NTNU | Norwegian University of Science and Technology

Department of Computer Science

TDT4290 - Customer Driven Project

Attractive City

Authors:

Anmol Singh
Arnstein Øyhus
Bjørn Anders Flågen
Hanne Ødegård
Leonard Schøyen
Royce Lu
Sander Francis

Customer:

Heyloft

Supervisor:

Orges Cico

November 12th, 2021

1 Executive Summary

This report gives an overview of the development project of a prototype called "Digital Marketplace for Experiences" by team 10 – consisting of Sander, Royce, Leonard, Hanne, Bjørn, Arnstein, and Anmol. The customer for the project is Heyloft, and the prototype is developed based on their demands. The motivation for the prototype is to reduce the lack of experiences to book and attend in Trondheim, and hopefully help connect small-scale providers and their secret knowledge and love of places, to people everywhere. The prototype is expected to be functional in order to be tested and improved upon, also to help answer the learning goals from Heyloft's preliminary studies.

The team began early with the planning phase of the project, where team organization and project work were discussed. In addition, the team also planned weekly meetings with Heyloft to discuss and define the project scope and requirements. The different tools for development and administration were also decided based on previous knowledge and preferences by the team. A group contract was created to ensure that every team member was on the same page.

After the first meeting with Heyloft, it was clear that the team needed to adopt an agile methodology. Because of the idea of a self-organizing team and focus on value generation through adaptive solutions, the team chose to follow the Scrum framework with elements from Kanban and Extreme Programming (XP). The team roles were assigned according to Scrum with a Scrum master, product owner, and developers. The project work was also planned around the different Scrum events, which the team followed as close as possible. During development, new ideas and needs came up, and thus changed the requirements, which is in the nature of the agile approach. Prior to every new sprint, the customer prioritized requirements using the MoSCoW method. During the sprint planning meetings, the team would refine and add details to the chosen user stories, including estimating the amount of effort needed to complete the tasks. At the end of sprints, in the sprint review, the team demonstrated the current increment and received feedback from Heyloft. The team rounded off the sprints with sprint retrospective meetings to discuss how things went, and what to improve in the next sprint.

To get a thorough overview of the project, different software architecture views were created. The team wanted to use an architecture to introduce terminology and facilitate communication and discussion about the design of the system. True to agile philosophy, additions or changes to the architecture was carried out closer to when the feature was implemented because the context and requirements surrounding the feature are better known.

In the end, the prototype became a responsive web-application, which the team created to help answer the learning goals from Heyloft. The customer was satisfied with the results and the team had a steep learning curve through the entire development process.

The prototype can be accessed here: https://attractive-city.web.app/. The source code can be accessed from https://tinyurl.com/9uusyrth and the group presentation can be accessed from https://tinyurl.com/5w2c5ehs.

Table of Contents

1 Executive Summary **i**

2 Introduction **1**

2.1 The Team 1

2.2 Heyloft – The Customer 1

2.3 The Project 1

3 Planning **2**

3.1 Project Schedule 2

3.2 Team Organization 2

3.3 Project Work Organization 3

3.4 Tools and Infrastructure 3

3.4.1 Collaborative and Administrative Tools 5

3.4.2 Development Tools 5

3.4.3 Application Framework and Web Server 5

3.5 Quality Assurance (QA) 6

3.6 Risk Management 6

3.7 Registration of Working Hours 6

4 Preliminary Study **7**

4.1 Heyloft's Vision 7

4.2 Solutions of Today 8

4.2.1 Tripadvisor 8

4.2.2 Expedia 9

4.2.3 Visit Norway 9

4.3 Pain Points 9

4.4 Possible Solutions 10

4.5 Choice of Solution 10

5 Development Methodology **11**

5.1 Agile Methodology 11

5.2 Scrum 12

5.2.1 Roles 12

5.2.2 Definition of "Done" 12

5.2.3 Scrum Events 12

5.2.3.1 Sprint Planning Meeting . . . 13
5.2.3.2 Daily Scrum . . . 13
5.2.3.3 Sprint Review . . . 13
5.2.3.4 Sprint Retrospective . . . 14
5.3 Kanban . . . 14
5.4 Extreme Programming (XP) . . . 14
5.4.1 Values . . . 14
5.4.2 Sit Together . . . 15
5.4.3 Pair Programming . . . 15

6 Requirements **15**
6.1 User Stories . . . 16
6.1.1 Changes to the Product Backlog . . . 16
6.2 Non-functional Requirements . . . 17
6.2.1 Modifiability . . . 18
6.2.2 Usability . . . 18
6.2.3 Availability . . . 18

7 Architecture **19**
7.1 Architectural Methodology . . . 19
7.2 Architectural Drivers . . . 19
7.2.1 Functional Drivers . . . 19
7.2.2 Quality Attributes . . . 20
7.3 Architectural Patterns and Tactics . . . 20
7.3.1 Modifiability . . . 20
7.3.2 Usability & Availability . . . 20
7.3.3 Additional Architectural Patterns . . . 20
7.4 View List . . . 21
7.4.1 Use-case View . . . 21
7.4.2 Development View . . . 22
7.4.3 Physical View . . . 23
7.4.4 Process View . . . 24
7.4.5 Logic View . . . 25
7.5 Architectural Reasoning . . . 26

8 Implementation/Sprints **26**

8.1 Sprint 1 . . . 27
8.1.1 Sprint Planning . . . 27
8.1.2 Increment . . . 28
8.1.3 Sprint Review . . . 29
8.1.4 Sprint Retrospective . . . 30
8.2 Sprint 2 . . . 31
8.2.1 Sprint Planning . . . 31
8.2.2 Increment . . . 32
8.2.3 Sprint Review . . . 33
8.2.4 Sprint Retrospective . . . 34
8.3 Sprint 3 . . . 35
8.3.1 Sprint Planning . . . 35
8.3.2 Increment . . . 35
8.3.3 Sprint Review . . . 36
8.3.4 Sprint Retrospective . . . 37
8.4 Sprint 4 . . . 39
8.4.1 Sprint Planning . . . 39
8.4.2 Increment . . . 40
8.4.3 Sprint Review . . . 40
8.4.4 Sprint Retrospective . . . 41
8.5 Sprint 5 . . . 43
8.5.1 Sprint Planning . . . 43
8.5.2 Increment . . . 44
8.5.3 Sprint Review . . . 44
8.5.4 Sprint Retrospective . . . 45

9 Security **46**
9.1 Threat Model . . . 46
9.1.1 Misuse-case . . . 47
9.1.2 Security Threats . . . 48
9.1.3 Security Vulnerabilities . . . 48

10 Testing **50**
10.1 Test Plan and Routines . . . 50
10.2 Integration Testing . . . 50
10.3 User Testing and Feedback from Customer . . . 50

11 Internal and External Documentation **51**

12 Evaluation **51**

12.1 The Internal Process and Results . . . 51

12.2 The Customer and the Project Task . . . 52

12.3 The Supervisors . . . 52

12.4 Further Work with the Project . . . 52

12.5 Suggestions for Improvement in the Course TDT4290 . . . 53

13 Reflection about the Lectures **53**

13.1 Project Planning and Management . . . 54

13.2 Group Dynamics . . . 54

13.3 Agile Development . . . 54

13.4 Presentation Skills . . . 54

13.5 Technical Writing . . . 54

13.6 IT Security . . . 55

Bibliography **56**

Appendix **58**

A Documentation . . . 58

A.1 Frontend . . . 58

A.1.1 Prerequisites . . . 58

A.1.2 Structure . . . 58

A.1.3 Testing . . . 58

A.1.4 Setup . . . 58

A.1.5 Execution . . . 58

A.1.6 Deploy website . . . 58

A.2 Backend . . . 59

A.2.1 Prerequisites . . . 59

A.2.2 Contents . . . 59

A.2.3 Deploy back-end . . . 59

A.2.4 Develop and debug locally . . . 59

A.2.5 Use Cloud SQL Auth proxy . . . 60

A.2.6 Testing . . . 60

A.3 Database . . . 60

A.3.1 Experiences Google Sheet . . . 60

A.3.2 Experiences Sheet Script 61
A.4 AI 61
A.4.1 Automatic setup 61
A.4.2 Manual setup 61
A.4.3 Update requirements.txt 62
B User journey map 62
C Product Backlog Increments 63
C.1 Product Backlog Before Sprint 1 63
C.2 Product Backlog Before Sprint 2 64
C.3 Product Backlog Before Sprint 3 65
C.4 Product Backlog Before Sprint 4 66
C.5 Product Backlog Before Sprint 5 67
C.6 Product Backlog After Sprint 5 68
D Changes to product backlog 69
D.1 User stories added to the product backlog 69
D.2 User stories removed from the product backlog 69
E Sprint backlogs 70
E.1 Sprint backlog 1 70
E.2 Sprint backlog 2 71
E.3 Sprint backlog 3 72
E.4 Sprint backlog 4 73
E.5 Sprint backlog 5 74
F MoSCoW Prioritization Example 75
G Group contract 76
H Feature List 79
I Project plan 80
J Working hours registration 82
K User testing feedback 84
L Burndown Charts 85
L.1 The Burndown Chart from Sprint 2 85
L.2 The Burndown Chart from Sprint 3 85
L.3 The Burndown Chart from Sprint 4 86
L.4 The Burndown Chart from Sprint 5 86

2 Introduction

This project report documents group 10's process during the course TDT4290 Customer-Driven Project in autumn 2021 at NTNU. In the course, students are divided into teams and assigned a customer. The goal of the course TDT4290 is for the teams to learn software engineering skills in the context of a development project to make a realistic prototype of a software-based project "on contract" for a real-world customer, also known as problem-based learning [32]. The team's project involves creating a digital marketplace for experiences in Trondheim.

2.1 The Team

The team consists of seven students from NTNU, pursuing a master's degree in Computer Science, where all team members were involved in developing the prototype. In addition to the team role as a (software) developer, some had more responsibilities as well (section 3.2). The team's main motivation for the project is to get practical experiences related to software engineering and project management through realistic scenarios. Since the team will be putting high work effort into the project, it is also expected that the prototype will provide value for the customer and the (end)users.

2.2 Heyloft – The Customer

The real customer the team has been working with is **Heyloft** – an international eco-tourism startup, which aims to revolutionize travelling and local experiences [12]. Their promise is *a realm of calm excitement. An introduction to booking activities organized by a new community of people, each one eager to share their secret adventures.* Currently, they are partnered up with Visit Trondheim to collaborate around a new digital tool for Trondheim.

Heyloft's motivation is to be "the friend that knows all the cool spots and hooks you up" [12], and expects the team to develop a functional prototype that can be used to validate and communicate the solution concepts they have been working on in their preliminary studies (section 4).

2.3 The Project

One of the requirements in the course TDT4290 is to deliver a minimum viable product prototype. Prior to the course TDT4290, Heyloft has been doing pre-project work, including workshops with different stakeholders, which resulted in three prototypes. The team was assigned to the "Digital Marketplace of Experiences" prototype, where one of the goals is: "How can we make it easy for visitors and providers to plan, book, and attend activities?". Since the project was still quite vaguely defined, the team had the opportunity to explore and suggest the possible features in the prototype. The project evolved as the team developed, mainly following the Scrum framework (the team's implementation is described in section 8).

After some months, the prototype resulted in a web-application with the main feature "Speed Dating of Experiences", inspired by the swiping experience famously used in the dating application Tinder. The idea is that a user can swipe – like, dislike or favourite – through different experiences that Trondheim has to offer, both alone or in a group context with other users.

The prototype can be accessed through this link: https://attractive-city.web.app/. The source code can be accessed from https://tinyurl.com/9uusyrth.

3 Planning

This section presents how the project was planned, including the project schedule, team organization, and which tools and infrastructure that have been used throughout the project. As the customer requested us to work with a design-thinking view, the project plan would naturally evolve during the project.

3.1 Project Schedule

During the planning phase of the project, the team divided the project into phases, consisting of a planning phase, five development sprints, and a final phase allocated to both the presentation and the report. A simplified timeline of the project can be seen in the Gantt Chart in Figure 1. A more detailed project schedule for this project can be found in appendix I.

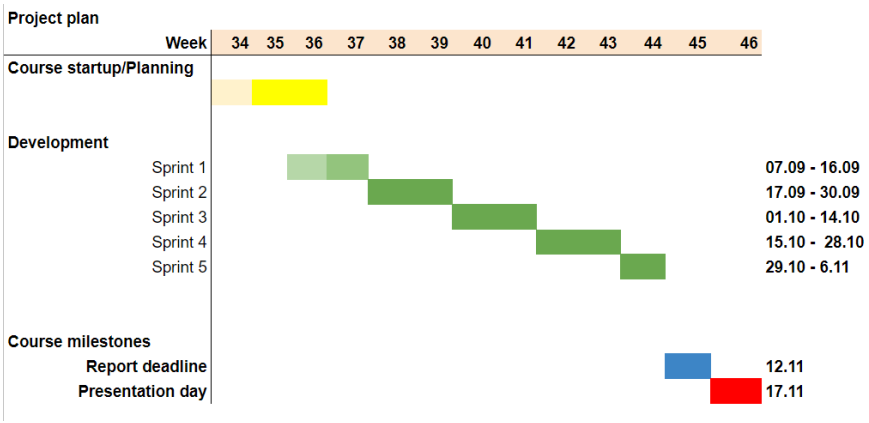


Figure 1: A Gantt Chart of the project overview.

The planning phase consisted of organizing the team, assigning roles to the team members, and meetings with Heyloft where the project scope was defined and requirements were discussed. Heyloft presented their visions for the requested prototype and challenged the team to come up with possible ideas that would fit their vision. The team came up with some ideas and proposed them as possible solutions to Heyloft, who reviewed the solutions and decided to go for the "Speed dating of Experiences" solution. The team also felt exited about this idea, as it would let users find experiences in a new context and way.

As soon as we had gotten the initial requirements from Heyloft, we began the first sprint. They wanted us to have as short sprints as possible to be able to iterate several times throughout the project. The team felt that a sprint length of less than two weeks could be too short, but decided to go for a 10-day sprint for the first sprint. The reason for having a shorter sprint was to test if it was possible to work at a velocity shorter than two weeks. A sprint length of ten days turned out to be too short and the sprint length was changed for the remaining sprints.

3.2 Team Organization

To make the work process efficient and functional, the team assigned members to different roles. As the team adapted Scrum, the team needed a Product Owner, a Scrum Master, and developers [25]. In addition, the team also assigned a Team Leader and a Quality Manager to distribute responsibility further. The team found it useful to delegate responsibility with roles. For instance, the Scrum Master took his role seriously and led the Scrum activities by always having a clear and structured agenda. Each team member was also working as developers during the project. Early in the project, the developers split into traditional roles such as front-end and back-end-developers, but throughout the project, the developers changed their role to a more modern full-

stack-developer. The team also agreed that each developer should be responsible for writing tests for their work. The team roles and their related responsibilities can be seen in Table 1.

Table 1: Team roles and responsibilities

Role	**Assignee**	**Responsibilities**
Product Owner (PO)	Anmol	Making sure that all requirements by the customer are fulfilled. Ensuring that the backlog is user-centered.
Scrum Master (SM)	Sander	Ensure that Scrum is followed. Create the sprint backlog and check that it is updated. Lead the scrum activities.
Team Leader (TL)	Hanne	Attend team leader meetings. Planning and coordinating team activities. Motivating team members.
Quality Manager (QM)	Bjørn Anders	Ensure the quality of the end product and the overall process. Check that project documents are consistent. Monitor and review all testing activities.
Developer	Anmol, Arnstein, Bjørn, Hanne, Leonard, Royce & Sander	Open dialog with TL and SM when problems appear. Execute plans made by the TL and the SM. Keeping track of time spent on various tasks.

3.3 Project Work Organization

In order to cooperate well and for the team spirit, the team decided to meet physically at NTNU Gløshaugen each Tuesday. The team decided to meet at 8:15 and begin the workday with a Daily Scrum meeting (section 5.2.3.2), before continuing working together on the project for the rest of the day. In addition, the team allotted time for the project each Friday from 8:15 to 12:00, also including a digital Daily Scrum meeting. For the rest of the week, each team member structured their time and effort for the project for themselves.

Naturally, this project included weekly meetings with Heyloft, and it was decided that the team should meet them each Thursday from 15:00 to 16:00. Since Heyloft has a Canadian contingent, these meetings were mainly digital, but sometimes also physical. The topic of these meetings included sprint reviews, approach discussions, and exchanging updates on the workflow. In addition, the team had weekly supervisor meetings on Tuesdays from 13:00 to 14:00, where the team discussed problems and was given guidance on the project and report. A snapshot from an example week containing the team's timetable can be seen in Figure 2. These were events repeating each week, with only a few exceptions. In addition, each team member structured their own work schedule themselves.

3.4 Tools and Infrastructure

Tools are a vital part of every project as they lay the foundation of collaboration and development between and by the people developing a project. Therefore, the team spent a major part of the planning phase discussing which administrative and development tools that were appropriate for this project. The team chose to adapt and use familiar tools and frameworks in order to achieve efficient collaboration as a team as well as a high-quality product. Additionally, it was important for the team that the tools were free and easily accessible. To keep the tool count low, the team decided that each tool should serve a unique purpose. An overview of the tools used in this project can be seen in Table 2, and will be further described in the following subsections.

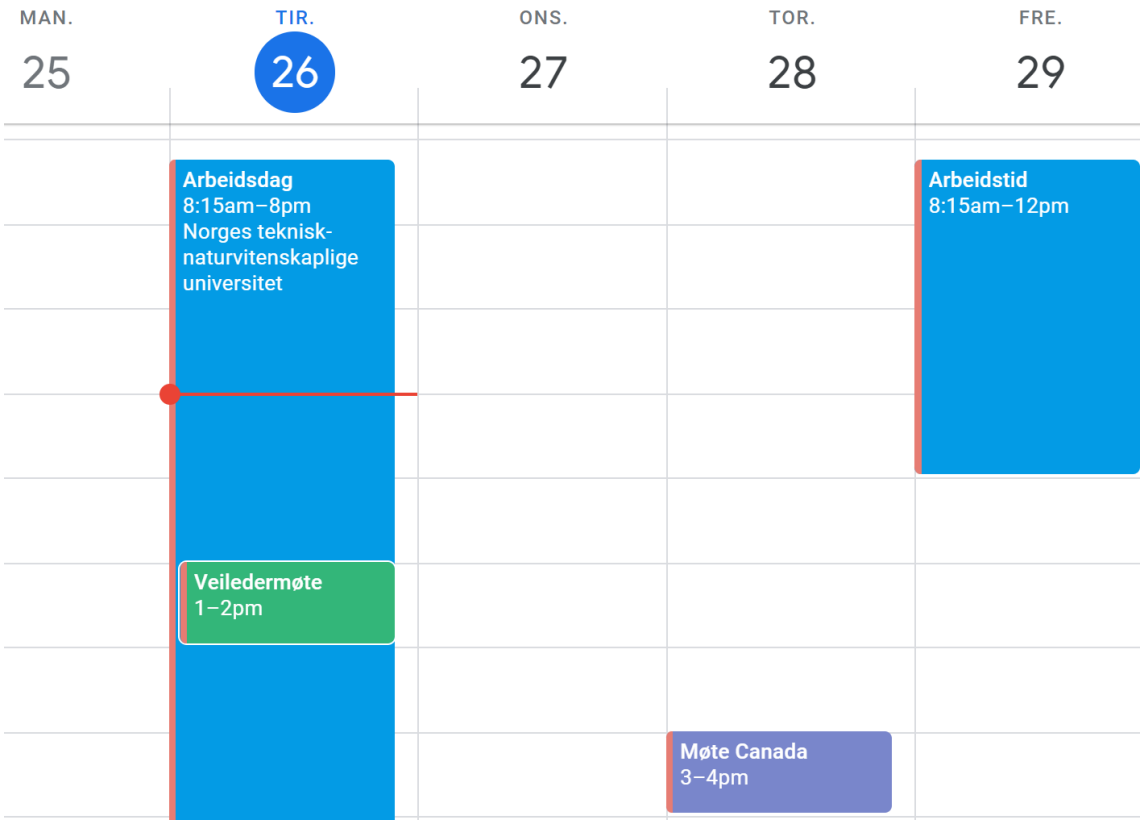


Figure 2: Overview of the team's official working hours and meetings during a typical week. Snapshot taken for week 43.

Table 2: Overview of tools used in this project.

Type	**Tools**
Collaborative and Administrative	
Communication	Slack, Facebook Messenger
Coordination	Google Calendar, GitHub Project Boards
Cooperation	Google Meet, Zoom
Common workspace	Google Drive
Text and multimedia editor	Google Docs / Sheet
Development	
Code repository	GitHub
Framework and Server	
Application Framework	React Typescript, Node.js Typescript, Python
Webserver	Google Cloud, Google Firebase
Database	Google Cloud SQL

3.4.1 Collaborative and Administrative Tools

The customer requested that we adapted Slack for communication between the team and Heyloft, which served the purpose of exchanging messages, documents, and planning meetings. The team also created an internal Slack workspace for communication, which was used for progress updates, questions, and discussions. In addition, as Slack is not the fastest communication channel, the team felt it necessary to have a Facebook-Messenger group for distributing urgent messages and important notifications quickly. To coordinate activities, meetings, and deliverables, the team used a combination of Google Calendar and GitHub Project boards.

Since parts of the customer company were based in Canada, there was a need for digital meetings between Heyloft and the team. The customer requested to use Google Meet for these meetings, which held the advantage of being browser-based with no need for installing any program as well as being easy to use. For these qualities, the team chose to use Google Meet for the internal meetings as well. In addition, the weekly meetings with the supervisor sometimes needed to be conducted digitally, using Zoom.

As the project evolved, it naturally generated a number of files like meeting minutes, scrum artifacts, and other administrative documents. In order to organize and keep track of them in a structured way, the team decided to create a Google Drive folder for easy access to the team's documents and artifacts. By doing so, documents and files could easily be created in the browser with Google Docs and Sheets from different devices.

3.4.2 Development Tools

When developing software as a team, the process can easily get unclear and messy as the code evolves during a project. In order to keep track of changes and additions from each team member to the codebase, the team decided to use a versioning control system. Then, if a mistake was to be made during development, it would be easy to go back to old versions of the code to fix the issue. The customer requested that we used GitHub as our code repository, which would also serve the purpose of versioning control. The entire team was familiar with GitHub from before and agreed to use it.

In order to keep track of the teams' progression and the status of tasks during development, the team decided to use Kanban boards as they provide a good visual representation of how tasks are going. The team wanted to keep the Kanban boards close to the code repository, and therefore decided to use GitHub's implementation of Kanban boards, the Project boards. The team experienced that it was also easy to coordinate work by assigning the developers tasks on the Project boards.

3.4.3 Application Framework and Web Server

As the intention behind this project was to create functional prototypes for Heyloft and to explore different features related to finding experiences, the team was given the freedom to choose the technologies to be used in the project. The team wanted to use a modern technology stack to make the team members competence competitive in a rapid-changing market. It was also important that the developers were familiar with the chosen technologies to ensure quick development of the prototypes. Several team members already knew Typescript and React, and the team decided to use React Typescript for the front-end part of the application.

In order to keep consistency throughout the source code of the application, the team decided to use Typescript as the language for the back-end of the application as well. This lowered the difference between front-end and back-end and also allowed the developers to work in a full-stack manner as the project evolved. The team chose to use the Node.js framework as there was some competence in the team with Node.js. In addition, Python was chosen for implementing a recommendation system. This was done because there are several open libraries written in Python for data processing and analysis, making it suitable for creating a recommendation system.

When it comes to storing data, the team assessed the data that needed to be stored based on its properties. As there were several relations between the data, the team decided to use a relational database like SQL. Early in the project, the database was hosted at NTNU's MySQL servers to get a functional development environment up and running as soon as possible. For the team, this was beneficial as it allowed the team to begin development early. By the time the entire application was hosted online, the database was migrated to run on Google Cloud SQL.

To host the software, Heyloft and the team decided to use Google Cloud Services with Firebase, as it provided good documentation, easy hosting, and access to user authentication which was needed. In addition, there was a free trial period making it a favorable option for the customer as well. As Google Cloud Services also provides access to relational databases, the team naturally decided to use Google Cloud SQL as mentioned above.

3.5 Quality Assurance (QA)

To ensure the quality of both the process and the end product, the team created templates for project documents, standards for cooperation, and agreed on working routines. Standards for cooperation and working routines were formalized in a group contract, appendix G. We created the contract on 2. Sep and it outlines the expectations within the group. It also contains conflict handling, to ensure a pleasant experience if conflicts were to arise. Luckily, no conflicts occurred during the project. Some of the QA-measures, like project documents, were created in the initial phase of the project, and other measures like naming convention of files was added during the project.

Initially, the team decided to code with a feature branch workflow. In this way, the main-branch of the project was always operational. When implementing new features, developers branched out a new branch, implemented the new functionality, and wrote tests (section 10). To merge the new code into the main-branch, both old and new tests needed to pass. A pull request would then be created and checked by another developer. Feedback would then be given to the changes before merging the new code into main.

In addition, to increase the quality and consistency of the code, the team decided to use Prettier for code formatting. Prettier was configured to format code on "save" and ensured that the codebase was easily readable and maintainable.

3.6 Risk Management

Unexpected events are inevitable when working on a software project of a certain size. In order to limit the effects of such events, risk management is an essential component of the initial phases of a project. Risk management is a process of identification, assessment, and prioritization of risks [24]. The team followed this process to be aware of possible risks that might occur and to be prepared and ready to take action. During this process, possible risks were assessed according to the probability of happening and the severity of the effect (Low (L), Medium (M), High (H)), the consequences of the risk, the strategy to mitigate the risk, and who was responsible for resolving the risk. This discussion is summarized in Table 3.

3.7 Registration of Working Hours

To monitor the teams working effort throughout this project, the team members logged their working hours in a timesheet each week. The reason for this was to ensure that the team invested the needed effort and to keep track of the working hours put into this project. The registration schema as of two days into week 45 can be found in Appendix J.

Table 3: Risk management

Risk	**Consequence**	**Probability / Severity**	**Strategy**	**Responsible**
Conflicts within the team.	Reduced motivation and morale. Reduced throughput and delays.	L/M	Stated in the group contract, Appendix G.	Team Leader
Valid absence for a group member, illness, etc.	Delays/Reduced throughput	H/M	Everyone should understand the entire project so we can cover for each other if needed.	The team
Miscommunication within the team.	Delays	M/M	Stand-up meetings two times a week where synchronization happens.	Scrum Master
Miscommunication with the customer.	Delays. Reduced value of the final product.	M/H	Frequent meetings and open communication with the customer.	Product Owner
Changes to requirements.	Have to redo work.	M/L	The agile methodology embraces change. Requirements are expected to change and the team should adapt accordingly.	The team
Loss of data.	Possibly large delays, need to redo work.	L/H	All important information is stored in Google Drive. Source code is kept in GitHub, which provides continuous backups and version control.	The team

4 Preliminary Study

To get a good understanding of the problem at hand, a thorough preliminary study is important. In this section, we discuss Heyloft's vision, solutions of today, possible solutions, and the final choice of solution.

4.1 Heyloft's Vision

Heyloft wants to create a user-driven community and marketplace for connecting small-scale providers [12]. Heyloft's goal is to build an end-to-end digital platform that supports small-scale experience providers around the globe. Heyloft has partnered up with Visit Trondheim, which aim is to promote the Trondheim region as an attractive destination. Together, they will collaborate to develop a new digital tool for the city of Trondheim. Heyloft and Visit Trondheim have set the following goals for this collaboration:

- New and strengthened partnerships with actors in the industry
- Coordinate with partners in the digital transformation of the travel and tourism sector
- Create independent collaboration between actors
- Product development, marketing, and digitalization

In order to achieve these goals, Heyloft and Visit Trondheim have come up with three different prototypes: (1) Digital Marketplace of Experiences, (2) "The HUB" and (3) a storytelling platform. We, the team, were assigned to work on the Digital Marketplace of Experiences and were given the following responsibilities:

- Design prototypes that can help answer our prototype learning goals
- Build out prototypes to be tested and improved upon
- Collaborate with the design team to integrate user research into prototypes

Heyloft wanted to involve the team in the process of sharing and exploring new ideas related to the digital marketplace of experiences. Therefore, they did not have any specific business requirements or evaluation criteria specified for the final solution. They did however provide possible solution concepts (Table 4) and goals (Table 5) for the Digital Marketplace of Experiences. Some of the goals were formed as questions in which our prototype(s) should aim to answer.

