SRS open day feature

**Table of Contents**

1. Introduction 1

1.1 Motivation 1

1.2 Scope 1

1.3 Definitions, and Abbreviations. 2

2. The Overall Description 2

2.1 Product Overview and Functionalities 2

2.3 Constraints and Assumptions 3

3. System Requirements 5

3.1 External Interface Requirements 5

3.2 Functional Requirements 6

3.3 Use Case Diagram 8

3.4 Non-Functional Requirements 8

4. User Interface Design 10

Appendix A 11

Appendix B 12

# 1. Introduction

This Software Requirements Specification (SRS) document outlines the scope, objectives, and functional requirements for a new "Open Days Tracking" feature within TravelMate, an internal Swisscom application for managing business travel days.

## 1.1 Motivation

TravelMate facilitates employees working from international offices for up to 10 days annually. To maintain legal compliance, each unit has a 180-day yearly limit, managed by a reviewer. The primary motivation for this new feature stems from a critical gap in the current system: TravelMate lacks a mechanism to track whether these allocated "Open Days" are utilized.

This absence of tracking leads to several significant problems:

* **Lack of visibility:** Administrators and reviewers cannot see if Open Days were used, hindering their ability to assess travel patterns and utilization.
* **Resource misallocation:** Unused travel days remain untracked and cannot be efficiently reallocated, leading to potential waste.
* **Limited data-driven decision-making:** Without usage data, optimizing travel day allocation based on actual needs is impossible.

The "Open Days Tracking" feature aims to resolve these issues by introducing a system to monitor Open Day usage. This will enhance transparency, efficiency, and planning within TravelMate. It will provide administrators with the necessary insights to view usage and redistribute unused days and optimize overall travel day management. While travel bookings remain in the existing system, this feature ensures reviewers gain clear visibility into these bookings and their impact on Open Day usage.

The intended audience for this document includes Product Owners, Developers, Testers, and System Administrators.

## 1.2 Scope

This feature introduces tracking and reporting for Open Days, allowing administrators to see if a scheduled day was used. It will provide the ability to reallocate unused days, thereby improving efficiency and enabling data-driven insights.

**Functionalities Include:**

* Tracking the usage status of Open Days.
* Reporting on utilized and unused travel days.
* Functionality to reallocate unused days.

**Out of Scope:**

* Automated rescheduling or reassignment of days.
* Enforcement of travel compliance policies.
* Forecasting future travel needs.

The feature will integrate into TravelMate, respecting existing functionalities and data privacy regulations.

## 1.3 Definitions, and Abbreviations.

#### **Key Definitions**

* **Open Day:** A designated travel day in TravelMate, allocated at the unit level, with statuses: Used, Unused, or Reallocated.
* **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. The Overall Description

This section provides a high-level view of the new "Open Days Tracking" feature, explaining its purpose, core functionalities, and how it integrates with the existing TravelMate system. It builds upon the motivation and scope detailed in the Introduction to describe the envisioned product.

## 2.1 Product Overview and Functionalities

#### Product Overview

#### The "Open Days Tracking" feature is an extension, not a standalone product, designed to be fully integrated into the existing TravelMate system. It specifically enhances TravelMate's ability to manage the 180 travel days allocated per unit. By providing visibility into whether opened days are used, it addresses the current system's lack of oversight. Implementing this feature requires modifications to the TravelMate database, user interface, and business logic

#### 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:

* **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.
* **Testing**
  + The system should implement tests for the new and old business logic, bugs are introduces as little as possible.

## 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.**
* **Dependency on External System (WFIDB)**  
  The feature will need to interact with an external system called **WFIDB**. This dependency may introduce **constraints related to API availability, rate limits, response times, authentication, and data synchronization.** Any limitations or changes in the WFIDB 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.
* **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 WFIDB**

The Open Days Tracking feature in TravelMate relies on WFIDB to retrieve employee data. WFIDB provides a list of employees along with additional details such as department, role, and employment status.

**Key Requirements**

* **Data retrieval and matching:** TravelMate must fetch employee data from WFIDB and match it with its internal records to determine Open Day usage.
* **Data unavailability and error handling:**
  + If WFIDB 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:**
  + WFIDB 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 WFIDB 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.  
**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.  
  **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 WFIDB**

**Input:** Request to WFIDB API to retrieve employee details.  
**Processing:** The system calls WFIDB’s API, retrieves employee data, and matches it with TravelMate records.  
**Output:** List of employees eligible for travel.  
**Error Handling:**

* If WFIDB is unresponsive, the system retries after a delay.
* When WFIDB is down, or returns empty values, logs must be made.

#### **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.  
  **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.  
**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.  
**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 WFIDB, and administrator modifications.  
**Processing:**

* The system logs actions such as:
  + Open Day status changes (e.g., used, unused, reallocated).
  + API calls to WFIDB, 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.

