 97ms
    ✓ market cap filtering small / mid / large 240ms
    ✓ market cap toggles off when already selected 65ms
    ✓ handles stocks with missing market cap during filtering 74ms
    ✓ sector filter handles multiple selections and removal correctly 95ms
    ✓ sector toggle on and off 123ms
    ✓ dailyChange gainers filters positive change 91ms
    ✓ dailyChange losers filters negative change 93ms
    ✓ dailyChangePercent gainers filters positive % 97ms
    ✓ dailyChangePercent losers filters negative % 92ms
    ✓ filters by minimum price only 91ms
    ✓ filters by maximum price only 79ms
    ✓ sort low-high by price 99ms
    ✓ sort high-low by price 89ms
    ✓ sort low-high percent 80ms
    ✓ sort high-low percent 106ms

 Test Files  1 passed (1)
      Tests  43 passed (43)
   Start at  04:19:52
   Duration  5.02s (transform 362ms, setup 187ms, collect 602ms, tests 2.97s, environment 976ms, prepare 27ms)
```

Test Coverage Summary:



Detailed Test results:

Test Case ID	Description	Test Steps / Action	Expected Result	Status
01	Initial Loading State	Render component and check for skeletons before data loads.	Skeletons are present in the DOM.	**PASS**
02	Loading State Disappears	Wait for data fetch to complete.	Skeletons are removed from the DOM.	**PASS**
03	Fetch Failure Handling	Mock a rejected API call for fetchWatchlist.	Displays "Nothing in this watchlist yet" and stops loading.	**PASS**
04	Missing Watchlist Prop	Mock API response with	Handles missing data gracefully without crashing.	**PASS**

		missing watchlist key.		
05	Display Watchlist Data	Mock API with valid stock data.	Displays stock names (e.g., Apple, Tesla) correctly.	**PASS**
06	Empty Watchlist State	Mock API with empty watchlist array [].	Displays empty state message.	**PASS**
07	Add Stock Success	Click "Add Stock" and mock success response.	API is called and list refetches.	**PASS**
08	Add Stock Failure	Click "Add Stock" and mock 500 error.	API called, error logged, no crash.	**PASS**
09	Add Stock Log Error	Mock specific error message from backend.	Console logs the specific error message.	**PASS**
10	Remove Last Stock	Remove the only remaining stock.	State updates to show empty watchlist message.	**PASS**

11	Stop Propagation on Remove	Click "Remove" button.	Does NOT trigger row click navigation.	**PASS**
12	Remove Failure (200 status)	Mock remove failure with status 200 (logical error).	Does NOT refetch the list.	**PASS**
13	Remove Failure (500 status)	Mock remove failure with status 500.	Triggers a refetch to restore data.	**PASS**
14	Remove Failure (Network)	Mock network error on remove.	Triggers a refetch to restore data.	**PASS**
15	Search by Company Name	Search for "Tesla".	Shows Tesla, hides Apple.	**PASS**
16	Search by Symbol	Search for "AAPL".	Shows Apple Inc.	**PASS**
17	Clear Search	Filter list then clear search input.	Full list is restored.	**PASS**

18	No Search Results	Search for non-existent string "xyz".	Displays "No stocks matched" row.	**PASS**
19	Row Navigation	Click on a stock row.	Navigates to `/stockdetails/:symbol`.	**PASS**
20	Open/Close Filter Modal	Click filter button, then click close button.	Modal opens and then closes.	**PASS**
21	Close Modal via Overlay	Click outside the modal (overlay).	Modal closes.	**PASS**
22	Modal Content Click	Click inside the modal content.	Modal remains open (propagation stopped).	**PASS**
23	Price Error Reset	Enter invalid price range, then fix it.	Error message appears, then disappears.	**PASS**
24	Price Error (Upto < From)	Set "Upto" lower than "From".	Displays "From cannot be greater than Upto".	**PASS**
25	Apply Button Disabled	Create a price error condition.	Apply button becomes disabled.	**PASS**

26	Clear All Filters	Click "Clear All" in modal.	Resets all filters and shows full list.	**PASS**
27	Apply Empty Filters	Click "Apply" without selecting options.	Modal closes, no changes to list.	**PASS**
28	Apply Sort & Close	Select sort option and click Apply.	List sorts and modal closes.	**PASS**
29	Market Cap Filtering	Select Mid, Small, and Large cap filters.	List filters correctly for each category.	**PASS**
30	Market Cap Toggle	Click selected market cap again.	Filter is deselected.	**PASS**
31	Null Market Cap	Filter list with stocks having null market cap.	Handles null values without crashing.	**PASS**
32	Sector Filter Logic	Select multiple sectors, remove one.	Updates active filters correctly.	**PASS**
33	Sector Toggle	Click active sector filter.	Filter is removed.	**PASS**

34	Daily Gainers Filter	Select "Gainers" (Daily Change).	Shows only stocks with positive change.	**PASS**
35	Daily Losers Filter	Select "Losers" (Daily Change).	Shows only stocks with negative change.	**PASS**
36	Percent Gainers Filter	Select "Gainers" (%).	Shows only stocks with positive %.	**PASS**
37	Percent Losers Filter	Select "Losers" (%).	Shows only stocks with negative %.	**PASS**
38	Min Price Filter	Set "From" price.	Shows stocks above minimum price.	**PASS**
39	Max Price Filter	Set "Upto" price.	Shows stocks below maximum price.	**PASS**
40	Sort Price Low-High	Select Price Low-High.	List sorts ascending by price.	**PASS**
41	Sort Price High-Low	Select Price High-Low.	List sorts descending by price.	**PASS**
42	Sort Percent Low-High	Select Percent Low-High.	List sorts ascending by percentage.	**PASS**

43	Sort Percent High-Low	Select Percent High-Low.	List sorts descending by percentage.	**PASS**

27. Dashboard-Header.jsx testing report

Unit Testing Report:

```
✓ test/unitTesting/Dashboard-Header.test.jsx (24 tests) 3926ms
  ✓ DashboardHeader Component — FULL COVERAGE (24)
    ✓ renders loading skeletons initially 57ms
    ✓ fetches fresh data when cache is empty 72ms
    ✓ uses cached data if timestamp is recent (< 5 mins) 15ms
    ✓ fetches fresh data if cache is expired (> 5 mins) 21ms
    ✓ renders stocks with correct positive/negative styling 46ms
    ✓ handles N/A values and missing names gracefully 25ms
    ✓ displays 'No active stock data available' when data array is empty 15ms
    ✓ navigates to details page when clicking a stock ticker 26ms
    ✓ does not navigate or change cursor when stock symbol is missing 59ms
    ✓ handles 401 Session Expired error 24ms
    ✓ handles generic API fetch failure 31ms
    ✓ handles invalid data format (non-array response) 15ms
    ✓ activates search popup on input focus 8ms
    ✓ closes search popup when overlay is clicked 13ms
    ✓ closes search popup on Escape key press 10ms
    ✓ does not close search popup on non-Escape key press 10ms
    ✓ debounces search input and fetches suggestions  400ms
    ✓ handles empty search result (No matches found)  392ms
    ✓ clears results when input is empty  738ms
    ✓ navigates when clicking a search result  392ms
    ✓ handles search API error gracefully  390ms
    ✓ handles invalid search response format  392ms
    ✓ renders 'Add to Watchlist' button when props are provided  392ms
    ✓ does NOT render 'Add to Watchlist' button if not on watchlist page  379ms

