**Comprehensive Test Plan: Open Days Tracking Feature in TravelMate**

**1. Executive Summary**

This document presents the quality assurance framework for the "Open Days Tracking" feature within TravelMate, based on the Software Requirements Specification (SRS). This feature enhances transparency and efficiency for Swisscom administrators by enabling them to monitor Open Day usage, review consumption patterns, and reallocate unutilized days.

The plan details a comprehensive approach to functional and non-functional testing, with a reinforced emphasis on robust error handling for external dependencies (like WFIDB), extensive UI and backend test coverage, and rigorous non-functional validation for performance, security, and scalability. The SRS introduces new functional requirements, such as distinct UI modes (view/edit), detailed traveler display, and critical refactoring efforts, alongside refined non-functional criteria for reliability and maintainability. This necessitates a rigorous and adaptable quality assurance strategy, prioritizing a "Shift-Left" approach and automation for early defect detection and a high-quality release.

**2. Feature Overview and Objectives**

This section provides a foundational understanding of the "Open Days Tracking" feature, its purpose, functionalities, and integration within TravelMate, as defined in the SRS.

**2.1. Feature Purpose and Motivation**

The "Open Days Tracking" feature addresses TravelMate's inability to monitor "Open Days" usage—up to 10 annual travel days for international employees, managed within a unit's 180-day yearly limit. This lack of tracking leads to poor visibility, inefficient reallocation of unused days, and hinders data-driven decisions. The feature aims to resolve these issues by monitoring Open Day usage, improving transparency, efficiency, and planning within TravelMate, allowing administrators to view usage and redistribute unused days. It is designed as an extension fully integrated into the existing TravelMate system.

**2.2. Core Functionalities**

The "Open Days Tracking" feature introduces several core functionalities, as detailed in the SRS:

* **Tracking Open Days Usage:** Tracks and visually displays whether an Open Day was utilized.
* **Reporting and Visibility:** Provides administrators a dashboard to view planned Open Days, indicating if the date has passed for quick assessment of availability.
* **Reallocation of Unused Travel Days:** Allows marking and reassigning unused travel days for better resource utilization.
* **Seamless Integration with TravelMate:** Fully embedded within the existing TravelMate UI, compatible with current workflows.
* **Display of Traveler Information:** Enables administrators to click on a date to view a list of employees traveling, aiding informed reallocation decisions.
* **Distinct View and Edit Modes:** Supports separate view and edit modes, with robust error handling to revert changes if saving fails.
* **Refactoring for Integration and Consistency:** Includes necessary codebase refactoring to unify inconsistent patterns, improve maintainability, enhance testability, and ensure conformity to modern coding standards.

**2.3. Key Definitions**

Consistent with the SRS, the following definitions are critical:

* **Open Day:** A designated travel day in TravelMate, allocated at the unit level, allowing travel bookings.
* **Unit:** A group of teams sharing a maximum of 180 Open Days per year.
* **Reviewer (Admin):** An administrator managing Open Days for a unit.
* **WFIDB:** An external system providing employee data.
* **TravelMate:** The internal travel management system being enhanced.

**2.4. In-Scope and Out-of-Scope for Testing**

Based on the SRS:

* **In Scope:** Open Day status tracking, administrator dashboard UI (view/edit modes), reallocation functionality, internal backend services/APIs, and refactoring efforts.
* **Out of Scope:** Automated rescheduling/reassignment, enforcement of travel compliance policies, and forecasting future travel needs.

**2.5. Constraints and Assumptions Impacting Testing**

The SRS clarifies constraints and assumptions influencing the testing strategy:

**Constraints:**

* **Integration with an Existing System:** Requires development within TravelMate, necessitating refactoring and extensive regression testing to maintain stability.
* **Active Development in the Same Area:** Continuous development by multiple contributors demands careful coordination and adaptable testing strategies.
* **Limited Existing Tests:** Few automated tests exist, requiring a robust testing framework (unit, integration, E2E) as a prerequisite for stability.
* **Company-Specific Component Library:** UI must use Swisscom’s internal component library, influencing UI testing for consistency and adherence to design guidelines.
* **Dependency on External System (WFIDB):** Relies on WFIDB, introducing potential limitations (API availability, rate limits, response times, authentication, data synchronization) that require specialized integration and error handling tests.

**Assumptions:**

* **System Refactoring Will Be Supported:** Necessary refactoring is feasible and approved.
* **Collaboration with Other Developers Will Be Managed:** Effective version control, code reviews, and team coordination will prevent conflicts.
* **Testing Will Be a Priority:** Establishing a comprehensive testing framework is acknowledged as necessary.
* **Sufficient Documentation for the Component Library Exists:** Adequate documentation or internal support for the component library is available.
* **Feature Flag Will Be Used to Control Feature Availability:** Functionality will be controlled by a feature flag for safe deployment and testing, enabling per-environment control and runtime configuration. If misconfigured, the feature will remain disabled by default.