**8. Display who is traveling on which day**

* **Input:** Admin clicks om a date and sees who is traveling on that day. This will allow them to make subtle decisions in the way the allocate, if only 1 person is flying on a specific day and HR needs an extra day, they can ask the person to reschedule.
* **Ouput**: Show the admin a list of travelers.
* **Error Handling**: Show error to the admin and leave the travelers list empty.

**9. Implement edit functionality in the admin dashboard**

* **Input:** Admin presses on the edit button. This is needed so we can implement requirement 8.
* **Ouput**: Admin can edit the days and save / cancel.
* **Error Handling**: Show error to the admin and leave the travelers list empty.

## 3.3 Use Case Diagram

A diagram of a diagram

AI-generated content may be incorrect.This use case diagram shows the open days tracking. To get a better view, the picture is stored in the Analysis directory of my graduation folder.

Figure 1: Use case diagram of the open days tracking feature

## 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 unit).
* 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 units** without performance degradation.
* The architecture should support **horizontal scaling** if additional units or teams require tracking.

#### **7. Compliance**

* The system must comply with **Swisscom's internal IT policies** regarding **data handling and retention.**

# 4. User Interface Design

The Open Days Tracking feature is presented as a calendar-based dashboard that supports two modes:

* **Edit Mode**:  
  Allows administrators to manage contingent days. A checkbox labeled *“Show reallocation options”* reveals which days are eligible for reallocation.  
  While there is currently no visual feedback (e.g., colored highlights), the design anticipates these for future implementations. See Appendix A for a visual design.
* **View Mode**:  
  Clicking on a date shows which employees traveled on that day. This helps admins make informed decisions about reallocations. See Appendix B for a visual design.

This design is discussed and created together with the UX designer.

**Note:**  
The designs include color-coded indicators (e.g., orange for reallocation eligibility), but these are not yet implemented and are not in scope for this SRS. However, the UI is structured to support these features in future releases with minimal rework.

The UI uses Swisscom’s internal component library and follows internal design standards for consistency, usability, and accessibility.

## Appendix A

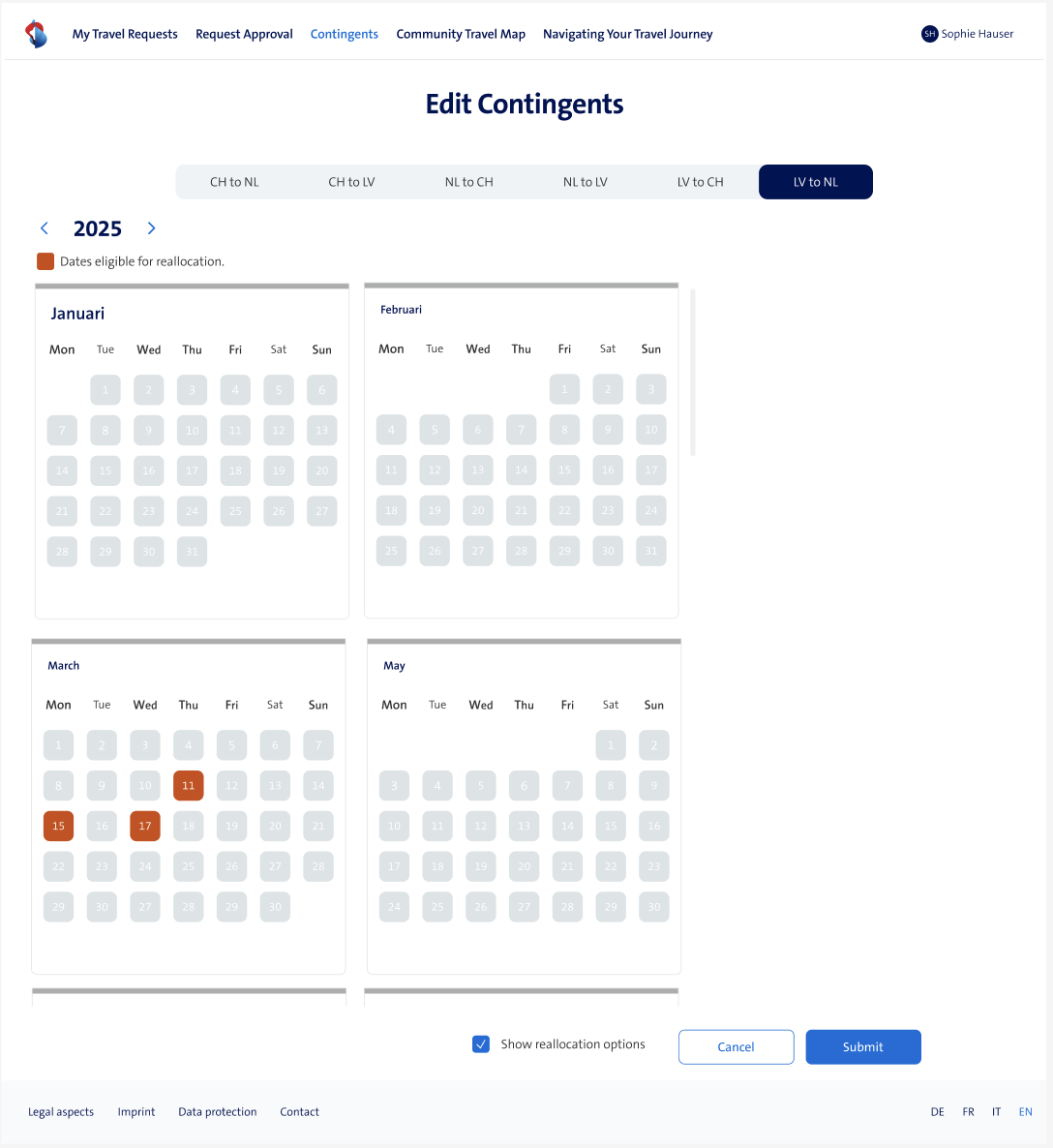
****This appendix shows the design of the edit mode in the dashboard with my feature enabled.

