Bachelor's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Anders Heftøy Carlsen
Adrian Wist Hakvåg
Eivind Strand Harboe

The Development and Implementation of a Ticket Assistant for AtB AS

Bachelor's thesis in Bachelor of Engineering in Computer Science
Supervisor: Jonathan Jørgensen
May 2023

NTNU
Norwegian University of
Science and Technology

Anders Heftøy Carlsen
Adrian Wist Hakvåg
Eivind Strand Harboe

The Development and Implementation of a Ticket Assistant for AtB AS

Bachelor's thesis in Bachelor of Engineering in Computer Science
Supervisor: Jonathan Jørgensen
May 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

This Bachelor's thesis focuses on the development of a ticket assistant for AtB, which is the public transport administrator in Trøndelag. The assistant is a new feature to AtB's app that aims to help users choose the correct ticket based on their specific needs and travel habits. The project's goal was to explore whether this digital assistant can simplify the ticket selection process, especially for edge case scenarios and cost optimization. Furthermore, another goal has been to ensure that the solution feels like a natural extension of the AtB application.

The assistant has been designed with a focus on user needs identified through a digital survey, as well as the current design language of the AtB app. Throughout the project, emphasis has been placed on user-friendliness and an intuitive user interface, which was tested iteratively using prototypes.

Agile development was employed in the creation of the assistant, with continuous communication with AtB and test users throughout the development process.

The final solution consists of two features, a questionnaire to recommend the most cost-effective ticket based on travel habits, and a page for tips and information, addressing questions about edge case scenarios.

The main finding is that the ticket assistant is likely to improve user satisfaction and streamline the ticket selection process. Future improvements include support for more ticket types, as well as incorporating historical data, which might enhance the overall effectiveness of the assistant.

Sammendrag

Denne bacheloroppgaven fokuserer på utviklingen av en billettassistent for AtB, den administrative enheten for kollektivtransport i Trondheim. Assistenten skal være en utvidelse til AtB sin app og har som mål å hjelpe brukere med å velge riktig billett basert på deres spesifikke behov og reisevaner. Prosjektets mål var å undersøke om denne digitale assistenten kan forenkle billettvalgsprosessen, både for spesielle scenarier og for å finne den billigste billetten for brukerens behov. Et annet mål var at løsningen skulle føles ut som et naturlig utvidelse av AtB sin applikasjon.

Billettassistenten ble utviklet med fokus på brukernes behov, kartlagt gjennom en digital undersøkelse, samt det nåværende designspråket i AtB-appen. Gjennom hele prosjektet ble det lagt vekt på brukervennlighet og et intuitivt brukergrensesnitt, som ble testet iterativt ved hjelp av prototyper.

Smidige utviklingsmetoder ble brukt for å lage den digitale assistenten sammen med kontinuerlig kommunikasjon med AtB og testbrukere gjennom hele utviklingsprosessen.

Den endelige løsningen består av to funksjoner: et spørreskjema som anbefaler det mest kostnadseffektive billettalternativet basert på reisevaner, og en side med tips og informasjon for spørsmål om særtilfeller.

Hovedfunnet fra oppgaven er at assistenten sannsynligvis vil forbedre brukertilfredshet og effektivisere billettvalgsprosessen. Ytterligere forbedringer kan være støtte for flere billettyper, samt å inkorporere historisk data, som vil forbedre den totale opplevelsen av den digitale assistenten.

Preface

This bachelor thesis represents the culmination of our collective efforts as a team. Its purpose is to highlight the research, design and development that went into the creation of a ticket assistant for the AtB application.

The task has been carried out on assignment from AtB AS. Initially, the idea was for assistive features in the AtB app, and the team chose it because of the project's ability to potentially touch the lives of people that travel in Trøndelag.

We would like to express our gratitude to our advisor, Jonathan Jørgensen, for his commitment, support and guidance. Additionally, we would like to acknowledge AtB AS and their development team for their support, resources and for giving us the task that enabled this project.

Trondheim | 21st May 2023

Adrian Wist Hakvåg | Anders Heftøy Carlsen | Eivind Strand Harboe

Task description

The primary objective of this bachelor's thesis was to design and develop a digital assistant for the new AtB mobile application. This assistant's purpose is to help users of public transport in Trøndelag choose the right ticket type.

One of the main goals has been to make a feature that feels like a natural extension of the already existing application. It should also be an optional feature that is not forced upon users. All the functional requirements of the solution can be found in the Vision document (Appendix C) and the rest of the requirements are found in the requirements documentation (Appendix D). The design should also be intuitive and easy for users to understand. Given AtB's wide range of users, the solution must also be accessible and follow WCAG requirements (Appendix F).

Table of Contents

List of Figures

List of Tables

List of Acronyms

API Application Programming Interface

BaaS Back-end as a Service

GDPR General Data Protection Regulation

NoSQL Not only SQL or No SQL

NP-complete Nondeterministic polynomial-time complete

npm Node Package Manager

PR Pull request

QA Quality assurance

SSOT Single source of truth

VCS Version Control System

VoIP Voice over IP

WCAG Web Content Accessibility Guidelines

1 Introduction

Public transportation is essential to society since it promotes social inclusion, reduces traffic congestion, and increases sustainability. It promotes the use of shared vehicles and offers an effective and environmentally sustainable mode of transportation, consequently lowering air pollution and greenhouse gas emissions. A well-designed public transportation system connects members of all societal segments to essential services like healthcare, employment, and education. Thus, it provides the potential for social mobility for individuals.

1.1 Background

AtB is the public transport manager for Trøndelag county municipality and is responsible for the planning, marketing, and development of public transport.

AtB used to offer two separate apps for their services, one for purchasing tickets and an additional one for looking up departure times and travel options. However, recently they launched a brand-new app that combines both of these features. Users of the new app can purchase tickets, make travel plans, and view departures, all in one place. The new app currently provides a wide range of tickets, with prices varying according to what traveller type the user is and where they are travelling. At the time of writing, there are five traveller categories and 13 geographical zones. Users have the option of purchasing single or periodic tickets which come in durations of 24 hours, 7 days, 30 days, 60 days, 90 days, and 180 days. With all these different options, selecting the ticket that is the cheapest and fits the user's needs can be challenging.

In the context of this thesis, a digital ticket assistant revolves around providing personalised guidance for the ticket purchasing process. The development team previously explored this concept as a possible feature, but it was never prioritised. AtB conducted a survey that indicated that over 50% of the participants would like an assistant for purchasing tickets.

1.2 Problem

In this project, the goal was to create assistive features that help users of the AtB app choose the cheapest ticket that meets their needs. The features should always be accessible but shouldn't interfere with the app's current design. The interface should be simple to use and need as few clicks as possible. From both a coding and design standpoint, it should integrate smoothly with the app. After narrowing down the scope, the team together with AtB came up with the following problem:

> *How can we help users quickly and accurately select the right ticket based on their needs, while keeping consistency with the app's current design language?*

1.3 Project constraints

Throughout the project, the team had several limitations and constraints. The primary expectation from AtB was a functional implementation that could be directly integrated into the app, imposing a high standard for code quality and testing. Thus, the team had to adhere to AtB's existing tech stack. These expectations, coupled with the necessity to submit multiple mandatory assignments throughout the project's timeline, imposed strict deadlines and added pressure to the development process.

Furthermore, the necessity to adhere to all relevant regulatory and compliance standards brought an extra layer of complexity to the task. Lastly, the requirement for the solution to be scalable and maintainable necessitated careful design and planning, which added to the already tight timeline. These limitations combined to create a challenging environment, yet they also served to clearly define the project's scope and expectations.

1.4 Report structure

The report is structured in the following manner:

Chapter 1 - Introduction provides the background of the project and the issues the team aimed to address.

Chapter 2 - Theory contains all relevant theory to be able to understand the report.

Chapter 3 - Method explains how different aspects of the project were solved.

Chapter 4 - Results showcases the results from the research and development.

Chapter 5 - Discussion discusses why the results were as they were, and what was positive and negative about them.

Chapter 6 - Conclusion summarises the findings, reflects on the project's success and limitations, and suggests potential directions for future work.

Chapter 7 - Social impact delves into the broader societal implications of the project, highlighting both direct and indirect impacts, and addresses how the solution aligns with sustainability goals.

2 Theory

This section introduces relevant theory for the thesis. Because this is a development and design project, this section will cover relevant technologies, development methods and concepts used throughout the entire process.

2.1 Technology

2.1.1 App development for mobile devices

Mobile application development is the process of creating software for smartphones, tablets, and other mobile devices. These apps can be preinstalled, downloaded from app stores or accessed through mobile web browsers. With the growing use of mobile devices for connecting to the Internet, organizations across industries are developing mobile apps to meet the needs of their customers, partners, and employees.

Apps can be developed either by using each platform's native development solutions or by using a cross-platform framework. Cross-platform development is the process of developing apps for two or more platforms from some or even all of the same source code (*What is cross-platform mobile development?* 2023).

2.1.2 Back-end as a Service (BaaS)

Back-end as a Service (BaaS) provides features like databases, user authentication, hosting, and serverless functions. Abstracting these into an easy-to-use Application Programming Interface (API) allows for rapid development as the BaaS takes care of scaling the relevant services (Cloudflare 2023).

2.1.3 Digital ticketing

Digital ticketing is a system where the ticket is virtual and grants the holder access to certain goods or services. Digital tickets should be secure, portable, widely accepted, user-friendly, viewable, and manageable. They can be implemented through an account-based or smart card-based system and have largely replaced paper ticketing systems (*Electronic ticket* 2023).

2.2 Optimisation problems

2.2.1 Dynamic programming

Dynamic programming is the process of breaking down complex problems into overlapping subproblems that often involve multistage, frequently recursive procedures. The subproblems are subsequently solved and entered into a dynamic programming table. In many situations, this method enables precise solutions in a very reasonable

amount of time. However, depending on the problem, dynamic programming solutions may be difficult to implement and may have higher time and space complexities than desired (Chumburidze et al. 2019).

