POLITECNICO
MILANO 1863

094214 - Software Engineering (for Automation)

Design Document

Student Names:

Kien Ninh Do
Julie Ronesen Landaas
Ole Mandius Harm Thorrud

June 2, 2025
Version 1

Contents

1 Introduction

1.1 Purpose

Planning a vacation for a large group can be overwhelming, with numerous suggestions to consider, expenses to track, and logistics to coordinate. As we have encountered this problem several times ourselves, we want to find a solution that gathers a lot of necessary and useful functionalities, creating an efficient platform for all participants of the vacation.

2 Architectural Design

2.1 Structural view

2.1.1 Full-stack system

The structural design of the full-stack system is shown in Figure 1. The figure shows all components of the full-stack system and the dependencies between them.

A lower level class diagram of the backend is shown in section 2.1.2. More detailed information about the frontend pages is given in section 3.

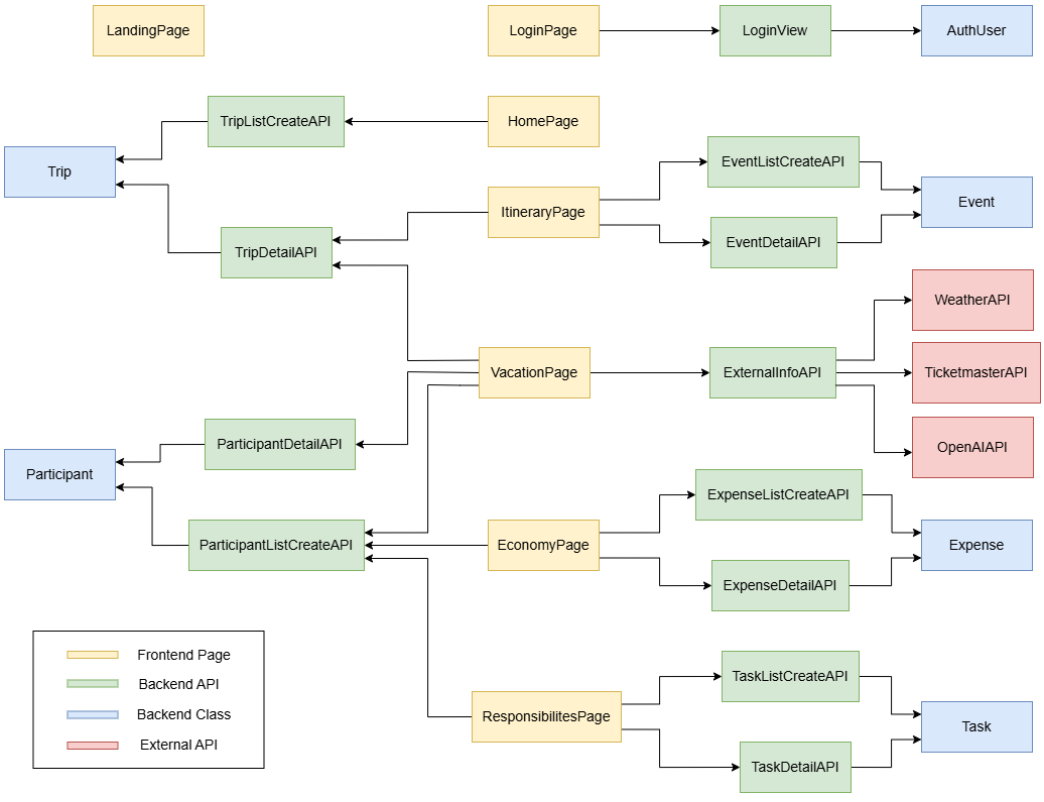


Figure 1: High-level component diagram of the full-stack system

2.1.2 Backend

The backend consists of the following classes:

- **User:** Contains user information and allows users to log in and out.
- **Participant:** A participant is represented by a user profile and is participating in a trip. Only participants in a trip are allowed to manage that trip.
- **Trip:** Contains all the information about a trip. Allows participants to manage a trip by adding and removing participants, events and tasks.
- **Event:** Contains information on events that are planned/suggested for the trip. This can, for instance, be activities or restaurants.
- **Task:** Contains information about the tasks that needs to be performed. Allows participants to add a responsible for the task and change status of the task.
- **TaskStatusType:** Status that can be set on tasks. Can be requested, in progress or done.
- **Expense:** Contains information about expenses incurred during the trip.

