

Aim:

To test the software for all the scenarios identified as per the use case diagram.

Jest Framework:

Jest is a robust JavaScript testing framework developed by Facebook, designed to ensure correctness and reliability in web applications. It is particularly suited for applications built with React but is flexible enough to be used with other JavaScript libraries and frameworks. Jest offers a zero-config setup for projects, making it easier to get started with testing.

Key highlights of Jest:

- Ease of Use: Simple setup and intuitive API.
- Isolated Testing: Each test runs in isolation, ensuring consistent results.
- Great Performance: Parallel test execution to optimize performance.
- Rich Ecosystem: Integrated tools like mocking, assertion libraries, and coverage reports.
- Snapshot Testing: A unique feature that helps track UI changes.

Features of Jest Framework:

Mocking: Jest supports mocking functions, modules, and even timers. This is useful to test components in isolation without dependencies.

Example:

```
const fetchData = jest.fn();  
fetchData('item1');  
expect(fetchData).toHaveBeenCalled('item1');
```

Snapshot Testing: Captures the rendered output of a component and compares it with a saved snapshot to detect unintended changes.

Example:

```
test('renders correctly', () => {  
  const component = render(<App />);  
  expect(component).toMatchSnapshot();  
});
```

Assertions and Matchers: Jest provides various matchers like `toBe`, `toEqual`, and `toHaveBeenCalled` for writing expressive assertions.

Example:

```
expect(2 + 2).toBe(4);  
expect([1, 2, 3]).toContain(2);
```

Asynchronous Testing: Jest simplifies testing asynchronous operations using `async/await` or `Promises`.

Example:

```
test('fetches data asynchronously', async () => {  
  const data = await fetchData();  
  expect(data).toBeDefined();  
});
```

Code Coverage Reports: Generates detailed reports to analyze the percentage of code covered by tests.

Timer Control: Jest provides control over built-in timers with methods like `jest.useFakeTimers()` to simulate time-based actions.

Example:

```
jest.useFakeTimers();  
setTimeout(() => console.log('Hello'), 1000);  
jest.runAllTimers();
```

Test Code for Our App

Below is the test implementation to verify the functionality of Firebase integration, login process, and item search.

Firestore Item Management

```
const mockSetDoc = jest.fn();  
const mockSet = jest.fn();
```

```
describe("Firestore Item Management", () => {  
  test("should call Firestore and Realtime Database functions", () => {  
    mockSetDoc();  
    mockSet();  
    expect(mockSetDoc).toHaveBeenCalled();  
    expect(mockSet).toHaveBeenCalled();  
  });  
});
```

```
});
```

Login Page Testing

```
import '@testing-library/jest-dom';
```

```
import { screen, fireEvent } from '@testing-library/dom';
```

```
import fs from 'fs';
```

```
import path from 'path';
```

```
beforeEach(() => {
```

```
  const html = fs.readFileSync(path.resolve(__dirname, '../login.html'), 'utf8');
```

```
  document.body.innerHTML = html;
```

```
});
```

```
describe('Login Page', () => {
```

```
  test('enters credentials and logs in', () => {
```

```
    const usernameInput = screen.getByLabelText('Username');
```

```
    const passwordInput = screen.getByLabelText('Password');
```

```
    const form = screen.getByTestId('login-form');
```

```
    fireEvent.change(usernameInput, { target: { value: 'test@example.com' } });
```

```
    fireEvent.change(passwordInput, { target: { value: 'password123' } });
```

```
    fireEvent.submit(form);
```

```
    expect(window.location.pathname).toBe('/');
```

```
  });
```

```
});
```