2.2.2 Greedy algorithms

A greedy algorithm seeks straightforward, frequently multi-step solutions to complex problems. At each stage, they focus on the best local option in an effort to find the best overall solution. Greedy algorithms don't always result in the best solution, but they can deliver an approximation in a reasonable amount of time (Chumburidze et al. 2019).

2.2.3 Knapsack-problem

The knapsack problem is a widely studied optimization problem in computer science and mathematics. It involves choosing a subset of items to maximize their total value while ensuring that their combined weight does not exceed a given capacity constraint. This problem has practical applications in various fields such as scheduling, resource allocation, financial portfolio optimization, and cryptography. Moreover, it often serves as a benchmark for evaluating the effectiveness of optimization algorithms and heuristics (*Knapsack problem* 2023).

Definition:

> Given items of different values and volumes, find the most valuable set of items that fit in a knapsack of fixed volume. - Blac 2021a

Formal Definition:

> There is a knapsack of capacity $c > 0$ and N items. Each item has value $v_i > 0$ and weight $w_i > 0$. Find the selection of items ($\delta_i = 1$ if selected, 0 if not) that fit, $\sum_{i=1}^{N} \delta_i \cdot w_i \leq c$, and the total value, $\sum_{i=1}^{N} \delta_i \cdot w_i$, is maximized. - Blac 2021a

The knapsack problem is an Nondeterministic polynomial-time complete (NP-complete) problem which means that a solution to the problem can be checked in polynomial time, but finding the solution is not necessarily possible in polynomial time for big values of n. Therefore there is no known algorithm that is both fast and correct in all cases, only approximations exist. To approximate and/or solve the knapsack problem multiple approaches can be taken such as; greedy programming, dynamic programming or using meet-in-the-middle methods (*Knapsack problem* 2023).

2.2.4 Unbounded knapsack vs classical knapcsack

An Unbounded Knapsack is very similar to the classical knapsack problem. The core difference is that you are allowed repeats of items. As with the classical knapsack problem, the unbounded knapsack problem is also an NP-complete problem.

Formal Definition of unbounded knapsack problem:

> There is a knapsack of capacity $c > 0$ and N types of items. Each item of type t has value $v_i > 0$ and weight $w_i > 0$. Find the number $n_t > 0$ of each type of item such that they fit, $\sum_{t=1}^{N} n_t \cdot w_t \leq c$, and the total value, $\sum_{t=1}^{N} n_t \cdot w_t$, is maximized. - Blac 2021b

2.3 Agile devlopment methods

Agile methodology is a set of ideals and principles that guides teams through software development. It serves as a structured and iterative approach to management and product development (Rehkopf 2023).

2.3.1 Scrum

When using Scrum methodology teams complete increments of work in time intervals called sprints. The overarching goal of Scrum is to create learning loops where teams learn from previous sprints and feedback from customers (*What is Scrum?* 2023).

Scrum consist of three predefined roles:

> The Scrum Master is accountable for establishing Scrum as defined in the Scrum Guide. - Schwaber and Sutherland 2020

> The Product Owner is accountable for maximizing the value of the product resulting from the work of the Scrum Team. - Schwaber and Sutherland 2020

> Developers are the people in the Scrum Team that are committed to creating any aspect of a usable Increment each Sprint. - Schwaber and Sutherland 2020

Scrum comprises five events, primarily the Sprint, which sets the stage for all other events (Schwaber and Sutherland 2020). Sprints, lasting a month or less, provide rhythm and consistency. Each new sprint starts immediately after the previous one concludes. Sprint planning determines the value, objectives, and execution plan of the sprint. Daily Scrum, a 15-minute meeting, allows developers to update on progress and discuss issues. The end of each sprint involves a Sprint review to assess the outcome and a retrospective to strategise improvements for future sprints.

Scrum artefacts, representing work or value, include the product backlog, sprint backlog, and increment. The product backlog is an ordered list of what is needed to improve or further develop the product. The sprint backlog is a list of tasks, selected from the product backlog, that the development team plans to complete during a specific sprint to meet the sprint goal. An increment is a usable piece of work towards the product goal, completed within a Sprint. It builds upon all previous increments, is thoroughly tested and is ready for use.

2.3.2 Kanban

The central idea in Kanban is the use of a board to visualise and organise the work into specific categories like Stories, Todos, In progress and Done. Teams using Kanban are more focused on visualizing work and maximising efficiency (Radigan 2023).

2.4 Software Development Life Cycle

2.4.1 Version control

A software project's development is often managed via a Version Control System (VCS). It is widely used by software businesses to enable developer communication. Through the use of branches and merging, it enables concurrent work on the same project (Zolkifli et al. 2023). A version control system is often used in conjunction with a centralised hub which serves the repository (Source code management platforms). These platforms offer important services like the ability to request reviews of submitted code, effective branch management, continuous integration/deployment features and even static site hosting.

Code reviews and pull/merge requests are essential for maintaining code quality and fostering collaboration in software development. When performing code reviews, reviewers look at suggested code changes, offer feedback, and identify potential problems. Developers can submit changes via pull or merge requests for review, and after they are accepted in the code review, they can be merged into the main codebase. These features support knowledge sharing among team members and help guarantee high coding standards (Docs 2023).

2.4.2 Prototyping and wireframing

In software development, a prototype is a high-fidelity basic implementation of the vision of the product. This prototype is often constructed in a collaborative interface design tool. The software utilises a variety of vector graphics editors and prototyping tools to give the user the ability to graphically create a mock-up or even entire design schemes, as well as giving these mock-ups very basic functionality.

Wireframing is a low-fidelity, quick process in web or app development that creates a stripped-down, 2D representation of an interface. It prioritises content and functionalities without styling or graphics and establishes template relationships. This simplicity allows for rapid stakeholder approval and easy alterations if required.

Prototyping and wireframing facilitate the creation of a mock-up to evaluate planned features without investing time and effort into a complex solution that might later require significant alterations or be discarded altogether. Moreover, prototyping enables early user testing during the development process, offering essential insights into potential issues or enhancements that can be made (*The Importance of Prototypes in Product Design* 2019). Additionally, this approach fosters effective communication of project intentions and vision among the broader team and encourages contributions and feedback from fellow designers.

2.4.3 User testing

User testing is a critical component of software development as it helps to identify usability issues and gather feedback from users. There are various methods of user testing that can be used, depending on the research questions and the resources available. Some common methods used in software development are:

- **Usability testing:** This involves observing users as they perform tasks and collecting feedback on ease of use, efficiency, and satisfaction. Additionally, usability testing facilitates the identification of problems in the design and uncovering of opportunities to improve. One can also learn a great deal about user behaviour and the different preferences of different user groups (Moran 2019).
- **A/B testing:** A/B testing is the process of comparing two versions of an implementation to see which one performs better in terms of user engagement, conversion rates, or other metrics. What's more, A/B testing can be a powerful tool for software development, as it provides a data-driven approach to making design decisions (Nielsen 2005)

Furthermore, testing requires a diverse group of people. If a website/app has several highly distinct groups of users, 3-4 users from each category of users should be tested to ensure that most user needs are covered (Nielsen 2000). A lot of applications today are used by both young and old people, and people who are comfortable and uncomfortable with apps and technology in general. Nielsen 2000 therefore recommends testing 3 users from each category of users. Categories can be sorted in a multitude of ways, for instance by competence, age, or usage patterns.

2.4.4 QA-testing (Quality assurance testing)

Quality assurance (QA) testing is a methodical process of assessing a software product's functionality, performance, and usability to make sure it satisfies the user's expectations and meets requirements before deployment. A high level of QA stringency is required for products with large and diverse user bases or those that serve important societal functions. To guarantee that the solution works as expected in all scenarios, such products require extensive testing in various environments. Large corporations frequently assign an entire department to oversee the QA testing process and keep in close contact with the product's developers. Performance, integration, and usability testing are some crucial QA testing procedures (Feldman 2005).

2.5 Design

2.5.1 Universal Design

Universal design is a design principle with the overarching goal of ensuring content accessibility for all users, regardless of age, potential handicaps, or education level (*UUtilsynet* 2023). This principle helps guarantee that everyone can participate and utilise the various technical solutions available.

To achieve these objectives, the World Wide Web Consortium has developed guidelines known as the Web Content Accessibility Guidelines (WCAG). According to *Web Content Accessibility Guidelines (WCAG) 2.1* 2018, WCAG is defined as:

> WCAG covers a wide range of recommendations for making Web content more accessible. Following these guidelines will make content more accessible to a broader range of people with disabilities, including accommodations for blindness and low vision, deafness and hearing loss, limited movement, speech disabilities, photo-sensitivity, and combinations of these, as well as some accommodation for learning disabilities and cognitive limitations; but will not address every user need for people with these disabilities. - (*Web Content Accessibility Guidelines (WCAG) 2.1* 2018)

WCAG has various levels—A, AA, and AAA—to inform users of the accessibility standards met by the content.

UUtilsynet, the Norwegian regulatory body for Universal Design, mandates that all apps connected to the internet comply with WCAG 2.0. This is because such apps are considered web solutions and are therefore covered by *Lov om likestilling og forbud mot diskriminering, Kapittel 3, § 17 & § 18* 2017. As long as the implementation of the WCAG standards does not pose an unreasonable burden on the app distributor, universal design should be implemented in the main section of the app, as stipulated by *UUtilsynet* 2023. In regards to the specific criteria these apps should meet, public applications must adhere to all 47 points of the WCAG 2.0 checklist (Appendix F), while private applications need to comply with 35 of these points.

2.5.2 Screen readers

Screen readers are a necessity for blind, partially sighted, and people with reading difficulties. They allow for navigation of websites and applications with the use of voice assistants and other tools rather than the traditional tapping of buttons or mouse/keyboard input. The use of aria-labels and equivalents for iOS/Android allows screen readers to convey the meaning and function of elements in voice form or through other means to users (*An introduction to screen readers | AbilityNet* 2021).

2.6 Scientific theory

2.6.1 Deductive approach

Traditional researchers taking a deductive research approach, start with a hypothesis or theory, analyse data or samples and then check if the hypothesis is supported or not (Schmitz 2012). In the context of software development, a deductive approach can be defined as first formulating a hypothesis or assumption regarding user needs, preferences, or pain points. Then methodically gathering and analysing data via user testing, surveys, or other research techniques to verify or disprove these initial hypotheses. Ultimately guiding the design and development of the software.