The structural design of the backend is shown in Figure 2.

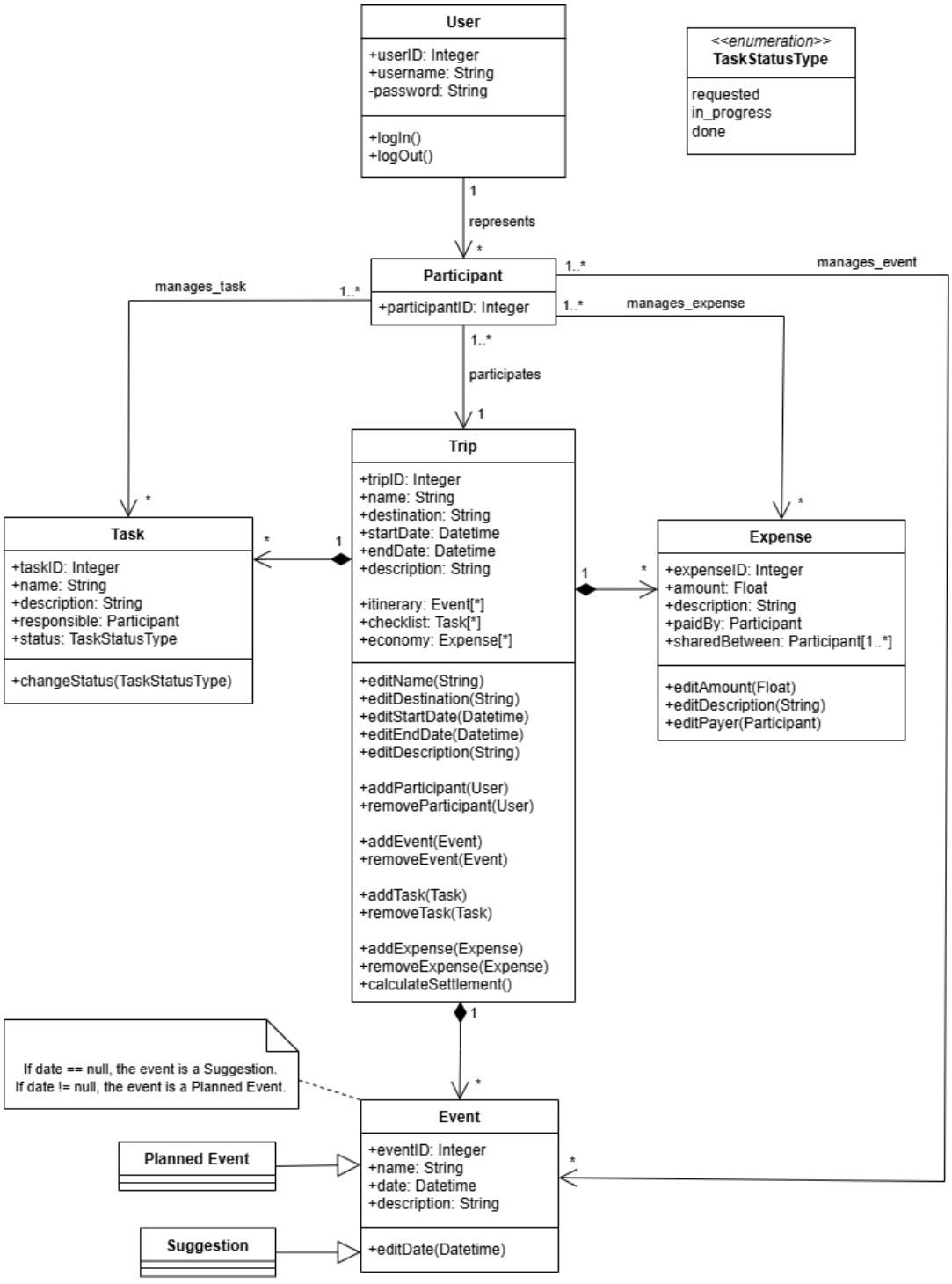


Figure 2: Class diagram of the backend

2.2 Runtime view

The following sequence diagrams are based on the scenarios that have been created in the *Requirement analysis*.