**3. Overall Test Strategy**

This section outlines the layered testing methodology and commitment to automation for comprehensive coverage and early defect detection.

**3.1. Layered Testing Approach**

A multi-layered testing approach will be employed, progressing from isolated components to comprehensive end-to-end workflows:

* **Unit Testing (Backend):** Validates isolated units of backend code, such as business logic for status determination and booking matching.
* **Integration Testing (Backend):** Validates interactions between backend components, including REST API endpoints, database interactions, and integration points with WFIDB, covering error handling and retry mechanisms.
* **Component Testing (Frontend):** Tests individual UI components in isolation, such as status badges, reallocation buttons, and traveler display pop-ups.
* **End-to-End Testing (Frontend):** Simulates full administrative workflows, covering scenarios like dashboard viewing, the reallocation process, comprehensive error handling, and mode switching.
* **Non-Functional Testing:** Assesses critical system qualities including performance, security, maintainability, scalability, and compliance.

**3.2. Automation First**

Given the "Limited Existing Tests" constraint, quality assurance activities are integrated early in the SDLC to build quality in from the outset.

* **Automation:** Prioritized across all testing layers (unit, integration, E2E) for rapid feedback and efficient regression testing. Specific tools like Cypress/Playwright for frontend and Spring Boot Test/Testcontainers for backend will be utilized.
* **Continuous Integration (CI):** Automated tests are integrated into CI/CD pipelines, executing on every code commit to enable early detection of regressions and mitigate risks from "Active Development in the Same Area". The SRS explicitly states that deployment is blocked if any pipeline tests fail, making robust automated testing indispensable.

**3.3. Test Items**

The specific items designated for testing include:

* REST API endpoints for Open Day data, including any new data structures required for traveler information.
* All Open Day dashboard components on the frontend, encompassing view/edit modes, reallocation options, and traveler details.
* Complete reallocation workflows and their impact on underlying data.
* Logging and audit trails, especially for WFIDB interactions and error conditions.
* Integration points with WFIDB, including Grafana visibility for errors, backend caching, and batch requests.
* Refactored code modules, ensuring consistency, maintainability, and absence of regressions.

**4. Functional Requirements Testing**

This section details the testing approach for each functional requirement, ensuring comprehensive coverage, incorporating requirements from the SRS.

**4.1. Functional Test Cases Matrix**

This matrix maps each functional requirement from the SRS to specific test cases, ensuring complete coverage and serving as a blueprint for automated tests. The Functional Requirement (FR) IDs align with the numbering in the "SRS OPEN days after feedback.docx".

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FR ID** | **Functional Requirement** | **Test Case ID** | **Test Case Description** | **Test Type** | **Preconditions** | **Pass/Fail Criteria** |
| FR3.2.1 | Retrieve Travel Data | TC-FR3.2.1-001 | Validate successful retrieval and display of Travel data. | E2E / Unit | Admin logged in. | Dashboard loads within 2s and displays correct data. |
| FR3.2.2 | Determine Open Day Status | TC-FR3.2.2-001 | Validate past Open Days without bookings marked 'unused'. | Component | Past Open Day, no bookings. | Status is 'unused'. |
| FR3.2.2 | Determine Open Day Status | TC-FR3.2.2-003 | Validate past Open Days with bookings no showing up | Component | Past Open Day, existing bookings. | Day is not showing up as unused |
| FR3.2.2 | Determine Open Day Status | TC-FR3.2.2-004 | Verify behavior when booking data unavailable. | E2E | Booking data unavailable. | Data not displayed as per error handling. |
| FR3.2.3 | Fetch Employee Data from WFIDB | TC-FR3.2.3-001 | Test business logic in the BE, and first name and last name should show in the UI | Unit / E2E | WFIDB available. | Employee data accurate and matched. |
| FR3.2.3 | Fetch Employee Data from WFIDB | TC-FR3.2.3-003 | Check if the BE endpoint is giving back errors | Grafana | WFIDB simulated down/empty. | System logs error; errors visible in Grafana. |
| FR3.2.3 | Fetch Employee Data from WFIDB | TC-FR3.2.3-004 | Validate caching for WFIDB API calls. | Backend | Multiple requests. | Subsequent requests served from cache. |
| FR3.2.4 | Allow Admin to Reallocate Unused Days | TC-FR3.2.4-001 | Validate successful deselection of 'unused' Open Day. | E2E | Admin logged in, Open Day 'unused'. | Reallocation successful; status updated. |
| FR3.2.5 | Display Reallocation Days | TC-FR3.2.5-001 | Validate correct un selection of travel day after saving from reallocation view | E2E | Admin in view mode. | Dashboard loads correctly with accurate data. |
| FR3.2.6 | Display Travelers on Day | TC-FR3.2.6-001 | Validate successful display of traveler list on date click. | E2E | Admin views dashboard, clicks date with travelers. | Traveler list displayed correctly. |
| FR3.2.6 | Display Travelers on Day | TC-FR3.2.6-002 | Verify empty traveler list for no bookings. | E2E | Admin views dashboard, clicks date with no travelers. | Empty list or "No travelers" message. |
| FR3.2.7 | Differentiate View/Edit | TC-FR3.2.7-001 | Validate transition from view to edit mode. | E2E | Admin in view mode. | Mode changes correctly; edit enabled. |
| FR3.2.7 | Differentiate View/Edit | TC-FR3.2.7-002 | Validate transition from edit to view mode. | E2E | Admin in edit mode. | Mode changes correctly; edit disabled. |
| FR3.2.8 | Refactoring for Integration | TC-FR3.2.8-002 | Validate business logic of the code | Unit | Refactored components. | Tests are green after the refactor. |