2.6.2 Qualitative vs Quantitative methodology

Irrespective of the specific testing methodology chosen, certain aspects remain universally significant. The choice between qualitative and quantitative testing methods plays a critical role in determining the insights gained. Both approaches offer distinct perspectives, with qualitative methods emphasising subjective understanding and rich contextual data, while quantitative methods focus on objective measurement and statistical analysis. It is essential to carefully consider the merits of each method, as the selected approach will yield different insights that can shape the direction of the project (Jick 1979).

2.6.3 Triangle methodology

Triangulation is the practice of using multiple different methodologies to study the same question or phenomenon. Denzin defined it as:

> The combination of methodologies in the study of the same phenomenon - Denzin 1978

In the context of performing user surveys and user testing, this can be performing tests with both qualitative and quantitative methods. Maintaining focus on the same phenomena across different methodologies is essential for triangulation to be effective (Jick 1979).

3 Method

The task of this project was to develop a ticket assistant for AtB. After narrowing the scope, the primary task became to create a questionnaire that recommends the cheapest ticket based on a user's answers. The second task was to create a solution for edge case scenarios, for example when a user is travelling with a bike or at night.

A key aspect of the project was making a feature that looks and feels like the existing AtB application. The feature should also be perceived as valuable and beneficial by the users.

As a result, the team utilised a deductive approach to design the solution, making assumptions about what users would need before testing the theory and making adjustments. From a concept to a prototype to the final product, every development stage included user testing which led to changes that made the product user-friendly and easy to use.

3.1 Research method

Research methodology involves strategies, processes, and methods used to gather, examine, and evaluate data. It's essential as it provides a framework for collecting, analyzing and interpreting data.

In this thesis, a triangular approach utilising qualitative and quantitative methods was incorporated. This included a user survey, user testing and an evaluation of the final product. Every aspect of the research methodology enhanced the understanding of the issue and enabled the development of an appropriate solution to the research question.

3.1.1 Survey for mapping user-needs

The first method employed was a user-needs survey. This method is primarily quantitative and was used to gather preliminary data about user needs and the existing solution. The survey also served as a test for the assumption of what the users need.

It was distributed among acquaintances who were known to use public transportation in Trondheim, and they were asked to share it with others. In an effort to get answers from a broader range of users, participants were encouraged to share the survey with people from a different age group than themselves. To keep responses anonymous and be in compliance with General Data Protection Regulation (GDPR), the questions asked were not personally identifiable and did not collect sensitive personal information. Additionally, the survey was only sent to individuals over the age of 18 to avoid needing consent from any parents or guardians.

The questions in the survey primarily focused on the ticket-purchasing experience and the usage of the new app. This was to map current issues and find areas of improvement within the current ticket selection process. Another goal was to find out how many people still use the old app and why they haven't changed. This was done in hopes of figuring out if the solution could serve as motivation for people to make the change to the new app. The questions were constructed in a way that promoted

both numerical answers and more in-depth text answers to get both qualitative and quantitative responses.

For the creation and distribution of the survey, Nettskjema was chosen. Nettskjema is a flexible and secure tool for designing and collecting data (*Nettskjema* 2023). It is highly regarded when it comes to data privacy and conforming with applicable user data storage laws. This made it a good fit for the project.

To analyse the data from the user survey, quantitative responses were evaluated to identify trends, patterns and potential correlations between different factors. Qualitative responses from open-ended questions were categorized based on recurring themes. This helped identify common pain points and create a better visualisation of the user's needs (Appendix H).

3.1.2 User testing with digital prototype

Following the survey, a digital prototype was developed for user testing with the intention of finding the most user-friendly and intuitive approach. This method is mainly qualitative, including extensive, in-depth information about user interactions and preferences. To facilitate this process, a simple template with guidelines was created for those conducting user tests, along with a set of questions to prompt subjects. A criterion for the test subjects was that they regularly used one of the AtB's apps, while also having a focus on testing a wide range of age groups. To expedite the recruitment process, the team decided to engage with individuals easily accessible.

To record responses, at least one team member was present during each test. Their role was to document the user's impressions and responses, allowing for an unfiltered, firsthand account of the user's experience. This made it possible to observe both how the user interacted with the solution as well as having an ongoing dialogue in which the users expressed their opinions. It's noteworthy that the user was not required to write anything themselves; their only task was to test the features and engage in a conversation with the team member.

At the end of the tests, A/B testing was implemented to evaluate different versions of certain pages. This quantitative approach complemented the qualitative user tests, allowing for an evaluation of various design and functionality options based on user preferences.

During the user testing phase, the data analysis focused on the recorded observations and notes from the user sessions. The team reviewed each user's verbal and non-verbal feedback, highlighting any issues encountered, suggestions made, and positive experiences. The team then compared the feedback across multiple users to identify patterns and common challenges and made a document summarising the findings (Appendix I).

A qualitative approach together with usability tests was chosen since the objective was to understand how the prototype was used in a natural setting with as little intervention as possible (Lowhorn 2007). This approach allowed for a deeper understanding of how users navigated the app as well as potential design and flow issues. The modifications made to the prototype based on this feedback prepared it to serve as a foundation for developing a fully-fledged solution.

3.1.3 User testing of final product

After finishing the final product, another user test was conducted. This was another qualitative method used to gain detailed feedback on the finished solution. The user tests were conducted in the same manner as the prototype user tests to ensure consistency in the approach. Even though there wasn't time to make changes, the tests served as a foundation for future work and improvements to the solution. The participants were given a specific task to complete, which involved navigating to the ticket assistant feature, completing the questionnaire, and reviewing the recommended ticket. After completion, the questions that were asked were designed to elicit feedback on various aspects of the feature, such as ease of use, clarity, relevance, and user-friendliness (Appendix J).

3.1.4 Reflection on the research methodology

Combining the methods mentioned resulted in a deeper understanding of the users' needs. It also enabled an iterative approach to design and develop the final product. The survey provided a broad understanding of the problem, while the prototype testing provided more concise insight into user preferences. Finally, the user tests of the final product unveiled the strengths and weaknesses of the solution, as well as suggestions for future improvement.

Some alternative research methods were for example ethnography, the process of observing users interacting with the app undisturbed, or interviews instead of the user survey. These approaches are qualitative and time-consuming and therefore don't fit the purpose of getting a broad understanding of the user's needs. Instead of conducting user tests, using expert analysis could have been utilised. However, a priority was to get the opinions of the average user of the application. These methods ensured the triangular approach, which facilitated a robust understanding of the research question and helped produce an extensive and thoroughly researched, and user-centric solution.

3.2 Choice of technology

3.2.1 Tech stack

Since AtB already had a well-defined infrastructure in place, the technology stack for the project was mostly predetermined. By utilising the preselected tech stack, the team was able to focus on delivering a top-notch user experience while maintaining compatibility with AtB's existing ecosystem.

React Native with TypeScript

React Native is a cross-platform framework for application development. Several alternatives to React Native exist, including Ionic, Flutter, and Cordova. However, given that AtB already utilises React Native and has deep integrations with build tools like Metro, changing frameworks or build tools would be a considerable undertaking. As a result, React Native became a natural choice for the team, particularly when combined with TypeScript.

Combining React Native and TypeScript yielded the team seamless integration with the existing codebase and provided the needed cross-platform capabilities. This combination ensured developer productivity, code reliability, and broad compatibility, making it the ideal choice for building high-quality applications that align with the teams and AtB's goals.

Node.js

Developing and testing JavaScript applications locally requires a form of JavaScript runtime environment and there are multiple different solutions that exist today. For example Bun (ZIG and V8 based runtime) and Deno (Rust and V8 based runtime). However, the most widely used is called Node.js (C/C++ and V8 based runtime). Node has an extensive community with a lot of packages available through its package registry and manager called Node Package Manager (npm). Node was chosen because of its reliability, and it is also required for using development tools such as Metro and React Native, as well as a lot of essential packages that are only available through Node and npm.

Firebase

The usage of a BaaS has the potential to greatly speed up development. There is a multitude of different BaaS solutions, however, Firebase is one of the most popular choices. Other popular alternatives are Supabase, AWS Amplify or NHost, all with different pricing schemes.

AtB uses Firebase as their BaaS of choice because of its ease of use, a plethora of different products, and extensive community. Firebase however also comes with some limitations, for example, Firestore (Not only SQL or No SQL (NoSQL) Document database) is propriety, and Firestore is known for being hard to migrate away from (Pauly 2020). Firestore doesn't support complex querying of data, which leads to overfetching and the need for filtering either client-side or in-cloud functions. Despite some issues, Firebase's intuitive and powerful UI is ideal for front-end-focused developers. Coupled with AtB's integration requirement, Firebase was the preferred choice.

Rust with Actix

Rust is a fast and memory-efficient low-level programming language, with an extensive type system as well as an ownership guarantee model that guarantees memory-safe programs. The ownership model, or borrow checker, allows for checking ownerships at compile time, reducing the number of memory bugs encountered at run time helping new and inexperienced low-level developers greatly (Klabnik and Nichols 2023). Other low-level programming languages like C/C++ are also fast but don't offer the same kind of memory safety that Rust provides.

Furthermore, there exists a plethora of useful packages, like Actix for HTTP servers. Actix is a fast HTTP framework with built-in extractors and easy form handling (*Welcome | Actix* 2023). It is widely used in a wide range of environments from enterprise to hobby projects. Actix and Rust were a natural choice because of their speed, ease of use, as well as Rust ownership guarantee model that ensures memory-safe execution thus resulting in a better and faster development experience.

3.2.2 Design tools

Figma

Figma is a free online design and prototyping tool where you can work as a team on interactive prototypes or wireframes. The platform was chosen because it encourages collaboration, is cost-free, has a sizable user base, and makes it simple to make interactive prototypes that almost feel like the finished product. One of the most critical elements is live collaboration, which makes it possible for the team to work together on the project with ease. When creating a feature for a mobile application, having the option to run the wireframe on a mobile device is also a valuable tool (*Figma* 2023).