Table 4: Solution concepts

ID	**Solution Concept**
SC1	Platform to allow passionate locals to offer "pop-up" tours(i.e uber of experiences)
SC2	Customizable itinerary of atypical activities for different amounts of time
SC3	Innovative digital tools for planning and attending experiences

Table 5: Prototype Goals

ID	**Prototype Goal**
PG1	How can we make it easy for visitors and providers to plan, book, and attend activities?
PG2	How can we reduce the frustration that people report from finding, planning, booking, and attending experiences in Trondheim?
PG3	How can we make a bookable system that has an intuitive and sophisticated way to support users?
PG4	What are the requirements for a booking system that is generic and can be used by small-scale and medium-scale providers with a varying type of services?
PG5	Researching what we need to create this type of system to be successful
PG6	Determine if there are existing APIs we can use

4.2 Solutions of Today

Today, there are many existing online marketplaces for experiences. The team investigated some of the competitors in order to get a better understanding of the problem at hand and to see how we could make the prototype unique as this was a strong desire from Heyloft. Heyloft has also identified pain points which are described in subsection 4.3.

4.2.1 Tripadvisor

Tripadvisor is one of the world's biggest travel guidance platforms [1]. When it comes to showcasing experiences, Tripadvisor has a section called "Things to Do". Here the user is shown different categories of experiences like "Walking Tours", "Art Galleries" and "Outdoor Activities". After selecting one of these categories, the user is shown the top experiences for that category based on user reviews. The experience itself has a short description and multiple subsections of information that can be valuable to the user. These subsections vary depending on the type of experience, some examples are: "Know before you go", "Cancellation policy", "Available languages", "Inclusions",

"Exclusions". Tripadvisor is built upon users providing feedback, therefore, user reviews are highly visible.

Advantages A wide selection of experiences. Some subsections of information about the experience are highly informative and in the form of bullet lists, which make them concise and easy for the user to read. Many experiences can be booked directly from Tripadvisor without the need for redirection.

Disadvantages The user is shown a lot of information, this makes the site appear almost chaotic and makes browsing for experiences more stressful than it has to be.

4.2.2 Expedia

Expedia is also one of the worlds biggest travel guidance platforms [33], and one of Tripadvisor's main competitors. Expedia has a section called "Things to do" where they showcase experiences to the user. After indicating what city the user wants to browse experiences in, the user is shown the top 10 experiences in this city. There are also various experience categories that the user can choose from. The experience itself has multiple information subsections. The first subsection the user is shown is called "overview". Here, Expedia lists practical information like "Features", "Cleaning and safety practices" (Covid-19 related), and a bullet list that contains short sentences describing the experience. More detailed information about the experience is available in the "About this activity" subsection. All in all, Expedia's way of showcasing experiences is quite similar to Tripadvisor. Expedia provides exclusively on-site booking.

Advantages The filters and various categories are easy to navigate and use. The experience information is well structured, which makes it easy for the user to read, even though there is a lot of information available.

Disadvantages Small selection of experiences, especially outside the big cities. The user has to set a start and end date while searching for experiences. The start and end date can not be more than two weeks apart, this imposes unnecessary constraints on the user.

4.2.3 Visit Norway

Visit Norway is a state-owned web portal whose goal is to promote Norway as a travel destination [2]. On the Visit Norway website, the user can filter experiences based on multiple categories such as city/area, age, level of difficulty, type of experience, and so on. The user can also filter based on quite an extensive range of subcategories. The experiences themselves are shown with a header picture, two subsections of information called "overview" and "details". "Overview" contains a general description of the experience, while "details" contains short bullet lists containing essential information. There is also a sidebar containing additional information, like opening hours and price range. Visit Norway is a web portal, therefore, the user has to be redirected to the experience's own website in order to make a booking.

Advantages: The user is not bombarded with extensive information, on the contrary, the information about the individual experiences is very concise and to the point. This makes the browsing experience as a whole more engaging compared to the other two competitors investigated.

Disadvantages: It is impractical for the user to filter experiences based on subcategories due to the extensive amount of options to choose from.

4.3 Pain Points

Through various workshops, Heyloft had identified a great number of pain points, which are defined as specific problems faced by current or prospective customers in the marketplace [21]. In this case, the specific problems were related to finding things to do in Trondheim. The team considered the following pain points to be the most relevant to the Digital Marketplace of Experiences:

- Spent time and energy contacting friends for tips
- Finding only activities they've done before or are uninterested in
- Not knowing where to look for fun activities that they would like
- Using google to find activities without success
- Not having happy butterflies in their stomach when they plan a holiday
- Difficult to be impulsive
- Being a big group of people that can not think about anything to do
- Searching on google maps for activities and only finding the typical popular ones

4.4 Possible Solutions

Heyloft encouraged swift prototyping. Consequently, the team decided to hold an internal meeting where the team members would present ideas and possible features for the Digital Marketplace of Experiences. This meeting was held on 1. Sep, 2021. The following day, the team held a presentation for Heyloft where we presented selected features from the internal meeting. After the meeting, the team sent Heyloft a PDF containing a list of all the features and their related user stories (Appendix H).

Artificial Intelligence

Heyloft wanted to incorporate artificial intelligence (AI) into the prototype and as such the team kept this in mind when looking into possible features for the Digital Marketplace of Experiences. Since the main idea of the prototype was to create a marketplace for experiences, the most obvious way to utilize AI was to create some kind of recommendation system. Two of the team members had AI as their main field of study, so they were given the responsibility of exploring how recommendation systems could be built.

During this exploration, they discovered that most recommendation systems relied on large quantities of data on user interactions. Especially if machine learning or other more advanced intelligent models were to be used [11] [18]. Heyloft did not have any data of sufficient scale, and we did not see the prototype being able to generate this data either. As a result, the goal of the exploration shifted focus. Instead of looking into specific recommendation systems and how they were built, we looked into the kind of data that would be beneficial in the future. After looking at several recommendation systems available on the internet, for example Airbnb's recommendation system [11], we understood that user preferences for different experiences was the most important data. The prototype would therefore need to store and track how a user interacted with experiences. We also noted that collaborative-filtering and content-based filtering formed the baseline for many more advanced models [6] [18]. These had the benefit of requiring less data and they were fairly simple to implement. So if we were to prepare for the future, it could be beneficial to implement collaborative-filtering and/or content-based filtering. It was also clear that utilizing AI would be of great business value if the prototype were to launch as a service. In a paper by Netflix, they state that approximately 80% of their streams come from their recommender system [10], while Airbnb reports that the implementation of a recommendation system increased booking of experiences by around 30% [11].

4.5 Choice of Solution

On 3. Sep, 2021 Heyloft sent us a feedback document where they concluded that "Speed Dating of Experiences" (Table 6) would be the feature behind the Minimum Viable Product that Heyloft and the team would create together. Heyloft stated that the selection criteria were the following:

- Supports the most "shifts" identified during the ideation phase of our project out of the options presented on the feature list
- It has a higher level of "unknowns" to it and is a feature we could learn a lot about
- From a UX perspective, it gives a great degree of user control for a lower user input
- Easy to collect user preferences for experiences, which is needed for some AI recommendation systems
- Greatest degree of excitement for this feature (from Heyloft and NTNU team)

The requirements for this solution are discussed in detail in section 6.

Choice of AI Technique

As Heyloft had no previous experience with AI, the choice of recommendation system was up to the team. The collaborative-filtering technique requires data about multiple users' preferences, which meant that the team needed data about different users' preferences for the different experiences [16]. Heyloft did not have this type of data, so the choice of recommendation system naturally fell on content-based filtering, which requires data on a single user's interaction with experiences, something the team had access to.

Table 6: Speed Dating of Experiences

Speed Dating of Experiences	
Description	**User stories**
Show an inciting summary of one experience at a time. Tinder-esque swipe right/left to show interest or not. Show more information when user is interested, continue to another experience when not.	• As a user I think that there are too many options presented when looking for experiences. • As a user I want to get to know the experiences I might be interested in. • As a user I want to be inspired to do something.

5 Development Methodology

When working on a larger project it is important to have a clear development methodology in mind. It lays the foundation for cooperation with the customer, the structure of the development team, and eventually the quality of the end product. This section gives an introduction to the core concepts of agile methodology, including some frameworks that it contains. It also describes how we utilized these frameworks during the project.

5.1 Agile Methodology

After the first meeting with the customer, it was clear that the team needed to adopt an agile methodology. The agile methodology, in contrast to traditional development processes, focuses on being adaptive by following an iterative and incremental process. During the initial phases of the project, Heyloft followed the process of design thinking. Design thinking is a process for creating desirable, viable, and feasible solutions through an iterative process of inspiration, ideation, and implementation [13]. As a result, it was natural for us to adopt a similar mindset. The customer also stated early on that they were open to multiple prototypes and wanted us to get started right away and deliver a simple, but working sketch. This is something that would not be possible using traditional methods, e.g the waterfall model. The possibility to alter requirements and adapt during the process was also an important reason for choosing the agile methodology. Several different frameworks fit within the agile methodology. We decided to use Scrum in combination with some concepts of Kanban and Extreme Programming.

5.2 Scrum

Scrum is a workflow-centric agile framework that focuses on value generation through adaptive solutions for complex problems [25]. This made it a good candidate for our project. The idea of value generation is important in design thinking as well. Another important aspect of Scrum is the idea of self-organizing teams [15]. The customer did not want to govern the development of the project, so a high level of independence was necessary. The team also consisted of students with different schedules and as such, it was important to be able to adapt and organize accordingly. Scrum is also a framework that was familiar to all the team members. It had all the benefits of agile that we needed, and allowed us to get started quickly. One possible downside of Scrum is the fact that it was designed for teams working full-time in a shared office, but we felt that the benefits out-weighted the presumed challenges.

5.2.1 Roles

The Scrum Master, the Product Owner, and the development team are three essential roles in Scrum [25]. The role of the Scrum Master is to oversee the Scrum process and to make sure the team's implementation of Scrum runs smoothly and effectively. The Scrum Master should also work as a servant for the team by coaching the team and removing impediments. The Product Owner is the team's main communication channel with the customer and is responsible for the product backlog. It is important that the product backlog is user-centric and that it includes every requirement from the customer. It should be ordered by priority and be transparent and understandable. In addition to these roles, we defined some additional roles (section 3.2). These roles are not defined in Scrum but were added because the team felt it was necessary to share responsibility within the team. Scrum encourages self-organization and this is one of the ways we chose to organize ourselves.

5.2.2 Definition of "Done"

An important aspect of Scrum is the definition of done. During our first team meeting, the team agreed that a user story is not fully implemented until:

- The code implementing the functionality has been written
- The code has been documented and commented
- The code has been tested, ensuring that it works.

Having a clear definition of done is important so that everyone in the team, and the Product Owner, have a common understanding of the quality that is required for each increment. This definition is not intended to be final but can evolve and change over time. During our five sprints, we did not experience a need to change this definition. In hindsight, we think that a less strict definition would have increased our turn-around time in the beginning phases of the project without compromising the quality of the end product. For instance, we could have dropped the requirements of documentation and testing for the first couple of sprints. It is also worth mentioning that this definition of done reflects the fact that the project is just a prototype.

5.2.3 Scrum Events

In Scrum, the development process is divided into sprints. Each sprint follows a clear structure starting with the sprint planning meeting, where the sprint backlog is created. During a sprint, the Daily Scrum is the most important event to synchronize the team. At the end of the sprint, a sprint review is held, and finally, the sprint is rounded off with a sprint retrospective. Breaking the process into sprints allows quick development, frequent feedback from the customer, but most importantly it facilitates constant improvement through empiricism. The events are outlined in figure 3 and further described in the following subsections.

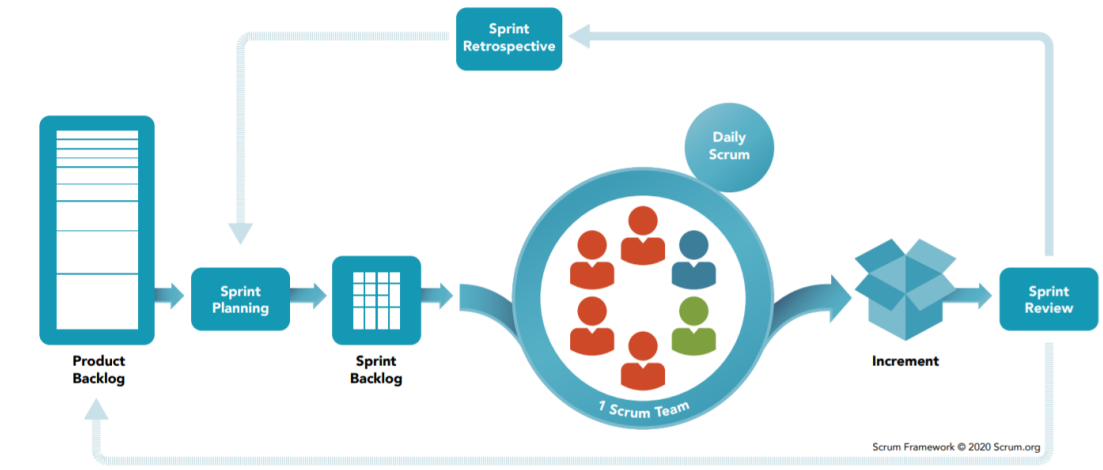


Figure 3: A graphical representation of a sprint within the Scrum framework, from [28].

5.2.3.1 Sprint Planning Meeting

During the sprint planning meeting, a sprint goal is defined, as well as the length of the sprint. The sprint goal is important and should encompass the value the sprint generates for the customer. In general, a sprint should not be longer than a month but can be as short as one week. The main output from the sprint planning meeting is the sprint backlog. This is created from the highest priority items in the product backlog containing the user stories that should be implemented during the upcoming sprint. For each user story in the sprint backlog, the team evaluates the task at hand and sets an initial time estimate for the implementation. To get good estimates we used planning poker. The purpose of planning poker is to reach an estimate that takes all the team members' experience into account. This is achieved by assigning individual estimates, discussing why we reached our estimates, and then agreeing on a final estimate together. For the estimates, we decided to use a relative scale between 1-5, instead of person-hours. The reasoning behind this is that we felt person-hours were difficult to estimate when we do not work a full-time job.

5.2.3.2 Daily Scrum

During the sprint, the most important event is the Daily Scrum meeting. This is a short meeting, around 15 minutes, that ideally should take place at the same time each day. During this meeting, each team member shares their current status. This involves telling the team about what they have done since the last meeting, what they plan on doing until the next meeting, and any foreseeable problems are presented. The goal of the meeting is to improve team cohesion and to adjust the process in order to achieve the sprint goal. Since our team consists of students with different time schedules, a Daily Scrum meeting was not possible. Instead, we tried to uphold two meetings, on Tuesday and Friday mornings, which were our main development days. We found that this meeting was very crucial for team synchronization. The meeting often cleared up any misunderstandings, identified any impediments, and kept everyone up to date on how far we had come.

5.2.3.3 Sprint Review

At the end of each sprint, a sprint review is held where the team presents the current increment to the customer. It is also natural to discuss the next steps and to update the backlog during this meeting. The sprint review is an important event where the product increment is inspected and

crucial feedback is received from the customer. Additional details on how we did the sprint review are presented in section 8.

5.2.3.4 Sprint Retrospective

The final event, that concludes the sprint, is the sprint retrospective. This is the part where the process is inspected and improved upon. During this meeting, each team member shares their thoughts on the sprint, including what was good or bad and what can be improved upon. The sprint retrospective is possibly the most important Scrum event. The whole point of Scrum is to be adaptable and to improve for each increment, and at the end of the retrospective, the team should commit to the most influential changes that increase effectiveness. Additional details can be found in section 8, where each sprint is explained in greater detail.

5.3 Kanban

We decided to implement a Kanban board, from the Kanban framework, in order to get a better understanding of how we were doing in each sprint and as a point of coordination. Kanban is a framework made to manage and improve flow systems [14]. It was originally made to balance demand and capacity and to catch bottlenecks in a factory [17]. The Kanban board is an important artifact within this framework, designed to optimize the workflow in a team [22]. A quick glance at the board gives an overview of who is doing what, how far they have come, but also what no one is working on that should be implemented.

For the implementation, we used the board functionality in GitHub, figure 4. The left-most column contains tasks that are in the sprint backlog, but no one is working on. The middle column contains tasks that are currently in progress, while the last column contains tasks that are completed. The avatars at the bottom of the cards indicate the people assigned to that task, while the number on each task is the estimated effort for the task, see section 5.2.3.1 for more details on how this was done. The other labels indicate if a task involves front-end, back-end, or both. The Kanban board has a lot of important information condensed into a small format which played a central role in the communication of tasks within our team.

5.4 Extreme Programming (XP)

Scrum on its own is mainly a framework that establishes good workflow and cooperation within the team, and also with the customer. It does not specify anything regarding the quality of the code or how the team should organize when programming. As a result, we incorporated some of the ideas and processes from XP to complement Scrum.

5.4.1 Values

XP is also an agile framework, but it focuses on producing high-quality software and improving the quality of life for the development team [8]. The team considered these three XP values most relevant:

- **Communication** – Face-to-face discussion is important to improve knowledge transfer among the team members, but it also helps synchronize the team. This is why we decided that everyone needed to meet physically during a certain time frame on Tuesdays. Communication with the customer is also important. Instead of waiting for the weekly meetings to discuss something we kept an open communication channel on Slack. Here we could ask questions and often got a response within the day.
- **Simplicity** – Keep things simple to avoid waste and only do what is absolutely necessary. This creates a system that is easier to maintain and scale.

- **Feedback** – Constant feedback is crucial in order to continually improve the solution. We often created rough sketches and got important feedback before we started implementing any functionality. This helped us create a solution with a higher value for the customer and often led to a simpler design.

These values are also incorporated in some of the practices we "borrowed" from XP.

5.4.2 Sit Together

The idea of having to meet physically, and sit together once a week, allowed good communication flow in the team. This made us synchronized and kept us up to date on what people were doing, and what needed to be done.

5.4.3 Pair Programming

We also used pair programming a couple of times. This is a practice that often leads to a simpler, more refined solution than when one person develops alone. It is also a practice that integrates continuous code review which improves code quality. Pair programming is also useful if someone is less familiar with the chosen programming language, which was the case for us.

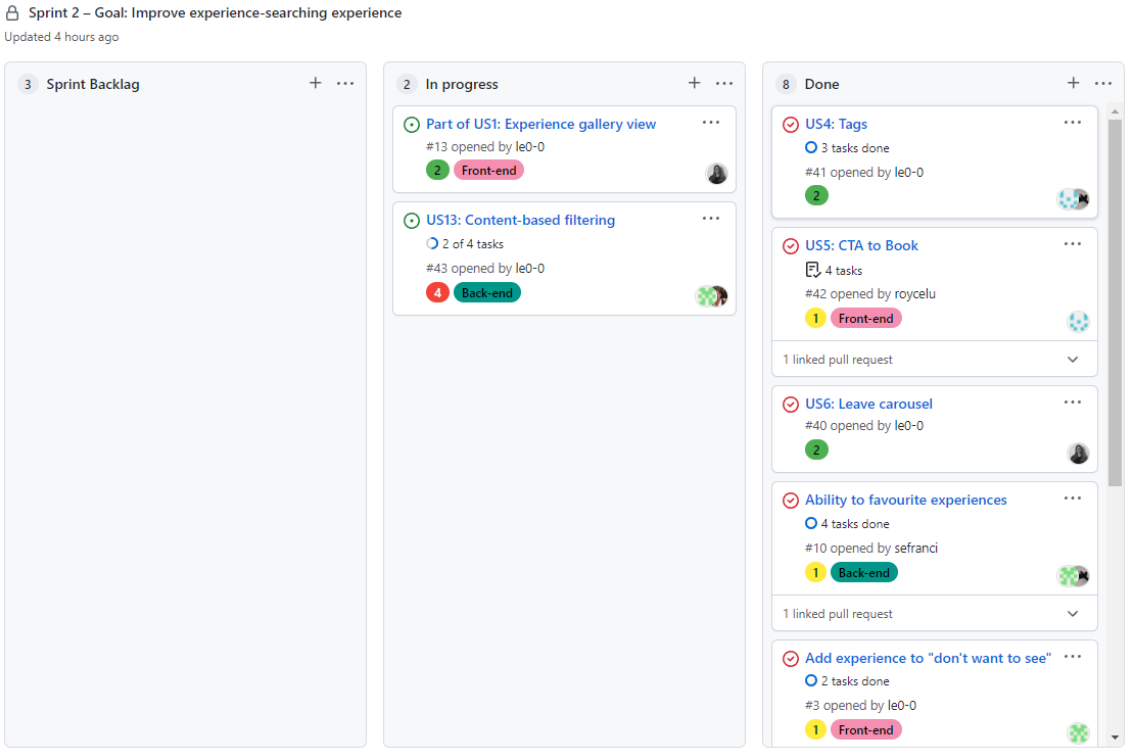


Figure 4: The kanban board implemented in Github, a snapshot from the end of sprint 2 (1. Oct).

6 Requirements

Because of the iterative nature of the agile approach, new ideas and needs might come up during development, and thus change the requirements. Therefore, the team iteratively discussed with the customer and updated requirements based on what the team and the customer had learned from the previous sprints. Instead of defining functional requirements, the team operated with user

stories (section 6.1). The reason behind this choice was that both the team and the customer felt that the user stories implicitly described the functional requirements. This is because there was only one defined user group, the "users", which all the functionality revolved around. Therefore, neither the team nor the customer found it useful to explicitly define functional requirements. Another reason for using user stories is that in Agile, user stories keep the requirements user-oriented, which helps the team to keep the users in mind when developing the prototype [23]. The user stories were kept in a product backlog, which was updated after the sprint review, and refined during the sprint planning meetings.

Through continuous feedback on our prototype from the customer, the user stories in the product backlog, as well as the non-functional requirements evolved, and became more and more defined (increments of the product backlog are in Appendix C).

6.1 User Stories

User stories are short and informal descriptions of features, given from the perspective of the intended users [23]. User stories describe what feature is needed, and how that feature is adding value to the product.

As mentioned in section 4.1, Heyloft did not have any specific business requirements as they wanted to involve the team in exploring new ideas related to their digital marketplace of experiences. To develop an initial list of user stories, the team used a user journey map given by the customer, which described some of the use case scenarios of the prototype the team was going to develop (example in Appendix B). The team also used the description of "Speed Dating of Experiences" from the feedback document (Table 6) to develop the initial list of user stories.

The customer prioritized the initial list of user stories using the MoSCoW method. The MoSCoW method is a way of prioritizing requirements by placing each requirement under "M – Must have", "S – Should have", "C – Could have", or "W – Will not have" (MoSCoW) [20]. During this prioritization, the customer could also add new user stories. The MoSCoW method was used by the customer after every sprint review, except for the sprint review in sprint 5, as that was the last sprint (example of MoSCoW prioritization list in Appendix F. After receiving the MoSCoW list, the Product Owner would update the product backlog by adding the new user stories, remove those who were under the "W – Will not have" group, and re-prioritize the product backlog. During the following sprint planning meeting, the team would refine the product backlog by choosing some user stories to be in the sprint backlog, adding more details to them, and splitting them into more technical tasks. The first increment of the product backlog is shown in Figure 5.

6.1.1 Changes to the Product Backlog

Appendix D.1 shows a list of all the user stories and when they were initially added to the product backlog. Since the product backlog is an artifact subjected to constant change, some user stories were later removed based on feedback from Heyloft, Appendix D.2.

Product Backlog - Ordered by priority (Highest priority on top)						
ID	User stories	Estimated amount of effort (1-5)	How to demo	Details	Scheduled for sprint	Status
US1	Swiping on experience cards			As a user, I should be able to swipe left on an experience card to dismiss an experience, and swipe right to show interest, so that I can save the ones I am interested in.		Not completed
US3	Experience card			As a user, I should be able to see an experience card to learn about the key details of an experience so that I can decide if I like that experience or not.		Not completed
US7	Detailed Experience view			As a user, I want to be able to scroll down on an experience card to learn more about it so that I can learn more about it now rather than later.		Not completed
US2	Superlike on experience cards			As a user, I should be able to superlike an experience to show more interest and add it to my favorited experiences.		Not completed
US5	CTA to Book			As a user, I want to be able to indicate that I want to book an experience right from the experience card so that if I like an experience enough to book it I have the option to do so without seeing more		Not completed
US6	Choosing Experiences			As a user, I want to be given the opportunity to choose between the experiences I have already found and liked so that I can make a choice rather than becoming stuck in the endless swipe.		Not completed
US4	Experience markers			As a user, I want to see "experience markers" so that I can see quick visual information about that experience and make a decision faster / without having to read a lot of content.		Not completed
US13	Recommendation system			As a user, I want to swipe right on an experience I like so that I see more experiences that I like just as much.		Not completed
US8	Create User			As a user, I want to go to a website and create a user, so that my personal preferences and data is saved		Not completed
US9	Log in			As a user, I want to go to a website and log in to a user I have created, so that the website is personalized to me.		Not completed
US10	Search for experiences			As a user, I want to do text search for specific experiences to heyloft's database of experiences and get matching results, so that I can find specific experiences or groups of experiences faster.		Not completed
US12	Unexpected hug			As a user, I want to give/receive "unexpected hug" to a experience so that I can show my interest to that experience.		Not completed
US11	Give reviews			As a user, I should be able to review experiences, so that I am able to give feedback to the arranger of the experience		Not completed
US14	Filter experiences			As a user, I want to apply filters to heyloft's database of experiences and get matching results, so that I can find specific experiences or groups of experiences faster.		Not completed

Figure 5: The first increment of the product backlog. The IDs were arbitrary and given at the time the user story was created. The product backlog was ordered by priority, with user stories higher up in the backlog having higher priority than those below it.

6.2 Non-functional Requirements

The non-functional requirements define the expected quality and desired characteristics of the prototype. Like the user stories, these were also developed through iterations and in collaboration with the customer. The final list of non-functional requirements is given in Table 7. Other non-functional requirements like performance and security were of secondary importance as they would not directly contribute to the focus of the project, which was to explore solutions to the problem given by the customer.

Table 7: Final list of non-functional requirements for the prototype.

ID	Name	Description	Reasoning
NFR1	Modifiability	Make it easy to implement new features and modify old code	Heyloft has a focus on learning and exploring the problem through prototyping. If the solution is to be changed after the course is over, new developers have to be able to make changes. It is therefore preferred that probable and possible changes are limited in scope to certain parts of the project/codebase. It is also preferred to keep the software as modular as possible to allow components to be used elsewhere.
NFR2	Usability	Make the platform easy to navigate and use for the user	As the service has competitors, it is important that it stands out as easy to use. The team must edge out an advantage through unique features but also through a seamless and intuitive user experience.
NFR3	Availability	Make the prototype easily accessible for end-users	Heyloft planned on performing user tests on the prototype, and having the prototype accessible from the internet would make it easier for Heyloft to perform user tests whenever it fits them best and without any technical setup.

6.2.1 Modifiability

Architecture Design

There are several architectural design choices that can contribute to making the prototype more modifiable. Some of these are taken and described in section 7.3.1.

6.2.2 Usability

Software Design Patterns

Software design patterns are templates for solving a specific problem that can be used in a variety of situations [7]. Design patterns are therefore easily recognizable for users and help with faster user familiarity, which in turn betters the product usability [31]. The team has implemented some design patterns from other systems that many users are already familiar with, like the carousel feature from Tinder.

User Testing

The customer let the team know early in the project that they planned on performing user testing of the prototype sometime during the project. Through feedback user testing the team could get insight into issues or complications a user could have when using the prototype, and increase the usability by solving these issues. User testing was not done until the end of the last sprint, but it confirmed some of the usability issues the team predicted, and had worked on fixing.

6.2.3 Availability

Multiple Platforms

The prototype website was developed for both desktop and mobile, with the main focus being on mobile. Having the prototype working on multiple platforms made it available for more end users.