Figure 2: Edit page with Show reallocation options enabled, showing my feature

## Appendix B

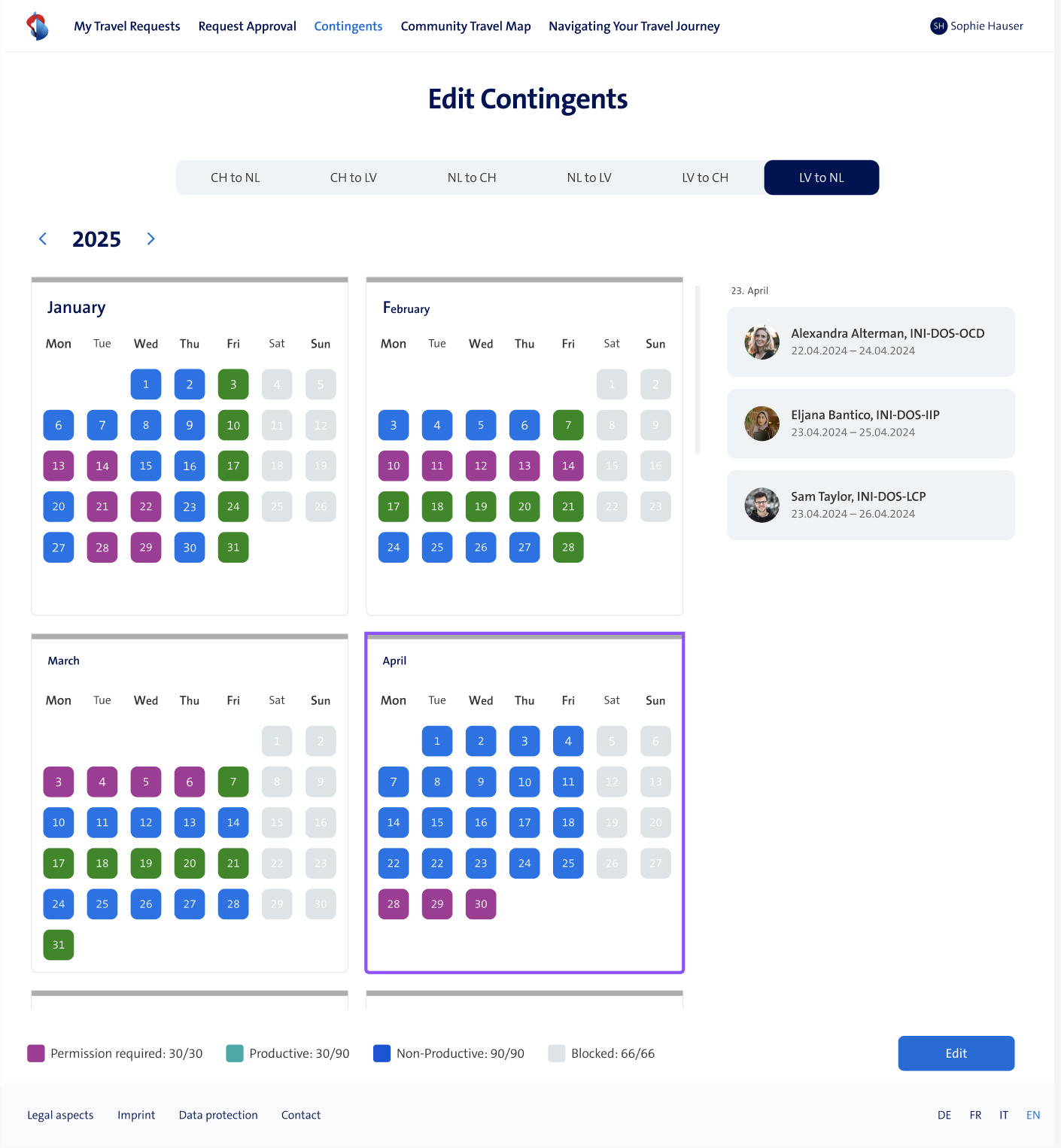
This appendix shows the design of the view page in the dashboard, with the new feature where you can select a date and see who is traveling there.

Figure 3: View page with a selected date, showing the people who are traveling on the specific day