In addition, Figma is used by AtB to design and document their entire design process. This made it possible to reuse components, ensuring that the solution followed the design patterns of AtB and enabling easy collaboration with their internal designers.

Lucid Chart

Lucid Chart is a web-based tool for creating diagrams and charts. It was chosen as the preferred diagramming tool because it promotes real-time collaboration, is free, easy to use, and fits the need of the project. Some other appreciated features that Lucid Chart provides are a drag-and-drop interface and templates for many different types of charts (Chart 2023). In this project, Lucid Chart is mainly used to create flow charts for the ticket assistant and recommended tickets.

3.2.3 Administrative/collaboration tools

Slack

Slack was used for communication between team members and the rest of AtB. This was primarily because Slack was the existing solution at AtB. Everyone in the AtB development team, clocks in via Slack and indicates whether they will be working from home or in the office. This made it possible to always know who was in the office at any given time (*Features* | *Slack* 2023).

Discord

Discord, a Voice over IP (VoIP) communication platform, provides instant messaging, voice calls, and multimedia sharing. Its ability to create private servers and voice or text channels made it ideal for communication with the supervisor and the sharing of documents among team members (*Discord* | *Your Place to Talk and Hang Out* 2023).

Git

Git is the industry standard when it comes to Distributed VCS and is widely used in software development. There are alternative version control systems available, such as Mercurial or Apache Subversion. However, given Git's integration with IDEs, its large community with a lot of resources and tutorials and its adoption by AtB, it was the best choice for this project (*About - Git* 2023).

GitHub

The team utilised GitHub's Project feature as an issue board. This issue board was used to manage the product backlog, track progress during sprints, and keep track of who worked on what part of the project. GitHub's project feature ties well into the rest of the Git ecosystem, and updates with Pull request (PR) and closing of issues. GitHub also served as a centralised repository for all code and allows AtB to keep parts of the app open source. This allows external contributors to submit a pull request if they see a potential improvement or fix (*Features | GitHub* 2023).

Clockify

To log and keep track of working hours Clockify was chosen. Clockify is a free solution that allows for easy tracking of hour distribution. There are a lot of different solutions for hour tracking, but Clockify was seen by the team as the best and easiest solution as its free tier allows for all the features deemed necessary (*Clockify* 2023). This included tagging working hours with labels, seeing all hours from the team in a calendar, and being able to export all data easily.

Google Drive

As a cloud storage solution, Google Drive enabled all team members to access important files and documents while facilitating collaboration in the development and writing process (*Features | Google Drive* 2023).

Overleaf

Overleaf is a tool that allows for co-writing using LaTeX. Overleaf also serves as a form of version control, that made it possible to keep track of changes relating to important documents (*Overleaf documentation* 2023).

3.3 Development method

3.3.1 Wireframing and prototyping

Based on results from the user survey that was conducted to map user needs, a wireframe was created. This wireframe, with a primary focus on formulating a questionnaire and a Q&A page, was tested among potential app users before initiating any coding processes for the solution. It is essential to obtain input from the product's actual users. Additionally, this approach makes it simpler to experiment with various solutions. When compared to how much time writing code would take, making significant changes or adding new features is easy and time effective when creating a wireframe.

During the third sprint, the initial wireframe of the questionnaire and a Q&A page were created using feedback from the user survey. Along with the wireframe, functional requirements and user stories were made to define the different functionalities that the solution should provide (Appendix C & D). This wireframe was only shared with AtB's internal designers, and not distributed to actual users. The questions of the questionnaire in the wireframe were:

- How often do you travel?

- What type of traveller are you?
- For how long are you going to travel with AtB?
- What zones are you going to travel between?

Throughout the project, these questions largely remained consistent, with only minor modifications in their wording and sequence. To calculate a recommended ticket, this was the minimum required information needed. There was a dialogue about whether to add more detailed questions, but the simplicity of the questionnaire was prioritised instead.

The team then modified the wireframe in response to the designers' suggestions before it was made into a complete prototype. For users to be able to fully test the solution the prototype needed to be interactive. Feedback from these user tests prompted further refinements to the prototype. Ultimately, this prototype served as a foundation for the final solution (Appendix D).

3.3.2 QA testing

AtB's QA testing process is partially automated, ensuring that every pull request merged into the main branch triggers an automatic build of the application. This system allows the QA tester to quickly evaluate newly added functionality. Using Github's issue board, tasks are tracked from the design phase to approvals and production, providing a visual representation of the workflow and enabling efficient quality control.

For the purpose of this thesis, a task related to the project was introduced on the board and a pull request was created, to merge into the main branch. Upon receiving code approval, the pull request was merged and a build was acquired of the app incorporating the new features. Based on the functional requirements (Table 2), a list of acceptance criteria was developed, which the testers used as a reference during the testing phase. This comprehensive approach not only tested the application against predefined criteria but also included exploratory testing, where the tester attempts to identify potential bugs in ways a developer may not anticipate.

3.3.3 Ticket assistant algorithm

To provide a ticket recommendation, a calculation was required to determine the cheapest ticket combination based on user input. The problem at hand is finding the cheapest combination of tickets based on a few parameters. This problem turns out to be a version of an unbounded knapsack problem, where the max weight is the total travel duration, the values are the cost of the tickets, and the weights are the lengths of each ticket. The problem can be divided into two sub-problems, solving for combinations with a duration less than or equal to the total travel duration, and combinations with a duration of more than or equal to the total travel duration (Appendix K).

To solve both sub-problems, a bare-bones project was first developed in a separate Python environment with the only focus being on the algorithms. Using this method made debugging and testing the solution in isolation more straightforward. Additionally, this approach also ensures reusability and better documentation. After an initial

version was created in Python, work began on integrating it into AtB's Rust back-end (Appendix L).

3.3.4 Work distribution

To optimise each team member's productivity and capitalise on their individual strengths, the team opted to divide responsibilities. This approach enabled everyone to focus on the areas where they believed they could make the most significant contributions. However, as is common in many projects, some overlap transpired, resulting in each team member participating in all the different processes. Essentially, two team members focused their work on the front-end part of the project, while the last focused on the endpoint.

After creating a prototype in Figma, the next step was to implement the solution into the AtB app. This integration process consisted of two primary components: front-end implementation and back-end implementation. The back-end portion involved developing an endpoint within the AtB back-end to calculate the most cost-effective ticket combination. The main challenge centred around connecting to the EnTur API and ensuring that the implemented algorithm functioned effectively for all scenarios and within the context of the AtB back-end.

The front-end team focused on writing the code for the solution, including the questionnaire and the Q&A page. In constructing the questionnaire, a significant emphasis was placed on ensuring the solution's appearance and user experience aligned with the rest of the application. The primary challenges involved creating custom components that did not previously exist and guaranteeing that state management and communication with the back-end functioned as anticipated.

3.3.5 Agile development

Agile development techniques were used in the form of SCRUM, along with a digital Kanban board (Appendix G). Scrum was followed with some modifications like not having a Scrum master and not doing daily Scrum. However, the sprint system was strictly followed with sprints, sprint reviews, and sprint planning (Appendix B). In the planning phase, the team created a backlog with major tasks for the project period. These were used as a starting point for creating smaller more concrete tasks later on. Issues for the upcoming period were listed on the Kanban board, which was managed on GitHub. The project's milestones and several working areas were used to categorise and group issues. Group members were also assigned their own issues, which made it possible for everyone to know what each member was working on at any time.

3.3.6 Integration with AtB

In developing the solution, AtB's adopted practices were followed to ensure effective integration. Central to the workflow were the version control and branch management strategies on GitHub.

During the project, the team worked together with the development team at AtB's office. The goal was to learn how to operate in a professional environment and adhere to tried-and-true work ethics in order to prepare for a career as software developers. Despite managing to integrate smoothly into the AtB team, there were some limitations because of the Bachelor thesis. For example, the team couldn't follow the same sprint structure as the rest of the office, because a lot of time had to be spent writing and documenting the process.

Each team member was responsible for a specific feature, developed on an individual branch. This branch-based approach streamlined parallel development while also minimising potential code conflicts. These feature-specific branches were periodically merged into a central branch for the ticket assistant, maintaining an organised and manageable codebase. Merging required mandatory code reviews to maintain code quality and uphold the integrity of AtB's codebase. This not only ensured high coding standards but also encouraged knowledge sharing and team collaboration.

4 Results

The results section presents the various results produced during the project. The scientific result focuses on the results achieved while trying to answer the problem at hand. This includes the user survey, the user prototype with user tests, and the final product with user tests. The engineering results show the results of the goals set before the project started, both functional and non-functional. Lastly, the administrative results show how the team has worked during the project, including the hour distribution report and other administrative results.

4.1 Scientific results

4.1.1 Results from user survey

The user survey received 146 answers covering all ages from 18 to 65 and older (Figure 2), with the majority falling into the 18 to 25 category. There was also a wide spread in technical ability, ranging from 0 to 10, where 10 was best. Most people gave themselves a 7 while only 2 gave themselves a rating of less than 4 (For full result, see appendix H).

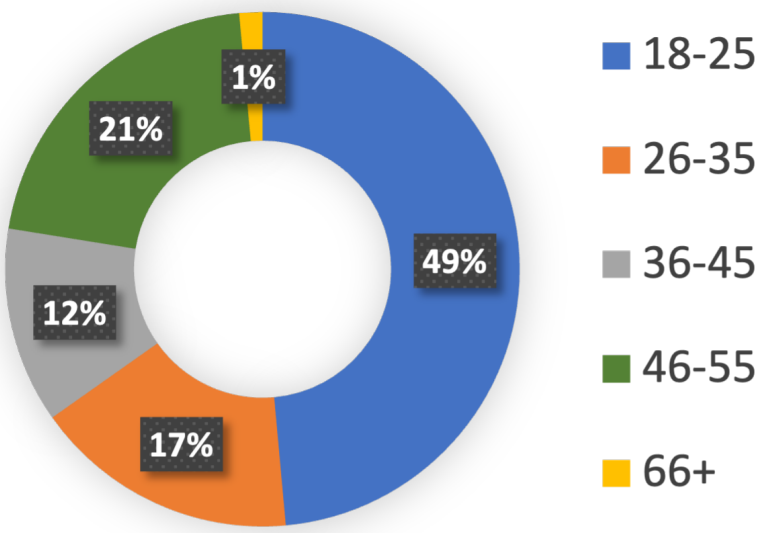


Figure 2: Figure showing what type of subjects participated in the user survey

Usage of the new app

The participants were then asked whether they used the new AtB app or another solution. 59% of respondents said they used the new app. Participants who said no to the previous question were asked why. The primary reasons for not using the new app have been grouped and are presented in Figure 3.