Cloud Server

Setting up a cloud server where the prototype was continuously deployed let Heyloft perform user tests whenever they wanted, and without help from the team. It also lets Heyloft review the prototype for themselves.

7 Architecture

This section is about the software architecture and how it has been used in the project. The team wanted to use an architecture to introduce terminology and facilitate communication and discussion about the design of the system. The software architecture developed can be described as a traditional client-server-based architecture, with separation of the user interface, communication and API, business logic, and handling of persistent data.

7.1 Architectural Methodology

The team has utilized an agile methodology, specifically the Scrum framework, and this has shaped how the idea of architecture developed in the project. The architecture was not meticulously crafted upfront, as this would have been closer to the waterfall method of project structuring, and would have led to decisions being taken without enough information. Instead, changing the architecture to reflect a new feature has been considered part of implementing that feature. On the other hand, some attention was paid to the features that are soon to be implemented when making changes to the architecture, so as to not make implementing those features more difficult by accident. True to agile philosophy, additions or changes to the architecture was carried out closer to when the feature was implemented because the context and requirements surrounding the feature are better known at that time.

The architecture is stored in the project's GitHub repository and is therefore easily accessible by the whole development team during implementation, but also accessible by Heyloft. The architecture documentation consists of MarkDown files and PlantUML files that can be compiled to Portable Network Graphics files – PNG. Both MarkDown and PlantUML files are text files, and can be changed right in the coding environment – Visual Studio Code. Changes must propagate through the version control system as any other change to the repository would.

The team has attempted to keep the architecture free from implementation details and independent of, for example, which programming language is used.

7.2 Architectural Drivers

In the following section, we will outline circumstances that have shaped the design of the architecture.

7.2.1 Functional Drivers

The functional drivers consist of the parts of the architecture needed to fulfill the technical requirements of the finished product. The final version of these can be found in Appendix C. The architecture had to be designed and changed to be able to fulfill these requirements throughout development.

7.2.2 Quality Attributes

Quality attributes, outlined in section 6.2, explain categories of non-functional properties to prioritize when developing the system. The architecture determines many properties of the system, so the quality attributes must be taken into account when changing the architecture.

7.3 Architectural Patterns and Tactics

These tactics are used to accomplish the aforementioned quality attributes.

7.3.1 Modifiability

Semantic Cohesion

Maintaining high semantic cohesion means that code in the same module, or any other natural separation of code, should relate to the same topic or functionality to a higher degree than to code outside that boundary. By choosing a narrow selection of responsibilities or functionality to be included in a module, and by relocating all code related to these responsibilities into the same module, this effect is achieved. The effects of this tactic are that changes are more localized [3, Chapter 5.3].

Restrict Communication Paths & Layered pattern

To restrict communication paths is to limit the number of possible data and control flow transfers between modules. This will reduce the number of other modules that need to change because of a change in one module, which improves modifiability [3, Chapter 5.3]. In this project, this is only achieved by convention. The layered pattern and development view in general is the location where allowed dependencies are communicated. The limitations are not automatically enforced, nor impossible to breach for a developer; code reviews are relied on to spot accidental noncompliance and fix breaches of the conventions.

The modules in the back-end are separated into layers. These layers are organized in a sequence. Modules in a layer are only allowed to depend on modules in layers after that layer in the sequence. This enforces a directed acyclic dependency graph, and higher cohesion, as modules of similar nature are conceptually grouped.

7.3.2 Usability & Availability

We have not committed to any architectural tactics to explicitly improve the usability or availability of the system. However, some non-architectural tactics have been discussed earlier on.

7.3.3 Additional Architectural Patterns

Architectural patterns are known solutions or templates to common problems. These solutions are often tried and tested and the details of their implementation have been developed by many experts. In addition, many developers have experience with them; this makes communication around the system easier. Below are some patterns used in the architecture of this software system.

Client-Server Pattern

The nodes of a system using the client-server pattern[4] are connected in such a way that client nodes exclusively communicate with a separate type of nodes – server nodes. The server nodes, as the name suggests, provide services to the client nodes. In this project, the client sends all communicates with the back-end server and Firebase servers.

Singleton Pattern

The singleton pattern[26] is used for elements that there should only exist one of during run-time. The creation and access of the element is set up in such a way that only one instance is ever created. This pattern helps reduce complexity and simplifies interactions with the element in question.

7.4 View List

This sections contains an overview of views used to communicate the current architecture. Views are descriptions of the software architecture from different perspectives. The information within a view is not necessarily unique among the views, but it is framed in a unique way. Combined, the views give a more complete picture of the architecture. We have used the 4+1 view model to describe our architecture, Table 8.

Table 8: Views and descriptions used to communicate the software architecture.

Name	**Stakeholders**	**Use-cases**	**Description**
Use-case view	Architects	Configuration, design, implementation, system analysis	Contains fundamental use-cases – intended interactions with the software system.
Development view	Developers	Implementation	Describes modules separating code and functionality into coherent design-time entities with strict restrictions for code dependency.
Physical view	Developers, system engineers	Configuration, implementation, system analysis	Shows physical compute nodes and communication channels that exist between them run-time.
Process view	Developers, system engineers	Implementation, system analysis	Detailed description of components the software system is separated into run-time, communication channels between them, and properties of both.
Logic view	Developers, end-user, UX designer	Designing, implementation, usage	A collection of circumstances and information relevant for end-users interacting with the system run-time or UX-designers designing this experience.

7.4.1 Use-case View

The use-case view is in some ways a redundant view, and therefore constitutes the "+1" in the name of the 4+1 View Model; it does not contain more information than what the other four combined does, but is used to guide the development of the other views. The view should contain essential use-cases of the software system – in this project, just one – so that the efforts of architects and other stakeholders have a common direction.

Figure 6 shows the system behavior after a user navigates to the saved experiences page. When navigating around the website, information has to be retrieved from the persistent data storage. The events in the figure show how different components of the system interact. The figure follows the UML sequence diagram format. This use case, consisting mainly of a data retrieval, was chosen because it resembles or is representative of many or most other processes the architecture must be designed to accommodate. A quick summary: User intention is captured and transmitted in the front-end, the intention is received by the API, processed by the controllers, which make use of and changes persistent data through the database connector. The case in question shows a successful processing of the request. The system being developed is explicitly a prototype, so there are few patterns and techniques dedicated to fault recovery. Because of this, a sequence diagram showing the behavior during a failure was not included.

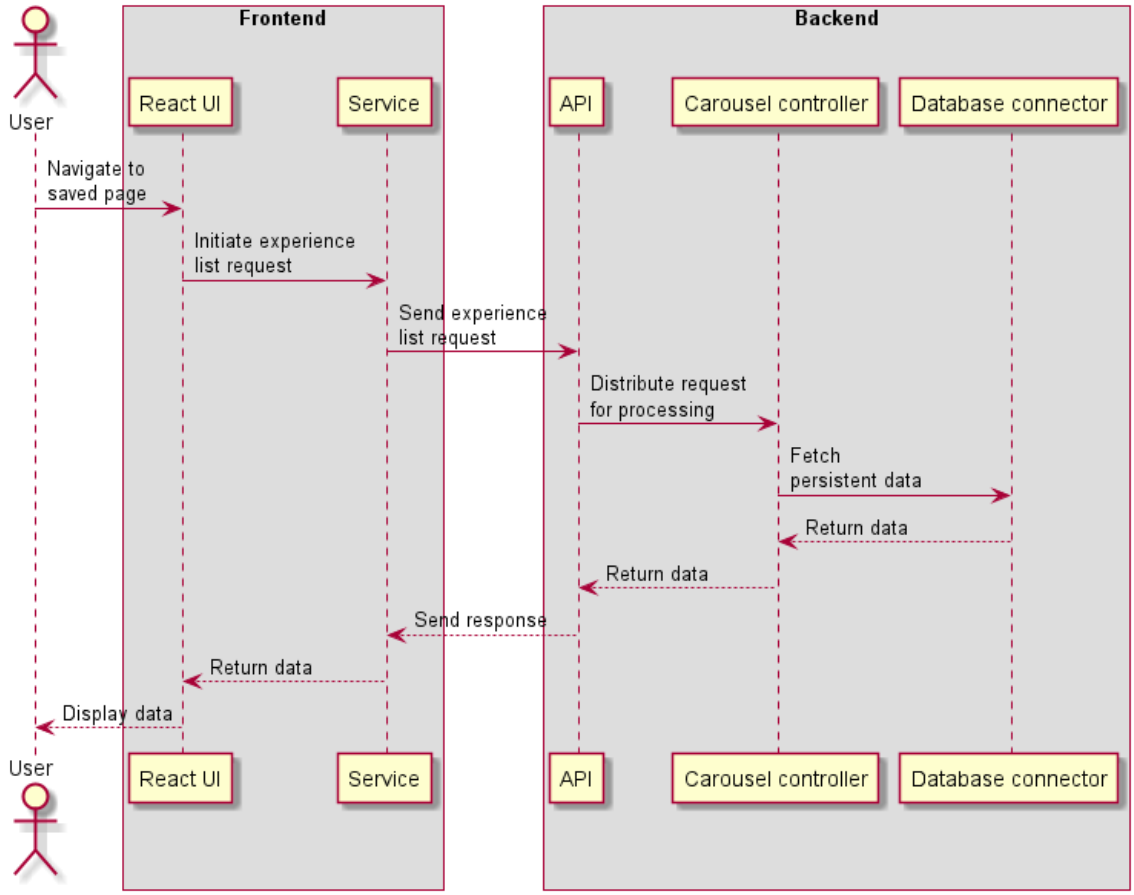


Figure 6: A UML Sequence diagram that shows method calls and data transmissions in the system after the user wants to see their liked experiences; specifically, a successful processing of the request.

7.4.2 Development View

The development view communicates modules in the system, separation of code and functionality into coherent design-time artifacts.

Figures 7 and 8 use a proprietary format. Packages represent layers, rectangles represent modules, and arrows represent dependencies: The source of the arrow is the module that is dependant, and the destination is the module that is being depended on.

Figure 7 shows layers and modules in the back-end. The network layer contains only the API endpoint module. This module is responsible for exposing an API on the network interface of the node running the back-end artifact. The network layer is allowed to depend on the logic layer. The logic layer processes requests received from the API. Requests are separated into a number of controllers. The experience controller is responsible for processing requests related to experience information. The group controller is responsible for group information and manipulation, as well as the group experience exploration functionality. The user controller is responsible for user information and manipulation. The carousel controller processes request related to the experience exploration functionality for individuals, which we defined as the "Speed Dating of Experiences". All controllers are allowed to depend on the controller utility module, containing logic that might be shared by many controllers. They also make use of the database connector to retrieve and manipulate persistent application data. The AI models module is separated from the carousel controller because it differs in nature and style from the carousel controller. It is responsible for all machine learning algorithms. It is technically in a layer above the data layer, thus it can make use of the database connector, but at the moment, it does not; instead, data is provided to the model by the carousel controller.

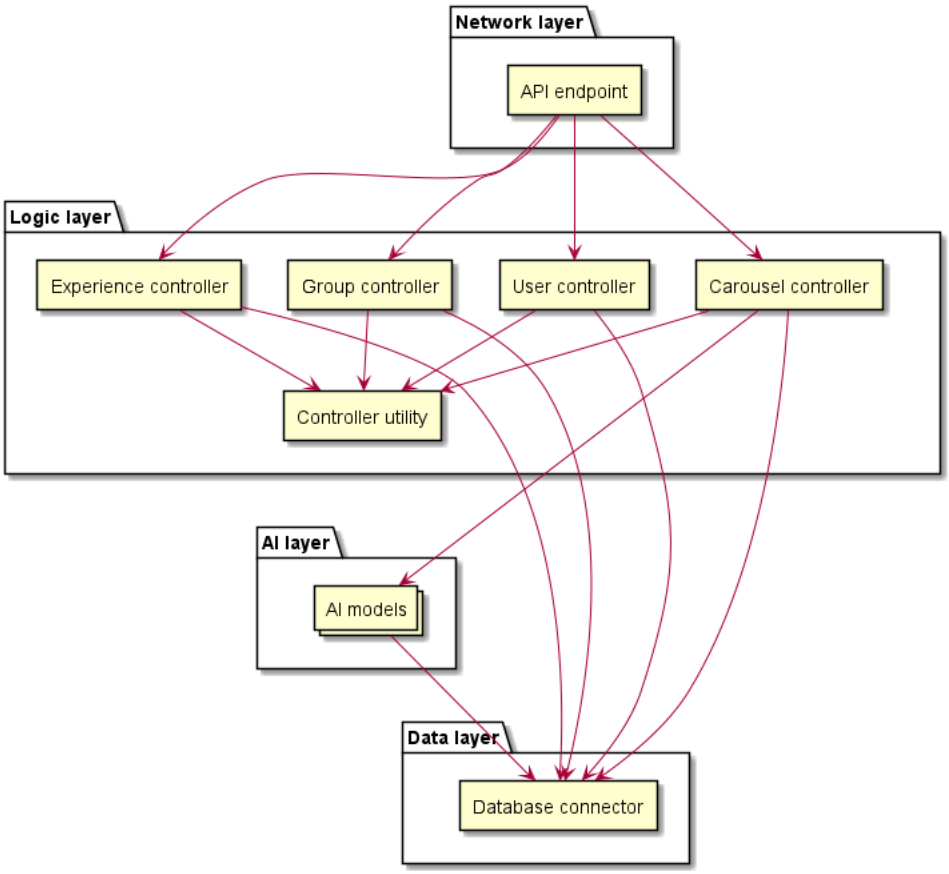


Figure 7: Layers, modules, and dependencies in the back-end artifact.

Figure 8 shows modules in the front-end. The front-end contains fewer modules than the back-end. Although different elements in React can be sorted into a hierarchy, like the folder structure, most or all React elements are so similar in content and function that it is sub-optimal to separate them into modules; the module boundaries would be less clear than normal. Most of the React elements are therefore contained in only one module – React UI – responsible for the visualization of the application. The user context module is responsible for maintaining the user state and communicating with the authentication provider. The services module is responsible for communicating with the API exposed by the back-end.

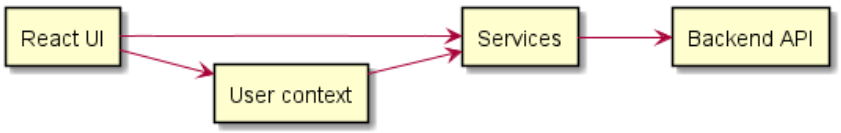


Figure 8: Modules, and dependencies in the front-end artifact.

7.4.3 Physical View

The physical view communicates nodes present, artifacts running on them, and communication channels at run-time and during development. The view is used to analyze the properties of the system as a whole, such as security, performance, and availability. The view contains two diagrams following the Deployment diagram format in UML.

At run-time, the front-end artifact, running on the client computer, communicates securely over HTTPS with the back-end artifact, running on the Google Cloud Platform (GCP). The Back-end communicates with the live database securely over proprietary means inside the GCP. This is

shown in Figure 9.

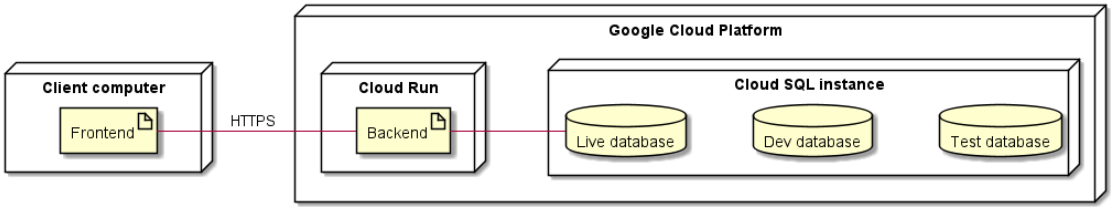


Figure 9: A UML Deployment diagram: Nodes, artifacts, and their communication for the live version.

During development, both the frontend and backend artifacts are running on the developer's computer, communicating over HTTP. The backend can reach the development database and test database through Cloud SQL Auth proxy – a piece of software distributed by Google Cloud for connecting to Cloud SQL over the internet. The security of this communication channel depends on Cloud SQL's offering, Figure 10

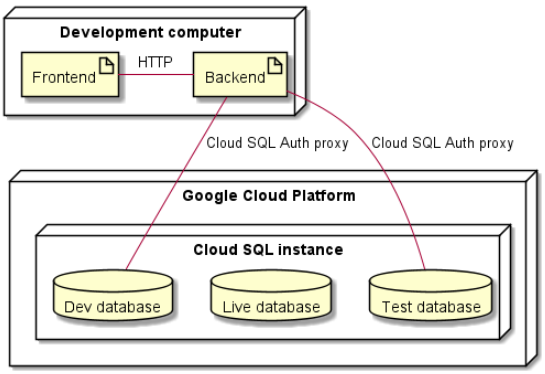


Figure 10: A UML Deployment diagram showing nodes, artifacts, and their communication during development and automatic testing.

7.4.4 Process View

The process view illustrates how different components of the software system will interact run-time. It is used for system analysis and implementation.

In Figure 11, artifacts are represented by package notation, components are rectangles, and control flows are arrows with a solid stroke. Arrows with a dotted stroke mean that control is also returned the same way it was transferred. Elements that there might exist multiple of in their parent context are marked with green. The diagram shows properties of different elements that exist run-time. While the backend artifact is only meant to be running on one instance worldwide, there might be many clients running at the same time, trying to communicate with the backend. Furthermore, the only component type with the possibility of concurrent instances is recommendation models, which are created for each request of certain types. Other components therefore utilize the singleton pattern to simplify interactions. For control flows, it can be noted that control flows start from the React UI component and spread inwards in the system.

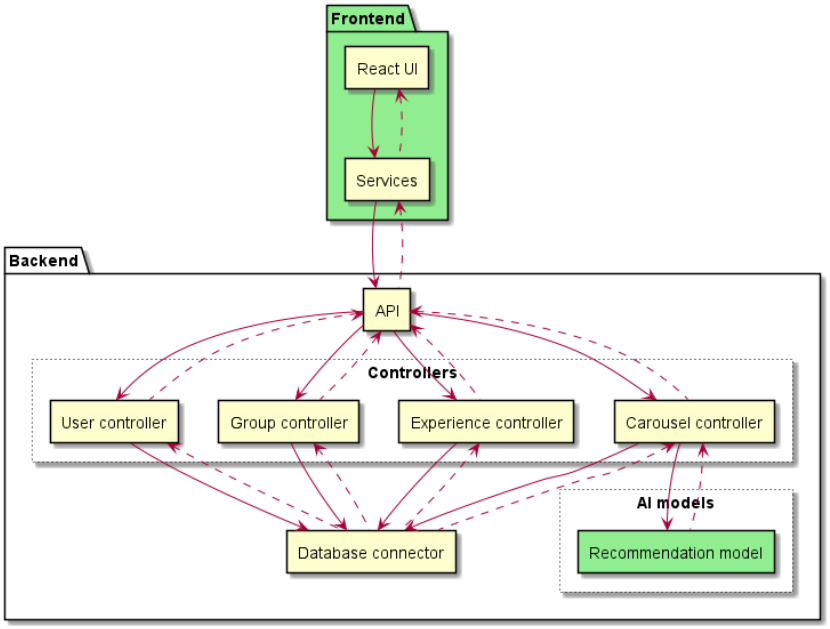


Figure 11: Components and control flows, sorted by artifact and groups.

7.4.5 Logic View

The logic view concerning the functionality and usage the system provides to users. Figure 12 shows a sitemap for the website. The sitemap sections the website into pages, which each contain different functionality. It therefore serves as the natural separation of user-centered functionality. This can be used to justify or design orderings and structure of the codebase or individual modules. A user will arrive at the home page when they first open the site. All pages have a navigation bar. Figure 12 shows which pages a user can navigate to from the navigation bar. Furthermore, all other links are also shown. All pages that require the user to be logged in are marked as such. Pages in the diagram ending with /GroupID adapt to the group that is being shown at the time.

The home page is used for information and introduction to the application, as well as a hub to navigate around the page. The experience swiper page helps the user explore and save experiences. The saved experiences page presents the users with the experiences they have previously saved. The groups list page and other group-related pages help the users create and make use of groups to explore experiences as a group. The features of the application are with this divided into three. This separation can be found in the code base as well; for example with the three backend controllers: (1) the carousel controller, (2) the experience controller, and (3) the group controller.

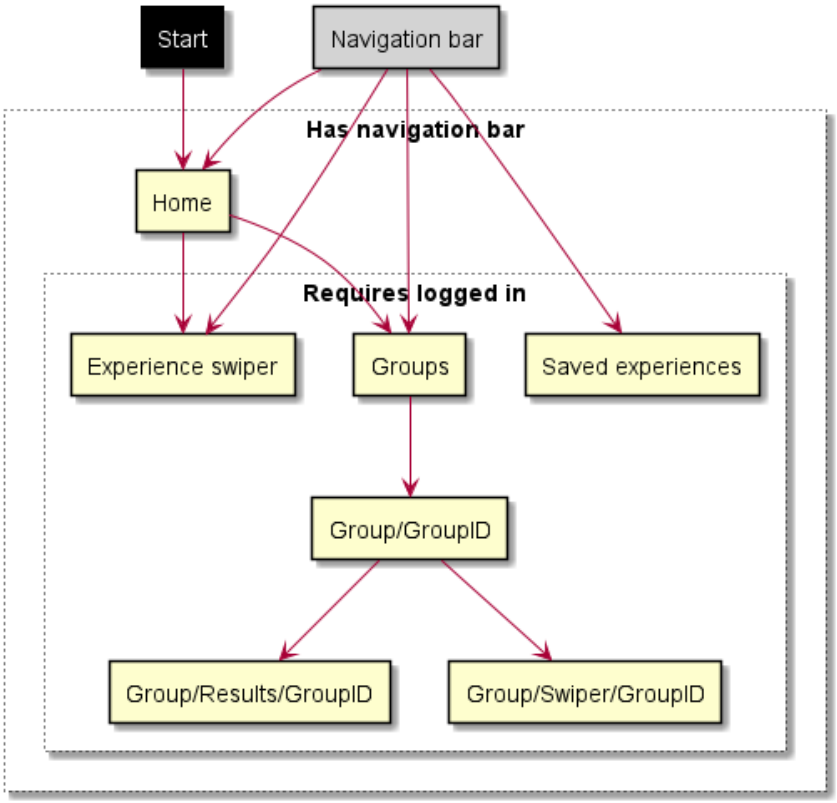


Figure 12: A site map of the application.

7.5 Architectural Reasoning

As mentioned previously, the architecture has developed to be quite traditional. The separation of responsibilities in the backend, into communication, processing, and data, is common; thus, many developers will be able to understand and work on the software system without studying the conventions and concepts in the project to great lengths. This was a goal for us because the project, from start to finish, is quite short, and we wanted to get started quickly. It will presumably also help potential developers from Heyloft in a similar manner. Because of the commitment of Heyloft and the student team to agile methodologies, the functional requirements of the project were bound to change. This meant that tried and tested architecture elements which fit well with a plethora of different functionality were also favored.

From the architectural tactics and patterns committed to by the team, it's clear that modifiability was a high priority. There are other quality attributes which were prioritized in this project, but which no architectural tactics and patterns were committed to improve. By keeping the code-base, and system in general, modifiable, the team was able to efficiently solve problems and improve the other quality attributes when there were indications it was needed. This way, development efforts will be focused on what generates the most value at the moment, instead of prematurely optimizing the system for needs that do not exist.

8 Implementation/Sprints

In this section, each sprint is described in more detail. For each sprint, we explain how we performed the different Scrum events, the state of the different artifacts, and other implementation details, which includes the use of Kanban and Miro boards, as well as burndown charts. It is worth mentioning that every sprint was done as a part-time work effort with an average of 20 hours per week for each team member. The sections are divided into sprints and follow the same format: An

introduction of the sprint, the sprint planning meeting, the increment, the sprint review, and the sprint retrospective.

8.1 Sprint 1

The first sprint started as soon as we had the initial user stories from the customer. We decided on a sprint length of 1.5 weeks from 7. Sep to 17. Sep. The sprint goal was to *experiment with the "Tinder of Experiences"*, which was the main feature of the prototype. Later the feature changed name to Speed Dating of Experiences. Based on this goal we decided to implement the user stories shown in Appendix E.1 and made an initial workload estimate between 1 and 5 using planning poker (as described in Section 5.2.3.1). Subsequently, we broke these down into technical tasks and inserted them into the kanban board on Github. We decided to split the tasks into front-end and back-end specific tasks.

During the first sprint, we had three daily scrum meetings; one each at the beginning of our official workdays and one update meeting with the customer. Do note that we use the term daily scrum for a 15-minute meeting that only took place two times a week. For more information see section 5.2.3.2. As this was our first sprint, a crucial part was to set up a working technical project, where the team members could easily work efficiently with tasks. Naturally, some tasks were dependent on others. For instance, an operational database was needed in order to store experiences that would be displayed as a part of US1, US3, and US7. The daily scrum meetings were therefore important for the team in order to cooperate, track progression and update each other on the status of each team member and their tasks.

Table 9 shows information on all the Scrum events that took place during the first sprint.

Table 9: Sprint 1 events

Sprint 1 (7. Sep - 17. Sep)		
Event	**Date**	**Duration**
Sprint planning	7. Sep	4 hours
Sprint review	16. Sep	1 hour
Sprint retrospective	17. Sep	1 hour
Daily scrum	10. Sep, 14. Sep, 17. Sep	15 minutes each

8.1.1 Sprint Planning

Prior to the start of sprint 1, the team had collected requirements in the form of user stories from the customer through the MoSCoW prioritization process (section 6). The product owner was then supposed to add these user stories to the product backlog and prioritize them before the sprint planning meeting. However, because of inexperience with the scrum methodology, the product backlog was updated in the beginning of the sprint planning meeting (appendix C.1) where several team members contributed with updating and prioritizing the product backlog, which is only the product owner's task. Since other team members updated and prioritized the product backlog, the scrum guidelines were not followed [25]. In later sprints, the product backlog was updated and prioritized only by the product owner, and before the sprint planning meeting. The meeting was held on 7. Sep and took four hours. The outcome of the meeting was the sprint goal, *experiment with the "Tinder of Experiences"*, and the sprint backlog (appendix E.1).

After updating the product backlog, the team refined the product backlog. This meant filling out the "Estimated amount of effort", "How to demo" and "Scheduled for sprint" columns in the product backlog. The "Estimated amount of effort" column could have values ranging from 1 to 5 (section 5.2.3.1). In the "Scheduled for sprint" column we filled in "sprint 1" for the user stories we wanted to include in the sprint backlog. We also filled in the "How to demo" column which described how to demonstrate the features that would be added with the user story. The sprint backlog contained the same columns as the product backlog, except for "Scheduled for sprint"

and "Status" as these were self-explanatory. Following this, we split the user stories into smaller, independent technical tasks, and put them into the Kanban board on GitHub (figure 13). The team also estimated the amount of effort needed for the technical tasks.

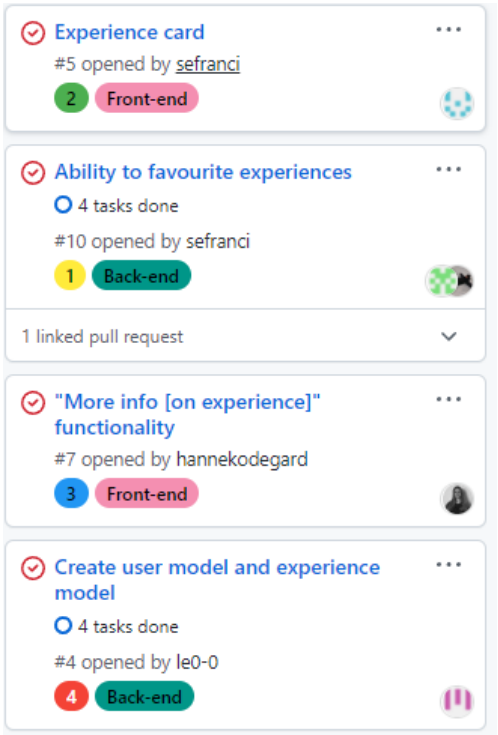


Figure 13: A snapshot of some of the technical tasks on the Kanban board from Sprint 1. The tags correspond to the estimated effort and whether it was a front-end or back-end task. Note: This snapshot was taken on 23. Sep, which was after the end of sprint 1.

