SRS open day feature

**Table of Contents**

1. Introduction 1

1.1 Motivation 1

1.2 Scope 2

1.3 Definitions, and Abbreviations. 3

2. The Overall Description 3

2.1 Product Overview and Functionalities 3

2.3 Constraints and Assumptions 5

3. System Requirements 6

3.1 External Interface Requirements 7

3.2 Functional Requirements 7

3.3 Use Cases 9

3.4 Non-Functional Requirements 9

# 1. Introduction

This Software Requirements Specification (SRS) document outlines the scope, objectives, and functional requirements for a new feature in TravelMate, an internal application used within Swisscom to manage and track business travel days. TravelMate enables employees to work from an international office for up to 10 days per year, with Swisscom covering travel and accommodation costs. To ensure compliance with legal regulations, each tribe (a group of multiple teams) is limited to a maximum of 180 travel days per year. The reviewer (admin) of a tribe is responsible for opening these 180 days in the system and approving employee travel requests.

Currently, TravelMate lacks a mechanism to track whether the allocated Open Days are used. This results in inefficiencies such as unused travel days that are not reallocated and a lack of visibility for reviewers and administrators regarding travel day utilization. The goal of this project is to develop a system that tracks Open Days usage and provides this information in an admin panel, allowing reviewers and administrators to:

* View which days have been used and which are still available.
* Reuse or redistribute unused travel days.
* Optimize travel day planning and allocation.

The travel day bookings will continue to be managed within the existing TravelMate system, but this new feature must ensure that reviewers have clear visibility into employeebookings and travel day usage.

## 1.1 Motivation

The Open Days Tracking feature is designed to enhance the functionality of TravelMate by providing administrators with a way to monitor the utilization of Open Days. Currently, TravelMate allows administrators to schedule Open Days, which are designated days when employees or teams are permitted to travel. However, there is no existing mechanism to track whether these days were actually used. This lack of visibility creates inefficiencies in travel planning, as administrators cannot determine whether the allocated travel days are effectively utilized.

Without this feature, several problems arise:

* **Lack of visibility:** Administrators do not have insight into whether Open Days were used, making it difficult to assess travel patterns.
* **Resource misallocation:** Since unused travel days go untracked, organizations may allocate travel resources inefficiently.
* **Limited data-driven decision-making:** Without tracking, administrators cannot optimize travel day allocation based on real usage data.

This feature is aimed at improving transparency, efficiency, and planning within the organization’s travel management system. By tracking Open Days, administrators can make better-informed decisions, ensuring that travel days are not wasted and can be reallocated when necessary.

The intended audience for this document includes:

* **Product owners and stakeholders,** who need to ensure that the feature aligns with business objectives and contributes to improved resource management.
* **Developers and engineers,** who will be responsible for designing and implementing the new functionality within TravelMate.
* **Testers,** who must validate that the feature meets the defined requirements and functions correctly.
* **System administrators,** who will manage and oversee the feature’s usage within the organization.

## 1.2 Scope

This new feature will introduce tracking and reporting capabilities for Open Days, ensuring that administrators have a clear overview of whether scheduled travel days were actually used. The system will allow administrators to see whether an open travel day was used or not and provide the ability to reallocate unused travel days when necessary. These capabilities will improve the efficiency of travel planning and enable organizations to optimize their resource allocation.

The feature will include the following functionalities:

* **Tracking of Open Days usage,** enabling administrators to see whether an open day has been used by someone or not.
* **Reporting capabilities,** allowing administrators to view utilized and unused travel days.
* **Reallocation functionality,** providing the ability to redistribute unused travel days based on need.
* **Improved planning and transparency,** giving organizations data-driven insights into their travel day usage.

However, the feature will **not** include:

* **Automated rescheduling:** While administrators can reallocate unused days, the system will not automatically reassign them.
* **Policy enforcement:** The feature will provide visibility into travel day usage but will not enforce travel compliance policies.
* **Future travel forecasting:** The system will only track past and current Open Days but will not predict future travel needs.

This feature is expected to be developed and implemented within a defined timeline, ensuring that it integrates smoothly into TravelMate without disrupting existing functionalities. Data tracking will be restricted to Open Days within an organization to maintain compliance with internal policies and data privacy regulations.