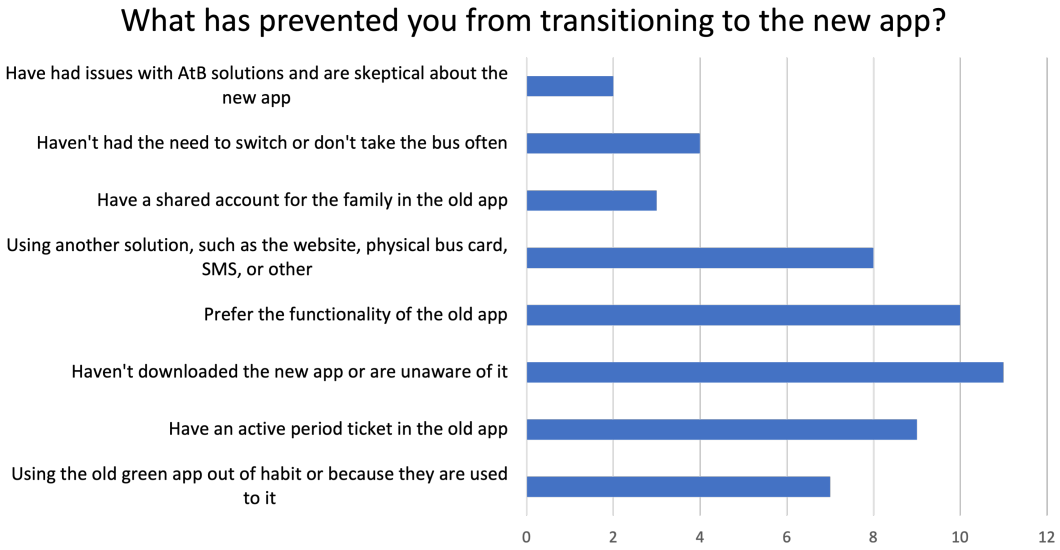


Figure 3: Figure showing text answers grouped into general categories

Ticket purchasing experience

One of the last questions was ”Do you find it easy to determine the appropriate ticket to purchase for any given situation?” Most participants reported that they find buying a ticket in the new app easy. Less than 25% of respondents gave themselves a rating of 5 or below for their capacity to identify the right ticket to buy in every circumstance. 10 represents an easy experience and 0 represents a very tough one (Figure 4).

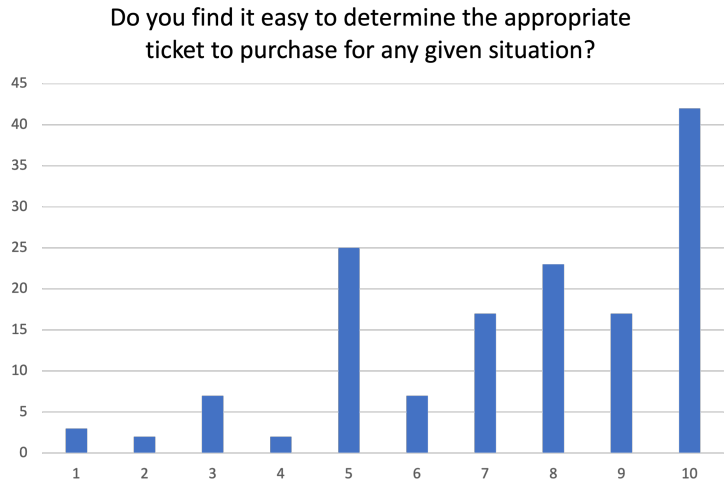


Figure 4: Figure showing the answers to the question ”Do you find it easy to determine the appropriate ticket to purchase for any given situation?” where 10 is very easy and 0 is very hard.

Next, people were asked how the ticket-purchasing experience could be improved. The answers have been grouped together in Figure 5

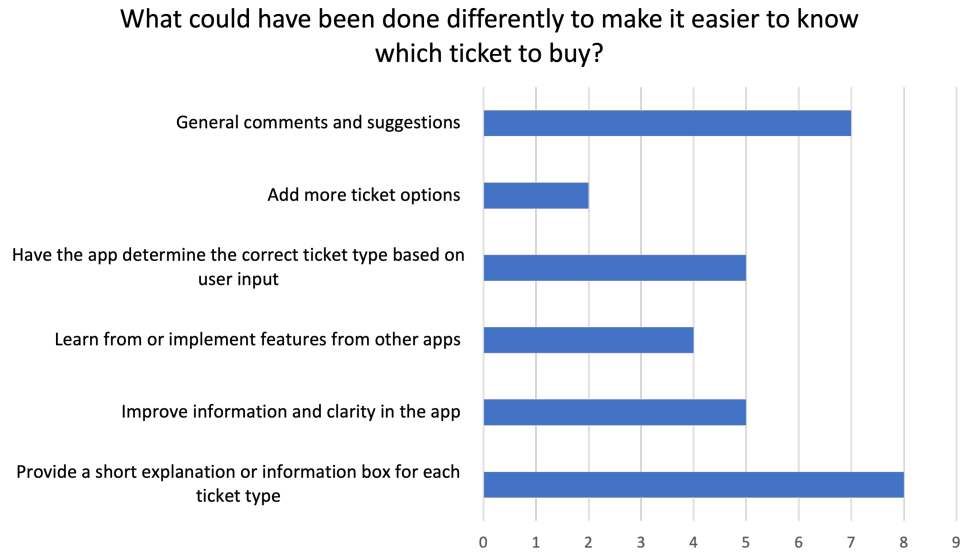


Figure 5: Figure showing text answers to how the ticket purchasing experience could be improved, grouped into general categories

Finally, when asked if they think that they always buy the cheapest tickets for themselves, two-thirds of respondents said they did. When asked if they wished for the app to recommend the cheapest ticket based on their travel history, 87% said yes. The last question asked if the user had any other suggestions on how to improve ticketing, the answers have been grouped into several categories (Figure 6).

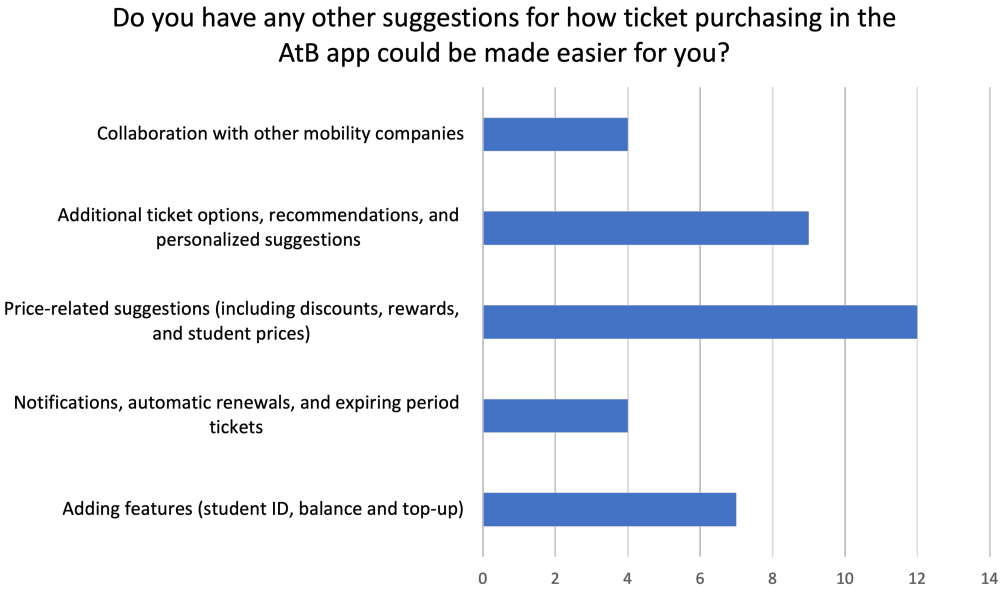


Figure 6: Figure showing text answers to how the app experience could be improved, grouped into general categories

4.1.2 Results from user test with digital prototype

After creating the digital prototype, a small group of people gave their feedback. The group consisted of people from different age groups (Figure 7).

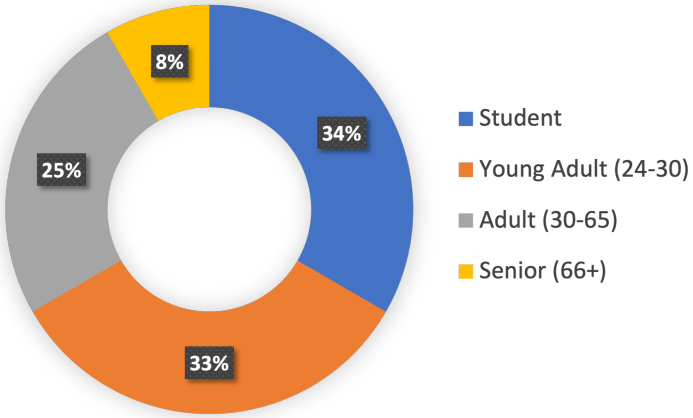


Figure 7: Figure showing what type of subjects participated in the user tests, 4 were students, 4 were young adults, 3 were adults and 1 was a senior

Digital ticket assistant

The following prototype is what was created based on the mapped user needs. All headings are in Norwegian for the first prototype because it was only tested on Norwegian-speaking people.