8.1.2 Increment

During this first sprint, a subpage for the swiping functionality and the experience display was created, these pages are displayed in Figure 14.

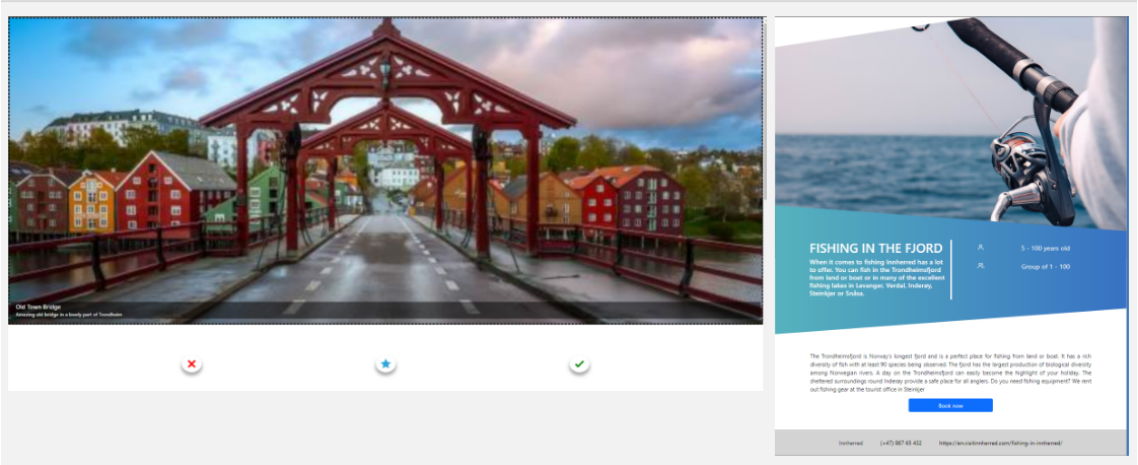


Figure 14: Snapshot of the sprint 1 increment

8.1.3 Sprint Review

At the end of the sprint, we had a one-hour sprint review where we showed the customer our current increment. We started the sprint review by demonstrating the functionality we had been able to implement in this sprint. We managed to complete four out of seven user stories according to our definition of done. This is defined in section 5.2.2. The finished user stories from Sprint 1 are shown in Table 10.

Table 10: Finished user stories from Sprint 1.

ID	Name	Description
US3	Experience card	The user can now see a card with the name of the experience, a picture and a description.
US7	Detailed Experience view	The user can now scroll down to see a more detailed description of the experience.
US2	Favourite on experience cards	The user can now click a button with a blue star icon to indicate that he or she wants to mark this experience as a favorite.
US5	CTA to Book	At the bottom of detailed experience view, there is now a "book now" button.

The main value we provided in this sprint was a website running locally that showed experience cards and also a more detailed description of the experiences if you scrolled down. In addition to this, we implemented a possibility to favorite experiences and provided a "book now" button. We also managed to connect front-end and back-end so that we could retrieve experiences via the back-end. The user stories we did not complete this sprint are listed in Table 11.

Table 11: Unfinished user stories from sprint 1.

ID	Name	Explanation
US1	Swiping on experience cards	Liking, dismissing and favoriting experiences was enabled, but the actual click and drag swiping animation was not finished
US6	Leave carousel	Not started
US4	Tags	Tags were visible to the user, but we had not yet finalized what tags should be used and what their symbols would look like

US1, Swiping on experience cards, was the highest prioritized user story this sprint. This user story was not finished because we focused on implementing the functionality for swiping, and not the visual animation associated with liking, disliking or favouriting an experience. US6 and US4 were not finished because we had underestimated the time and effort it would take to complete the user stories that were finished. Two of the unfinished user stories, US1 and US4, had many of their sub tasks done, but they did not meet our definition of done and were therefore not considered completed. As stated earlier, the user stories in the sprint backlog are sorted by prioritization. The reason US4 had some work done to it before we started on the higher prioritized US6, was that tags (US4) were closely related to the design of the experience cards and therefore, easy to swiftly implement when the experience card itself was finished. These uncompleted user stories were put in the product backlog for the next sprint.

Together with the customer we then had a brief discussion on what user stories the team would like to work on for the next sprint. The customer already had a list of user stories but wanted our input. For the most part, they wanted to make it simple and easy for users to find and book experiences. Finally, the team concluded that a 1.5-week duration for sprints would be too short. Therefore, we informed the customer that the next sprint will have a duration of two weeks.

After the meeting, the customer did the MoSCoW priority by themselves without the product owner. This was done since both parties had agreed on how the MoSCoW prioritization process should be done.

Following the sprint review, the product backlog was updated (appendix C.2).

8.1.4 Sprint Retrospective

After the sprint review with the customer, the team got together and had a sprint retrospective. The meeting was held on 17. Sep and lasted for one hour. The entire team was present. During this meeting, we made the rounds letting everyone speak. Each person listed what they thought was good, what could have been better, and what we should do differently in the next sprint. While everyone spoke, the Scrum master wrote down the points on artificial post-it notes in Miro, figure 15.

Figure 15: A Miro board was used during the sprint retrospective to summarize what was being said.

Overall the team felt that we got a lot of tasks done, we had set up a good technical structure for the project which allowed us to implement features efficiently. This was also due to the fact that we had a thorough sprint planning meeting. Further, the team felt that everyone made a good effort. We were also in agreement that we were using the sprint board on GitHub efficiently, which allowed us to get an overview of who was doing what, and how far they had come. As a final point, we felt that we stuck to the backlog and only implemented functionality that the customer wanted.

We also noted some things that were not as good. During this sprint, we did not complete as many user stories as we would have hoped. We did agree that we got a lot of functionality done, but some user stories according to our definition of done were not fully completed. This was partly due to having big development branches making the final integration difficult. Communication between people doing different tasks was also not optimal.

After everyone had spoken, we discussed the different points and pointed out which improvements we should focus on during the next sprint. We decided to let everyone give a 'heart' to the improvements they thought were the most valuable, as can be seen in the improve column in figure 15. After voting, we discussed the improvements with the most votes and decided how we wanted to implement them and why they were important.

- Mainly, we wanted to try to use continuous integration of code in order to avoid having separate branches with important code. This would also make it easier to complete tasks since they often lacked final integration with other parts.
- To improve communication we decided that we should not divide user stories into back-end

and front-end tasks, since this naturally led to a split in communication between the people doing back-end tasks and the people doing front-end tasks. Instead, we decided to keep the full user stories as tasks on the Kanban board allowing people to work full-stack, but also making it easier to collaborate. This would also make it easier to understand when a user story was "done" which might help us complete more tasks.

- We also decided to use a burn-down chart in order to implicitly improve communication. The idea was to update the burn-down chart continuously as the team wrote code, giving a good overview for everyone to see how we were doing.

8.2 Sprint 2

For the second sprint, the team agreed to extend the sprint length to two weeks. The team felt that 1.5 weeks was just a bit too short and identified this as the main reason for incomplete user stories. The sprint was scheduled from the 17. Sep to the 1. Oct and the sprint goal was to *improve the user experience when searching for experiences*. In accordance with the sprint goal, the sprint backlog (appendix E.2) was created by pulling the most important and relevant items from the product backlog. Four daily scrum meetings were held this sprint, in addition to one customer update meeting. Table 12 summarizes the Scrum events from sprint 2.

During the sprint planning meeting, we chose not to split the user stories into technical tasks in the Kanban board on GitHub. The reason for this stems from the retrospective of Sprint 1: We wanted to avoid splitting the development into front-end and back-end. Having the full user story as a task forces a full-stack mindset during implementation. Since the user stories alone obviously are larger than broken down technical tasks we decided to allow several people to assign themselves to the same story. This resulted in better communication within the team.

Table 12: Sprint 2 events

Sprint 2 (17. Sep - 1. Oct)		
Event	**Date**	**Duration**
Sprint planning	17. Sep	2 hours
Sprint review	1. Oct	1 hour
Sprint retrospective	1. Oct	1 hour
Daily scrum	21. Sep, 24. Sep, 28. Sep, 1. Oct	15 minutes each

We also decided to implement a burndown chart during this sprint, Appendix L.1, as discussed during the retrospective of Sprint 1. A burndown chart is a tool for visualizing the velocity of the team during the sprint. We wanted to implement this in order to keep a continuous report on the progress of the sprint. While traditionally made based on the number of person-hours, we decided to base it on the assigned relative points. From the burndown chart, it is clear that most of the tasks are completed on our main development days, which were on 21. Sep and 28. Sep.

8.2.1 Sprint Planning

The sprint planning for sprint 2 was held on 17. Sep and lasted for two hours. The product owner had received the MoSCoW prioritized list from the customer after the sprint review for sprint 1 and updated the product backlog. During the meeting, the team refined the product backlog and decided on what user stories to focus on for sprint 2. This resulted in a sprint backlog (appendix E.2). One of the suggestions made during the sprint 1 retrospective was to put user stories directly into GitHub (figure 15). In sprint 1, the team had put smaller more technical tasks into GitHub, but felt that it became hard to keep an overview of the tasks and how they related to the user stories. Hence, in the sprint planning meeting, the user stories were put directly into GitHub and were given technical subtasks as part of their description. An example can be seen in figure 16. This meant that the team only estimated the amount of effort needed for the user stories, and not for smaller, more technical tasks.

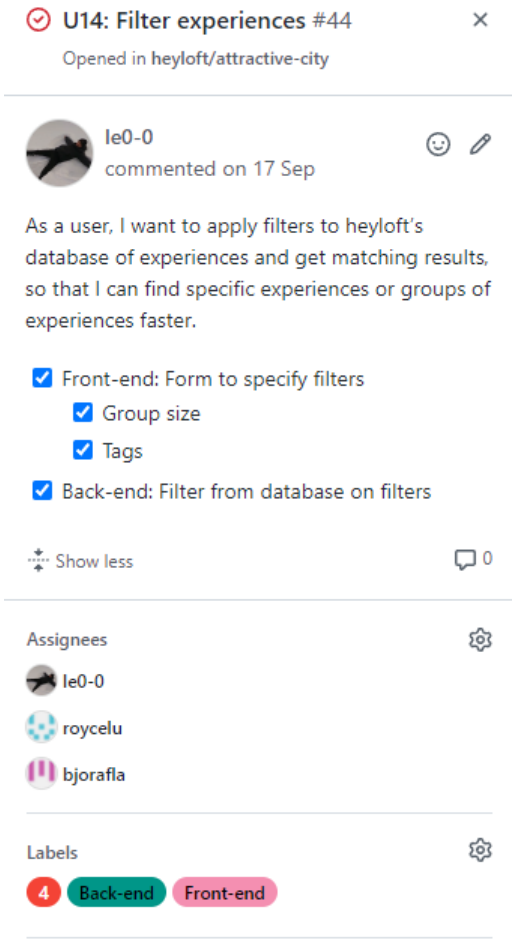


Figure 16: An example of how a user story was entered as an issue in the kanban board on GitHub.

As mentioned in the introduction, the sprint goal for sprint 2 was to *improve the user experience when searching for experiences.*

8.2.2 Increment

During this sprint, tags were added to the experiences along with a booking button and a prompt to leave the carousel after having swiped on 5 experiences. The main views are displayed in Figure 17.

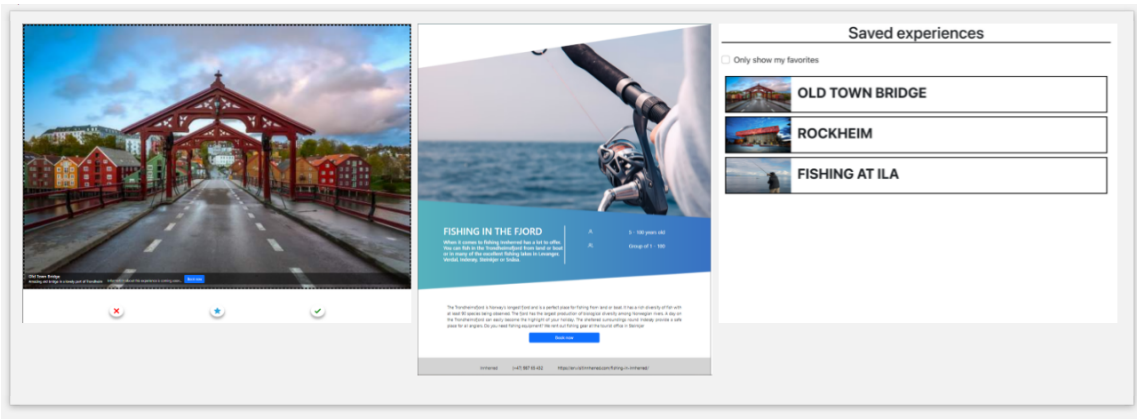


Figure 17: Snapshot of the sprint 2 increment

8.2.3 Sprint Review

We had a one-hour sprint review on 1. Oct. This started with a demo for the customer showcasing the increment of this sprint. During the demo, we ran into some confusion regarding who was to present and who had working code, but it sorted itself out in the end. We managed to complete five out of six user stories this sprint. Table 13 shows the finished user stories for sprint 2.

Table 13: Finished user stories in sprint 2

ID	Name	Implementation
US1	Swiping on experience cards	A user can now swipe left or right on an experience card to indicate whether they like the experience or not.
US6	Leave carousel	After liking five experiences, a modal displaying your recently liked experiences pops up. The user can click on these experiences to see more information about them.
US4	Tags	There are now tags visible on every experience card. The tags are keywords that describe the experience (e.g "music").
US14	Filter experiences	There is now a blue button in the top right corner of the screen. If the user presses this button, a menu where you can filter experiences based on tags is shown.
US5	CTA to Book	A big "book now" has been added to the experience cards.

The value we provided in this increment was enabling swiping on the experience cards, to give the user an option to look at recently liked experiences, display appropriate tags, filter experiences on tags, and also a more visible "book now" button on the experience card itself. In this sprint, we had one unfinished user story, which is listed in Table 14.

Table 14: Unfinished user stories from sprint 2

ID	Name	Explanation
US13	Content-based filtering	Functionality finalized, but missing tests. Therefore, not merged into the main branch of the project yet.

US13 was completed locally, but we did not manage to write tests and merge it into the main branch of the repository in time for the sprint review. Therefore, US13 did not meet the definition of done and was accordingly considered not complete and scheduled for the next sprint.

The feedback from the customer was positive and the comments suggested that they felt we had managed to implement what they wanted. At the end of the sprint review, we discussed the next sprint and the customer asked us what user stories we would like to focus on. This resulted in us sending them a list of user stories that they would take into consideration when creating a

MoSCoW prioritized list of user stories for the next sprint. These prioritized user stories would then be added to the sprint backlog for sprint 3 during the sprint planning meeting on 2. Oct. We decided to do it this way as it proved to be effective in sprint 1.

Following the sprint review, the product backlog was updated (appendix C.3).

8.2.4 Sprint Retrospective

Like the last sprint, the team got together digitally and used a Miro board for the sprint retrospective, figure 18. The meeting was scheduled for 1. Oct and lasted one hour. During the sprint retrospective for sprint 2, the team agreed that the 'problems' from sprint 1 had been improved. Several team members pointed out that communication was better and people got quick responses when needed. Instead of dividing into a front-end task and a back-end task, we decided to base the tasks on the user stories and divided ourselves accordingly. This made it easier to work full-stack, which resulted in better communication and better flexibility in tasks.

However, the team also discussed things that could be better. Just like in sprint 1, the team had some difficulties with merging new changes to the main branch right before the sprint review. With little time to prepare for the meeting, we experienced the meeting as chaotic. In addition, several team members pointed out that we should be better at updating the burndown chart and the status on the GitHub board during the sprint, and not only at the end, in order to keep track of the current status of the sprint.

For the next sprint, the team discussed some improvements:

- Before the next sprint review, the team wanted to introduce "code freeze", which means that no code can be edited or modified during the freeze period [9]. The suggested time was 12 o'clock before the sprint review, which hopefully gives enough time for debugging and making sure that everything works as expected, including time to plan the meeting.
- The team has decided to have more transparency into task progress. The reason for this was that the scrum master mentioned that the documents were not always up to date, which made it difficult to know the progress of the sprint. By ensuring that the documents and the status on the GitHub board are correct, it may be easier for the team to have a better overview of the bigger picture.

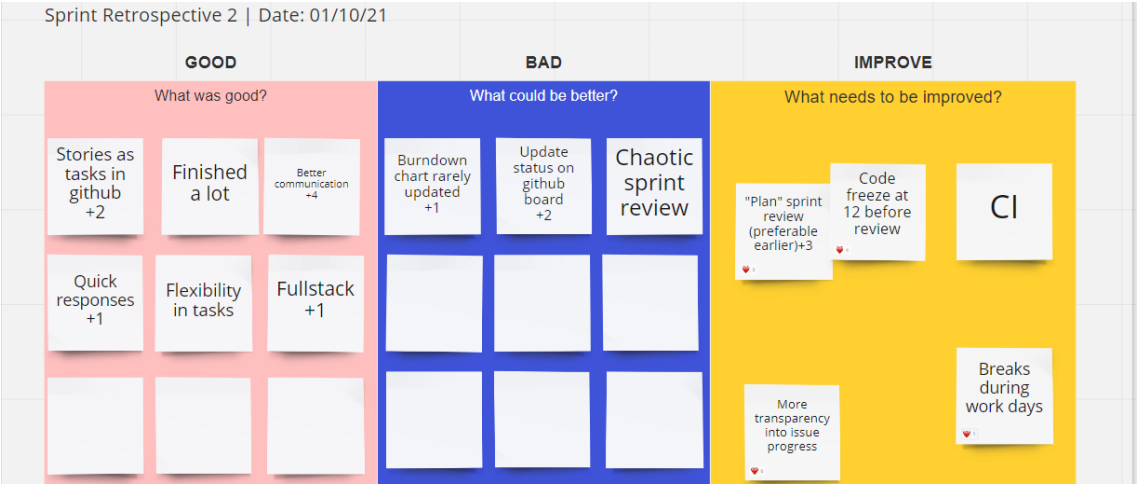


Figure 18: The Miro board was used during the sprint retrospective for sprint 2.

8.3 Sprint 3

Continuing with a sprint length of two weeks, the third sprint was scheduled from 2. Oct to 15. Oct. Each of the Scrum events are summarized in Table 15. The sprint goal was to *improve usability and accessibility*, since the customer wanted to user test the prototype after the sprint. For instance, that included the possibility to have users, so that personalized data and preferences were saved. In order for the users to use the prototype, we had to deploy it to a website. The main focus during the sprint was mobile-first, since the users would most likely be using smartphones when using the prototype.

Table 15: Sprint 3 events

Sprint 3 (2. Oct - 15. Oct)		
Event	**Date**	**Duration**
Sprint planning	2. Oct	2 hours
Sprint review	14. Oct	1.5 hours
Sprint retrospective	15. Oct	1 hour
Daily scrum	5. Oct, 8. Oct, 12. Oct, 15. Oct	15 minutes each

To keep the team updated, we continued to have the user stories in the Kanban board on GitHub and use a burndown chart, Appendix L.2, to keep a continuous report on the sprint progress. Feedback from previous sprints indicated that the overall communication within the team was good, due to the high transparency of the progress. During this sprint, we had four daily scrum meetings to keep the team updated, but no meetings with the customer except the sprint review.

8.3.1 Sprint Planning

The sprint planning meeting for the third sprint was held on 2. Oct and lasted for two hours. After the sprint review for sprint 2, the team sent their suggestions for new user stories to the customer, who then finalized and prioritized the user stories based on their MoSCoW prioritization process. The product owner then updated the product backlog. Then the team followed the process of earlier sprint planning meetings and refined the product backlog before deciding on what to focus on for this sprint. The user stories chosen for sprint 3 can be seen in the sprint backlog (appendix E.3). The user stories were taken from the product backlog in accordance with the sprint goal: *improve usability and accessibility*. As in the earlier sprints, planning poker was used to discuss and decide the "estimated amount of effort" for each user story.

8.3.2 Increment

This increment added a navigation bar to the website along with a filtration pane and a homepage. This can be seen in Figure 19.

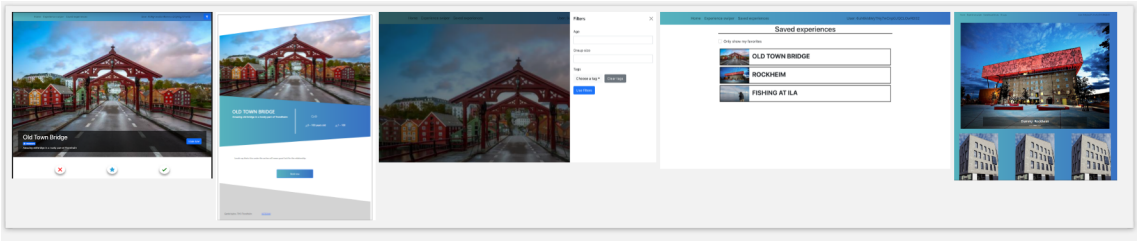


Figure 19: Snapshot of the sprint 3 increment

8.3.3 Sprint Review

As stated in the sprint 2 retrospective, the team had agreed upon a code freeze at 12:00 on the day of the sprint review. At 12:15 the team had a meeting where we merged all our finished code and pushed it to the production branch which is the version of the code that is deployed online. After merging, we encountered numerous technical challenges related to login while viewing the website's mobile version. This resulted in that the progress made on US16, mobile-first, was not shown to the customer during the sprint review.

For sprint 3, the customer had asked that the sprint review should be physical. The customer also wanted to discuss what new features to implement in the next sprint and how these features should be designed. Therefore, the meeting was scheduled for one and a half hour, half an hour longer than usual, starting at 14:30.

The sprint review was held at Digs, a startup community located in the city center of Trondheim where Heyloft has its offices. The sprint review started off with us giving the customer a link to the website. Since the prototype was accessible online, the customer had the possibility to test the functionalities themselves. The customer was very enthusiastic about being able to interact with the product. The customer was happy with the progress that was made this sprint, but also pointed out that they wanted to start user testing after the next sprint and that the website should be user-friendly on mobile by then.

We managed to complete four out of six user stories this sprint. These are shown in Table 16, while the unfinished user stories can be seen in Table 17.

Table 16: Finished user stories in sprint 3

ID	Name	Implementation
US13	Content-based filtering	When swiping on an experience, there is now an algorithm that registers what type of experiences a user has liked and uses this information to display similar experiences.
US17	Navigation on website	There is now a navbar visible to the user. The navbar contains links to "Home", "Experience Swiper", "Saved Experiences" and "Login"
US18	Website working on the internet	The website is now deployed and can be visited online at https://ac-testenv-328009.web.app/
US8	Create user	A user can now press the "anonymous login" button or "Create user" to start swiping as a user. The data from the swiping is then registered to this user.

The value we provided in this increment was implementing the content-based filtering algorithm to show the user more relevant experiences, easier navigation on the website, online deployment, and user creation. This sprint had two unfinished user stories.

Table 17: Unfinished user stories from sprint 3

ID	Name	Explanation
US16	Mobile-first	Various parts of the website have been made responsive and are now more user-friendly on mobile. However, not all components have been made responsive, and the swiping animation is not easy to use on mobile.
US9	Log in	A user can log in from a computer, but not from a mobile device.

While working on US16, mobile-first, Google Chrome developer tools for mobile was used to emulate how the website would look and behave while using a mobile device. During the sprint, the team discovered that there were various differences in behavior between the mobile version of the website viewed in Google Chrome developer tools and the actual behavior of the website while viewed on a mobile device. Therefore, multiple parts of the website had to change. Due to this

unforeseen difference in behavior being discovered late in the sprint, US16 was not finished before sprint 3 concluded.

When implementing US9, Google Firebase Authentication was used to login users. During development, the newly implemented functionality behaved as intended on a computer. However, when running the application on a mobile device, the implementation failed due to what the team later found out to be bad placement of the Firebase API config keys. As the functionality worked properly from a computer, the team struggled and spent time debugging resulting in US9 not being completed this sprint.

The last item on the agenda for the sprint review was the discussion of possible new features to implement in sprint 4. Together with the customer, we concluded that swiping on experiences as a group would be the main focus for the next sprint. Therefore we had a quick brainstorming session on what the design of the group swiping feature could look like. There was a five-minute session where everyone present in the meeting sketched out their individual design suggestions using pen and paper. We then discussed some of the suggestions. As the scheduled meeting time came to an end, all the sketches were finally uploaded to the slack channel we share with the customer.

This was our only physical sprint review. The team experienced that in this physical meeting, the flow of the conversation was more natural than in the digital meetings. It was also easier to share the individual design ideas as these were sketched out on paper and easy for everyone physically present to see. The team also found the meeting to be more personal. Obviously, being physically present is more personal than meeting online. However, both before and after the "official meeting", we had a more casual conversation with the customer about personal day-to-day topics not related to the project. This casual conversation was a nice way to get to know the customer a little better on a personal level as there is seldom any casual conversation in the digital meetings. That being said, this physical meeting was more time-consuming than the digital meetings, given that the team members had to travel to the city center in order to participate.

Following the sprint review, the product backlog was updated (appendix C.4).

8.3.4 Sprint Retrospective

Following the structure of the previous retrospectives, we met digitally and used a Miro board to summarize the discussion, figure 21. The meeting was held on 15. Oct and took one hour. Most of the team agreed that this sprint was highly productive and we had a good workflow. We were able to complete most of the user stories assigned to this sprint, and we managed to set up our website in the cloud. During this sprint, we continued to base the tasks in GitHub on the complete user stories. Naturally, this led to several members working on tasks together, which some noted as an important motivational factor.

Even though the team experienced overall communication as good, several team members still mentioned problems like lack of transparency and status updates. For instance, many of us still forgot to update the burndown chart and the GitHub board during development. This made it difficult for the rest of the team, who were not working on the specific task, to keep up to date on the progress. This may lead to another problem: the review preparation, which the team experienced as problematic. This issue has been mentioned during every sprint retrospectives. The improvement point from the previous sprint about "code freeze" did not work as intended. Due to huge differences in the codes when merging them together, the code freeze was broken in order to fix the merge conflicts.

In order to refine the development process for the upcoming sprint, the team mentioned several improvements. The heart on the notes in figure 21 indicates that a member wanted to commit to the specific improvement. We then discussed the most voted improvements in more detail.

- In order to improve transparency and the lack of updates, we decided that everyone had to update the issues they were working on in GitHub. Concretely, we should add a checklist of subtasks as we are discovering them and update them when they are completed. This was previously done by some team members, as seen in figure 20, but we felt the need for

everyone to follow. We believe that this will have a great effect on transparency, but it would also allow more seamless collaboration. Further, we implemented a new channel on Slack called updates. The point of this channel was to be an open space where the members were encouraged to post their progress. This includes completion of subtasks, any problems that have been encountered, or any questions regarding the progress of other members. Hopefully, this channel will result in increased communication and transparency.

- Implementing code freeze for this sprint did not fix the problems when preparing for the sprint review. We still encountered several problems when merging functionalities that we could not resolve before the review. To improve this we agreed that we should do continuous integration to a bigger extent than previously. This involves making changes directly into the main development branch, where this is natural but also pushing code from branches when subtasks are completed.

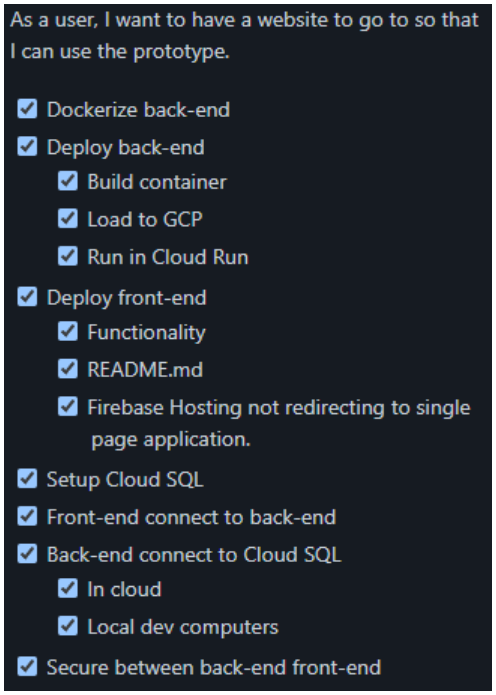


Figure 20: Following the third sprint review, we decided that everyone should try to keep the issue in GitHub up to date.