2.2.1 Check the weather

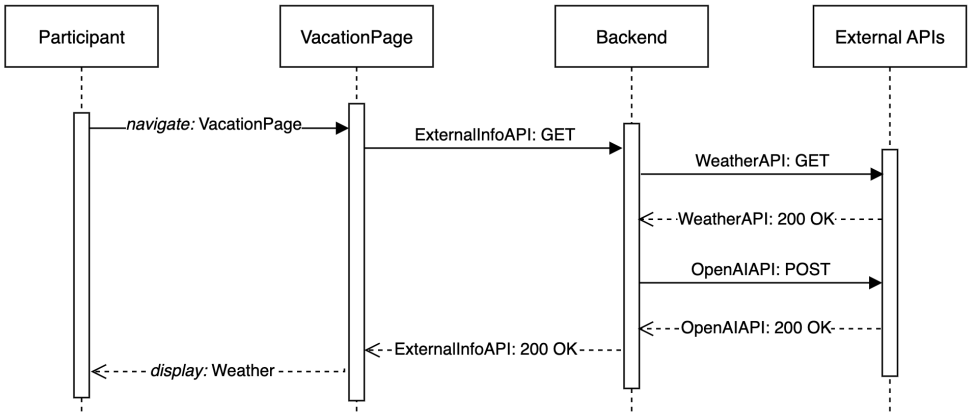


Figure 3: Check the weather through the Vacation-page.