## 1.3 Definitions, and Abbreviations.

#### **Key Definitions**

* **Open Day** – A designated day within **TravelMate** when employees are allowed to travel and work from an international office. These days are allocated at the **tribe level** (a group of multiple teams) and must be approved by an administrator. Open Days can have different statuses, including:
  + **Used** – The Open Day was utilized for an employee’s travel.
  + **Unused** – The Open Day was not utilized and remains unclaimed.
  + **Reallocated** – The Open Day was unused and has been reassigned for future use.
* **Tribe/Unit** – A group of multiple teams within Swisscom that share a pool of Open Days. Each tribe is assigned a **maximum of 180 Open Days per year**, which are managed by an administrator (reviewer).
* **Reviewer (Admin)** – A designated **administrator** responsible for managing open days within a **tribe**. The reviewer has permissions to **open, track, and reallocate open days** within the TravelMate system.
* **AnyOrg** – An **external system**that provides **employee data** to TravelMate. It is used to substract employee data since we can only save active employees in the database. However, AnyOrg is known to occasionally return **incorrect or incomplete data**, requiring validation.
* **TravelMate** – The **internal travel management system** used within Swisscom to schedule, approve, and track employee business travel. The Open Days Tracking feature will be an enhancement to TravelMate.

# 2. The Overall Description

This section provides a high-level description of the new **Open Days Tracking** feature in TravelMate. It explains the feature’s purpose, how it integrates with the existing system, and the key functionalities it introduces. The goal of this section is to provide a foundational understanding of the product before detailing specific requirements in later sections.

## 2.1 Product Overview and Functionalities

#### Product Overview

The **open days Tracking** feature is an enhancement to **TravelMate**, an internal business travel management system used within Swisscom. TravelMate allows employees to work from an international office for up to 10 days per year, with travel and accommodation costs covered by Swisscom. Each **tribe**, a group of multiple teams, is allocated a maximum of **180 travel days per year**, which must be managed by an assigned **reviewer (admin)** who opens these days in the system and approves employee travel requests.

Currently, TravelMate does not provide visibility into whether these **open days** are actually used. This results in inefficiencies, such as unused travel days not being reallocated and a lack of oversight for administrators. The new feature introduces tracking and reporting capabilities to solve these issues.

This feature is **not a standalone product**; it is an extension of TravelMate and will be fully integrated into the existing system. It does not require separate installation but will require modifications to the **TravelMate database, user interface, and business logic** to support the new functionality.

#### Functionalities

The **open days tracking** feature introduces the following core functionalities:

* **Tracking Open Days Usage**
  + The system will track whether an Open Day was actually used.
  + The status of each Open Day (used or unused) will be stored and displayed.
* **Reporting and Visibility**
  + Administrators will have access to a **dashboard** that displays all planned Open Days.
  + Each Open Day will be marked with an **indicator** showing whether the date has **passed or is upcoming**.
  + The dashboard will allow reviewers to **quickly assess**which Open Days are still available and which have already occurred.
* **Reallocation of Unused Travel Days**
  + Unused travel days can be marked and reassigned to ensure better resource utilization.
* **Seamless Integration with TravelMate**
  + The feature will be embedded in the existing TravelMate interface.
  + It will work with existing travel booking and approval processes.

#### Dependencies and Integration

The new feature will **rely on existing TravelMate components** and will require changes in the following areas:

* **Database Modifications**
  + A new field or flag in the database will track the status of each Open Day.
  + Existing data models will be updated to accommodate usage tracking.
* **User Interface Enhancements**
  + A new dashboard section will display Open Days data.
  + Reviewers will have access to options for marking and reallocating unused travel days.
* **Integration with Business Logic**
  + The system must determine whether a travel day was actually used, based on employee booking records.
  + Administrators must have the ability to update travel day statuses manually if needed.

## 2.3 Constraints and Assumptions

#### **Constraints**

The development of the **Open Days Tracking** feature is subject to several constraints that influence its design, implementation, and integration within the existing TravelMate system:

* **Integration with an Existing System**  
  The feature must be developed within the**existing TravelMate system,** which requires refactoring to accommodate the new functionality. This limits the flexibility of introducing entirely new architectures or significant redesigns.
* **Active Development in the Same Area**  
  The part of the system where this feature is being implemented is currently undergoing **continuous development by multiple contributors.** This means that frequent code changes may impact the implementation, requiring careful coordination and adaptability.
* **Limited Existing Tests**  
  The current system has **few automated tests**, meaning a robust testing framework must be established before developing the new feature. This includes setting up **unit tests, integration tests, and potentially end-to-end tests** to ensure system stability.
* **Use of a Company-Specific Component Library**  
  The feature’s user interface must be built using the **company’s internal component library**. Familiarizing with this library is necessary to maintain **consistency, usability, and adherence to design guidelines.**
* **Data Integrity and Migration**  
  Since this feature introduces new data tracking (i.e., marking Open Days as used or unused), the **existing database schema may need modifications**. Ensuring **data integrity**will be critical, and potential **migration scripts** or updates might be required to apply these changes to historical travel records without affecting existing functionality.
* **Dependency on External System (AnyOrg)**  
  The feature will need to interact with an external system called **AnyOrg**. This dependency may introduce **constraints related to API availability, rate limits, response times, authentication, and data synchronization.** Any limitations or changes in the AnyOrg system could impact how Open Days data is tracked and processed within TravelMate.
* **Integration with an Existing System**  
  The feature must be developed within the existing **TravelMate system**, which requires **refactoring** to accommodate the new functionality. This limits the flexibility of introducing entirely new architectures or significant redesigns.