**5. Non-Functional Requirements Testing**

This section details the validation of each Non-Functional Requirement (NFR) to ensure performance, reliability, security, maintainability, scalability, and compliance, incorporating the requirements from the SRS.

**5.1. Non-Functional Testing Details Matrix**

This matrix maps each NFR to its specific testing approach, tools, and success criteria, reflecting the SRS.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **NFR Category** | **NFR Description** | **Metric/Target** | **Testing Approach** | **Tools/Techniques** | **Test Environment** | **Success Criteria** |
| Performance | Dashboard loads <2s | < 2s (95%) | Frontend Perf. Testing; E2E Tests | Cypress | Local, Staging | Dashboard loads <2s for 95% of E2E runs. |
| Performance | Queries determining status <1.5s (250 req/unit). | < 1.5s | Backend Load/Stress Testing | Integration load test | Local | Query response times meet target under load. |
| Reliability | Uptime 99.5%. | 99.5% uptime | Continuous Monitoring | Prometheus, Grafana | Staging, Prod-like | Uptime metrics consistently meet target. |
| Reliability | Deploy only when all tests pass. | 100% pipeline success | CI/CD Gate Checks; Automated Test Execution | GitLab CI | CI/CD Pipeline | Deployment pipeline green; no manual overrides. |
| Security | Only authorized admins update/reallocate. | Role-based access | Currently have no way to test this automatically, we will test this by hand. |  |  | Unauthorized access blocked; policies met. |
| Maintainability | 80%+ unit test coverage. | 80%+ coverage | Code Coverage Analysis; Static Code Analysis; Peer Reviews | Idea ide build in tooling | Dev, CI/CD | Coverage reports meet target; static analysis minimal. |
| Scalability | Handle 10,000+ Travels without degradation. | 10,000+ travels | Load Testing; Stress Testing | Integration performance test. | Test Env, Staging | Response times/resource use stable under load. |
| Compliance | Comply with Swisscom IT policies. | Policy adherence | Security Champion Review; Data Privacy Audit; Legal Review | Internal audit tools | Test Env, Prod-like | Policies documented and adhered to. |

**6. User Interface Testing Considerations**

The SRS provides explicit UI design specifications that inform the UI testing strategy.

* **Calendar-based Dashboard:** Ensure the calendar view correctly displays Open Days and their statuses, maintaining visual consistency.
* **Edit Mode:** Comprehensively test seamless transitions into and out of edit mode, the ability to select/unselect days, and the robustness of the submission process. verify the "Show reallocation options" checkbox correctly filters and displays only eligible (unused) days.
* **View Mode:** Ensure accurate display of traveler lists when clicking on a date, proper handling of empty lists.
* **Component Library Adherence:** Conduct visual regression testing and thorough checks for strict adherence to Swisscom’s internal component library and design standards for consistency, usability, and accessibility.
* **Responsiveness:** Ensure the UI maintains usability and visual integrity across various screen sizes and devices.
* **Error Message Display:** Verify error messages related to UI interactions are clearly displayed and provide actionable feedback.

The detailed UI design specifications in the SRS provide a strong foundation for comprehensive UI/UX testing, with explicit distinction between "Edit Mode" and "View Mode" necessitating dedicated test cases for mode switching and unique functionalities.

**7. Conclusion and Recommendations**

The "Open Days Tracking" feature requires a robust QA strategy due to its scope, refactoring needs, limited existing tests, and active development. Functional and non-functional testing are vital for its quality and long-term success. Addressing monitoring gaps and achieving 80% unit test coverage are fundamental.

**Recommendations:**

* Address Monitoring Deficiencies: Resolve monitoring issues urgently for objective NFR validation.
* Check Code Quality and Test Coverage: Integrate automated coverage checks (80% target) and foster developer testing.
* Leverage UI Design for E2E Testing: Use detailed UI specs to inform E2E and component test design
* Run tests as much as possible in the pipeline, this is not feasible right now for integration and E2E test, but this will need to happen if we want to be sure of safety.
* Dedicated Refactoring Testing: Test the refactoring manually and automatically in the BE.

Adhering to this plan will ensure a reliable and efficient "Open Days Tracking" feature for Swisscom administrators, seamlessly integrated into TravelMate and built upon a foundation of high-quality, maintainable code.