Test Files  1 passed (1)
     Tests  24 passed (24)
  Start at  04:56:46
  Duration  6.19s (transform 237ms, setup 214ms, collect 545ms, tests 3.93s, environment 1.20s, prepare 27ms)
```

Test Coverage Summary:

Detailed Test results:

Test Case ID	Description	Test Steps / Action	Expected Result	Status
01	Initial Loading State	Render component with a pending data request.	Loading skeletons are displayed while data is being fetched.	**PASS**
02	Fresh Data Fetch (Empty Cache)	Render component when context cache is empty.	API is called to fetch fresh stock data.	**PASS**
03	Cached Data Usage	Render component with valid, recent cached data (< 5 mins).	API call is skipped; data is displayed directly from the cache.	**PASS**
04	Expired Cache Refresh	Render component with old cached data (> 5 mins).	API is called again to refresh the stale data.	**PASS**

05	Stock Styling (Positive/Negative)	Mock stock data with positive and negative price changes.	Positive stocks are styled green; negative stocks are styled red.	**PASS**
06	Data Fallback Handling	Mock stock data with missing names or prices.	Displays "N/A" for missing fields instead of crashing or showing blank space.	**PASS**
07	Empty Data State	Mock API returning an empty data array `[]`.	Displays "No active stock data available" message.	**PASS**
08	Ticker Navigation	Click on a valid stock ticker in the header.	Navigates correctly to the `/stockdetails` page for that symbol.	**PASS**
09	Missing Symbol Handling	Click on a stock item that has no symbol (e.g., invalid data).	Click is ignored; no navigation occurs. Cursor style remains default.	**PASS**

10	Session Expiry Handling	Mock a 401 Unauthorized response from the API.	Displays "Session expired. Please log in again." error message.	**PASS**
11	Generic API Failure	Mock a standard Network Error (500).	Displays "Failed to load market data." error message.	**PASS**
12	Invalid Data Format	Mock API response with non-array data.	Displays "Invalid data format from server." error message.	**PASS**
13	Search Focus Interaction	Click/Focus on the main search input field.	Search popup/overlay opens and becomes active.	**PASS**
14	Close Search via Overlay	Click the background overlay while search is open.	Search popup closes.	**PASS**
15	Close Search via Escape	Press the "Escape" key while search is open.	Search popup closes.	**PASS**

16	Ignore Other Keys	Press keys other than "Escape" (e.g., "Enter").	Search popup remains open.	**PASS**
17	Search Debounce Logic	Type in search bar. Wait 350ms.	API call is delayed until typing stops (debounced), then suggestions are fetched.	**PASS**
18	Empty Search Results	Type a query that returns no matches.	Displays "No matching stocks found." message.	**PASS**
19	Clear Search Input	Clear text from the search bar.	Search results list is cleared immediately.	**PASS**
20	Search Result Navigation	Click on a stock suggestion in the search results.	Navigates to the details page for the selected stock.	**PASS**
21	Search API Error	Mock a failure response for the search API.	Error is logged safely; application does not crash.	**PASS**

22	Invalid Search Response	Mock search API returning malformed data.	Handles invalid format gracefully without breaking the UI.	**PASS**
23	Watchlist Add Button	Render with `isWatchlistPage=true` and valid handler.	"Add to Watchlist" button is visible and functional next to search results.	**PASS**
24	Hidden Watchlist Button	Render with `isWatchlistPage=false` (default).	"Add to Watchlist" button is NOT rendered.	**PASS**

28. DataPrivacy.jsx Testing Report

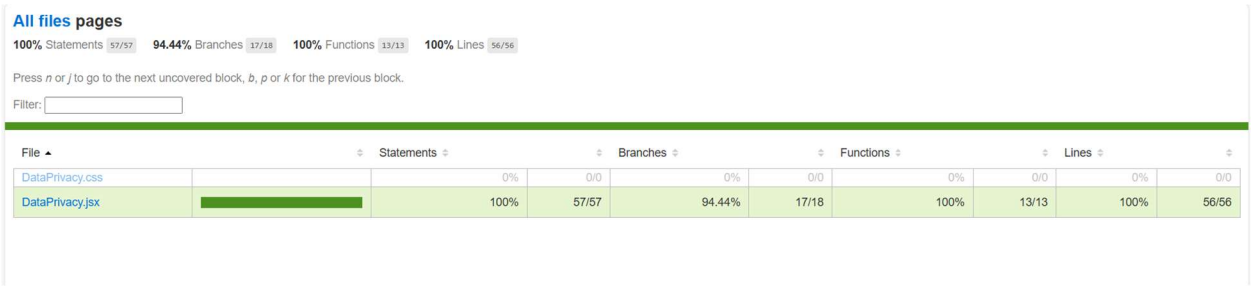
Unit Testing Report:

```
✓ test/unitTesting/DataPrivacy.test.jsx (25 tests) 891ms
  ✓ DataPrivacy Component Tests (25)
    ✓ Happy Paths (8)
      ✓ renders DataPrivacy component with correct initial AI state 102ms
      ✓ toggles AI suggestion switch and calls patch API 39ms
      ✓ opens and closes Privacy Policy modal 98ms
      ✓ opens and closes Terms and Conditions modal 60ms
      ✓ closes modal using ESC key 32ms
      ✓ initiates data download 23ms
      ✓ initiates account deletion with confirmation 25ms
      ✓ cancels account deletion when user declines confirmation 12ms
    ✓ Edge Cases (14)
      ✓ reverts AI toggle on patch failure 35ms
      ✓ resets download button on failed download 31ms
      ✓ resets delete button on failed account deletion 32ms
      ✓ handles error when fetching data privacy settings fails 12ms
      ✓ sets AI toggle to false when API returns false 20ms
      ✓ handles missing user details gracefully 12ms
      ✓ shows loading state during download request 30ms
      ✓ handles empty API response for user data 22ms
      ✓ switches between different modals correctly 46ms
      ✓ handles missing backend URL gracefully 11ms
      ✓ closes modal when clicking on backdrop/overlay 30ms
      ✓ handles undefined aisuggestion in API response 23ms
      ✓ maintains toggle state during API call 30ms
      ✓ handles user with partial details 14ms
    ✓ Comprehensive Coverage (3)
      ✓ covers all modal interaction scenarios 78ms
      ✓ covers all API error scenarios 45ms
      ✓ covers all user interaction states 24ms

 Test Files  1 passed (1)
      Tests  25 passed (25)
   Start at  21:02:07
   Duration  3.92s (transform 404ms, setup 303ms, import 813ms, tests 891ms, environment 1.52s)

 PASS  Waiting for file changes...
       press h to show help, press q to quit
```

Test Coverage Summary:



Detailed Test Report:

Test Case ID	Description	Test Steps / Action	Expected Result	Status
01	Initial Render State	Render DataPrivacy component with default AI toggle state.	Component loads with correct initial AI state.	**PASS**
02	AI Suggestion Toggle Patch	Toggle AI suggestion switch.	Sends PATCH API call and updates UI.	**PASS**
03	Open Privacy Policy Modal	Click “Privacy Policy”.	Privacy Policy modal opens.	**PASS**
04	Open Terms & Conditions Modal	Click “Terms and Conditions”.	T&C modal opens.	**PASS**
05	Close Modal via ESC	Press ESC key while modal open.	Modal closes.	**PASS**
06	Start Data Download	Click “Download Data”.	Data download begins.	**PASS**
07	Confirm Account Deletion	Confirm deletion when dialog appears.	Account deletion API triggered.	**PASS**
08	Cancel Account Deletion	Decline deletion in confirmation modal.	Deletion cancelled with no action.	**PASS**
09	AI Toggle Reverts on Failure	API patch fails.	Toggle returns to previous state.	**PASS**

10	Reset Download Button	Simulate failed download.	Download button resets to default state.	**PASS**
11	Reset Delete Button	Simulate failed account deletion.	Delete button resets.	**PASS**
12	Fetch Settings Error Handling	Throw error while fetching data privacy settings.	Error handled gracefully, UI does not crash.	**PASS**
13	Toggle False on API Returning False	Mock API returning false.	AI toggle set to false.	**PASS**
14	Missing User Details	Provide incomplete user details.	UI handles missing fields safely.	**PASS**
15	Loading State During Download	Trigger data download.	Loading spinner shown.	**PASS**
16	Empty API User Response	Mock API returning empty user data.	UI displays fallback values.	**PASS**
17	Modal Switching	Open one modal, then another.	Correct modal switches smoothly.	**PASS**
18	Missing Backend URL	Provide null/undefined backend URL.	UI handles gracefully.	**PASS**
19	Close Modal via Backdrop	Click overlay/backdrop.	Modal closes.	**PASS**
20	Undefined AI Suggestion API Response	API returns malformed suggestion data.	UI handles safely.	**PASS**

21	Maintain Toggle During API	Toggle while API is pending.	Toggle remains stable during request.	**PASS**
22	Partial User Details	Provide partial user profile.	Renders without breaking.	**PASS**
23	Modal Interaction Coverage	Test all modal open/close states.	All modal behaviors verified.	**PASS**
24	API Error Coverage	Simulate multiple error types.	All errors handled correctly.	**PASS**
25	User Interaction Coverage	Test inputs, clicking, toggles, ESC, backdrop.	All interactions work as expected.	**PASS**

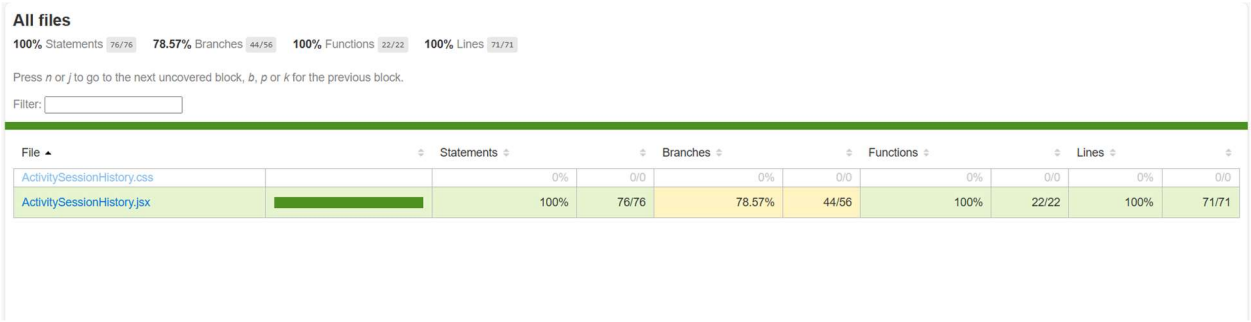
29. ActivitysessionHistory.jsx testing report

Unit Testing Report:

```
✓ test/unitTesting/ActivitySessionHistory.test.jsx (16 tests) 857ms
  ✓ ActivitySessionHistory — FULL 100% Coverage (16)
    ✓ renders all sections 131ms
    ✓ loads active sessions 41ms
    ✓ signs out one session 53ms
    ✓ handles signout error 110ms
    ✓ logs out all devices successfully 32ms
    ✓ handles logout-all failure 95ms
    ✓ toggles security alerts see-more 34ms
    ✓ toggles activity see-more 44ms
    ✓ downloads activity report successfully 32ms
    ✓ handles download error 94ms
    ✓ clears activity history 31ms
    ✓ handles clear history error 94ms
    ✓ handles empty sessions list 18ms
    ✓ handles empty security alerts 15ms
    ✓ handles empty activity history 17ms
    ✓ handles profile fetch error 12ms

Test Files  1 passed (1)
     Tests  16 passed (16)
  Start at  22:00:46
  Duration  3.27s (transform 283ms, setup 219ms, import 505ms, tests 857ms, environment 1.14s)
```

Test Coverage Summary:



Detailed Test Report:

Test Case ID	Description	Test Steps / Action	Expected Result	Status
01	Component Rendering	Render component with full API success mocks.	All main UI sections (navbar, sidebar, footer, titles) are displayed.	**PASS**

02	Load Active Sessions	After mock API resolves session list.	Session items (Chrome–Windows, Firefox–Linux) render correctly.	**PASS**
03	Sign Out Single Session	Click first “Sign Out” button.	Calls /logoutSession API with correct token and updates UI.	**PASS**
04	Handle Sign Out Error	Mock logoutSession to throw error.	Error is logged; UI does not crash.	**PASS**
05	Sign Out All Devices	Click “Sign Out from All Devices”.	Calls /logoutAllSessions API successfully.	**PASS**
06	Sign Out All Error	Mock logoutAllSessions to throw error.	Error is logged; UI remains stable.	**PASS**
07	Toggle See More (Security Alerts)	Click “See More” under alerts.	Button toggles to “See Less”; alerts expand.	**PASS**
08	Toggle See More (Activity History)	Click “See More” under activity history.	Button toggles to “See Less”; activities expand.	**PASS**
09	Download Activity Report	Click “Download Activity Report (PDF)”.	Calls /downloadActivityHistoryReport API successfully.	**PASS**
10	Download Error Handling	Mock download API to throw.	Error logged; UI unaffected.	**PASS**
11	Clear Activity History	Click “Clear Activity History”.	Calls /clearActivityHistory API successfully and clears UI list.	**PASS**
12	Clear History Error	Mock clearHistory to throw.	Error logged; UI continues working.	**PASS**

13	Empty Sessions Handling	Mock sessions API returning [].	Displays “No active sessions” fallback message.	**PASS**
14	Empty Security Alerts Handling	Mock alerts API returning [].	Displays “No security alerts” message.	**PASS**
15	Empty Activity History Handling	Mock history API returning [].	Displays “No activities” message.	**PASS**
16	Profile Fetch Error	First API call (myProfile) throws error.	Error logged; main heading still renders (no crash).	**PASS**

30. Toggle.jsx testing report

Unit Testing Report:

```
RERUN  src/components/Toggle.jsx x2
       Filename pattern: test/unitTesting/Toggle.test.jsx

✓ test/unitTesting/Toggle.test.jsx (12 tests) 288ms
  ✓ Toggle Component (12)
    ✓ Happy Paths (3)
      ✓ should render the toggle switch with the correct initial state 118ms
      ✓ should call onChange with the correct value when toggled 26ms
      ✓ should toggle from true to false correctly 19ms
    ✓ Edge Cases (5)
      ✓ should handle undefined value prop gracefully 10ms
      ✓ should handle null value prop gracefully 11ms
      ✓ should not throw error if onChange is not provided 14ms
      ✓ should handle multiple rapid clicks with controlled component behavior 11ms
      ✓ should handle multiple clicks with updated props 13ms
    ✓ Component Structure (2)
      ✓ should render with correct DOM structure 14ms
      ✓ should update when value prop changes 18ms
    ✓ Error Handling (2)
      ✓ should handle missing onChange prop safely 11ms
      ✓ should call onChange only when provided 18ms

Test Files  1 passed (1)
     Tests  12 passed (12)
  Start at  23:27:03
  Duration  764ms

PASS  Waiting for file changes...
      press h to show help, press q to quit
```

Test Coverage Report:

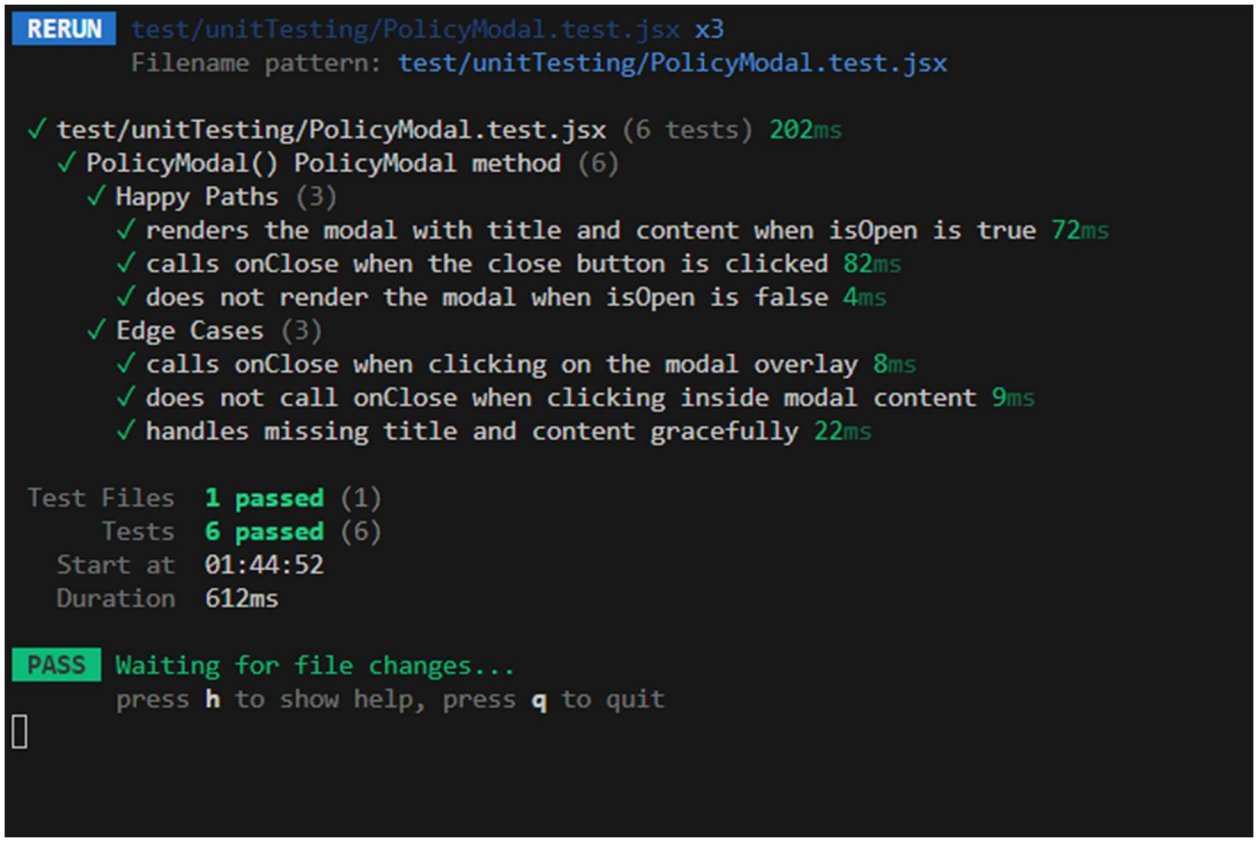
Detailed Test Report:

Test Case ID	Description	Test Steps / Action	Expected Result	Status
01	Initial Render State	Render toggle with default initial state.	Toggle renders with correct initial value.	**PASS**

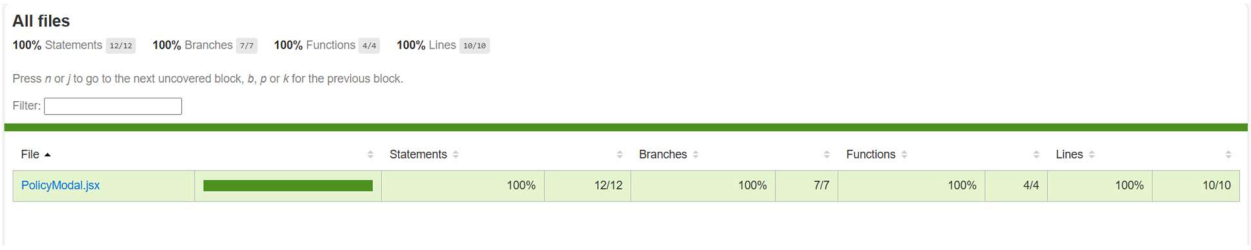
02	Toggle Change Event	Click toggle.	Calls onChange with correct updated value.	**PASS**
03	Toggle True → False	Simulate toggle from true to false.	State updates correctly.	**PASS**
04	Undefined Value Prop	Provide value={undefined}.	Component handles gracefully without breaking.	**PASS**