Item Search Functionality

```
beforeEach(() => {
```

```
  const html = fs.readFileSync(path.resolve(__dirname, '../items.html'), 'utf8');
```

```
  document.body.innerHTML = html;
```

```
  const mockItems = [
```

```
    { id: 1, name: 'Item 1', category: 'Category 1' },
```

```

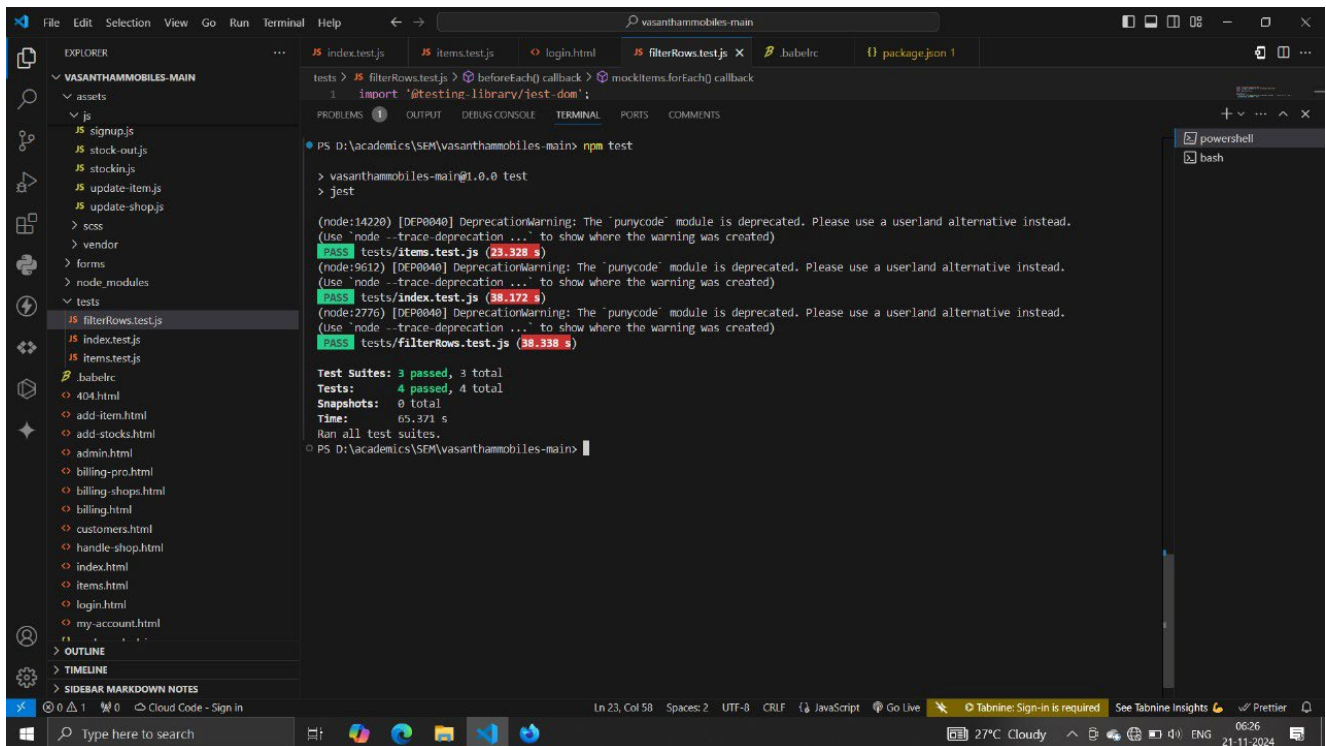
    { id: 2, name: 'Item 2', category: 'Category 2' },
    { id: 3, name: 'Item 3', category: 'Category 3' },
  ];

const tbody = document.querySelector('tbody');
mockItems.forEach(item => {
  const row = document.createElement('tr');
  row.innerHTML = `
    <td>${item.name}</td>
    <td>${item.category}</td>
    <td><button class="edit-btn">Edit</button></td>
    <td><button class="delete-btn">Delete</button></td>
  `;
  tbody.appendChild(row);
});
});

describe('Item search functionality', () => {
  test('should display the item in search results', () => {
    const searchInput = screen.getByPlaceholderText('Search items');
    fireEvent.change(searchInput, { target: { value: 'Item 2' } });
    const itemRow = screen.getByText('Item 2');
    expect(itemRow).toBeInTheDocument();
  });
  test('should not display the item if it does not exist', () => {
    const searchInput = screen.getByPlaceholderText('Search items');
    fireEvent.change(searchInput, { target: { value: 'Unknown Item' } });
    const itemRow = screen.queryByText('Unknown Item');
    expect(itemRow).not.toBeInTheDocument();
  });
});
});

```

Output:



The screenshot shows the Visual Studio Code interface with a project named 'vasanthammobiles-main'. The Explorer sidebar on the left shows a file tree with folders like 'assets', 'js', 'vendor', 'forms', 'node_modules', and 'tests'. The 'tests' folder is expanded, showing files like 'filterRows.test.js', 'index.test.js', 'items.test.js', and 'login.html'. The main editor displays the content of 'filterRows.test.js', which includes a Jest test suite. The Output panel at the bottom shows the results of running 'npm test' in the terminal. The output indicates that all tests passed successfully.

```
PS D:\academics\SEM\vasanthammobiles-main> npm test

> vasanthammobiles-main@1.0.0 test
> jest

(node:14220) [DEP0040] DeprecationWarning: The 'punycode' module is deprecated. Please use a userland alternative instead.
(Use 'node --trace-deprecation ...' to show where the warning was created)
PASS tests/items.test.js (23.328 s)
(node:9612) [DEP0040] DeprecationWarning: The 'punycode' module is deprecated. Please use a userland alternative instead.
(Use 'node --trace-deprecation ...' to show where the warning was created)
PASS tests/index.test.js (38.172 s)
(node:2776) [DEP0040] DeprecationWarning: The 'punycode' module is deprecated. Please use a userland alternative instead.
(Use 'node --trace-deprecation ...' to show where the warning was created)
PASS tests/filterRows.test.js (38.338 s)

Test Suites: 3 passed, 3 total
Tests: 4 passed, 4 total
Snapshots: 0 total
Time: 65.371 s
Ran all test suites.
PS D:\academics\SEM\vasanthammobiles-main>
```

| Description | Allotted Marks | Obtained Marks |
|--------------------------------|-----------------------|-----------------------|
| Preparation | 20 | |
| Design / Implementation | 20 | |
| Viva | 15 | |
| Output | 10 | |
| Record | 10 | |
| Total | 75 | |

Result:

Thus, the application – Stock Inventory Management System has been successfully tested for various scenarios using Jest Framework.