Streamlined Event Scheduling Application

A Test plan report by – Mohith Tummala

Test Plan: *Fetching Verified User events based on React state Management*

1. Test Plan Identifier

TP-EventApp-MyEvents-001 (Verified User Specific Events)

2. Introduction

This test plan outlines the testing approach for the MyEvents Module in our **Event Scheduling Application**. The module plays a critical role in retrieving, filtering, and displaying user-specific events while ensuring smooth navigation and state management.

This responsible test aims to verify its correctness, performance, and reliability based on defined requirements and maintains usability and accessibility standards.

3. Test Items

The primary test item in this document is the MyEvents React component, responsible for:

- Fetching and displaying user-specific events from the Redux store.
- Handling pagination using infinite scrolling.
- Managing state with React Hooks (useState, useEffect).
- Storing user authentication data in localStorage.
- Enabling navigation through react-router-dom.

4. Features to be Tested

- **Redux Store Integration**
  - Ensure loadEvents() is dispatched when the component mounts.
  - Verify if useSelector correctly retrieves allEvents from the store.
- **Local Storage Handling**
  - Ensure loggedInUser is fetched correctly from localStorage.
  - Check behavior when loggedInUser is null (not logged in).
- **Filtering Events**
  - Ensure MyEvents filters only the events created by the logged-in user.
  - Verify correct **initial** event list (first 5 events).

- **Infinite Scroll (Pagination)**
    - Ensure clicking "Load More" appends new events correctly.
    - Check if hasMore is set to false when all events are loaded.
- **Navigation Behavior**
    - Ensure clicking **Back** (navigate(-1)) works as expected.
    - Verify that clicking an event redirects to "/events/{event.id}".
- **Loading State**
    - Ensure "Loading events" appears before events are loaded.
    - Verify "Loading more event" when scrolling down.

5. Features Not to be Tested

- UI Styling (CSS and animations)
- Backend API calls (Mocking will be used instead)
- Security testing (as authentication is handled locally)

6. Testing Approach

The testing approach includes:

- **Unit Testing** using Jest:
    - Verify Redux (useSelector, useDispatch).
    - Verify localStorage and useNavigate().
    - Test filtering logic (correct user events displayed).
- **Integration Testing** using Jest:
    - Render the component and verify UI updates correctly.
    - Simulate user interactions (scrolling, clicking buttons).
- **Regression Testing**:
    - Manually verifying that fixes do not break existing features.

7. Item Pass/Fail Criteria

- **If Pass:**
    - loadEvents() action is successfully dispatched and updates the Redux store with event data.The first 5 events are displayed correctly, and additional events load properly via infinite scrolling.

- The changes done in linked component will be updated promptly in MyEvents.
- The "Load More" button correctly appends more events when available
- Clicking an event navigates to the correct event details page
- The "Loading events." and "Loading more events." messages appear at appropriate times.

- **If Fail:**
  - Events are incorrectly filtered, showing events not created by the logged-in user.
  - Pagination fails, preventing more events from loading when scrolling.
  - Clicking an event does not navigate to its details page.

8. Suspension Criteria and Resumption Requirements

- Testing will be suspended if Redux store fails to load test data.
- Infinite scroll malfunctions, causing the event list to freeze or stop loading new data.
- Ensure loadEvents() correctly dispatches and populates the Redux store.

9. Test Deliverables

- **Test cases document:** Detailed test scenarios covering each feature.
- **Test execution report:** Summary of passed and failed test cases.
- **Bug report:** Documentation of defects found during testing.
- **Test logs and screenshots:** Evidence of failures and successful test cases.

10. Testing Tasks

- Develop unit test cases for Redux, local storage, and event filtering.
- Execute test cases and log results if there is bad behaviour.
- Report bugs and re-test after fixes.

11. Environmental Needs

- **Development Framework:** React.js
- **Testing Framework:** Jest, Redux storage, react state management, Chrome.
- **Test Data:** User created events, User Authentication and db.json

12. Responsibilities

- **Fullstack Developer:** Majorly worked on React State Management.
- **Lead QA Engineer:** Implemented test strategy and fixed all bug reports.
- **Integration Expert:** Works on Continuous Integration (GitHub Actions).

13. Risks and Contingencies

- **Risk:** Redux store fails to provide test data.
  Mitigation: Use a Redux mock store to simulate state changes in unit tests.
- **Risk:** Unreliable UI state updates due to missing dependencies in React hooks.
  Mitigation: Ensure all dependencies are properly managed in useEffect hooks.

14. Test Reporting

- **Bug Lifecycle Tracking:**
  - Bugs are reported in an issue tracker (e.g., GitHub, Teams).
  - Issues are categorized by **Priority (P1, P2, P3)** and **Severity (Critical, Major, Minor)**.
- **Incident Reporting:**
  - Each issue must include:
    - Summary & Description
    - Steps to Reproduce
    - Logs & Screenshots
    - Environment Details (OS, Browser)
- **Test Coverage:**
  - Ensure **100% of core functionalities** are tested.
  - Validate that all **business requirements** are covered.

15. Test Result Analysis

- **Defect Analysis:**
  - Defects are analyzed based on **impact and root cause**.
  - If a bug cannot be reproduced, testers must provide additional logs and environment details.
- **Pass/Fail Metrics:**
  - Percentage of tests passed vs. failed.

- Number of high-priority defects vs. resolved defects.
- Performance trends over multiple test runs.

- **Final Test Summary:**
  - Document **all major findings**, fixes, and final approval status.

16. Approval

- **Professor:** *Prof. Peter Dillinger*
- **Lead Fullstack Engineer:** *Mohith Tummala*
- **Lead Automation Tester:** *Mohith Tummala*