05	Null Value Prop	Provide value={null}.	Component handles gracefully.	**PASS**
06	Missing onChange	Render without onChange prop.	No error thrown.	**PASS**
07	Rapid Click Handling	Simulate multiple rapid toggle clicks.	Controlled behavior remains consistent.	**PASS**
08	Multiple Updates With Props	Click multiple times with changing props.	Component remains stable and updates correctly.	**PASS**
09	DOM Structure Check	Inspect DOM.	Component renders correct structural HTML.	**PASS**
10	Prop Change Handling	Update value prop.	Toggle updates UI accordingly.	**PASS**
11	Missing onChange Handling	Remove onChange prop.	Component handles safely without crashing.	**PASS**
12	Conditional onChange Call	Call only when onChange is provided.	Prevents crashes; works correctly.	**PASS**

31. PolicyModal.jsx testing report

Unit Testing Report:



Test Coverage Report:



Detailed Test Report:

Test Case ID	Description	Test Steps / Action	Expected Result	Status

01	Render Modal (Open State)	Render component with isOpen = true along with title & content.	Modal displays correctly with provided title and content.	**PASS**
02	Close Button Handler	Click the modal's close ("×") button.	onClose callback is triggered once.	**PASS**
03	Do Not Render When Closed	Render component with isOpen = false.	Modal does not render in DOM.	**PASS**
04	Overlay Click Close	Click anywhere outside modal content (background overlay).	Modal closes and onClose is called.	**PASS**
05	Inside Modal Click (No Close)	Click inside modal content area.	onClose is not triggered; modal remains open.	**PASS**
06	Missing Title & Content	Render modal without providing title/content props.	Component handles gracefully without errors and renders safely.	**PASS**

32. PrivacyPolicy.jsx testing Report

Unit Testing Report:

```
✓ test/unitTesting/PrivacyPolicy.test.jsx (5 tests) 199ms
  ✓ PrivacyPolicy() PrivacyPolicy method (5)
    ✓ Happy Paths (3)
      ✓ renders the PrivacyPolicy component with all sections 108ms
      ✓ renders list items under "Information We Collect" section 23ms
      ✓ renders list items under "How We Use Your Information" section 29ms
    ✓ Edge Cases (2)
      ✓ renders without crashing when no props are passed 17ms
      ✓ handles unexpected text content gracefully 16ms

 Test Files  1 passed (1)
      Tests  5 passed (5)
   Start at  01:54:48
   Duration  2.42s (transform 138ms, setup 229ms, import 295ms, tests 199ms, environment 1.17s)
```

Test Coverage report:

Detailed Test Report:

Test Case ID	Description	Test Steps / Action	Expected Result	Status
01	Render All Sections	Render the <PrivacyPolicy /> component with full props.	Component displays all main sections: headers, paragraphs, lists.	**PASS**
02	Render "Information We Collect" Items	Inspect list under the “Information We Collect” section.	All provided list items appear correctly.	**PASS**

03	Render "How We Use Your Information" Items	Inspect list under this section.	All usage-information list elements render properly.	**PASS**
04	Render Without Props (Graceful Fallback)	Render component with no props passed.	Component does not crash; fallback rendering works.	**PASS**
05	Handle Unexpected Text Content	Pass malformed/edge-case text content.	Component renders gracefully without throwing errors.	**PASS**

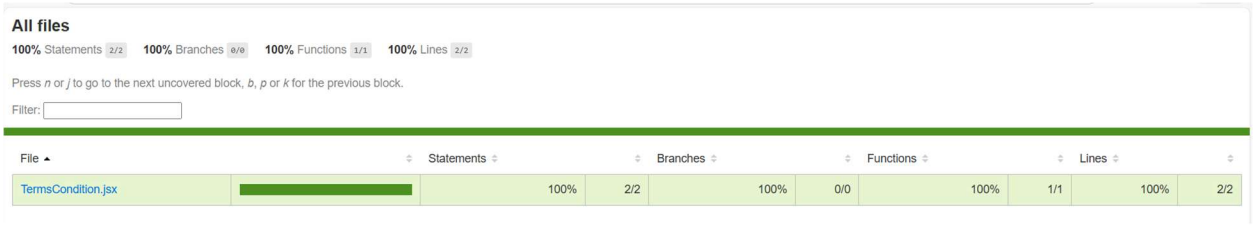
33. TermsCondition.jsx testing Report

Unit testing report:

```
✓ test/unitTesting/TermsCondition.test.jsx (5 tests) 177ms
  ✓ TermsCondition() TermsCondition method (5)
    ✓ Happy Path Tests (3)
      ✓ should render the TermsCondition component with all sections 103ms
      ✓ should render list items under Acceptable Use section 17ms
      ✓ should render list items under Investment Disclaimers & AI Insights section 16ms
    ✓ Edge Case Tests (2)
      ✓ should handle rendering without crashing even if no props are passed 17ms
      ✓ should handle unexpected HTML structure gracefully 18ms

Test Files  1 passed (1)
     Tests  5 passed (5)
  Start at  02:03:32
  Duration  2.75s (transform 125ms, setup 290ms, import 321ms, tests 177ms, environment 1.35s)
```

Test coverage report:



Detailed Test report:

Test Case ID	Description	Test Steps / Action	Expected Result	Status
01	Render All Sections	Render <TermsCondition /> with valid props.	Full Terms & Conditions layout and all major sections render correctly.	**PASS**
02	Render "Acceptable Use" List Items	Inspect the list items in the “Acceptable Use” section.	All provided list entries appear as expected.	**PASS**

03	Render "Investment Disclaimers & AI Insights" Items	Inspect the disclaimers and AI notices.	List items render correctly within the section.	**PASS**
04	Render Without Props	Render component without providing props.	Component renders safely without crashing.	**PASS**
05	Handle Unexpected HTML Structure	Provide malformed or unexpected content.	Component handles content gracefully without errors.	**PASS**