2.2.2 Suggest and add to itinerary

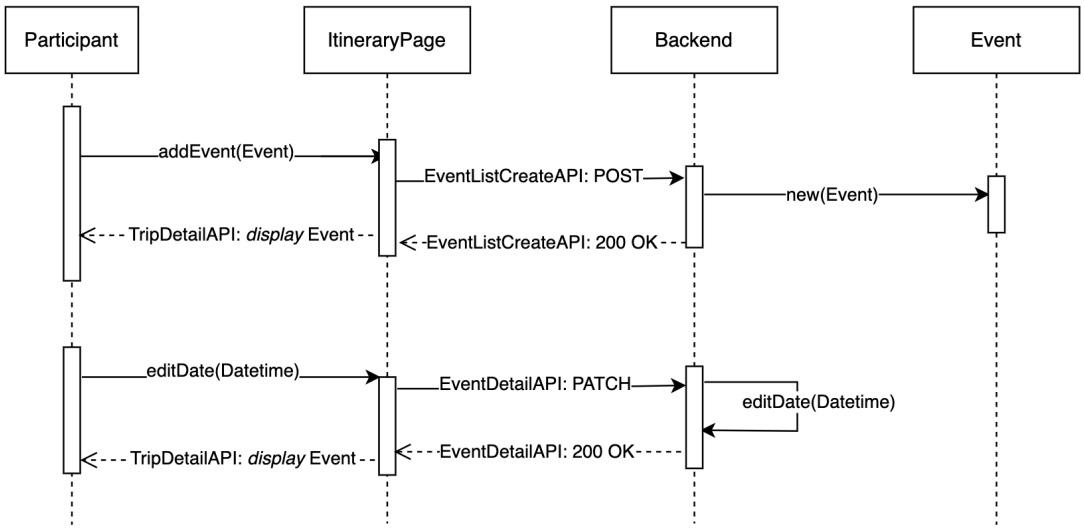


Figure 4: Suggest and add to itinerary

2.2.3 Assignment of task and mark as done

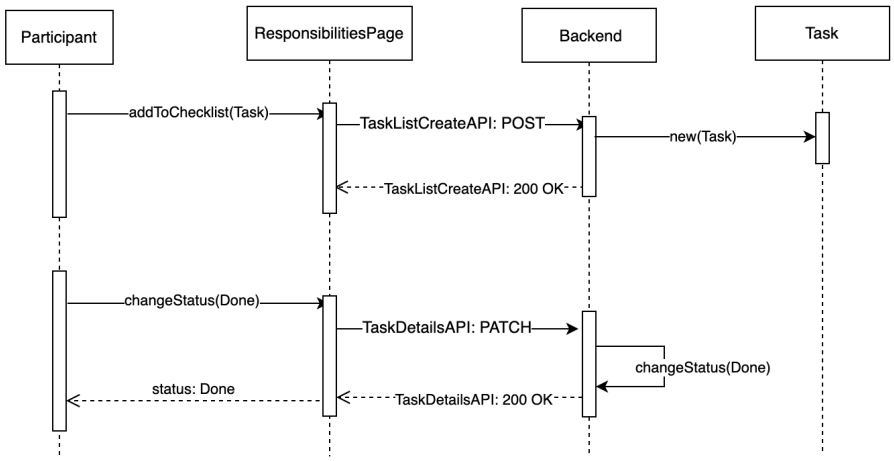


Figure 5: Assignment of task and mark as done

2.2.4 Add expense to settlement and close it

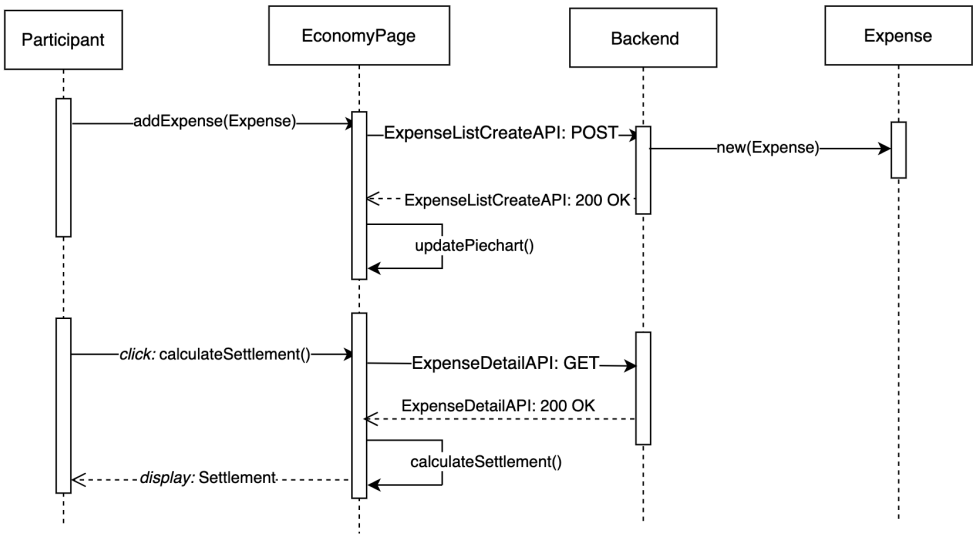


Figure 6: Add expense to settlement and close it

3 User interface design

As shown in Figure 7, the user interface enables users to switch between multiple pages, each corresponding to a distinct function of the application.

a) The first page is the **landing page**, which appears when visiting the website. It includes simple functionality such as a "Get started" button to begin navigating the platform.

b) The second page is the **login page**, where users can log in using a traditional username and password combination.

c) The third page is the **main home page**, which displays upcoming vacations and those planned for the future. It includes buttons for logging out, a task overview, and a view of all previously visited destinations. Clicking on the name of a specific vacation takes the user to its detailed home page.

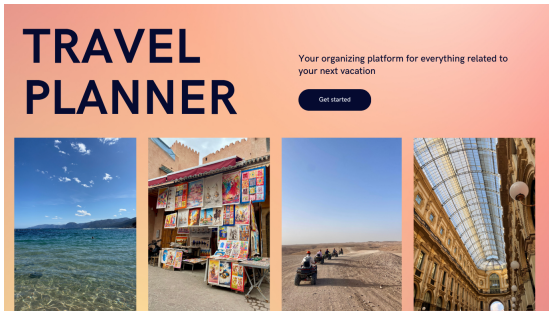
d) The **vacation-specific home page** offers all key functionalities including itinerary, economy, and responsibilities. Users can also get an update on the weather, happenings in the city through Ticketmaster.com, add participants and review own expenses.

e) The **itinerary page** functions like a calendar, where users can fill routes with activities using drag-and-drop from the suggestions section. By clicking on "+ Add suggestion," users can suggest activities in a brainstorming style.