Ticket overview page

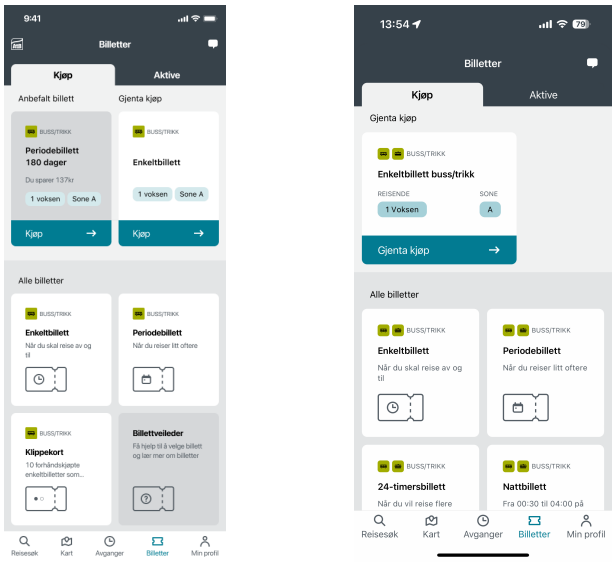


Figure 8: Ticket overview page of the first prototype compared to the app

This is how the digital assistant was first integrated into the app. The feature was located at the bottom of the ticket overview page and had a slightly different colour to the tickets, making it stand out, but not too much. In addition, at the top, the user could find a recommended ticket based on historical data.

Ticket assistant welcome page

Figure 9: Landing page of the ticket assistant

Once the user opened the ticket assistant, they were greeted by the welcome page before they started answering questions. This is also where the user could access the tips and information page.

Q&A page

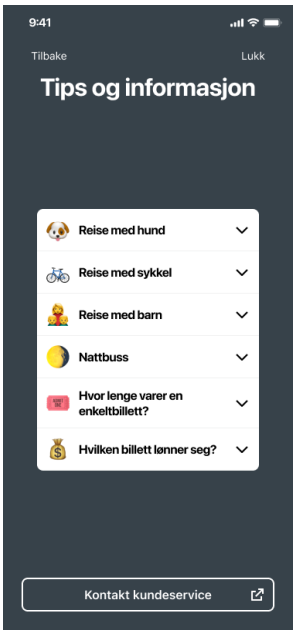


Figure 10: Tips and information page

The first Q&A page consisted of multiple dropdown components with frequently asked questions. At the bottom, there was also a button to contact customer support.

Travel frequency

Figure 11: Travel frequency selection screen

The user could indicate their frequency of travel on the travel frequency screen. It was answered with a slider that the user could move to a value they found most correct.

Traveller type selection

Figure 12: Traveller type selection screen

Selecting a traveller type was done on the traveller type selection screen by expanding one of the categories and clicking on the select button. After expanding a traveller category tile, the user was presented with more detailed information about the traveller type.

Travel duration

Figure 13: Travel duration selection screen

Choosing a duration for the user's travel could be done through two methods: using the date picker or opting for one of the quick answer alternatives. This allowed the user to either select a specific date or provide an estimated duration.

Zone selection

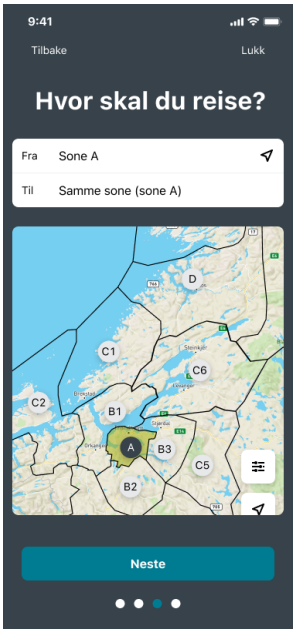


Figure 14: Zone selection screen

The zone selector page is where the user chose what zones they were going to travel through, during the time frame specified. To select a zone the user could either use the map or the buttons at the top. This component is essentially the same as the zone selector that has been used in other places in the app for consistency.

Recommended ticket screen

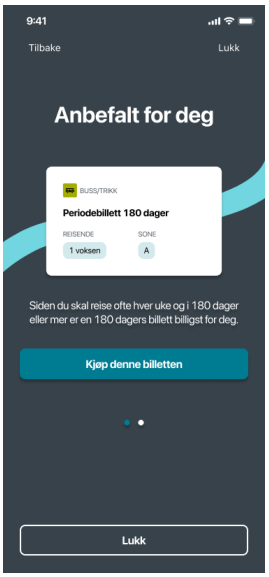


Figure 15: The screen displaying the recommended ticket based on the previous answers

The last page was the ticket summary page, which is where the recommended tickets were displayed. The intention of the primary button was that it would take the user directly to where they could purchase the recommended ticket.

Summary of findings

The user tests of the digital prototype provided a lot of good feedback and important notes (Appendix I). During the initial phase of user testing, some major usability issues were identified. Specifically, there were issues related to unclear language in some of the headings and descriptions, which caused confusion. Subjects also reported difficulty with finding the assistant. However, users generally had a positive perception of the prototype and found the flow natural and intuitive to navigate.

Some common complaints were:

- The ticket assistant is not easy to locate
- Language throughout the solution is not natural and hard to understand
- The first page of the flow is very bulky and takes up a great deal of space
- Subjects wanted to see more clearly how much they save by choosing a specific ticket
- The duration picker was overwhelming and at the same time lacking a quick choice that was specific enough for the user.

4.1.3 Changes made to the prototype

Integration

Feedback from the user tests led to the separation of the ticket recommendation questionnaire and the tips and information screen. Now, the latter was located at the top of the tickets overview page for easy access, while the questionnaire remained at the bottom. This time, with a design that made it easier to differentiate from the tickets. It also resulted in the removal of the recommended ticket at the top of the screen.

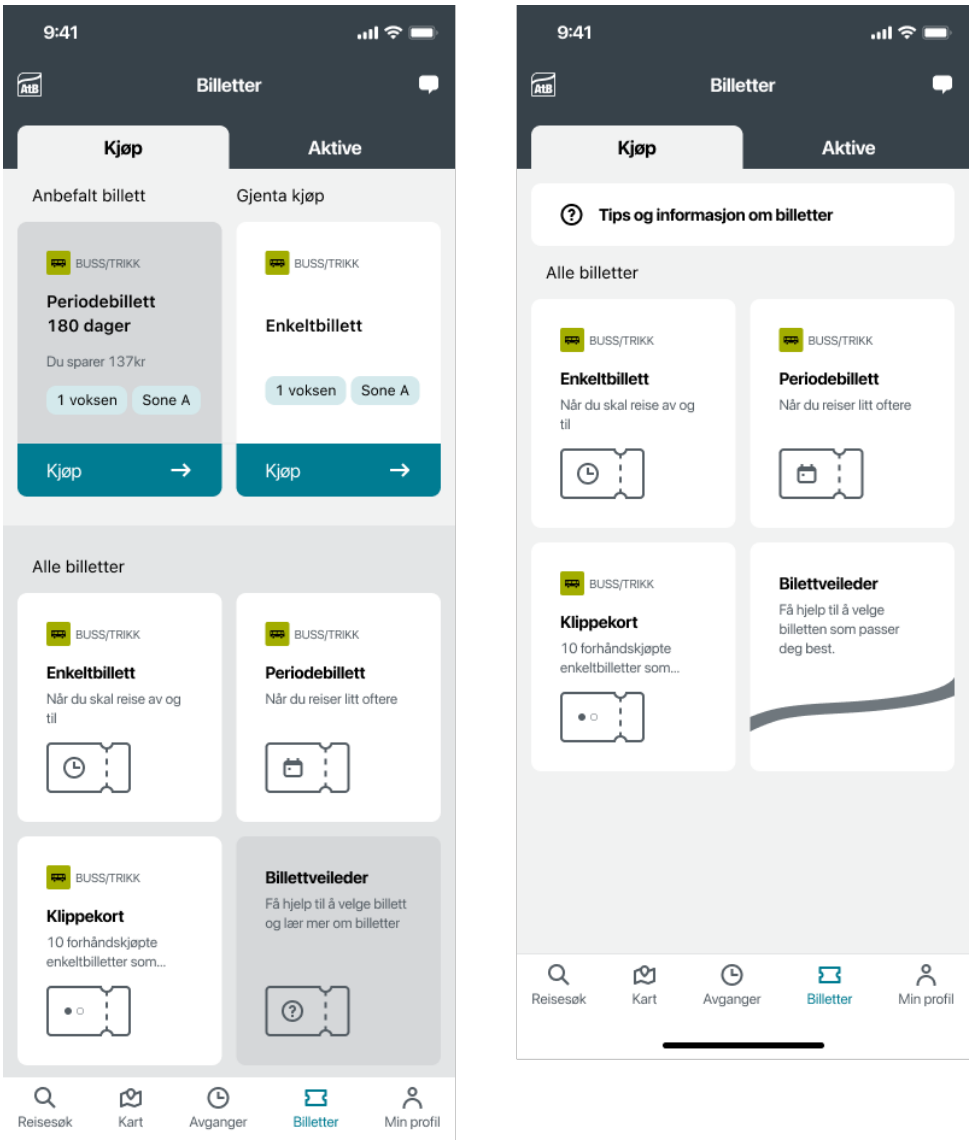


Figure 16: First(left) and second(right) iteration of the ticketing page

Ticket recommendation questionnaire

After going through the results from the prototype tests, the following changes were made (Figure 17). The biggest changes included a redesign of the duration picker and adding more information to the summary page. Including ticket cost and savings was one of the most requested additions from the user tests. Based on the results of A/B testing (Figure 18 & Table 1), the category picker remained with the expandable tiles. Also, the A/B testing results of the duration picker showed that the user wanted the quick option alternatives but wanted it to be more subtle (Figure 19 & Table 1). Therefore, none of the options were used, but a new duration picker component was made with an emphasis on the feedback received (Figure 27).