Figure 21: The Miro board from the sprint retrospective of sprint 3.

8.4 Sprint 4

Sprint 4 was scheduled from 16. Oct to 29. Oct and the sprint goal was to *experiment with group swiping*. Group swiping was a new and interesting feature that Heyloft wanted to explore in order to see its viability. An overview of the Scrum events during the 4th sprint is shown in Table 18.

Table 18: Sprint 4 events

Sprint 4 (16. Oct - 29. Oct)		
Event	**Date**	**Duration**
Sprint planning	16. Oct	2 hours
Sprint review	28. Oct	1 hours
Sprint retrospective	29. Oct	1 hour
Daily scrum	19. Oct, 22. Oct, 26. Oct, 29. Oct	15 minutes each

As with previous sprints, a burndown chart was used to give an overview of the progress, Appendix L.3. From the burndown chart, it is quite clear that we over-committed this sprint, which is further discussed in section 8.4.4. Other than that the burndown chart shows a nice linear progression, especially in the middle of the sprint. Following the retrospective of sprint 3, we also updated the issues in Github extensively, which gave a nice overview of how everyone was doing.

8.4.1 Sprint Planning

The sprint planning meeting for sprint 4 was held on 16. Oct and took two hours. The outcome of the meeting was a sprint goal, *experiment with group swiping*, and the sprint backlog (appendix E.4). Before the sprint planning meeting for sprint 4, the team had received a MoSCoW priority list of user stories for sprint 4 from the customer. Unlike previous sprints, the team did not send their suggestions for user stories to the customer before the sprint planning. Instead, they had, in collaboration with the customer, a brainstorming session about the group swiping session. This then became the basis for the MoSCoW list the customer created. During the sprint planning meeting, the product backlog was refined. In the product backlog, the unfinished user stories from sprint 3 were prioritized since they aimed to improve usability, and the customer was planning to start user testing during sprint 4. Therefore, it was natural to have these user stories in the sprint

backlog. The team used planning poker to estimate the amount of effort for each user story, and then put them into the kanban board on GitHub.

8.4.2 Increment

This increment added the initial group functionality along with a way to guide the user through the site. This was also the sprint where mobile-first was added as shown in Figure 22.

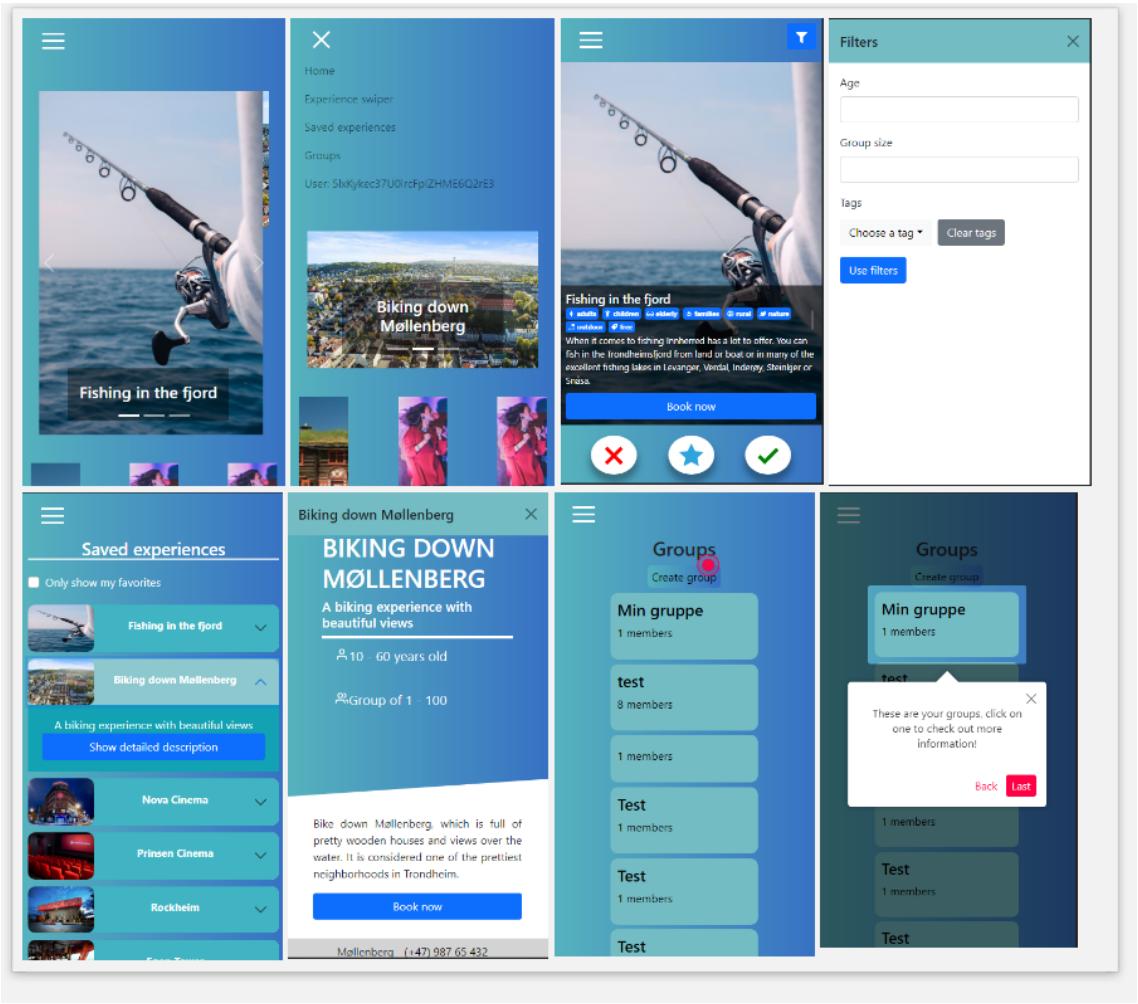


Figure 22: Snapshot of the sprint 4 increment.

8.4.3 Sprint Review

On 28. Oct we had a one-hour digital sprint review to conclude sprint 4. On the day of the sprint review, the team had a preparation meeting at 12:15 to prepare for the upcoming sprint review at 15:00. In the preparation meeting, we merged all finished code into the main branch of the repository. After the merge, we discovered that there were conflicting names in the CSS files which led to multiple components not being styled the way we intended. This was fixed and we pushed the progress we had made in sprint 4 to the production branch, which is the version of the code that is deployed online.

The sprint review started off with the team sending the customer a new URL for the website which is now hosted at https://attractive-city.web.app. We then let the customer explore the new

website's functionalities. These functionalities corresponded to the finished user stories of sprint 4, figure 19.

Table 19: Finished user stories in sprint 4

ID	Name	Implementation
US16	Mobile-first	The website is now responsive and easy to use for mobile users.
US9	Log in	The user can now log in with a personal or anonymous account.
US20	Group creation	The user can now create a group which they intend to swipe on experiences with.
US21	Group sharing	The user is now given a URL they can share with people they want to join the group. If the URL is visited the user gets the choice to either dismiss the invitation or to join the group.

The value provided in this sprint was a responsive website, a finalized log in functionality, and the possibility for a user to create a group. When a group is created, the owner of the group is now provided with a URL that can be shared with people that the group owner wants to join the group. This link is fully functional and allows other users to join the group. In sprint 4, we had four unfinished user stories, figure 20.

Table 20: Unfinished user stories from sprint 4

ID	Name	Explanation
US26	Group swiping introduction	The user is now shown some information on how to create a group and share the invitation link, but the introduction as a whole is not finalized.
US22	Group swiping	Not started
US23	Group results	Not started
US25	Group preferences	Not started

During this sprint, we had underestimated the workload of the group-related user stories. Regardless of the high amount of unfinished user stories, we as a team felt that we had done much of the necessary backend work required in order for the group-related user stories to be completed by the next sprint.

The customer was pleased with the new functionality regarding group creation and sharing. The customer also liked how the website now looked on the mobile version. While exploring the new functionality, the customer was confused by an issue regarding redirection after logging in and pointed out that this also might a problem for the users during user testing. The team took note of this.

Following the sprint review, the product backlog was updated (appendix C.5).

8.4.4 Sprint Retrospective

To finish the sprint the team met online and used a Miro board for the sprint retrospective, figure 23. The meeting followed the same format as previous sprints with all members present and was scheduled for 29. Oct. It lasted for one hour. During this sprint, many of the team members reported good communication and better status updates. The team felt that we had a better overview of who was doing what and when tasks were completed. This improvement can, to some extent, be attributed to the implementation of a new update channel on Slack, following the retrospective of sprint 3. The team also agreed that we got a lot of functionality done and into production. Some team members pointed out that this was due to more frequent use of continuous integration (CI). During this sprint, the team felt that we were better at taking breaks on our workdays, which was something we have been trying to improve since it was brought up during the retrospective of sprint 2. As the project is coming to an end, some members reported that

they are pleased with how the prototype has turned out. We also felt that we have created value for Heyloft.

Many things went well this sprint, but we also experienced things that could have been better. Several team members were not happy with the number of user stories that were not completed in this sprint. After discussing this for a bit we agreed that this was mostly because many of us got sick during the first week of the sprint, which in turn affected our productivity. However, we also felt that we might have underestimated the effort needed for some user stories during the sprint planning meeting. As a result, we committed to more user stories than we were capable of completing. As the project has grown in complexity we have experienced a degrading code quality making it harder to implement new features. We believe that this was an important factor as to why we could not complete as much as usual during this sprint. Unwanted behavior due to conflicting names in our CSS files is one example. The sprint reviews have been a topic in many of our previous sprint retrospective meetings. Even though some things have improved, we still believe there are things we could do better. This time we experienced the structure of the sprint review to be a bit lacking, which led to an awkward and cumbersome experience for Heyloft. Another issue we have noted is that we, as a team, are too silent during meetings with the customer or with our supervisor. When a question is asked of us or feedback is required, the team tends to stay silent waiting for someone else to speak. We identified that this happened because we did not state who should speak on behalf of the group. This lead to awkward silences since no one felt they had the authority to represent the group.

The most important part of the retrospective is to decide what we can do to improve the development process. Again, the heart in figure 23 indicates that a team member wants to focus on this for the next sprint. To conclude the sprint retrospective we discussed the most voted improvements and decided to focus on the following:

- To improve the structure of the sprint reviews we want to write an agenda for the next sprint review. This agenda should include a summary of the sprint, which stories were completed, and a structured demo of the increment.
- To improve the degrading code quality, we think it is important to refactor some of our old code. This includes restructuring of components, rewriting functions, and creating better folder structure. Hopefully, this will increase the modifiability of the system. As an additional point to this, we decided to implement a more specific CSS naming convention to avoid CSS conflicts.
- To avoid silence in meetings, we think it would be a good idea to introduce a facilitator. The role of the facilitator is to take responsibility for breaking the silence and redirecting questions to the appropriate team member.

Figure 23: The Miro board, summarising the discussed point, after the sprint retrospective meeting.

8.5 Sprint 5

The final sprint, sprint 5, was scheduled for 29. Oct to 5. Nov. This was a short sprint that only lasted one week. The team was not satisfied with the remaining user stories from sprint 4 and felt it was necessary to do a final sprint to complete the group swiping functionality. The goal of the sprint was therefore to *finish and polish* the prototype. The team also felt that the code quality had degraded a bit and wanted to refactor some code in this sprint as well. Table 21 shows an overview of the Scrum events from sprint 5.

Table 21: Sprint 5 events

Sprint 5 (29. Oct - 5. Nov)		
Event	**Date**	**Duration**
Sprint planning	29. Oct	1 hour
Sprint review	5. Nov	1 hour
Sprint retrospective	5. Nov	1 hour
Daily scrum	2. Nov, 5. Nov	15 minutes each

Even though the sprint was a bit shorter than usual, we used a burndown chart to track progress, Appendix L.4. As with many of the previous sprints, it is clear, from the burndown chart, that most of our work is done on the main development days. For this sprint, this was on 2. Nov.

8.5.1 Sprint Planning

Being that sprint 5 was the last sprint, the team suggested to the customer that they focus on finishing the unfinished user stories from the last sprint as well as focusing on minor improvements to the usability of the product. The customer agreed, and therefore there was no need for new user stories from the customer. The new user stories that were added to the product backlog were developed by the team themselves and aimed at improving usability, based on their own and the customers' experience with the product. During the sprint planning meeting, the product backlog was refined similarly to the previous sprint planning meetings and a sprint backlog was created (appendix E.5). These were chosen in line with the sprint goal: *finish and polish*.

8.5.2 Increment

This increment added the final functionality for the group swiping and improved the homepage, this is displayed in Figure 24.

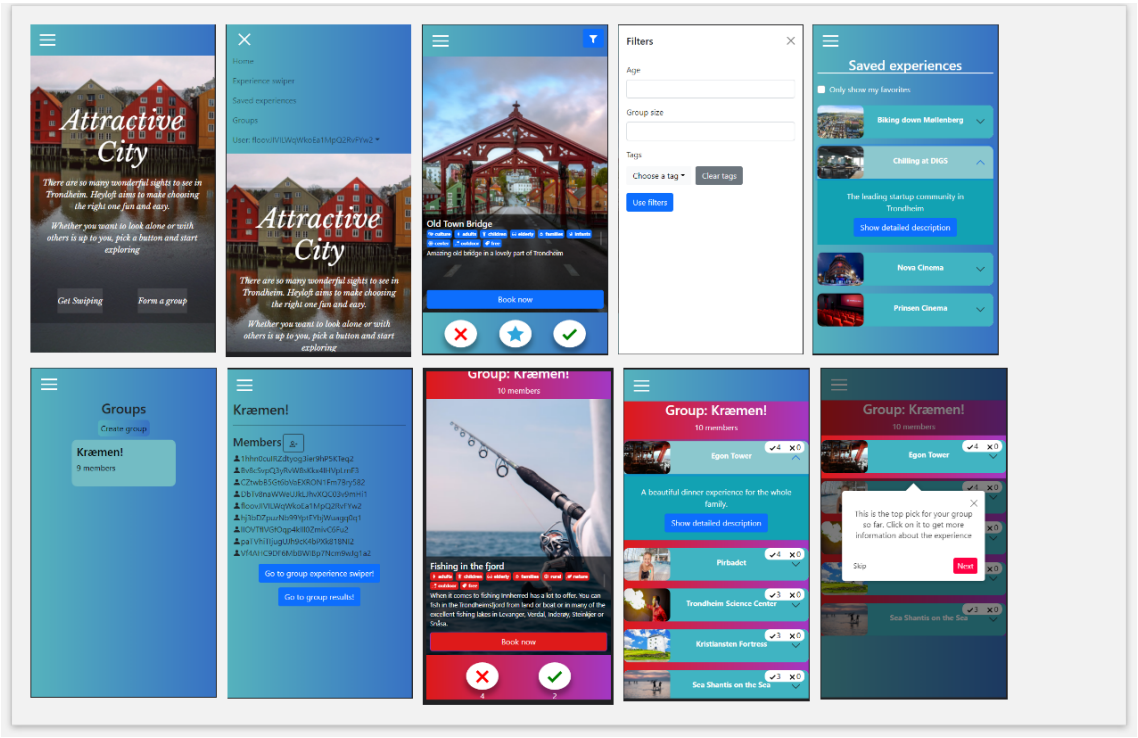


Figure 24: Snapshot of the sprint 5 increment

8.5.3 Sprint Review

The final sprint review was held digitally alongside a technical hand-off on 5. Nov. Due to this, it lasted half an hour longer than usual and was set to Friday at 13:00 instead of Thursday at 15:00 upon the customers' request. Before the review, the team sat down to go through all of the user stories to make sure everything was working as intended on the production server. This time was also used to get rid of the final bugs and run through the flow of the software.

At 13:00 the demonstration started with the customer trying out the new functionality. This was done by having a team member go through the different user stories while having the customer try to locate and use the corresponding features. The customer voiced his thoughts while navigating through the user stories, and seemed to do so easily. The customer was also very happy about the progress made in this sprint, and the quality of the implementation. During this sprint, the team managed to complete six out of eight user stories. These user stories are shown in Table 22.

The provided value in this sprint was implementing the group swiping functionality, results visualization, and introduction to the page. The team had two unfinished user stories listed in Table 23, these were the ones with the lowest priority from the sprint planning.

Since this was the final sprint, no further user stories were discussed. It was, however, mentioned that the current improvements corresponded well with the results from the external user tests concluded by the customer prior to the sprint review. After the review, the team had a technical hand-off where they walked the customer through the technical solutions and the information listed in section 11.

Following the sprint review, the product backlog was updated for the final time (appendix C.6).

Table 22: Finished user stories in sprint 5

ID	Name	Implementation
US22	Group Swiping	The user can now swipe on experiences in a group setting.
US23	Group Results	The user is now shown a list of the experiences which got the most likes from the group the user is a part of. How many likes and dislikes each experience received from the group is also visible to the user.
US26	Group swiping introduction	When creating and joining a group for the first time, the user is now given short introduction to how to use the group swiping functionality. The information is displayed in the form of tooltips.
US29	Log out	There is now a button which the user can click to log out.
US30	Redirections	After logging in, the user is now redirected to the last path clicked before being redirected to the login screen.
US31	Information on homepage	The homepage of the website now displays a short informative text about the website and provides you with the option to start swiping individually or to create a group.

Table 23: Unfinished user stories from sprint 5

ID	Name	Explanation
US32	Anonymous user nick-name	Not started
US25	Group preferences	Not started

8.5.4 Sprint Retrospective

The sprint retrospective was held 5. Nov and lasted for one hour. Like previous retrospectives, it was held digitally, with Miro as the working tool and with everyone present. The Miro board can be seen in figure 25. Most of the team members agreed that the team had been very productive during this sprint. The focus on improving communication, which had come up in multiple previous sprints, has given results, and most team members believed the communication and status updates were great. The introduction of a facilitator role, which was given to the Scrum Master, led to more natural conversations in the meetings with the customer and the supervisor.

Even though the members had mostly positive things to say about this sprint, some members expressed some problems with organizational tasks such as updating the burn-down chart and pushing the code to the production branch. The former could be attributed to the frequent use of the update channel in Slack, and that the members therefore felt the other members were sufficiently informed through that.

Since this was the last sprint, the improvements mentioned in the retrospective will not be implemented, but were found to be a good pointer for later projects. These improvements consisted of:

- To improve the use of the burndown chart the team wanted to either be more consistent in updating the chart or drop it completely in favor of other ways of updating the group.
- The team has decided to update the definition of done to include having the code pushed to the production branch. The reason for this is to keep the online website as up-to-date as possible so that all functionality can be tested in a production environment.

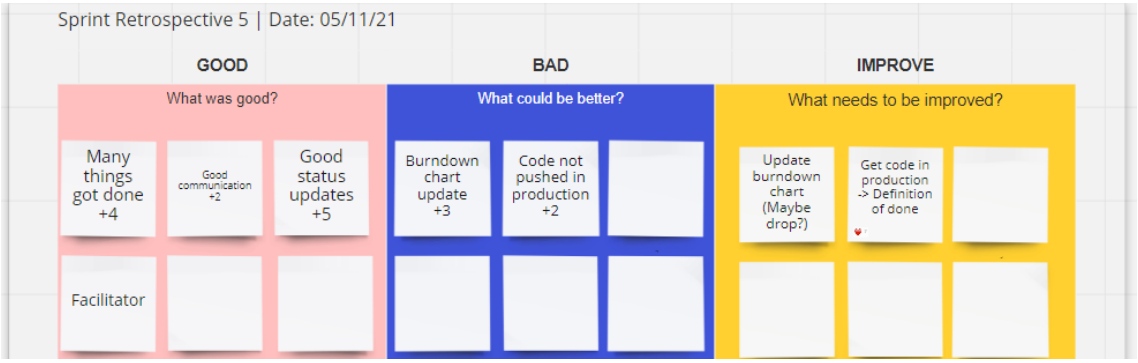


Figure 25: The Miro board was utilized during the retrospective of sprint 5.

9 Security

When developing a software system, it is necessary to take security into account and be aware of the security risks. Since there are many potential threats and vulnerabilities to the system, the team has tried to identify the most critical ones. That said, security was not the main concern while developing the project as the team was developing a prototype with dummy data. In addition, the customer wanted a functional prototype, which is why the team prioritized implementing functionality features before security features.

However, the prototype still has some simple security measures. For instance, the system requires a proper e-mail address and password in order for a user to create and/or log in to a user account to fully experience the functionalities the prototype has to offer. Some of the security is also provided by Google itself, through the use of Google Cloud Run and Firebase. Also, both the user IDs and the group IDs are created using unique identifiers (UID). These are randomly generated values that are difficult to guess and brute force.

In case of further development of the prototype, the team has tried to identify the potential security issues that may be critical in a production-ready system. The following sections consist of a threat model of the prototype, including possible security threats and vulnerabilities.

9.1 Threat Model

Threat modeling is a structured process to identify and mitigate the security issues associated with a system [5]. The execution of the process found place Saturday 6. Nov and lasted for three hours, which resulted in a threat model.

The main goal of the threat modeling was to think like an attacker and be aware of the security risks in the prototype. In order to do that, the team first identified the critical assets in the system (Table 24). Since the assets are the resources of value [30], the team also reflected on why the assets needed to be protected against potential attackers. After those were pointed out, the potential risks were identified and associated with the assets. Later on, the team moved on to map out all the places that an attacker can enter and (mis)use the system – the attack surfaces (Table 25). Also, a misuse-case diagram was drawn (section 9.1.1). Lastly, the team identified the security threats (section 9.1.2) and vulnerabilities (section 9.1.3).

Table 24: Overview of the critical assets, a description of them and potential risks.

ID	Asset	Description	Risks
A1	Application	The Firebase hosting website (link: https://attractive-city.web.app/) that the users access when using the prototype.	Service temporarily denied. Service lost. Service tampered.
A2	Codebase	The collection of source code (link: https://github.com/heyloft/attractive-city) that involves the application's logic and functionalities written by the developers.	Codebase disclosed. Codebase lost.
A2	Information on experiences	The application's main feature is to swipe on experiences. Without correct information, the application may not work optimally.	Experience information disclosed. Experience information lost. Experience information tampered.
A4	User credentials	The credentials that a user will use to log into the application.	User credentials disclosed. User credentials lost. User credentials tampered.
A5	User information	The application will store personal data about the user and how they use the functionalities, like e-mail address and the results from swiping on experiences.	User information disclosed. User information lost.

Table 25: Overview of the attack surfaces and possible attacks from an potential attacker

Attack surface	Possible interaction
Front-end application	Spoofing, brute force, malicious content.
Back-end API	Spoofing.
Google Cloud Platform	Malicious actions toward the database.

9.1.1 Misuse-case

A misuse-case diagram (Figure 26) was drawn by the team to visualize and understand how an attacker may attack the system, and hence identify the threats, while actions were taken by the system [19]. Figure 26 shows (mis)usecase of the system from a developer's, a user's, and an attacker's perspective. The misuse-case diagram also allows other non-tech stakeholders to get a better understanding of the security threats.

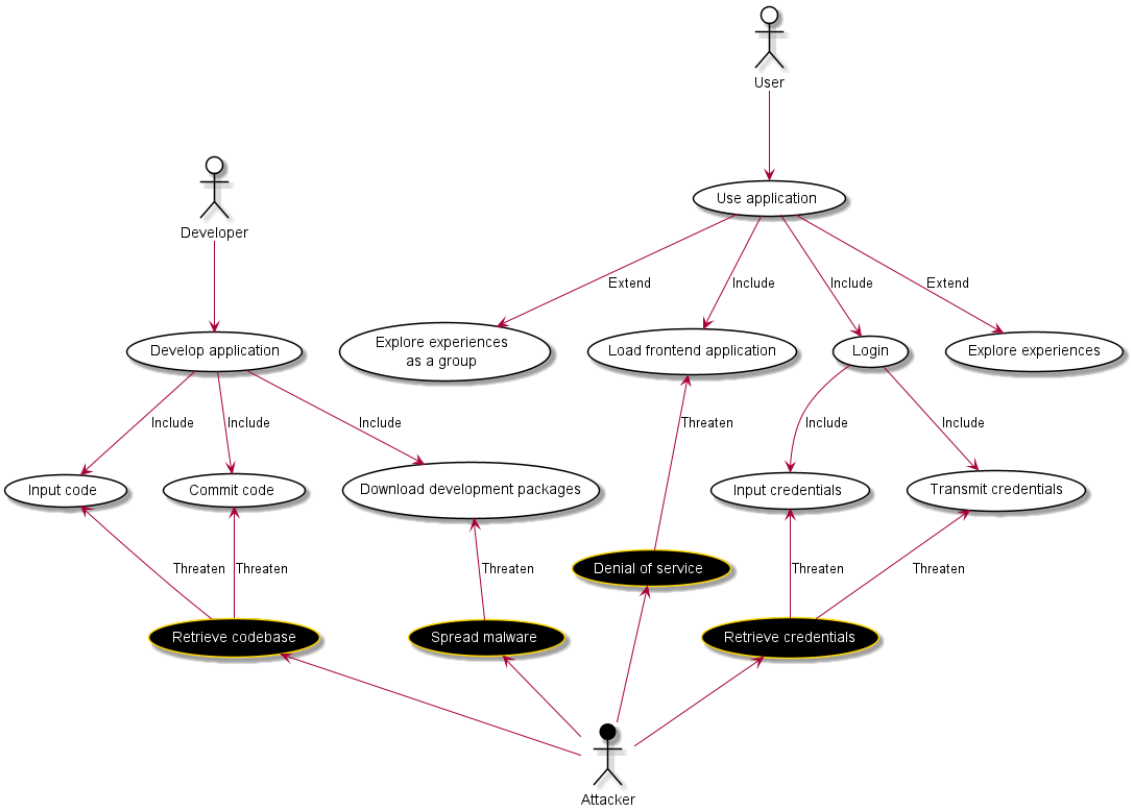


Figure 26: Misuse-case diagram of the system

9.1.2 Security Threats

Microsoft defines a threat as "an undesired event. A potential occurrence, often best described as an effect that might damage or compromise an asset or objective" [30]. The team used the threat mnemonic STRIDE, developed by Microsoft, to identify and uncover the threats [27]. STRIDE is an acronym for the six main threat categories: spoofing (S), tampering (T), repudiation (R), information disclosure (I), Denial of service (D), and elevation of privilege (E). STRIDE helped the team answer questions like "what can possibly go wrong with the system?" and "how bad can it be?". The team found spoofing – an attacker pretends to be someone or something else, information disclosure – an attacker sees data without permission and Denial-of-Service – an attacker reduces the availability of the service to legitimate users [29] as the most relevant threat categories (Table 26). After identifying the potential threats, the team also thought about mitigation mechanisms.

9.1.3 Security Vulnerabilities

Since many software are crippled with vulnerabilities, it is fair to assume that this is the case with the prototype as well. Microsoft defines vulnerability as "a weakness in some aspect or feature of a system that makes an exploit possible" [30]. In a prototype, the risks are less serious than if it was a system in production. Nevertheless, in the context of a production-ready system with actual users and actual data, these points should definitely be taken into consideration.. Below are the security vulnerabilities (Table 27) the team found critical and suggestions on how to mitigate them.