#### **Assumptions**

The following assumptions are made to define the requirements and guide the development process:

* **System Refactoring Will Be Supported**  
  It is assumed that the necessary refactoring of the TravelMate system will be feasible and approved to accommodate the new feature.
* **Collaboration with Other Developers Will Be Managed**  
  Given that multiple developers are working on the same part of the system, it is assumed that **version control, code reviews, and team coordination** will help prevent conflicts.
* **Testing Will Be a Priority**  
  Since the existing test coverage is low, it is assumed that setting up a **testing framework** will be a **necessary and accepted** part of the development process.
* **Sufficient Documentation for the Component Library Exists**  
  It is assumed that there is **adequate documentation or internal support** available for the company’s component library, making it possible to implement the UI in line with design standards.
* **AnyOrg Will Provide a Stable and Well-Documented API**  
  It is assumed that **AnyOrg will have a reliable API**with**proper documentation** for integration. If AnyOrg's API changes or has issues, it could affect the feature's ability to track and synchronize Open Days accurately.
* **Feature Flag Will Be Used to Control Feature Availability**  
  It is assumed that the Open Days Tracking functionality will be gated behind a feature flag. This allows the development team to safely deploy and test the feature without exposing it to all users. The flag will enable per-environment control and runtime configuration. If the feature flag system fails or is misconfigured, the feature will remain disabled by default to prevent unintended exposure.

# 3. System Requirements

This will be the largest and most important section of the SRS. The user and customer requirements are expanded and detailed to be used to guide the project design, implementation, and testing.

Each requirement in this section should be clear and unambiguous, correct, verifiable, complete and consistent.

It is important to note that you should not specify the implementation details here.

## 3.1 External Interface Requirements

#### **Integration with AnyOrg**

The Open Days Tracking feature in TravelMate relies on AnyOrg to retrieve employee data. AnyOrg provides a list of employees along with additional details such as department, role, and employment status. This integration is critical for verifying whether an Open Day was used.

##### **Key Requirements**

* **Data retrieval and matching:** TravelMate must fetch employee data from AnyOrg and match it with its internal records to determine Open Day usage.
* **Data accuracy issues:** AnyOrg has been known to return incorrect or outdated data. To handle this:
  + Implement validation and logging for inconsistencies.
  + Allow manual corrections by administrators if needed.
* **Data unavailability and error handling:**
  + If AnyOrg fails to return data, the system should retry after a short delay.
  + If repeated failures occur, cached data should be used where possible.
  + Administrators should be notified of persistent errors.
* **Performance considerations:**
  + Implement caching to reduce redundant API calls.
  + Use batch requests where possible.
  + Fetch data asynchronously to avoid UI delays.
* **Security and communication:**
  + AnyOrg is accessed via a REST API using secure HTTPS connections.
  + Authentication is required using API keys or OAuth tokens.

By addressing these constraints, TravelMate can ensure stable and accurate integration with AnyOrg while mitigating potential data reliability issues.

## 3.2 Functional Requirements

#### **1. Retrieve Open Days List**

**Input:** Admin accesses the Open Days dashboard.  
**Processing:** The system queries the database for all scheduled Open Days.  
**Output:** A list of Open Days with their respective dates and statuses (upcoming, passed, used, or unused).  
**Database Changes:** None (read-only query).  
**Error Handling:** If the database is unreachable, an error message is shown, and cached data is used if available.

#### **2. Determine Open Day Status**

**Input:** System checks if the date of an Open Day has passed.  
**Processing:**

* If the date is in the future, it is marked as **upcoming**.
* If the date has passed, the system verifies if any bookings exist for that date.
* If a booking exists, the day is marked as **used**; otherwise, it is **unused**.  
  **Output:** Status update for each Open Day.  
  **Database Changes:** The Open Day’s status is **updated and saved** in the database.  
  **Error Handling:** If booking data is unavailable, the system temporarily marks the day as unknown and flags it for admin review.