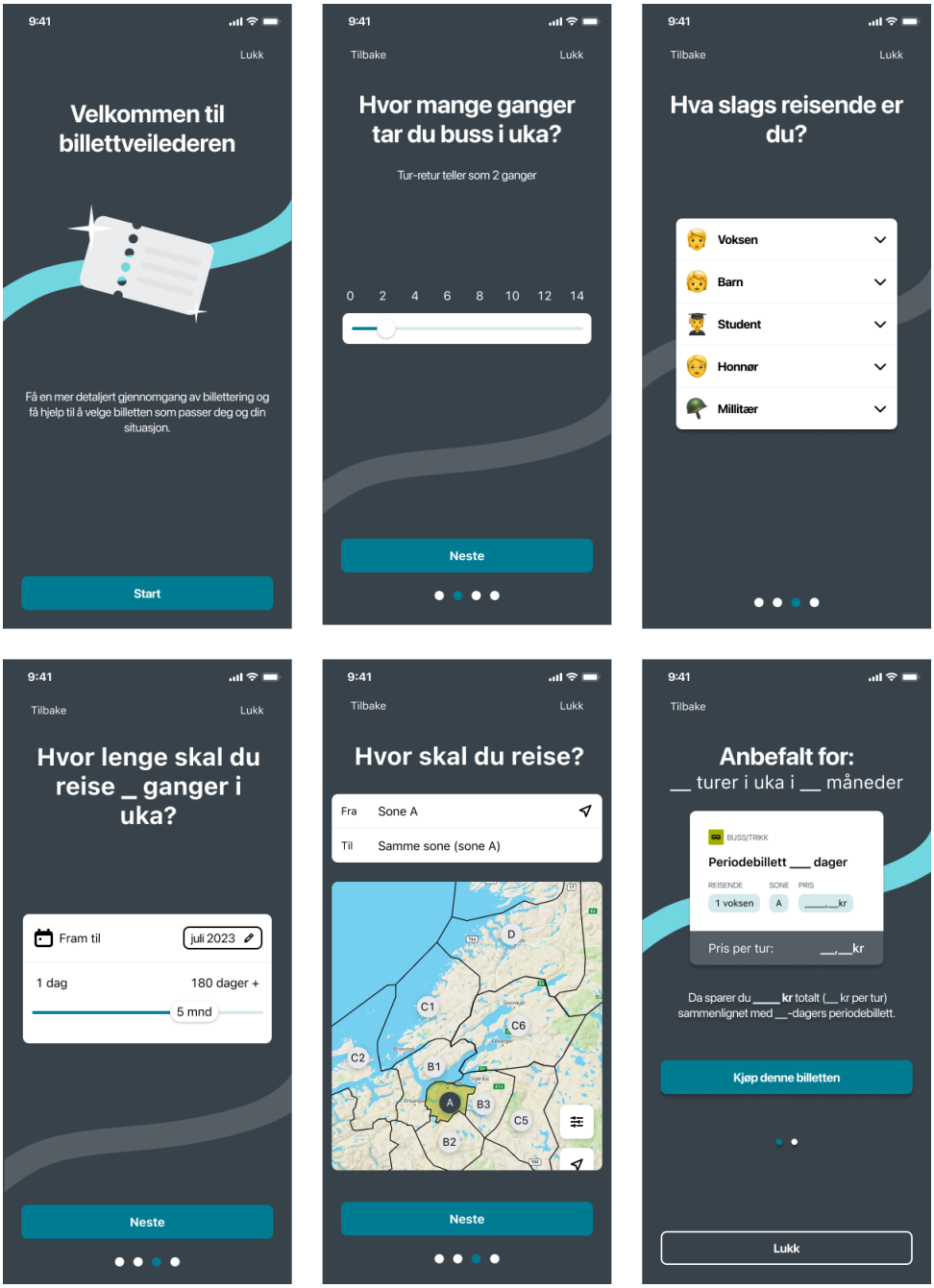


Figure 17: Final iteration of the questionnaire prototype

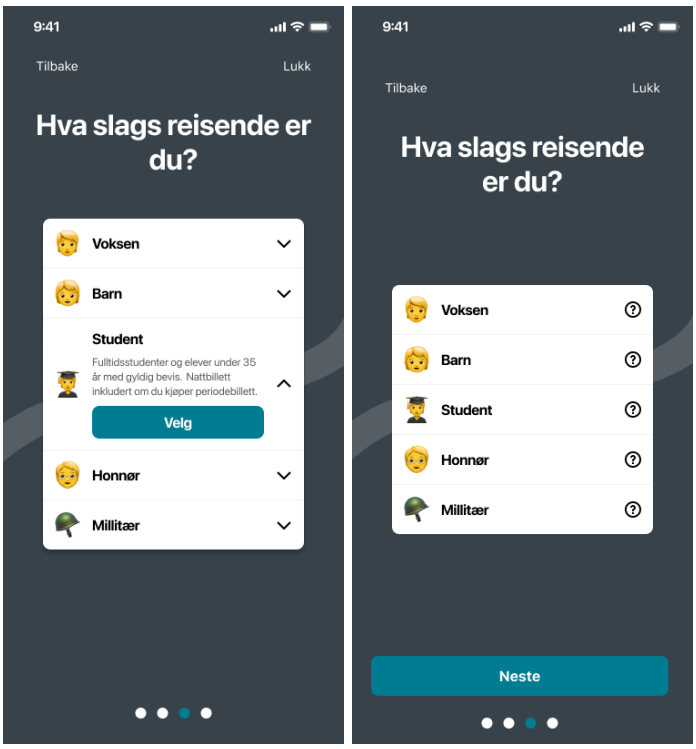


Figure 18: A/B testing of the category selection screen. Option 1 to the left, and option 2 to the right.

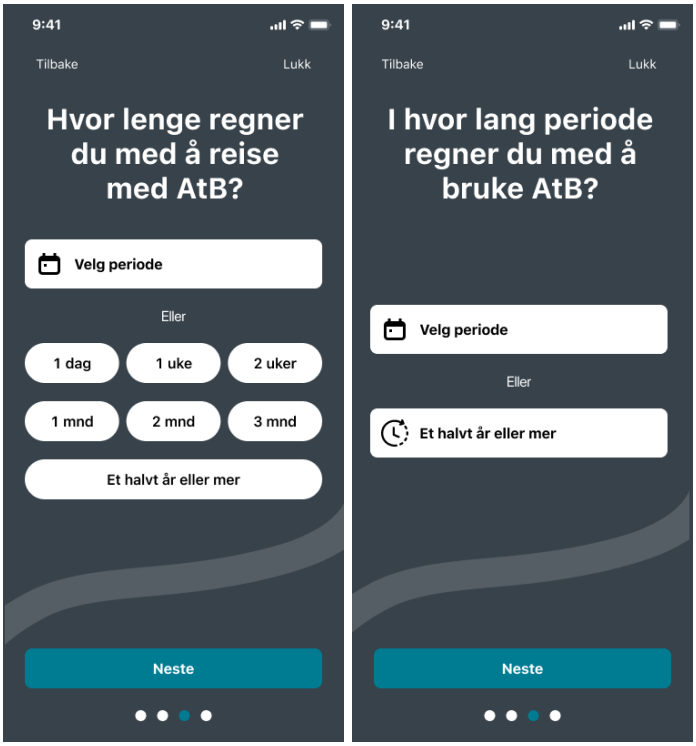


Figure 19: A/B testing of the duration selection screen. Option 1 to the left, and option 2 to the right.

Page	Option 1	Option 2
Category	11	1
Duration	9	3

Table 1: Results from A/B testing for the category and duration page during the prototype user tests.

Tips and information page

Feedback received following tests on a customer service representative suggested that the customer service team's workload should not be increased. Because of this, the customer service contact button was removed (Figure 20).

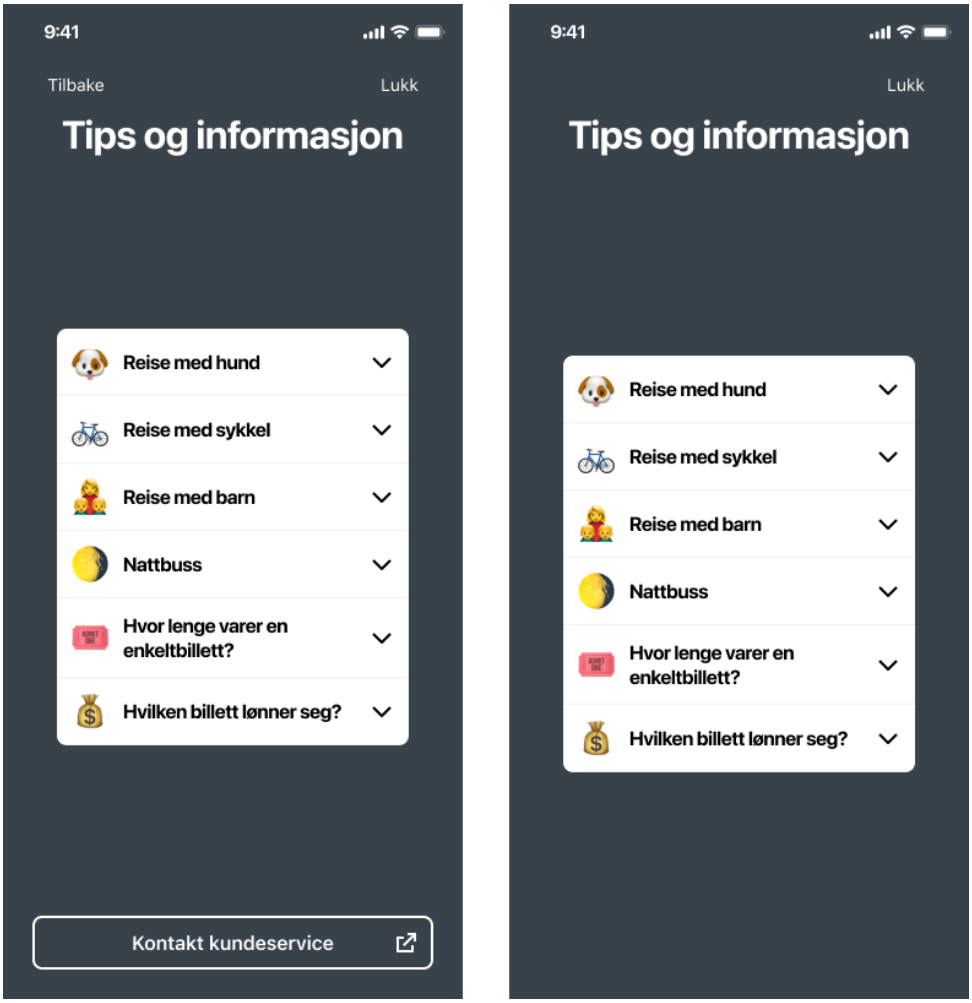


Figure 20: First(left) and second(right) iteration of the tips and information page

4.1.4 Final product

The final version of the solution is divided into two main parts: a Q&A page for common questions travellers with AtB has, and a ticket questionnaire for calculating what ticket is cheapest based on the user's travel patterns. Both solutions are on the tickets overview page in the AtB app. The Q&A page is a single page with questions and the questionnaire consists of six pages of which four are questions. The flow of the solution can be seen in figure 21.