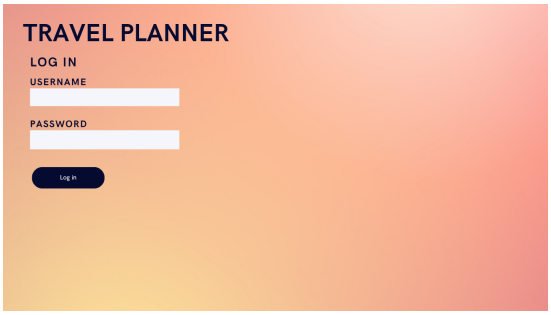
f) The **economy page** displays total expenses and allows participants to log payments into a settlement system. It dynamically updates a pie chart and shows how much each person owes. By clicking the "Close settlement" button, the system calculates the final settlement, showing each participant what they owe to others.

g) The **responsibilities page** features a drag-and-drop interface where tasks can be moved across different statuses: *Requested*, *In progress*, and *Done*, each indicated by a different color.

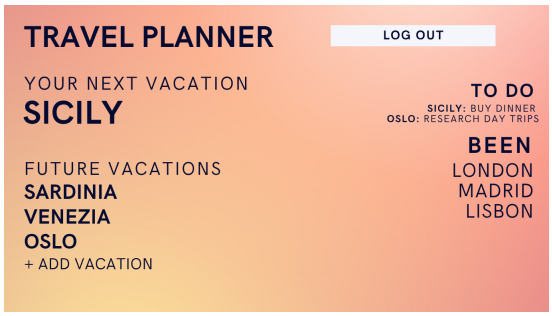
h) The final image shows an example of a pop-up window, such as the one that appears when clicking "Add expense" on the economy page.



(a) Landing page

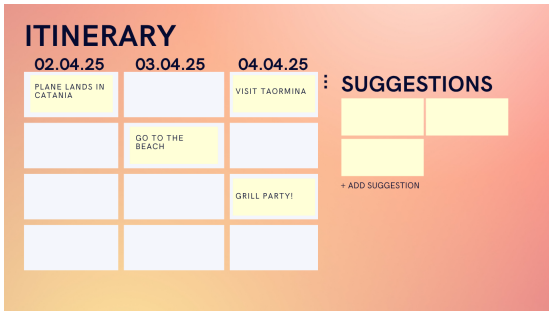


(b) Log in



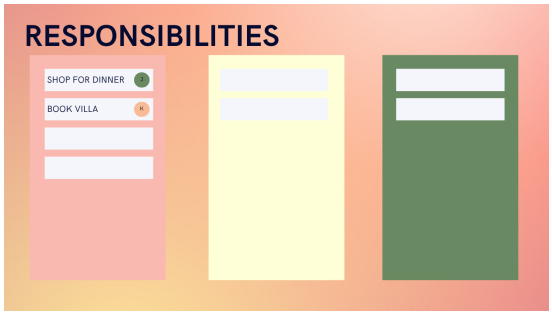
(c) Home page

(d) Home page of specific vacation

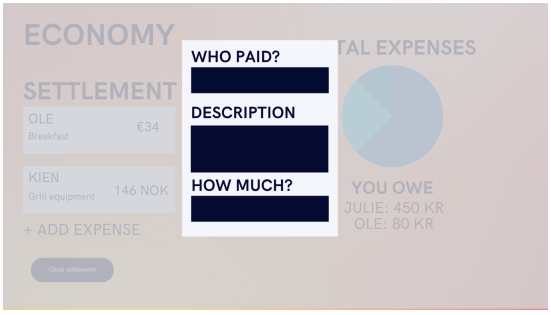


(e) Itinerary

(f) Economy



(g) Responsibilities



(h) Example of popup for expenses

Figure 7: User interface for the Travel Planner project.

4 Requirements traceability

Trip Planning Dashboard

- *The system shall allow users to manage trips with a shared itinerary:* This can be seen in the structural design, where multiple User instances can participate in the same Trip instance. In the UI-design, Users can add new vacations on the home page and add more participants later.
- *The system shall display the destination, travel dates, itinerary, to-do-list and economy of a trip:* This can be seen in the UI-design. The home page of a specific vacation shows the destination and travel date, and the other displays can be found on the respective pages.