Table 26: Identified threats using STRIDE and possible mitigations

Threat	**Category**	**Mitigation mechanism**
An attacker can register views on behalf of another user, both inside and outside of a group context.	Spoofing	Only allow request made by front-end to back-end.
An attacker can spread malicious software, like key loggers and background crypto miners, to the users of the system.	Spoofing	Take regularly backup of important data. Block and log unknown requests to both back-end and front-end.
An attacker can extract ratings of experiences by a user.	Information disclosure	Only allow requests made by front-end to back-end. Ensure that the user ID of other users is more difficult to obtain.
An attacker can get a user's e-mail.	Information disclosure	Apply encryption in the database. Ensure that the user ID of other users is more difficult to obtain.
An attacker can cause a Denial-of-Service attack during user testing.	Denial-of-Service	Follow and apply the guidelines from Firebase security checklist: https://firebase.google.com/support/guides/security-checklist

Table 27: Security vulnerabilities in the prototype and possible mitigations

Vulnerability	**Mitigation mechanism**
Possible to perform an HTML injection in the group name when creating a new group.	Validate and ensure that the input field does not contain HTML code and/or other script code.
Possible to blackmail/bribe team members into giving up credentials, for instance to Google Cloud.	Limit permissions by only giving access to those who need it. Monitor/log unusual activities.
Possible to brute force the password and hack into the user account.	Apply a maximum amount of tries. Require a strong password with a mix of letters, numbers, and symbols.
Possible to read user e-mail and password from the state in the browser.	Encrypt e-mail and password state.
Possible that the system uses packages with dependencies in front-end (figure 27) and back-end that may have security problems.	Keep the packages up to date. Install the newest version of the packages if possible, and check for vulnerabilities regularly.
Database credentials are stored in the GitHub repository.	Currently relying on GitHub's security. Move the credentials to a more secure and encrypted location with limited access.
Google account can be compromised, which gives attackers access to Google Cloud and the database.	Ensure that everyone with access permissions has two-step verification enabled.

```
audited 2587 packages in 51.018s

182 packages are looking for funding
  run `npm fund` for details

found 31 vulnerabilities (12 moderate, 18 high, 1 critical)
  run `npm audit fix` to fix them, or `npm audit` for details
```

Figure 27: Example of a security vulnerability: number of packages, related to front-end, installed in the project. Checked on 5. Nov 2021

10 Testing

Testing is a vital part of developing software. Effective test routines verify that the implemented functionality works as intended, and it detects bugs and faults in the source code that could be fatal if put into production. This section will elaborate on how the team dealt with testing and the type of testing that was done in this project.

10.1 Test Plan and Routines

As the purpose and focus of this project was to create functional prototypes for the customer, it was not necessary to focus as much on testing as if the team was to implement a fully working distributed end-product. However, to ensure the quality of the product, the team decided to write tests for the implemented features, and to enhance tested software as our definition of done (section 5.2.2).

Rather than creating a rigid test plan, the team agreed upon some routines in terms of testing. To ensure that software components and the data flow through the components worked as intended, integration tests were written when adding new features. In addition, to guarantee that the team implemented what the customer really wanted, acceptance testing was also conducted. Furthermore, the customer also arranged some user tests which gave the team valuable insights into the design of the product.

10.2 Integration Testing

The team's definition of done (section 5.2.2) required tested code, and integration tests were written for this purpose. The widely used Jest framework was used to write these tests. For the back-end part of the application, writing integration tests was fairly easy and effective. Here, tests were separated into test-suites, which contained tests that verified that the normal code's behavior was as intended, and that special and bad cases got handled accordingly. The tests called the back-end API of the application with differently constructed requests, tested the flow through the system, and that the database updated data as expected. To secure that the tests were not disturbing the production database, a separate replication of the database was created for testing. As seen in Figure 28, the overall coverage of the tests in the back-end is high, hopefully preventing faulty changes to the codebase in the future.

The team found it struggling and time-consuming to write good integration tests effective in the front-end part of the application. Furthermore, the team felt that it also added little value to the product as the code was naturally tested manually during development. Due to the time aspect as well as testing not being the main focus, the team agreed that integration testing in front-end was excess. In addition, as the customer wanted a mobile-friendly application, device testing was also performed on mobile as a part of the manual testing routine.

10.3 User Testing and Feedback from Customer

User testing is a great tool to ensure that the software is user-friendly, and to get valuable insights into how users interact with the product. The customer arranged some user tests at the end of sprint 4 and found some interesting aspects of improvement. During sprint 5, the team and the customer had a meeting where the results and finds of the user tests were presented, appendix K. The feedback contained both good points and points of improvement, and the team did changes to the product accordingly. In addition, during the sprint reviews, the customer also had some immediate feedback points that the team found valuable and acted upon.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	93.15	81.19	94.89	93.14	
backend	96.3	81.03	99.01	95.98	
carousel_controller.ts	92.85	100	100	92	38-40
database_connector.ts	98.79	83.33	100	98.68	56,448
endpoint.ts	100	100	100	100	
experience_controller.ts	84	66.66	87.5	83.33	26,57-61
group_carousel_controller.ts	98.55	88	100	98.43	134
index.ts	100	100	100	100	
python_connector.ts	86.66	50	100	86.2	20-25,51
user_controller.ts	100	100	100	100	
backend/routes	89.2	79.06	80	89.89	
carousel.ts	91.3	91.66	100	90.9	41-42
experiences.ts	86.36	100	50	90	44,76,84-85
group.ts	87.6	65.21	87.5	87.61	60,82,110,117-131,163-169,211,246,271
user.ts	100	100	100	100	
backend/utils	85.29	87.5	100	84.37	
config.ts	76.47	33.33	100	76.47	42-51,64-71
controllers.ts	94.11	100	100	93.33	31

Test Suites: 5 passed, 5 total
Tests: 51 passed, 51 total
Snapshots: 0 total
Time: 81.349 s, estimated 86 s
Ran all test suites.

Figure 28: Test-coverage in the back-end.

11 Internal and External Documentation

Since the team will not be working on the project forever, we have created a user- and installation guide. The necessary prerequisites and instructions on how to run the system both locally as well as on the server are provided in the Appendix A. This includes step-by-step instructions on the setup and execution of the different parts of the code.

12 Evaluation

This section provides an evaluation of several aspects of the project, including reflections around the internal process and results and evaluation of the course.

12.1 The Internal Process and Results

At the beginning of this project, it became clear that the members had roughly the same ambitions and expectations for this course. All team members have been reliable and trustworthy throughout the project and the workload has been distributed evenly. Therefore, the consensus of the team is that the team has been working well together. Through this project, the team members have been assigned tasks that were in the interest of the members working on them. For example, the team members that specialize in artificial intelligence worked on developing the AI-based recommendation system, and the team members with expertise in front-end development were assigned front-end tasks. By assigning tasks and having trustworthy team members, the internal process has been both exciting and valuable for the team members.

In the early phases of this project, the internal communication was not optimal. This led to some lack of internal updates and a slight reduction in productivity. The team thought that a reason for this could be that the team members did not know each other well in the beginning. To counteract this, the team could have held a team-building event at the beginning of the project, as this hopefully would have brought the team members closer.

As mentioned earlier, the team adapted Scrum, which held many advantages for the team. For instance, the team has not had any conflicts throughout the project. Problems were detected and resolved at an early phase during the retrospective meetings. In addition, the iterative nature of Scrum meant that the team could get a lot of feedback from the customer. The Product Owner

and the Scrum Master both learned and passed information and experiences regarding their roles to the rest of the team.

The team felt that they have created value for the customer by exploring and developing prototypes that fit with the "marketplace of experiences" visions.

12.2 The Customer and the Project Task

The team felt they had very good communication with Heyloft throughout the project. From the team's perspective, Heyloft were quick to respond to questions, flexible in when they could meet, and very engaged in the project. This made it easier for the team to clarify details and issues with Heyloft, while also making the planning and development of the prototype go much smoother. One of the designers at Heyloft lives in Canada, and because of differences in time, the communication was a bit delayed. However, this did not turn out to be a major issue.

The team felt that the project assignment was vague in the beginning as no specific requirements were given by Heyloft, but in retrospect the team found this to be positive as it gave the team freedom to brainstorm different ideas as well as letting them learn to concertize vague and open ideas. The freedom to discuss and suggest functionality and implementation that was given by Heyloft, was something the team really appreciated.

12.3 The Supervisors

The team as a whole, felt that we had excellent communication with our supervisor, Orges Cico. In the starting phase of the project, we found Orges to be somewhat demanding. In retrospect, we think that this was an important factor to help us swiftly build a solid foundation for the project. Orges also encouraged us to set multiple deadlines for the various sections of the report throughout the project. On each of these deadlines, Orges would read through what we had written and suggest improvements when necessary. This helped the report to become a document which was refined over time. His multiple thorough read throughs, quick replies and constructive criticism has benefited our project greatly. The team really think he as gone above and beyond in regards to supervision this fall.

12.4 Further Work with the Project

Both Heyloft and the team are satisfied with how the project turned out, but there is still room for improvements. Since the project is a prototype, the team finds it difficult to estimate how much effort is necessary to complete the project, because there will always be more features to explore and current features to maintain. Because the project is developed using known technologies, such as React in front-end, Node.js in back-end, and a Google Cloud database, the team considers the transition to new developers to be relatively easy. When working further on the project, the team suggests the following focus areas (not in prioritized order):

Technical challenges

Even though the prototype works well, the team has later discovered some minor bugs that should be fixed. For instance, the loading indicator in the experience swiper keeps spinning when the user has swiped through all available experiences. However, due to limited time on the project, the team prioritised exploring new features, instead of fixing those bugs while developing, even though it could improve the overall user experience.

Explore and Add New Features

During the last sprint, the team did not manage to complete all the user stories (Table 23) due to limited time on the project. The team believes that the unfinished user stories will contribute to an even better experience of the group swiping feature once they are finished. As the prototype is

still in an early phase, there are more features to explore. If the team had enough time, we would keep on brainstorming and find solutions to reduce the identified pain points (Section 4.3).

Apply the Mitigation Mechanisms

The team did not prioritize developing security features, as mentioned in section 9. After threat modeling, the team found several security vulnerabilities that may be critical in a production-ready system. Therefore, the team suggests focusing on the mitigation mechanisms in order to reduce the security risks that were identified. Although the project is still å prototype, ensuring the user's safety when using and/or testing the system is important.

Testing

To ensure the quality of the system, the team wrote tests for the implemented features (section 10). Although the test coverage for the back-end is high, the team found it struggling to write good and effective tests in the front-end. Further work is to focus more on testing the front-end to ensure quality and detect bugs before putting the code into production. Another solution is to continue with user testing, and thus receive both good and improvement feedback to act upon.

Further work with AI

In section 4, the different AI recommendation system possibilities are described, as well as the choices the team made. The team chose to use the content-based filtering technique in the recommendation system, as other techniques require data on user preferences, which Heyloft did not have. The prototype collects this type of user preference data, which means that given enough data, other AI techniques can be explored. Collaborative filtering is a technique that would be interesting to explore, as it is very popular and used by companies like Netflix and Amazon in their recommendation systems [18]. On top of collaborative filtering, one could apply machine learning when predicting users' preferences for experiences they have not rated yet. This could improve the performance of the recommendation system, and hence give better recommendations to the user.

12.5 Suggestions for Improvement in the Course TDT4290

Even though the team was mostly happy with the course, we have identified a few improvement points. These are described in the following paragraphs.

More Relevant Lectures

While some lectures proved to be very useful, there were a few the team found to have potential for improvement. Some of the lectures seemed quite irrelevant to the course, such as the one on working in major teams. Among the more relevant lectures like the ones on agile and security, the team believes they should have been earlier in the semester to easier implement the theory from the start. The team also found that some lecturers seemed to have a hard time connecting their topics to the course. The impressions of each individual lecture are listed in section 13.

More Consistent Compendium

The team found that the compendium succeeds in giving the students a good introduction to the course and the report. It did, however, include some inconsistencies when compared to the lecturers. This seemed especially clear in the agile/Scrum section. The compendium also seemed to give the impression that the report had to be written in a specific manner, although we have found the apparent hand-in specification to be a little more lenient.

13 Reflection about the Lectures

In addition to working on the project, the team has attended six lectures as part of the course. A variety of topics were covered during these lectures, which has proven useful during the project. This section provides a reflection from the team regarding the usefulness of these lectures, and how

they helped us during the course.

13.1 Project Planning and Management

The first lecture was about project planning and management by Terje Heen from NAV. This lecture was held on 1. Sep and introduced some interesting points that influenced the teams' decisions in the initial phases of the project. Particularly the part about pair programming, which led to us experimenting with this method. Pair programming is something we have used throughout the project and we believe it led to better solutions for some parts of our system. Apart from this and some other concepts, the lecture felt like a demonstration of how NAV has moved from working using the old model to the new model. Even though the points made were valid and interesting they did not seem to be relevant to the situation we were in. NAV is a major company with many cross-functional teams and the lecture focused on how to work in this context. This was not as relevant for us, who are working in a single team of seven people.

13.2 Group Dynamics

The second lecture was a workshop, which took place on the 8. Sep and was held by Kristian Drøsshaug. It was based on a questionnaire by Diversity Icebreaker, which is designed to help identify preferences when communicating and collaborating with others by assigning a color score (red, blue, and green). The lecture brought an awareness of trait differences and how people may react to and understand situations differently. As a result, the team was better prepared to resolve conflicting views. Still, the team felt that the lecture should include more theory on how to utilize the knowledge of trait differences to enhance the group dynamic in a team.

13.3 Agile Development

A lecture on Agile development and Scrum was held on 15. Sep by Orges Cico. Since the team had decided to use Scrum by this point in the course, we found the lecture to be highly relevant. The lecture gave a precise and exhaustive description of Agile and Scrum. Even though many of us had experience with these topics before, we felt that the lecture provided a clearer definition of the values and events of Scrum. These definitions were important, as they ensured that everyone in the team had the same conception of how to work within the Scrum framework. Kanban and XP were also introduced in the lecture, which influenced us into trying their methods, section 5.3 and 5.4.

13.4 Presentation Skills

On 22. Sep, a lecture about presentation skills was held by Eirik Sande. The lecture covered mostly familiar topics, but it was nice to be reminded of them. We were 'forced' to give a mini-presentation, which was a nice touch and forced us to step outside of our comfort zones. The team found this exercise to be helpful in removing some of the fear and anxiety that public speaking often introduces. Having pointers on how to structure a presentation was also useful when preparing for the presentation day.

13.5 Technical Writing

The fifth lecture was held by Serena Lee-Cultura about technical writing, on 29. Sep. The lecture contained a lot of useful information regarding technical writing in general and how to write the report. This lecture was not as relevant to the team. At this point in the course, we had already written two drafts of the report and gotten feedback and instructions from our supervisor.

13.6 IT Security

The fifth lecture about IT security was held by Martin Gilje Jaatun, on 13. Oct. For many in the group this lecture included new and valuable information. It also made us aware of security, which had been disregarded throughout the project. The reasoning for this was that we were developing a prototype. The lecture made us reassess this and understand that being aware of security threats is important, even if countermeasures are not necessary at this stage of the development. As a result, the lecture formed the basis for a security meeting as described in section 9.

Bibliography

[1] *About Tripadvisor*. URL: https://tripadvisor.mediaroom.com/us-about-us (visited on 11th Nov. 2021).

[2] *About visitnorway.com*. URL: https://www.visitnorway.com/info/about-visitnorway/ (visited on 11th Nov. 2021).

[3] L. Bass, P. Clements and R. Kazman. *Software Architecture in Practice*. 2nd ed. Addison-Wesley Professional, 2003. URL: https://people.ece.ubc.ca/matei/EECE417/BASS/index.html (visited on 9th Nov. 2021).

[4] *Client-Server Architectures*. URL: http://www.cs.sjsu.edu/~pearce/oom/ood/distArch/server (visited on 12th Nov. 2021).

[5] Larry Conklin and Victoria Drake. *Threat Modeling Process*. URL: https://owasp.org/www-community/Threat_Modeling_Process# (visited on 8th Nov. 2021).

[6] Houtao Deng. *Recommender Systems in Practice*. URL: https://towardsdatascience.com/recommender-systems-in-practice-cef9033bb23a (visited on 12th Nov. 2021).

[7] *Design Patterns*. URL: https://sourcemaking.com/design_patterns (visited on 9th Nov. 2021).

[8] *Extreme Programming (XP)*. URL: https://www.agilealliance.org/glossary/xp/ (visited on 3rd Oct. 2021).

[9] *Freeze (software engineering)*. URL: https://en.wikipedia.org/wiki/Freeze_(software_engineering) (visited on 6th Oct. 2021).

[10] Carlos A. Gomez-Uribe and Neil Hunt. *The Netflix Recommender System: Algorithms, Business Value, and Innovation*. URL: https://dl.acm.org/doi/pdf/10.1145/2843948 (visited on 12th Nov. 2021).

[11] Mihajlo Grbovic et al. *Machine Learning-Powered Search Ranking of Airbnb Experiences*. URL: https://medium.com/airbnb-engineering/machine-learning-powered-search-ranking-of-airbnb-experiences-110b4b1a0789 (visited on 12th Nov. 2021).

[12] *heyloft - Marketplace of experiences*. URL: https://heyloft.global/ (visited on 9th Nov. 2021).

[13] *Isn't design thinking a set, step-by-step process?* URL: https://designthinking.ideo.com/faq/isnt-design-thinking-some-set-step-by-step-process (visited on 3rd Oct. 2021).

[14] *Kanban*. URL: https://www.agilealliance.org/glossary/kanban/ (visited on 7th Nov. 2021).

[15] Henrik Kniberg. *Scrum and XP from the Trenches 2nd edition*. C4Media, 2015, p. 159.

[16] Shuyu Luo. *Introduction to Recommender System*. URL: https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26 (visited on 12th Nov. 2021).

[17] Rachaelle Lynn. *Introduction to Kanban*. URL: https://www.planview.com/no/resources/guide/introduction-to-kanban/ (visited on 7th Nov. 2021).

[18] Rico Meinl. *Recommender Systems: The Most Valuable Application of Machine Learning (Part 1)*. URL: https://towardsdatascience.com/recommender-systems-the-most-valuable-application-of-machine-learning-part-1-f96ecbc4b7f5 (visited on 12th Nov. 2021).

[19] *Misuse case*. URL: https://en.wikipedia.org/wiki/Misuse_case (visited on 8th Nov. 2021).

[20] *MoSCoW Prioritization*. URL: https://www.productplan.com/glossary/moscow-prioritization/ (visited on 7th Nov. 2021).

[21] *Pain points*. URL: https://www.gartner.com/en/sales/glossary/pain-points (visited on 9th Nov. 2021).

[22] Dan Radigan. *What is kanban?* URL: https://www.atlassian.com/agile/kanban (visited on 7th Nov. 2021).

[23] Max Rehkopf. *User stories with examples and a template*. URL: https://www.atlassian.com/agile/project-management/user-stories (visited on 11th Nov. 2021).

[24] *Risk management*. URL: https://en.wikipedia.org/wiki/Risk_management (visited on 11th Nov. 2021).

[25] Ken Schwaber and Jeff Sutherland. *The 2020 Scrum Guide*. URL: https://scrumguides.org/scrum-guide.html (visited on 6th Nov. 2021).

[26] *Singleton Design Pattern*. URL: https://sourcemaking.com/design_patterns/singleton (visited on 12th Nov. 2021).

[27] *STRIDE (security)*. URL: https://en.wikipedia.org/wiki/STRIDE_(security) (visited on 7th Nov. 2021).

[28] *The Scrum Framework Poster*. URL: https://www.scrum.org/resources/scrum-framework-poster (visited on 7th Nov. 2021).

[29] *Threat modeling framework - Threat categories*. URL: https://docs.microsoft.com/en-us/learn/modules/tm-use-a-framework-to-identify-threats-and-find-ways-to-reduce-or-eliminate-risk/1b-threat-modeling-framework (visited on 7th Nov. 2021).

[30] *Threat Modeling Web Application - Terminology*. URL: https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff648006(v=pandp.10)#terminology (visited on 7th Nov. 2021).

[31] *User Interface (UI) Design Patterns*. URL: https://www.interaction-design.org/literature/topics/ui-design-patterns (visited on 9th Nov. 2021).

[32] *What is PBL?* URL: https://www.pblworks.org/what-is-pbl (visited on 11th Nov. 2021).

[33] *Who We Are*. URL: https://www.expediagroup.com/who-we-are/our-story/default.aspx#module-tabs_item--7 (visited on 11th Nov. 2021).

Appendix

A Documentation

A.1 Frontend

The frontend is developed using ReactJS and Typescript. The hosting is provided by firebase and the testing frameworks utilized are Cypress and Jest.

A.1.1 Prerequisites

These pieces of software are required to run and develop the front-end.

- Node.js

A.1.2 Structure

- `./cypress` contains test for user interface.
- `./public` contains static files such as images and HTML.
- `./src` contains code files.
- `./package.json` contains configuration for Node.js project.

A.1.3 Testing

Run automatic tests to find errors in the front-end. This action should always be performed before pushing to main and deploying.

A.1.4 Setup

These steps must only be followed once, the first time this use-case is performed.

1. Ensure Cypress and Jest are installed with command `npm install`.

A.1.5 Execution

These steps must be performed every time you wish to perform the action in question.

1. Write `npm run cypress:open` in the terminal.
2. Run all Cypress test cases.
3. Run the command `npm test` to open the component tests.
4. Select to run all tests in the prompt that opens.

A.1.6 Deploy website

To deploy the back-end. Push changes to the prod branch. Ensure the front-end is working.

A.2 Backend

The backend utilizes nodeJS and ExpressJS with typescript. It is hosted using google clouad and a docker container and is tested using the Jest testing framework.

A.2.1 Prerequisites

These pieces of software are required to run and develop the back-end.

- Node.js
- Python 3.6 or newer

A.2.2 Contents

This document contains guides to perform a number of actions on the back-end.

- Deploy back-end
- Develop and debug locally
- Use Cloud SQL Auth proxy
- Testing

Each guide is separated into **Setup** and **Execution**; the former for instructions in the one-time setup of the task, and the latter for steps that is needed every time the action is performed.

A.2.3 Deploy back-end

Deploy the back-end to be run in Cloud Run. Note that the front-end will also be deployed.

Execution

1. Ensure the back-end is working by debugging and testing.
2. Push changes to the **prod** branch.

A.2.4 Develop and debug locally

Debug the back-end locally. Note that this still includes connecting to a database running on the Google Cloud Platform.

Setup

1. Ensure Cloud SQL Auth proxy is installed.
2. Ensure the python virtual environment is installed. Steps for this can be found in the README.md in the python folder.

Execution

1. Ensure the Cloud SQL Auth proxy is running. Steps for this can be found in the **Use Cloud SQL Auth proxy** guide.
2. Ensure all packages are installed with command `npm install`.
3. Start the back-end locally with the command `npm run debug`.

A.2.5 Use Cloud SQL Auth proxy

It is necessary to install and run this proxy to connect to the database instance running on the Google Cloud Platform. This access is used both in testing and debugging.

Setup

1. Ensure that your Google account has access to the attractive city project at console.cloud.google.com.
2. Install Google Cloud SDK following the guide at this page. Log in with the account above, region `europe-west1` and zone `europe-west1-d` when using the command `gcloud init`.
3. Follow the steps described here to install the Cloud SQL Auth proxy: https://cloud.google.com/sql/docs/mysql/c admin-proxy#install

Execution

1. Open a terminal at the location of your Cloud SQL Auth proxy and run command:

- macOS / Linux: `./cloud_sql_proxy -instances=ac-testenv-328009:europe-west1:ac-db-test=tcp:19001`
- Windows: `.\cloud_sql_proxy -instances=ac-testenv-328009:europe-west1:ac-db-test=tcp:19001`

A.2.6 Testing

Run automatic tests to find errors in the back-end. This action should always be performed before pushing to main and deploying.

Setup

1. Ensure Cloud SQL Auth proxy is installed.

Execution

1. Ensure the Cloud SQL Auth proxy is running. Steps for this can be found in the **Use Cloud SQL Auth proxy** guide.
2. Ensure Jest and other packages are installed with command `npm install`.
3. Run the command `npm test` to open the tests.

A.3 Database

Experiences are a key part of the attractive city prototype. To make it easier for non-technical people to add, remove and edit information about experiences they are written into a Google Sheet. For the experiences to be used in the prototype they need to be in the Cloud SQL database. For updating the Cloud SQL database with the latest information in the spreadsheet see section *Experiences Sheet Script*

A.3.1 Experiences Google Sheet

Experiences are written into this Google Sheet. There is also a seperate sheet for tags in the same spreadsheet. To view and edit this spreadsheet you need to be logged into a Google account that has permission to view and edit. Permissions can be given by the owner of the spreadsheet, Harald Stendal (harald.stendal@gmail.com).

A.3.2 Experiences Sheet Script

The Experiences Sheet Script is an Apps Script connected to the Experiences Google Sheet. The script takes information from the experiences sheet (not the tags sheet) and adds it to the Cloud SQL database that is specified.

How to open Experiences Sheet Script

To open the script follow these steps:

- Open the Experiences Google Sheet (see section *Experiences Google Sheet*)
- Click on **Tools**
- Click on **Script Editor** (**NOTE:** If you are signed into multiple Google Accounts, the script editor might not open. Make sure that you are only signed into one Google Account)

Transfering experiences to Cloud SQL

To transfer experiences to a Cloud SQL database, simply set the **db** string to the database you want to transfer the information to, and run the script.

Example - Updating the development database:

- Write "**var db = 'experiences_devdb';**"
- Run the script

NOTE: The following functionality has yet to be implemented - Removing experiences from the database, that have been removed from the spreadsheet. - Inserting new tags, that have been created in the tag sheet.

A.4 AI

A.4.1 Automatic setup

Follow these steps to create a Python Virtualenv automatically:

- Ensure that Python 3.6 or higher is installed.
- Ensure that pip is installed.
- Use a **Bash** terminal like **Git Bash**.

1. Open a terminal in the **backend** folder.
2. Run the command `npm run python-venv` in the terminal.

A.4.2 Manual setup

If that does not work try to install python dependencies in a virtualenv manually by following the steps below:

- Open a terminal in the **python** folder.
- Ensure that your version of Python is 3.6 or higher with the command `python --version`. If this is not the case, retry with `python3 --version`. If the latter does reveal a version of Python higher than 3.6, use `python3` instead of `python` for the rest of the guide. If neither worked, install Python and retry.
- Check that you have pip installed by calling `python -m pip --version`.

- Install virtualenv package to create a venv: `python -m pip install virtualenv`.
- Create a virtual environment: `python -m virtualenv venv`
- Activate the venv:
- macOS / Linux: `source venv/bin/activate`
- Windows: `.\venv\scripts\activate`
- Install the dependencies by calling `pyhton -m pip install -r requirements.txt`

A.4.3 Update requirements.txt

If you have added further dependencies to the code, update the requirements.txt file by doing the following NB! Remember to activate the virtual environment before doing this.

```
pip freeze > requirements.txt
```

B User journey map

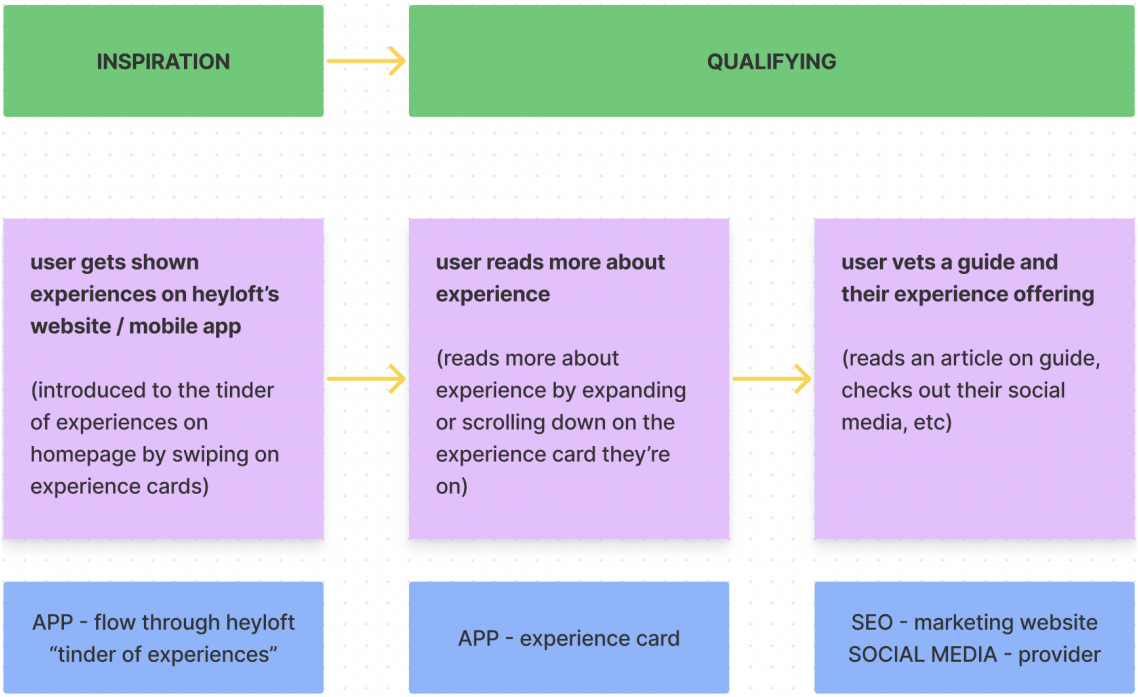


Figure 29: Snapshot of a fragment of the user journey map given to the team by the customer before sprint 1. The user journey map was created on Figma and it illustrates how the user may interact with the product. Snapshot taken on 7. Nov 2021.