#### **3. Fetch Employee Data from AnyOrg**

**Input:** Request to AnyOrg’s API to retrieve employee details.  
**Processing:** The system calls AnyOrg’s API, retrieves employee data, and matches it with TravelMate records.  
**Output:** List of employees eligible for travel.  
**Database Changes:** No persistent changes, but **temporary storage (caching)** may be used.  
**Error Handling:**

* If AnyOrg is unresponsive, the system retries after a delay.
* If incorrect data is returned, inconsistencies are logged, and admin intervention may be required.

#### **4. Identify Used and Unused Open Days**

**Input:** System cross-references Open Days with booking data.  
**Processing:**

* If an Open Day has a matching booking, it is marked as **used**.
* If no matching booking exists, it is flagged as **unused**.  
  **Output:** Updated Open Day statuses displayed in the dashboard.  
  **Database Changes:** The system **saves the updated Open Day status** in the database.  
  **Error Handling:** If booking data is missing, the system defaults to the last known status and logs a warning.

#### **5. Allow Admin to Reallocate Unused Days**

**Input:** Admin selects an unused Open Day and chooses to reallocate it.  
**Processing:** The system updates the Open Day's status to "reallocated" and allows it to be reassigned.  
**Output:** The Open Day is available for reassignment.  
**Database Changes:** The Open Day’s **status is updated and saved in the database**.  
**Error Handling:** If the update fails, an error is logged, and the admin is prompted to try again.

#### **6. Display Open Days on Dashboard**

**Input:** Admin views the dashboard.  
**Processing:** The system retrieves Open Days and their statuses from the database.  
**Output:** A user-friendly dashboard displaying Open Days, statuses, and action options.  
**Database Changes:** None (read-only query).  
**Error Handling:** If data retrieval fails, the dashboard displays an error message and prompts the user to retry.

**7. Log System Actions**

**Input:** Any significant system action, such as Open Day status updates, API requests to AnyOrg, and administrator modifications.  
**Processing:**

* The system logs actions such as:
  + Open Day status changes (e.g., used, unused, reallocated).
  + API calls to AnyOrg, including success, failure, and retry attempts.
  + Administrator actions (e.g., manual corrections, reallocation of Open Days).
* Logs should include timestamps, the responsible user (if applicable), and relevant system messages.  
  **Output:** A structured log entry stored in the system’s logging mechanism.  
  **Database Changes:** Logs are saved in the system’s logging service or database.  
  **Error Handling:**
* If logging fails, the system retries once and falls back to local storage if necessary.
* Critical failures are reported to administrators.

## 3.3 Flow diagram

This flow diagram shows the open days tracking flow. To get a better view, the picture is stored in the Analysis directory of my graduation folder.

## A diagram of a company AI-generated content may be incorrect.

## 3.4 Non-Functional Requirements

The **Open Days Tracking** feature in TravelMate must meet specific non-functional requirements to ensure performance, reliability, security, and maintainability. These requirements define system expectations and provide measurable criteria for validation.

#### **1. Performance**

* The dashboard displaying Open Days should **load within 2 seconds** for 95% of requests.
* Queries that determine Open Day status should complete in**under 1 second** for standard data loads (up to 500 Open Days per tribe).
* API calls should **not block UI operations** and must be handled asynchronously.

#### **2. Reliability**

* The system should maintain **99.5% uptime**, ensuring availability for administrators during working hours.
* The feature must include tests to ensure nothing breaks.
* Open Day status updates must be **consistent**, ensuring no duplicate or conflicting records are stored.

#### **3. Availability**

* The feature should be accessible **24/7**, except during scheduled maintenance.

#### **4. Security**

* Only **authorized administrators** should be able to update Open Day statuses or reallocate unused days.
* User authentication and access control should follow **Swisscom's internal security policies**.
* Sensitive data (such as employee information) must **not be stored locally** unless necessary for caching, and it should be **encrypted at rest.**

#### **5. Maintainability**

* The system should have **80%+ unit test coverage** to ensure code reliability.
* Tests should be automated, covering **key functionalities such as Open Day status determination and API integrations.**
* The feature should follow **modular coding principles** to allow easy modifications and improvements.

#### **6. Scalability**

* The system should handle at least **10,000 Open Days across multiple tribes** without performance degradation.
* The architecture should support **horizontal scaling** if additional tribes or teams require tracking.

#### **7. Compliance**

* The system must comply with **Swisscom's internal IT policies** regarding **data handling and retention.**
* must include **automatic retry mechanisms** to recover from transient failures