Group Collaboration

- *The system shall allow users to invite other members to a trip:* This can be seen in the structural design, where Participants can add other participants to a Trip instance. In the UI-design, this is easily accessible on the home page of a vacation.
- *The system shall allow users to assign responsibilities for events of a trip:* This can be seen in the structural design, where Participants can add Task instances and can assign responsibilities for each Task. This can be assigned when adding a new task on the Responsibilities page.

Suggestion System

- *The system shall allow members of a trip to recommend activities, restaurants, and attractions:* This can be seen in the structural design, where Event instances can be added to a Trip instance. The events are distinguished between suggestions and planned events in the UI-design, depending on whether they are in the suggestion area or in the calendar.
- *The system shall make it easy to add and delete events and edit the calendar:* This can be seen in the UI-design, where the events can be added by clicking the "+ Add Event" button and input the event details, they can be deleted by the click of a button, and an easy drag-and-drop feature is used to switch between suggestion and the dates of the calendar.
- *The system shall fetch data from external APIs to display information that can be useful for planning an itinerary:* This can be seen in the component diagram, where the Vacation page uses information from external APIs to display the weather and a list of events happening at the destination.

Expense Tracking

- *The system shall allow members of a trip to log expenses and who paid:* This can be seen in the structural design, where Participants can add Expense instances to a Trip instance. In the UI-design, expenses can be added in the Economy page, and all expenses details are displayed on the page.
- *The system shall automatically calculate balances and debts based on the logged expenses:* This can be seen in the structural design, where the Trip instance has a method for calculating the settlement. On the Economy page, this can be easily done by clicking a button.
- *The system shall display the economy in a clear way, preferably using diagrams:* This can be seen in the UI-design on the Economy page, the economy is displayed in a pie chart to increase readability. In addition, the details of all expenses are displayed.

5 Implementation, integration and test plan

The plan is to simultaneously implement the frontend and backend of the website. For the backend, we implement the database model, serializer and API one class at the time. For the frontend, we implement the design and functionalities of one page at a time. As each class is fully implemented in the backend, we can start using it in the frontend.

When it comes to the implementation order of the classes, we start with the most important and central classes, which are User, Participant, Trip and Event. Following that, we start by implementing the pages that first appear on the website and that uses the most central classes.

All components of the system will be implemented and extensively tested, ensuring that they work before starting with new components. When the basic components are implemented and working well, we will add more functionalities one by one, following the same approach. In the end, we will test whwther the website as a whole functions as planned and fulfills all requirements.

The backend classes, including database model and API, will be implemented in the following order:

1. User
2. Participant
3. Event
4. Task, together with TaskStatusType
5. Expense
6. External APIs

The frontend pages, including the design and belonging functionalities, will be implemented in the following order:

1. Landing page
2. Login page
3. Home page
4. Vacation page
5. Itinerary page
6. Responsibilities page
7. Economy page

5.1 Changes in the original proposal

In our original proposal, we included additional features such as a map displaying all the event locations and a memories page where participants could upload photos. As the project progressed, we decided to shift our focus toward the core functionalities of the platform like itinerary planning, task management, and expense tracking to ensure these features were well-implemented and stable. Since the map and memories page were not essential to the main planning experience, we chose to leave them out to better allocate our time and effort. We also initially considered including a currency converter, but after evaluating its relevance, we concluded that it wouldn't significantly improve the user experience and therefore opted not to implement it.

The idea of a user profile page was also part of our original plan. However, we realized that the travel history (both past and upcoming trips) is already displayed on the home page, and the profile

itself wouldn't offer much additional functionality. As a result, we decided it would be redundant to develop a separate profile page. Lastly, we planned to have a travel leader role with special permissions. In practice, though, we found it better to let all participants co-plan the trip equally. That way, everyone can contribute equally rather than limiting most functionality to a single person.

We have also included external APIs (openAI, Ticketmaster, WeatherAPI) to increase easiness and inspiration for planning a vacation. This has been implemented in the Vacation-page.