C Product Backlog Increments

C.1 Product Backlog Before Sprint 1

Product Backlog - Ordered by priority (Highest priority on top)						
ID	User stories	Estimated amount of effort (1-5)	How to demo	Details	Scheduled for sprint	Status
US1	Swiping on experience cards			As a user, I should be able to swipe left on an experience card to dismiss an experience, and swipe right to show interest, so that I can save the ones I am interested in.		Not completed
US3	Experience card			As a user, I should be able to see an experience card to learn about the key details of an experience so that I can decide if I like that experience or not.		Not completed
US7	Detailed Experience view			As a user, I want to be able to scroll down on an experience card to learn more about it so that I can learn more about it now rather than later.		Not completed
US2	Superlike on experience cards			As a user, I should be able to superlike an experience to show more interest and add it to my favorited experiences.		Not completed
US5	CTA to Book			As a user, I want to be able to indicate that I want to book an experience right from the experience card so that if I like an experience enough to book it I have the option to do so without seeing more		Not completed
US6	Choosing Experiences			As a user, I want to be given the opportunity to choose between the experiences I have already found and liked so that I can make a choice rather than becoming stuck in the endless swipe.		Not completed
US4	Experience markers			As a user, I want to see “experience markers” so that I can see quick visual information about that experience and make a decision faster / without having to read a lot of content.		Not completed
US13	Recommendation system			As a user, I want to swipe right on an experience I like so that I see more experiences that I like just as much.		Not completed
US8	Create User			As a user, I want to go to a website and create a user, so that my personal preferences and data is saved		Not completed
US9	Log in			As a user, I want to go to a website and log in to a user I have created, so that the website is personalized to me.		Not completed
US10	Search for experiences			As a user, I want to do text search for specific experiences to heyloft’s database of experiences and get matching results, so that I can find specific experiences or groups of experiences faster.		Not completed
US12	Unexpected hug			As a user, I want to give/receive “unexpected hug” to a experience so that I can show my interest to that experience.		Not completed
US11	Give reviews			As a user, I should be able to review experiences, so that I am able to give feedback to the arranger of the experience		Not completed
US14	Filter experiences			As a user, I want to apply filters to heyloft’s database of experiences and get matching results, so that I can find specific experiences or groups of experiences faster.		Not completed

Figure 30: Snapshot of the product backlog at the beginning of sprint 1.

C.2 Product Backlog Before Sprint 2

Product Backlog - Ordered by priority (Highest priority on top)						
ID	**User stories**	**Estimated amount of effort (1-5)**	**How to demo**	**Details**	**Scheduled for sprint**	**Status**
US1	Swiping on experience cards			As a user, I should be able to swipe left on an experience card to dismiss an experience, and swipe right to show interest, so that I can save the ones I am interested in.		Not completed
US6	Leave carousel			As a user, I want to be given the opportunity to choose between the experiences I have already found and liked so that I can make a choice rather than becoming stuck in the endless swipe.		Not completed
US4	Tags			As a user, I want to see tags so that I can see quick visual information about that experience and make a decision faster / without having to read a lot of content.		Not completed
US13	Content-based filtering			As a user, I want to swipe right on an experience I like, and see similar experiences, so that I see more experiences that I like just as much.		Not completed
US14	Filter experiences			As a user, I want to apply filters to heyloft's database of experiences and get matching results, so that I can find specific experiences or groups of experiences faster.		Not completed
US5	CTA to Book			As a user, I want to be able to indicate that I want to book an experience right from the experience card so that if I like an experience enough to book it I have the option to do so without seeing more		Not completed
US8	Create User			As a user, I want to go to a website and create a user, so that my personal preferences and data is saved		Not completed
US9	Log in			As a user, I want to go to a website and log in to a user I have created, so that the website is personalized to me.		Not completed
US10	Search for experiences			As a user, I want to do text search for specific experiences to heyloft's database of experiences and get matching results, so that I can find specific experiences or groups of experiences faster.		Not completed
US15	Booking form			As a user, I should be able to fill out a booking form after pressing the CTA to book		Not completed
US12	Unexpected hug			As a user, I want to give/receive "unexpected hug" to a experience so that I can show my interest to that experience.		Not completed
US11	Give reviews			As a user, I should be able to review experiences, so that I am able to give feedback to the arranger of the experience		Not completed
US3	Experience card				Sprint 1	Completed
US7	Detailed Experience view				Sprint 1	Completed
US2	Favourite on experience cards				Sprint 1	Completed

Figure 31: Snapshot of the product backlog at the beginning of sprint 2. It was updated after sprint review in sprint 1. NOTE: Details for completed user stories were filled in the original product backlog, but have been removed here for simplicity. For details on the completed user stories, see the sprint backlog for the sprint when the user story was completed. Sprint backlogs can be found in section 8.

C.3 Product Backlog Before Sprint 3

Product Backlog - Ordered by priority (Highest priority on top)						
ID	**Name**	**Estimated amount of effort (1-5)**	**How to demo**	**User story**	**Scheduled for sprint**	**Status**
US13	Content-based filtering			As a user, I want to swipe right on an experience I like, and see similar experiences, so that I see more experiences that I like just as much.		Not completed
US16	Mobile first			As a user, I want to access this application on my mobile device so that I can utilize the device I am already using when I want to access the application, rather than switching to the web version		Not completed
US17	Navigation on website			As a user, I want to have access to navigation controls so that I can have the freedom to go where I want when I want		Not completed
US18	Website working on internet			As a user I want to have a website to go to, so that I can use the prototype.		Not completed
US8	Create User			As a user, I want to go to a website and create a user, so that my personal preferences and data is saved		Not completed
US9	Log in			As a user, I want to go to a website and log in to a user I have created, so that the website is personalized to me.		Not completed
US19	Group Swiping			As a user that is part of a group, I want to be able to make group decisions easily so that I don't have to spend a long time going back and forth with my group of friends to figure out what's best for everyone		Not completed
US15	Booking form			As a user, I should be able to fill out a booking form after pressing the CTA to book		Not completed
US12	Unexpected hug			As a user, I want to give/receive "unexpected hug" to a experience so that I can show my interest to that experience.		Not completed
US10	Search for experiences			As a user, I want to do text search for specific experiences to heyloft's database of experiences and get matching results, so that I can find specific experiences or groups of experiences faster.		Not completed
US11	Give reviews			As a user, I should be able to review experiences, so that I am able to give feedback to the arranger of the experience		Priority: Will not have
US1	Swiping on experience cards				Sprint 2	Completed
US6	Leave carousel				Sprint 2	Completed
US4	Tags				Sprint 2	Completed
US5	CTA to Book				Sprint 2	Completed
US14	Filter experiences				Sprint 2	Completed
US3	Experience card				Sprint 1	Completed
US7	Detailed Experience view				Sprint 1	Completed
US2	Favourite on experience cards				Sprint 1	Completed

Figure 32: Snapshot of the product backlog at the beginning of sprint 3. It was updated after sprint review in sprint 2. NOTE: Details for completed user stories were filled in the original product backlog, but have been removed here for simplicity. For details on the completed user stories, see the sprint backlog for the sprint when the user story was completed. Sprint backlogs can be found in section 8.

C.4 Product Backlog Before Sprint 4

Product Backlog - Ordered by priority (Highest priority on top)						
ID	Name	Estimated amount of effort (1-5)	How to demo	User story	Scheduled for sprint	Status
US16	Mobile first			As a user, I want to access this application on my mobile device so that I can utilize the device I am already using when I want to access the application, rather than switching to the web version		Not completed
US9	Log in			As a user, I want to go to a website and log in to a user I have created, so that the website is personalized to me.		Not completed
US20	Group creation			As a group creator, I want to be able to create a new group so that I can make plans with my friends.		Not completed
US21	Group Sharing			As a group creator, I want to be able to share the group with my friends so that they can participate in the group I made.		Not Completed
US26	Group swiping introduction			As a new user, I want to have clear communication on how to use the app to make a decision with my friends.		Not completed
US22	Group Swiping			As a user, I want to be able to swipe on experiences that havent been disliked by my a big portion of my group, within the context of my group, at my own pace, so that I can decide with my friend what we want to do when we hang out.		Not Completed
US23	Group Results			As a user, I want to be shown the results of my group swiping experience so that I can see what our combined preferences are for experiences		Not completed
US25	Group Preferences			As a group creator, I want to add tags to my group so that only the experiences we're interested in show up.		Not completed
US27	Passive Profile Setup			As a user, I want the system to automatically create a suggestion to set preferences permanently that it's noticed, so that i can get better experiences for me and not have to go set up a profile separately.		Not completed
US28	Text Search			As a user, I want to do text search for specific experiences to heyloft's database of experiences and get matching results, so that I can find specific experiences or groups of experiences faster.		Not completed
US17	Navigation on website				Sprint 3	Completed
US18	Website working on internet				Sprint 3	Completed
US8	Create User				Sprint 3	Completed
US1	Swiping on experience cards				Sprint 2	Completed
US6	Leave carousel				Sprint 2	Completed
US4	Tags				Sprint 2	Completed
US5	CTA to Book				Sprint 2	Completed
US14	Filter experiences				Sprint 2	Completed
US3	Experience card				Sprint 1	Completed
US7	Detailed Experience view				Sprint 1	Completed
US2	Favourite on experience cards				Sprint 1	Completed

Figure 33: Snapshot of the product backlog at the beginning of sprint 4. It was updated after sprint review in sprint 3. NOTE: Details for completed user stories were filled in the original product backlog, but have been removed here for simplicity. For details on the completed user stories, see the sprint backlog for the sprint when the user story was completed. Sprint backlogs can be found in section 8.

C.5 Product Backlog Before Sprint 5

Product Backlog - Ordered by priority (Highest priority on top)						
ID	**Name**	**Estimated amount of effort (1-5)**	**How to demo**	**User story**	**Scheduled for sprint**	**Status**
US22	Group Swiping			As a user, I want to be able to swipe on experiences that havent been disliked by my a big portion of my group, within the context of my group, at my own pace, so that I can decide with my friend what we want to do when we hang out.		Not Completed
US23	Group Results			As a user, I want to be shown the results of my group swiping experience so that I can see what our combined preferences are for experiences		Not completed
US26	Group swiping introduction			As a new user, I want to have clear communication on how to use the app to make a decision with my friends.		Not completed
US29	Log out			As a user I want to log out, so that I am not stuck in one account every time I log in		Not completed
US30	Redirections			As a user I want to be redirected to the site I clicked the link to, even though I have to log in first, so that I end up where I inteded to go		Not completed
US31	Information on homepage			As a user I want to have information on the homepage, so that I know what the prototype is and roughly how it works		Not completed
US32	Anonymous user nickname			As a user I want to have a short nickname as my username when I log in anonymously, so that I have a simple username		Not completed
US25	Group Preferences			As a group creator, I want to add tags to my group so that only the experiences we're interested in show up.		Not completed
US27	Passive Profile Setup			As a user, I want the system to automatically create a suggestion to set preferences permanently that it's noticed, so that i can get better experiences for me and not have to go set		Not completed
US28	Text Search			As a user, I want to do text search for specific experiences to heyloft's database of experiences and get matching results, so that I can find specific experiences or groups of experiences faster.		Not completed
US16	Mobile first				Sprint 4	Completed
US9	Log in				Sprint 4	Completed
US20	Group creation				Sprint 4	Completed
US21	Group Sharing				Sprint 4	Completed
US17	Navigation on website				Sprint 3	Completed
US18	Website working on internet				Sprint 3	Completed
US8	Create User				Sprint 3	Completed
US1	Swiping on experience cards				Sprint 2	Completed
US6	Leave carousel				Sprint 2	Completed
US4	Tags				Sprint 2	Completed
US5	CTA to Book				Sprint 2	Completed
US14	Filter experiences				Sprint 2	Completed
US3	Experience card				Sprint 1	Completed
US7	Detailed Experience view				Sprint 1	Completed
US2	Favourite on experience cards				Sprint 1	Completed

Figure 34: Snapshot of the product backlog at the beginning of sprint 5. It was updated after sprint review in sprint 4. NOTE: Details for completed user stories were filled in the original product backlog, but have been removed here for simplicity. For details on the completed user stories, see the sprint backlog for the sprint when the user story was completed. Sprint backlogs can be found in section 8.

C.6 Product Backlog After Sprint 5

Product Backlog - Ordered by priority (Highest priority on top)						
ID	**Name**	**Estimated amount of effort (1-5)**	**How to demo**	**User story**	**Scheduled for sprint**	**Status**
US32	Anonymous user nickname			As a user I want to have a short nickname as my username when I log in anonymously, so that I have a simple username	Sprint 5	Not completed
US25	Group Preferences			As a group creator, I want to add tags to my group so that only the experiences we're interested in show up.	Sprint 5	Not completed
US27	Passive Profile Setup			As a user, I want the system to automatically create a suggestion to set preferences permanently that it's noticed, so that i can get better experiences for me and not have to go set up a profile separately.		Not completed
US28	Text Search			As a user, I want to do text search for specific experiences to heyloft's database of experiences and get matching results, so that I can find specific experiences or groups of experiences faster.		Not completed
US22	Group Swiping				Sprint 5	Completed
US23	Group Results				Sprint 5	Completed
US26	Group swiping introduction				Sprint 5	Completed
US29	Log out				Sprint 5	Completed
US30	Redirections				Sprint 5	Completed
US31	Information on homepage				Sprint 5	Completed
US16	Mobile first				Sprint 4	Completed
US9	Log in				Sprint 4	Completed
US20	Group creation				Sprint 4	Completed
US21	Group Sharing				Sprint 4	Completed
US17	Navigation on website				Sprint 3	Completed
US18	Website working on internet				Sprint 3	Completed
US8	Create User				Sprint 3	Completed
US1	Swiping on experience cards				Sprint 2	Completed
US6	Leave carousel				Sprint 2	Completed
US4	Tags				Sprint 2	Completed
US5	CTA to Book				Sprint 2	Completed
US14	Filter experiences				Sprint 2	Completed
US3	Experience card				Sprint 1	Completed
US7	Detailed Experience view				Sprint 1	Completed
US2	Favourite on experience cards				Sprint 1	Completed

Figure 35: Snapshot of the product backlog at the end of sprint 5. It was updated after sprint review in sprint 5. NOTE: Details for completed user stories were filled in the original product backlog, but have been removed here for simplicity. For details on the completed user stories, see the sprint backlog for the sprint when the user story was completed. Sprint backlogs can be found in section 8.

D Changes to product backlog

D.1 User stories added to the product backlog

Table 28: Details on when user stories were added to the product backlog.

ID	Name	Date added to product backlog
US1	Swiping on experience cards	7th Sep 2021
US2	Superlike on experience cards	7th Sep 2021
US3	Experience cards	7th Sep 2021
US4	Experience markers	7th Sep 2021
US5	CTA to Book	7th Sep 2021
US6	Choosing Experiences	7th Sep 2021
US7	Detailed experiences view	7th Sep 2021
US8	Create User	7th Sep 2021
US9	Log in	7th Sep 2021
US10	Search for experiences	7th Sep 2021
US11	Give reviews	7th Sep 2021
US12	Unexpected hug	7th Sep 2021
US13	Content-based filtering	7th Sep 2021
US14	Filter experiences	7th Sep 2021
US15	Booking form	21st Sep 2021
US16	Mobile first	2nd Oct 2021
US17	Navigation on website	2nd Oct 2021
US18	Website working on internet	2nd Oct 2021
US19	Group Swiping	2nd Oct 2021
US20	Group Creation	16th Oct 2021
US21	Group Sharing	16th Oct 2021
US22	Group Swiping	16th Oct 2021
US23	Group Results	16th Oct 2021
US24	Potential Algorithm for Group Swiping	16th Oct 2021
US25	Group Preferences	16th Oct 2021
US26	Group Swiping Introduction	16th Oct 2021
US28	Text Search	16th Oct 2021
US29	Log out	29th Oct 2021
US30	Redirections	29th Oct 2021
US31	Information on homepage	29th Oct 2021
US32	Anonymous user nickname	29th Oct 2021

D.2 User stories removed from the product backlog

Table 29: Details on when and why user stories were removed from the product backlog.

ID	Name	Date removed	Explanation
US19	Group Swiping	16th Oct 2021	User story too big, split into US20-US23 and US26
US24	Potential Algorithm for Group Swiping	16th Oct 2021	User story falls under US22

E Sprint backlogs

For each sprint backlog the ID refers to the user story in the respective product backlog, appendix C.

E.1 Sprint backlog 1

ID	User stories	Estimated effort (1-5)	How to demo	Details
US1	Swiping on experience cards	3	Swipe left on the first card, swipe right on the second card. At the end of the swiping session, check that the "saved experiences" contains the second card and not the first	As a user, I should be able to swipe left on an experience card to dismiss an experience, and swipe right to show interest, so that I can save the ones I am interested in.
US3	Experience card	2	Go to the gallery view and see that an experience card containing key information shows up.	As an experience goer, I want to be able to see an experience card to learn about the key details of an experience so that I can decide if I like that experience or not.
US7	Detailed Experience view	3	Go to the gallery view and scroll down on an experience card. This should contain more detailed information about the experience.	As a user, I want to be able to scroll down on an experience card to learn more about it so that I can learn more about it now rather than later.
US2	Favourite on experience cards	1	Superlike an experience from the gallery view and see that it has been added to the list of favorite experiences.	As a user, I should be able to favourite an experience to show more interest and add it to my favourited experiences.
US5	CTA to Book	1	Go to the gallery view and show a clickable button to book an experience.	As a user, I want to be able to indicate that I want to book an experience
US6	Leave carousel	2	Complete a swiping session, and check that all the liked experiences show up in the saved list at the end	As a user, I want to be given the opportunity to choose between the experiences I have already found and liked so that I can make a choice rather than becoming stuck in the endless swipe.
US4	Tags	2	Check that the experience cards contain some experience markers for example the suitable weather. Confirm that the markers are relevant.	As a user, I want to see tags so that I can see quick visual information about that experience and make a decision faster / without having to read a lot of content.

E.2 Sprint backlog 2

ID	User stories	Estimated effort (1-5)	How to demo	Description
US1	Swiping on experience cards	3	Swipe left on the first card, swipe right on the second card. At the end of the swiping session, check that the "saved experiences" contains the second card and not the first	As a user, I should be able to swipe left on an experience card to dismiss an experience, and swipe right to show interest, so that I can save the ones I am interested in.
US6	Leave carousel	2	Complete a swiping session, and check that all the liked experiences show up in the saved list at the end	As a user, I want to be given the opportunity to choose between the experiences I have already found and liked so that I can make a choice rather than becoming stuck in the endless swipe.
US4	Tags	2	Check that the experience cards contain some tags for example the suitable weather. Confirm that the tags are relevant.	As a user, I want to see tags so that I can see quick visual information about that experience and make a decision faster / without having to read a lot of content.
US13	Content-based filtering	4	Complete a swiping session, and confirm that the recommended experiences are similar to the ones that you swiped right	As a user, I want to swipe right on an experience I like, and see similar experiences, so that I see more experiences that I like just as much.
US14	Filter experiences	4	Apply filters to a collection of experiences and check if the right experiences are filtered in	As a user, I want to apply filters to heyloft's database of experiences and get matching results, so that I can find specific experiences or groups of experiences faster.
US5	CTA to Book	1	Go to the carousel and show a clickable button to book an experience.	As a user, I want to be able to indicate that I want to book an experience right from the experience card so that if I like an experience enough to book it I have the option to do so without seeing more.

E.3 Sprint backlog 3

ID	Name	Estimated effort (1-5)	How to demo	User story
US13	Content-based filtering	2	Complete a swiping session, and confirm that the recommended experiences are similar to the ones that you swiped right	As a user, I want to swipe right on an experience I like, and see similar experiences, so that I see more experiences that I like just as much.
US16	Mobile first	3	Open the prototype on mobile and confirm that all the functionality works and that the user interface is fits the device	As a user, I want to access this application on my mobile device so that I can utilize the device I am already using when I want to access the application, rather than switching to the web version.
US17	Navigation on website	2	Use the navbar to navigate through the website	As a user, I want to have access to navigation controls so that I can have the freedom to go where I want when I want.
US18	Website working on internet	4	Go to the prototypes website on the internet and confirm that all the functionality works as it should	As a user I want to have a website to go to, so that I can use the prototype.
US8	Create User	4	Go to the website, navigate to the "Create user" site, follow the steps to create a user, and get confirmation that the user has been made	As a user, I want to go to a website and create a user, so that my personal preferences and data is saved
US9	Log in	2	Go to the website, navigate to the "Log in" site, log in to an already existing user, and confirm that the preferences of the user account still are there	As a user, I want to go to a website and log in to a user I have created, so that the website is personalized to me.

E.4 Sprint backlog 4

ID	Name	Estimated effort (1-5)	How to demo	User story
US16	Mobile first	2	Open the prototype on mobile and confirm that all the functionality works and that the user interface fits the device	As a user, I want to access this application on my mobile device so that I can utilize the device I am already using when I want to access the application, rather than switching to the web version.
US9	Log in	1	Go to the website, navigate to the "Log in" site, log in to an already existing user, and confirm that the preferences of the user account still are there	As a user, I want to go to a website and log in to a user I have created, so that the website is personalized to me.
US20	Group creation	2	Log in with an existing user, go to the group "hub", fill in the necessary information for creating a group, and click "create group"	As a group creator, I want to be able to create a new group so that I can make plans with my friends.
US21	Group Sharing	3	Find your groups group code/link , share it to a friend who then goes to the group "hub", inserts the code and joins the group. Verify that your friend has joined your group	As a group creator, I want to be able to share the group with my friends so that they can participate in the group I made.
US26	Group swiping introduction	2	Verify, when you are logged in and go to the group "hub", that you get a quick and easy introduction on how to create and join groups aswell as starting a group swiping session	As a new user, I want to have clear communication on how to use the app to make a decision with my friends.
US22	Group Swiping	4	Go to one of your groups, and complete your swiping session inside that group at your own pace, verify that your swiping session has been registered	As a user, I want to be able to swipe on experiences that have not been disliked by my a big portion of my group, within the context of my group, at my own pace, so that I can decide with my friend what we want to do when we hang out.
US23	Group Results	3	Go to the group "hub" and verify that the results shown for the group swiping are correct	As a user, I want to be shown the results of my group swiping experience so that I can see what our combined preferences are for experiences.
US25	Group Preferences	2	Start creating a group, add some tags that interest you, finish creating the group, and verify in the swiping session that you only see experiences with matching tags	As a group creator, I want to add tags to my group so that only the experiences we're interested in show up.

E.5 Sprint backlog 5

ID	Name	Estimated effort (1-5)	How to demo	User story
US22	Group Swiping	3	Go to one of your groups, and complete your swiping session inside that group at your own pace, verify that your swiping session has been registered	As a user, I want to be able to swipe on experiences that have not been disliked by my a big portion of my group, within the context of my group, at my own pace, so that I can decide with my friend what we want to do when we hang out.
US23	Group Results	3	Go to the group "hub" and verify that the results shown for the group swiping are correct	As a user, I want to be shown the results of my group swiping experience so that I can see what our combined preferences are for experiences.
US26	Group swiping introduction	1	Verify, when you are logged in and go to the group "hub", that you get a quick and easy introduction on how to create and join groups as well as starting a group swiping session	As a new user, I want to have clear communication on how to use the app to make a decision with my friends.
US29	Log out	1	Log in, then log out and verify that you are logged out	As a user I want to log out, so that I am not stuck in one account every time I log in.
US30	Redirections	1	Click on one of the links that requires that you are logged in, then log in and verify that you are sent to the link that you originally clicked on	As a user I want to be redirected to the site I clicked the link to, even though I have to log in first, so that I end up where I intended to go.
US31	Information on homepage	1	Go to the homepage, and verify that there is information about the prototype	As a user I want to have information on the homepage, so that I know what the prototype is and roughly how it works.
US32	Anonymous user nickname	1	Log in with anonymous log in, and verify that your username is a short random name rather than a long user id	As a user I want to have a short nickname as my username when I log in anonymously, so that I have a simple username.
US25	Group Preferences	2	Start creating a group, add some tags that interest you, finish creating the group, and verify in the swiping session that you only see experiences with matching tags	As a group creator, I want to add tags to my group so that only the experiences we're interested in show up.

F MoSCoW Prioritization Example

MoSCoW Prioritization of Stories

Sprint 1 - 06.09.2021

Must Have

- Swiping Right
  - As an experience goer, I want to swipe right on an experience to indicate its an experience I might want to try.
- Swiping Left
  - As an experience goer I want to swipe left on an experience I don't have any interest in so that I don't see more experiences like those ones.
- "Superlike"
  - As an experience goer, I want to be able to "superlike" an experience so that I can have easier access to experiences I really like.
- Experience Card
  - As an experience goer, I want to be able to see an experience card to learn about the key details of an experience so that I can decide if I like that experience or not.
- "Experience Markers"
  - As an experience goer, I want to see "experience markers" so that I can see quick visual information about that experience and make a decision faster / without having to read a lot of content.
- CTA to Book
  - As an experience goer, I want to be able to indicate that I want to book an experience right from the experience card so that if I like an experience enough to book it I have the option to do so without seeing more experiences.
- Choosing Experiences
  - As an experience goer, I want to be given the opportunity to choose between the experiences I have already found and liked so that I can make a choice rather than becoming stuck in the endless swipe.
  -

Should have

- As a user, I want to go to a website and create a user, so that my personal preferences and data is saved
- As a user, I want to go to a website and log in to a user I have created, so that the website is personalized to me.

Could have

- As a user, I should be able to review experiences, so that I am able to give feedback to the arranger of the experience

Will not have

- As a user, I want to be able to book experiences, so that I get to try them out

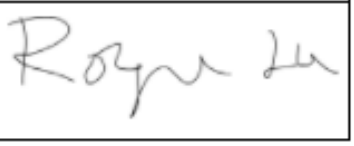
Figure 36: Snapshot of the MoSCoW Prioritization list given to the team by the customer for sprint 1. The document was handed to the team on 6th Sep 2021. NOTE: Some details have been omitted from the document, for the purpose of showing the structure of the MoSCoW prioritization document.

G Group contract

Date: 03.09.2021

Group contract - Group 10

Group members

Name	Role	Signature
Arnstein Thorstensen Øyhus	Development Team	
Hanne Kyllo Ødegård	Team Leader	
Bjørn Anders Flågen	Quality Manager	
Anmol Sarai	Product Owner	
Sander Francis	Scrum Master	
Leonard Nguyen Schøyen	Development Team	
Royce Lu	Development Team	

Date: 03.09.2021

Role Description

Role	Responsibilities
Quality Manager	Ensure the quality of the end product and the overall process. Check that project documents are consistent. Monitor and review all testing activities.
Product Owner	Make sure that all requirements by the customer are fulfilled. Ensure that the backlog is user-centered and order by priority.
Scrum Master	Ensure that Scrum processes are followed. Create the sprint backlog and check that it is updated. Lead the scrum activities
Team Leader	Plan and coordinate team activities. Motivate the team members.
Development Team Members	Open dialog with the Team Leader and Scrum Master. Execute plans made by the Team Leader and the Scrum Master. Keeping track of time spent on various tasks.

Deadlines

1. Absence from meetings: If one cannot attend a meeting, one needs to notify the other team members **at least 24 hours** before the meeting. If a situation arises after this limit, one should notify the group as soon as possible. The team decides if the reasoning for the late notice is valid.
2. Meeting agenda: A meeting agenda should be created and available in Google Docs at **least 24 hours** before the meeting.
3. Personal communication regarding the project should be answered **at least 24 hours** after the message has been received.
4. It is expected that all members arrive at meetings or other group activities on time. Arriving more than **15 minutes late** is not acceptable and must be notified in advance.

Guidelines for meetings

1. Every Tuesday is considered a workday. On this day it is expected that everyone is present, at least at some point during the day, at A4-138 in “Realfagsbygget”. The allocated time slot is 8-20.

Date: 03.09.2021

2. A meeting with the customer is held on Thursdays at 15:00.
3. The group should resolve any disagreements democratically.
4. Minutes should be written for each meeting by the person in charge.

Expected workload

1. It is expected that everyone tries to achieve 26 hours of work each week, but the minimum requirement is 20 hours. The workload includes all hours used in connection with the subject TDT4290, as well as meetings and other activities that are in connection with the project "Attractive City".

Communication

1. Communication should be performed over slack.

Breach of contract

1. Simple breaches of the contract, such as deadlines, result in having to write the minutes for the next meeting.
2. More serious breaches of the contract should be addressed internally within the group. If this does not solve the problem, it should be discussed with the supervisor.

H Feature List

Experience queue / Tinder for experiences	Show an inciting summary of one experience at a time. Tinder-esque swipe right/left to show interest or not. Show more information when user is interested, continue to another experience when not. • "As a user I think that there are too many options presented when looking for experiences" • "As a user I want to get to know the experiences I might be interested in" • "As a user I want to be inspired to do something"
Voting system for groups	Streamline the process of deciding what experience to choose from when in a group. Maybe through a recommendation system or perhaps matching through an experience queue. Aim to narrow down the available options making it easier to decide. • "As a group of users we want to be able to find experiences that the entire group wants to attend" • We want this process to be efficient and avoid having too many options, so that we actually end up going instead of just looking"
Planner Automatic schedule	Plans a day for the user. This can be done by the feature itself or by the user. The user can choose from broader categories, e.g "cultural events", or more specific categories, e.g "rock concert". The user can also pick how many activities they want to do in one day, and also at what time. The planner will then make a plan for the day. This can also be done completely automatic without any input from the user. • "As a user i want to be able to plan my day" • "As a user I want to get suggestions on activities to do and at what time, based on my input"
Personal suggestions	System to generate personal suggestions depending on age, previous activities, registered preferences, time slot. Show experiences based on previous bookings • "As a user I can access the site and get shown experiences that resemble things I have enjoyed in the past" Show possible experiences based on what similar people or groups have enjoyed. • "As a user I can get an unexpected hug by being shown what people or groups similar to me have enjoyed that is out of my regular scope"
Smart search	Search shows results based on keywords in the search, but also results that are in a similar category (for example a search on "theater" might also show cinemas) • "As a user I can search for specific experiences, and get matching results, so that I can find specific experiences faster"

Map	Show a selection of experiences for the user on a map. Can also be used to show the user's current position and other information, like transport information. • "As a user I want to see the experiences in my area" • "As a user I want to be able to be spontaneous and pick something wherever I am" • "As a user I want to know where the experiences are located and how to get to them"
Ratings for experiences, guides	Allow for guides to sign up and add experiences to their "showing list" • "As a user I can sign up as a guide" • "As a guide I can add places to my guide list to show others which places I feel confident guiding them too" Allow for rating of these guides and adding comments. • "As a user I can rate my guide and add comments to help other people choose a suitable guide for their experience"
360 degree display from experience location	Let the user get a sneak peak of the experience. • "As a user I want to have the option to see a 360 degree image of the experience, so that I know how it looks in real life" Help the user to decide if they want to experience the experience.
Experience markers	Mark experiences with small icons symbolizing something about the experiences, such as how well the experience works with current weather or if it is a "Unexpected hug". • "As a user I want to get information about the attraction's limitations. For instance if I have to be above a certain height limit, if the weather has to be nice or if I have to be a certain age" • "As a user I want to be made aware if the experience is hand picked for me or a 'shot in the dark'(unexpected hug"
Notifications system	Ability for Heyloft system to send users emails or push notifications with reminders, information, suggestions • As an experience provider, I can add certain important information to my provided experiences that will be shown to users in a timely manner to improve their experience with my offering. • As a user, I want to get information about upcoming experiences I have booked automatically presented to me so that the experience is seamless. • As a user, I want to receive relevant personal suggestions for activities/experiences to do before and after my booked experience to enhance or complete the experience/mood and make being an experience-consumer easier.

I Project plan

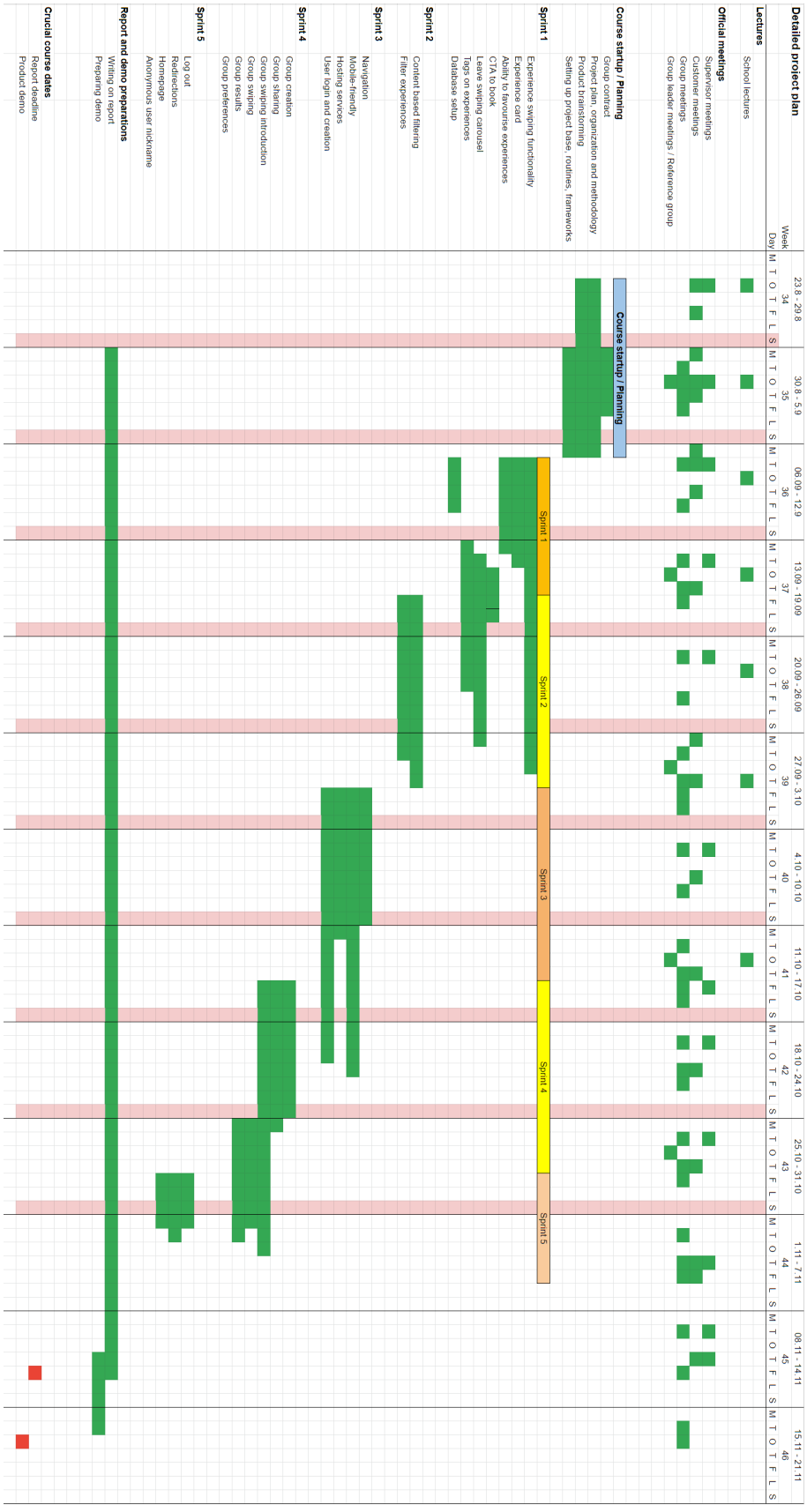


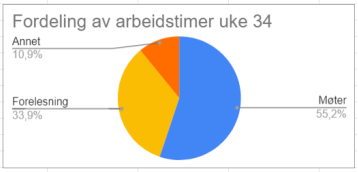
Figure 37: Snapshot of the project plan showing the events and activities for this project.

J Working hours registration

Timeliste Gruppe 10 — Alle enheter i timer

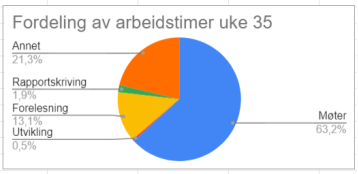
23.8 - 29.8

Uke 34	Prosjektfase:	Oppstart				
	Møter	Utvikling	Forelesning	Rapportskriving	Annet	Samlet
Anmol	3,25	0	2	0	0	5,25
Arnstein	3,25		2			5,25
Bjørn Anders	3,25		2		2	7,25
Hanne	3,25		2			5,25
Leonard	3,25		2			5,25
Royce	3,25		2		1	6,25
Sander	3,25		2		1,5	6,75
Samlet	22,75	0	14	0	4,5	**41,25**



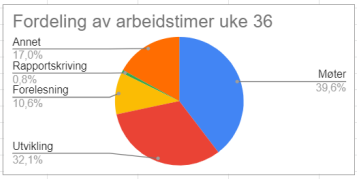
30.8 - 5.9

Uke 35	Prosjektfase:	Forstudie				
	Møter	Utvikling	Forelesning	Rapportskriving	Annet	Samlet
Anmol	9,5	0	2	0	3	14,5
Arnstein	9,5		2		2	13,5
Bjørn Anders	9,5		2		5	16,5
Hanne	10,5	0,5	2	2	4	19
Leonard	9,5	0	2	0	2,5	14
Royce	9,5		2			11,5
Sander	9,5		2		6,25	17,75
Samlet	67,5	0,5	14	2	22,75	**106,75**



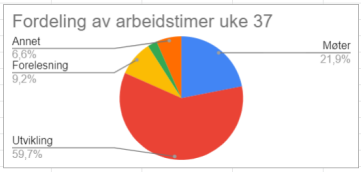
06.09 - 12.9

Uke 36	Prosjektfase:	Forstudie/Sprint 1 (07.09 - 16.09)				
	Møter	Utvikling	Forelesning	Rapportskriving	Annet	Samlet
Anmol	9,25	3	2	0	5	19,25
Arnstein	9	8	2			19
Bjørn Anders	9		2	1	9	21
Hanne	7	10	2		1	20
Leonard	6	10	2			18
Royce	6	5	2			13
Sander	6,25	6,5	2		7,5	22,25
Samlet	52,5	42,5	14	1	22,5	**132,5**



13.09 - 19.09

Uke 37	Prosjektfase:	Sprint 1 (07.09 - 16.09)				
	Møter	Utvikling	Forelesning	Rapportskriving	Annet	Samlet
Anmol	8	6	2		2	18
Arnstein	4	15	2			21
Bjørn Anders	3	11	2		4	20
Hanne	3,5	13	2		2	20,5
Leonard	4,75	16,5	2			23,25
Royce	4	12	2			18
Sander	6	17	2	3,75	2	30,75
Samlet	33,25	90,5	14	3,75	10	**151,5**



20.09 - 26.09

Uke 38	Prosjektfase:	Sprint 2 (17.09 - 30.09)				
	Møter	Utvikling	Forelesning	Rapportskriving	Annet	Samlet
Anmol	1,5	4	2	4	2	13,5
Arnstein	1,5	14	2			17,5
Bjørn Anders	1,5	18	2	1,5		23
Hanne	1,5	10	2	4		17,5
Leonard	1,5	12		4,75		18,25
Royce	1,5	11,25	2	1		15,75
Sander	1,5	15,5	2	3,5	0,5	23
Samlet	10,5	84,75	12	18,75	2,5	**128,5**

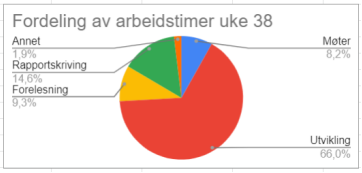
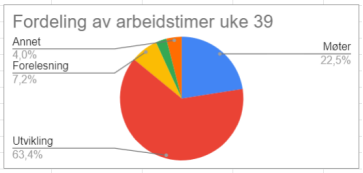


Figure 38: Snapshot of working hours registration sheet from week 34 to 38.

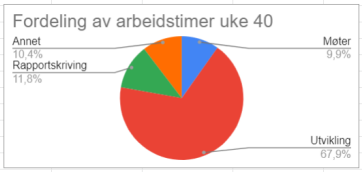
27.09 - 3.10

Uke 39	Prosiektfase:	Sprint 2 (17.09 - 30.09)				
	Møter	Utvikling	Forelesning	Rapportskriving	Annet	Samlet
Anmol	4	9	2			15
Arnstein	3	18		1		22
Bjørn Anders	3	7	2		2	14
Hanne	5,5	7	2		2	16,5
Leonard	3,5	12,75				16,25
Royce	3	8,5	2	1	0,5	15
Sander	6	16,5	1	1,5	0,5	25,5
Samlet	28	78,75	9	3,5	5	**124,25**



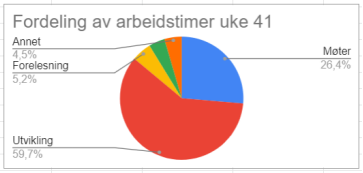
4.10 - 10.10

Uke 40	Prosiektfase:	Sprint 3 (02.10 - 15.10)				
	Møter	Utvikling	Forelesning	Rapportskriving	Annet	Samlet
Anmol	1	5		4	2	12
Arnstein	1	6		1	1,5	9,5
Bjørn Anders	1	4		2	4	11
Hanne	3	8		2	2	15
Leonard	1	12,5				13,5
Royce	1	10,5		0,75		12,25
Sander	1	16		1		18
Samlet	9	62	0	10,75	9,5	**91,25**



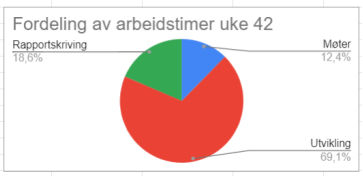
11.10 - 17.10

Uke 41	Prosiektfase:	Sprint 3 (02.10 - 15.10)				
	Møter	Utvikling	Forelesning	Rapportskriving	Annet	Samlet
Anmol	7	5	2	2	5	21
Arnstein	6,5	17		1,5		25
Bjørn Anders	5	21				26
Hanne	5	13,5	2		2	22,5
Leonard	5,75	13				18,75
Royce	4,5	11,75	2	1,5		19,75
Sander	7	11	2	1,5		21,5
Samlet	40,75	92,25	8	6,5	7	**154,5**



18.10 - 24.10

Uke 42	Prosiektfase:	Sprint 4 (16.10 - 29.10)				
	Møter	Utvikling	Forelesning	Rapportskriving	Annet	Samlet
Anmol	2			4		6
Arnstein	2	14		3		19
Bjørn Anders	2	14		5		21
Hanne	3,5	11		3		17,5
Leonard	2	13,75		3		18,75
Royce	1,75	15		2,5		19,25
Sander	1,75	16		2		19,75
Samlet	15	83,75	0	22,5	0	**121,25**



25.10 - 31.10

Uke 43	Prosiektfase:	Sprint 5 (29.10 - 5.11)				
	Møter	Utvikling	Forelesning	Rapportskriving	Annet	Samlet
Anmol	5,75	15		1		21,75
Arnstein	5	17				22
Bjørn Anders	5	12		2		19
Hanne	5	12		1		18
Leonard	3,75	13				16,75
Royce	5,75	6				11,75
Sander	5,75	18,5		1		25,25
Samlet	36	93,5	0	5	0	**134,5**

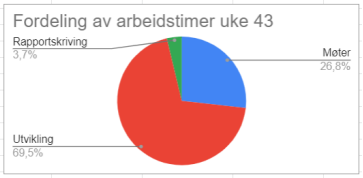
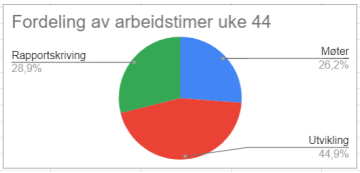


Figure 39: Snapshot of working hours registration sheet from week 39 to 43.

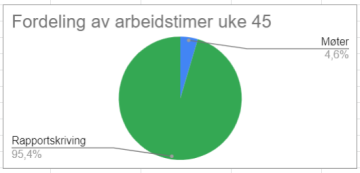
1.11 - 7.11

Uke 44	Prosiektfase:	Sprint 5 (29.10 - 5.11)				
	Møter	Utvikling	Forelesning	Rapportskriving	Annet	Samlet
Anmol	2	11		7		20
Arnstein	6,5	9		6		21,5
Bjørn Anders	6,5	5		8		19,5
Hanne	5,5	9		5		19,5
Leonard	3,5	5		6,5		15
Royce		10				10
Sander	9,5	8,5		4,5		22,5
Samlet	33,5	57,5	0	37	0	**128**



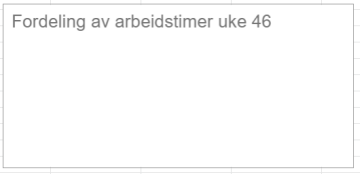
08.11 - 14.11

Uke 45	Prosiektfase:	Report				
	Møter	Utvikling	Forelesning	Rapportskriving	Annet	Samlet
Anmol				2		2
Arnstein						0
Bjørn Anders	1,25			15		16,25
Hanne						0
Leonard				16,5		16,5
Royce						0
Sander	1			13		14
Samlet	2,25	0	0	46,5	0	**48,75**



15.11 - 21.11

Uke 46	Prosiektfase:	Report				
	Møter	Utvikling	Forelesning	Rapportskriving	Annet	Samlet
Anmol						0
Arnstein						0
Bjørn Anders						0
Hanne						0
Leonard						0
Royce						0
Sander						0
Samlet	0	0	0	0	0	**0**



	Møter	Utvikling	Forelesning	Rapportskriving	Annet	Samlet
Total	351	686	85	157,25	83,75	1363

Figure 40: Snapshot of working hours registration sheet from week 44 to 46 taken 2 days into week 45.

K User testing feedback

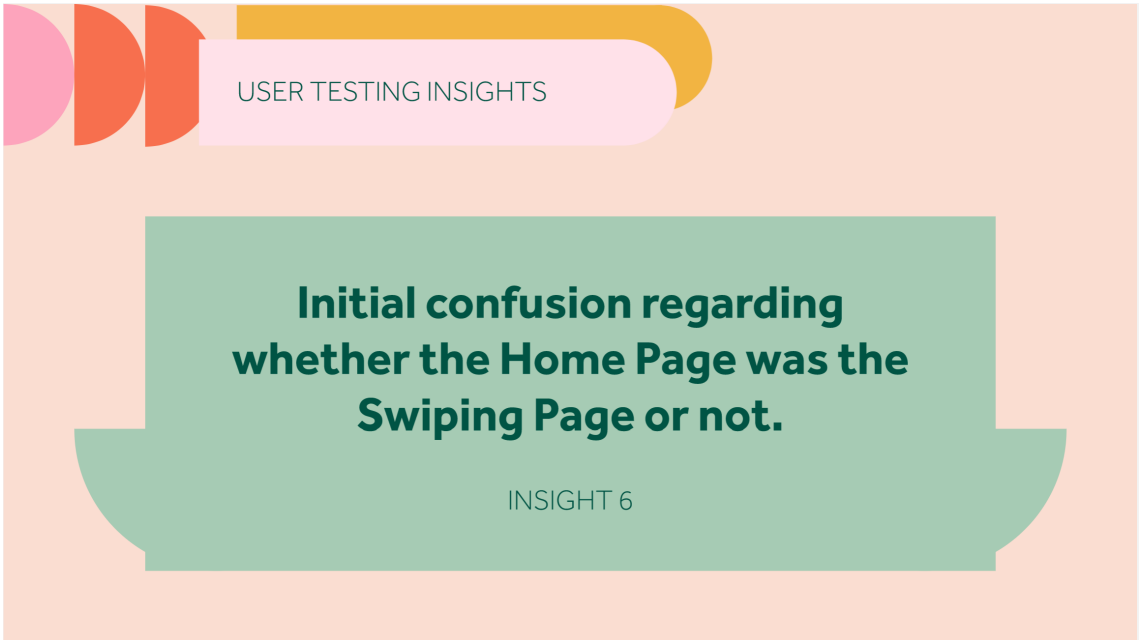
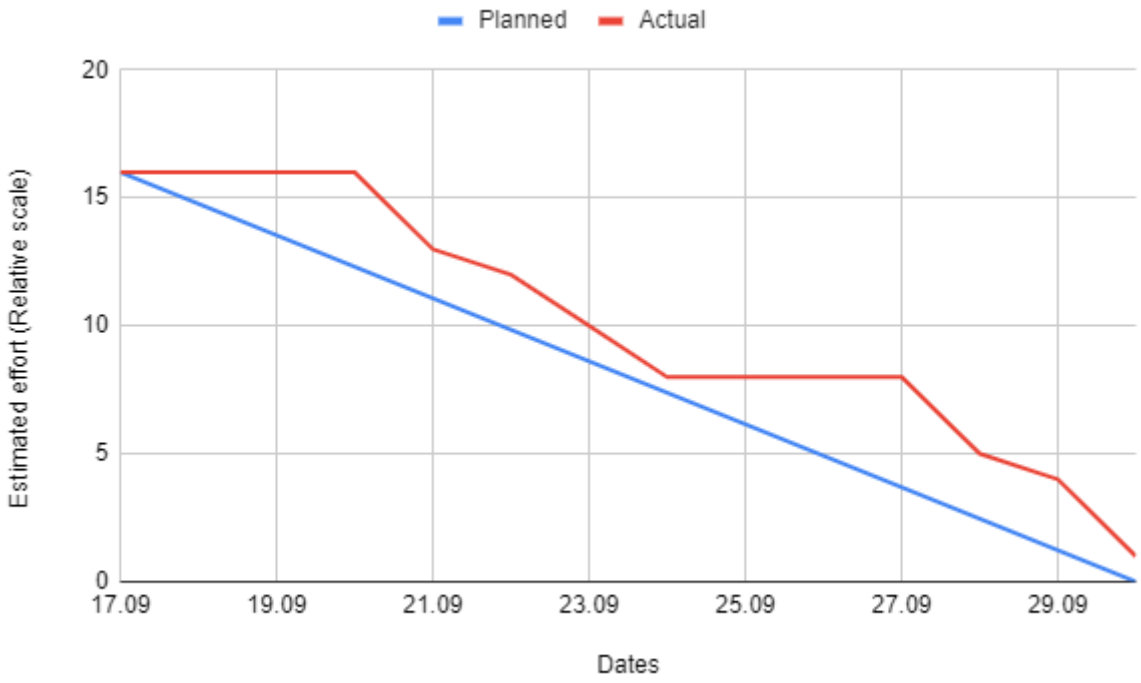


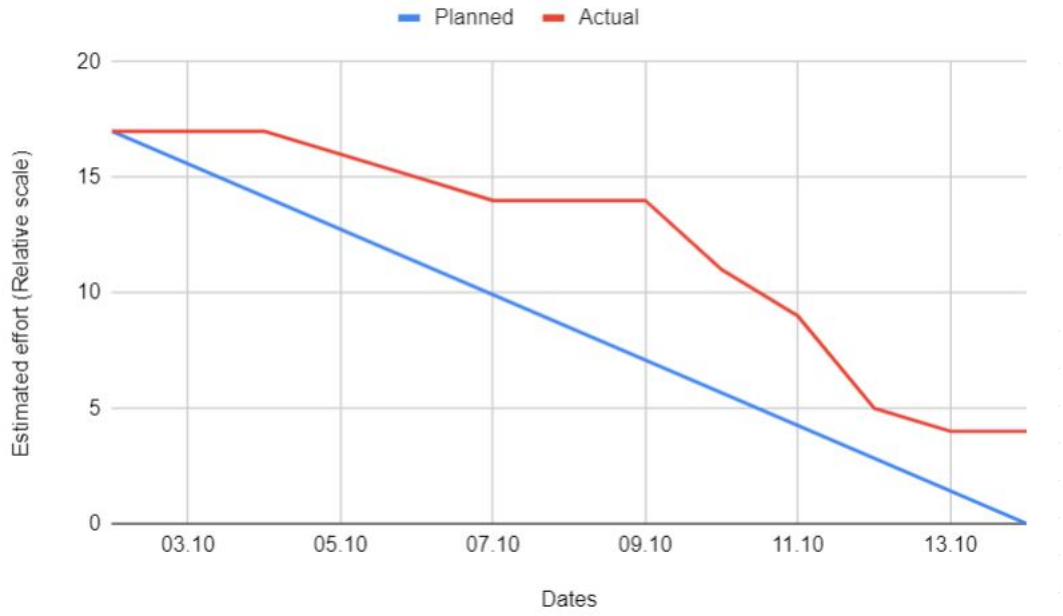
Figure 41: Snapshot of an improvement point from the user tests arranged by the customer.

L Burndown Charts

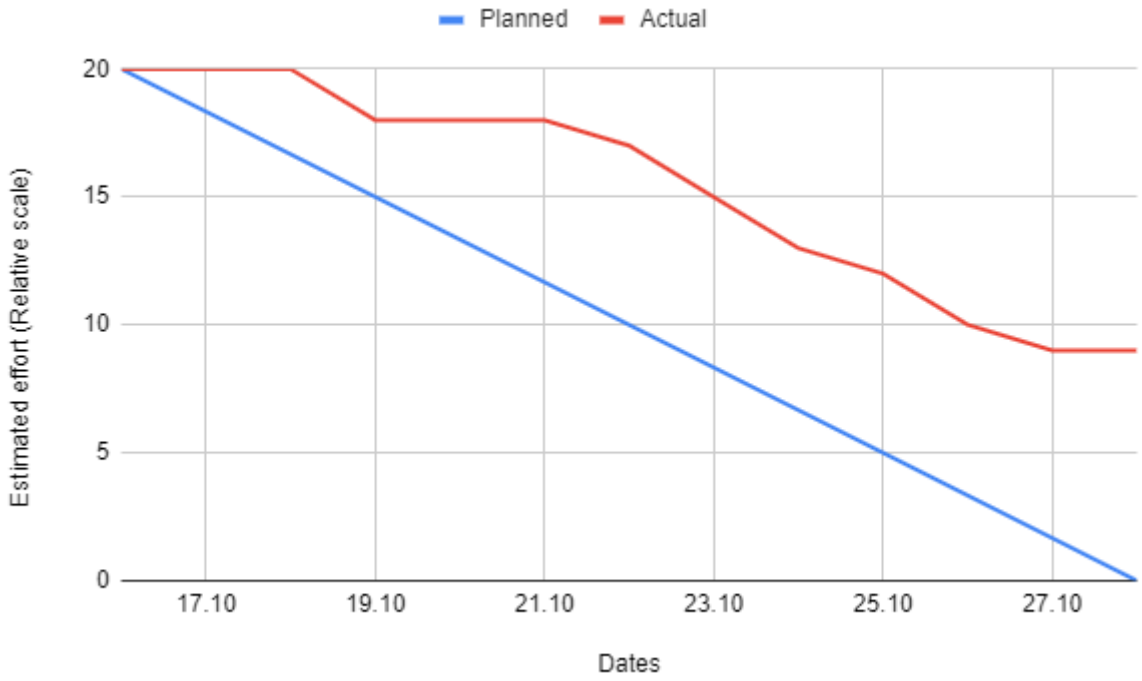
L.1 The Burndown Chart from Sprint 2



L.2 The Burndown Chart from Sprint 3



L.3 The Burndown Chart from Sprint 4



L.4 The Burndown Chart from Sprint 5