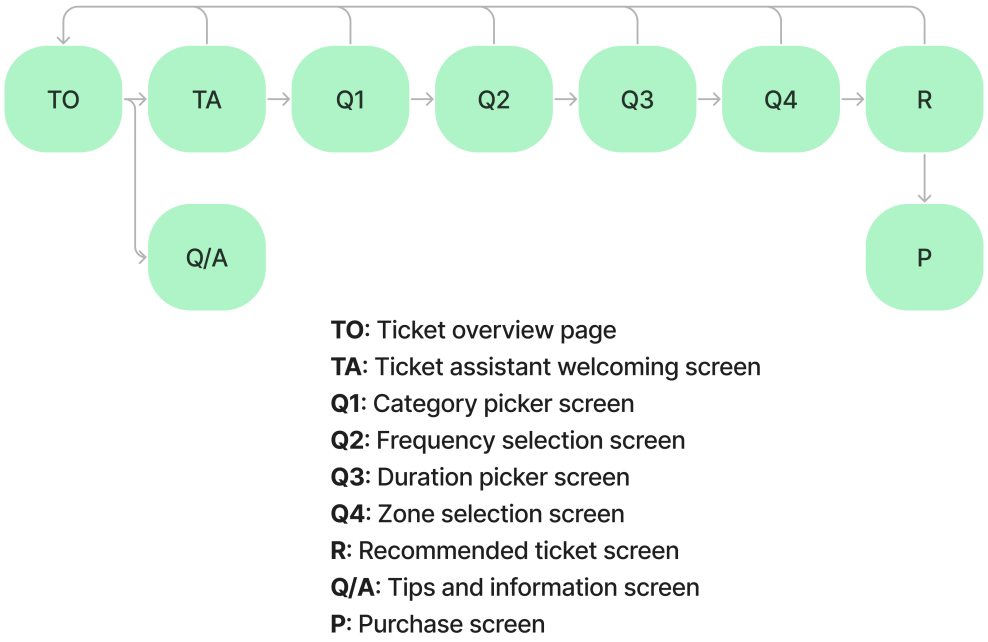


Figure 21: Flowchart of the solution

The solution focuses on employing many of the same components that the AtB already utilises. However, when a component didn't exist, it was created with an emphasis on being similar to the design system of the application. The final solution consists of the following pages:

Ticket overview page

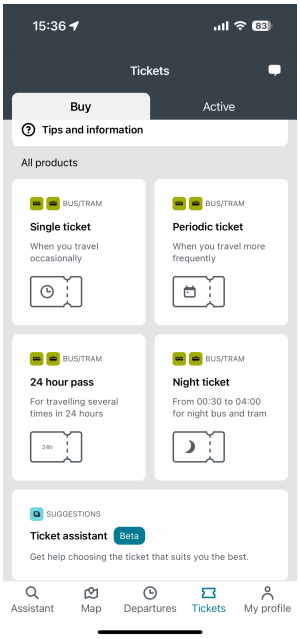


Figure 22: Ticket overview page

The ticket assistant is now labelled with a beta tag, suggesting that it is a feature still not fully developed. It also has a slightly different style to the tickets, making it slightly stand out.

Q&A page

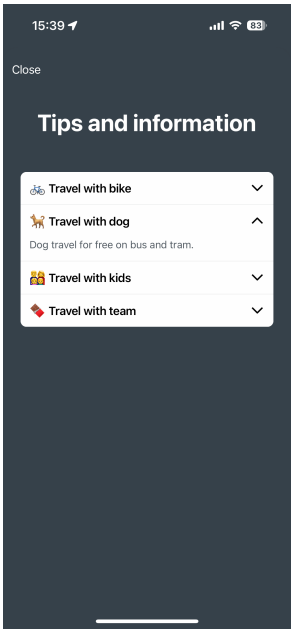


Figure 23: Tips and information page

The Q&A page is mostly the same with the biggest difference being that the content is fully modular and can be changed whenever desired from Firebase. The option for navigating to the questionnaire via the tips and information page is also removed.

Ticket assistant

To follow the existing navigation pattern in the app, both ”back” and ”close”, which in the prototype were on each side at the top, are now combined. If the user is on the first page, it shows ”close”, otherwise it says ”back”. Users can also swipe between pages.

To make it possible for the user to give feedback on the assistant or ask questions, a chat icon has been placed in the top right corner.

Welcome page

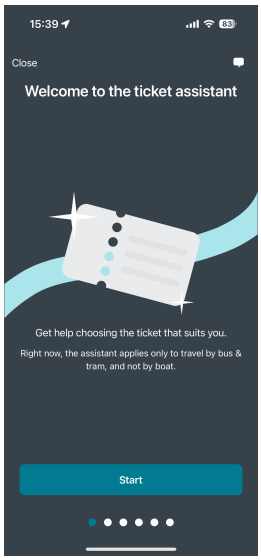


Figure 24: Landing page of the ticket assistant

The welcome page now explains what the ticket assistant is and some of its limitations to avoid misunderstandings.

Traveller type selection

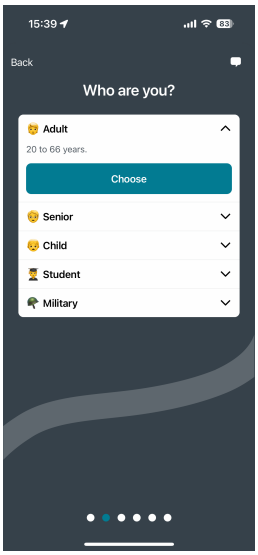


Figure 25: Traveller type selection screen

Traveller selection has had minimal visual changes.

Travel frequency

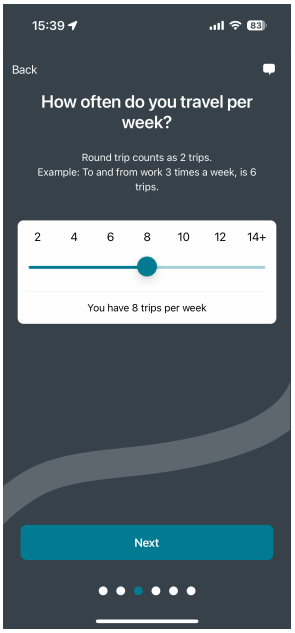


Figure 26: Travel frequency selection screen

The slider component received a change of colour on the thumb and the addition of the current selection at the bottom.

Travel duration

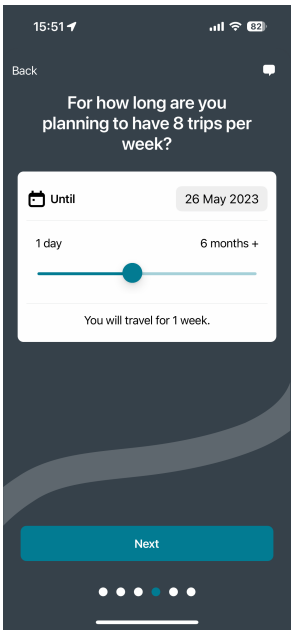


Figure 27: Travel duration selection screen

Selecting a duration for how long the user will travel, can be done in one of two ways; Using the date picker or a slider. Both elements are interconnected, using the date picker causes the slider to move, and vice versa, the slider causes the date picker to change. As with the frequency, a text previewing the current selection is added at the bottom. This page had the most changes from the first prototype to the final product.

Zone selection

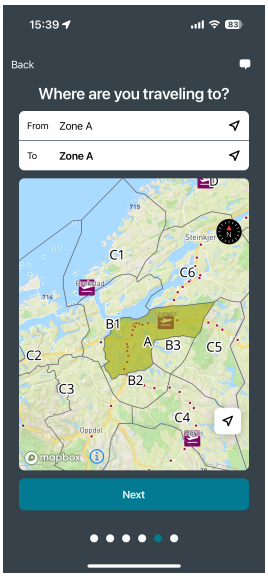


Figure 28: Zone selection screen

The zone selector page did not see any changes as it should still be the same as in the rest of the application.

Recommended ticket screen

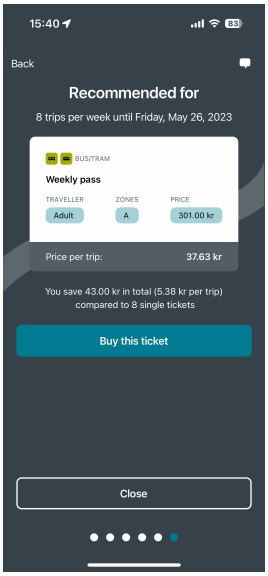


Figure 29: The screen displaying the recommended ticket based on the previous answers

Lastly, the summary page now displays a notice if the recommended ticket's duration is less than the duration provided by the user, saying that the recommended ticket doesn't cover the whole period. The primary button still takes the user to where they can purchase the ticket, and the secondary button is to close the assistant. Most importantly, the summary page now displays how much the user saves by choosing the recommended ticket rather than buying single tickets, both in total and per trip.

Adaptations for screen reader

Several pages within the assistant have been changed to improve navigation for users who rely on screen readers due to the highly visual nature of some components. When a screen reader is activated, these changes happen: replacing sliders with buttons, hiding the map and simplifying the dropdown category selector

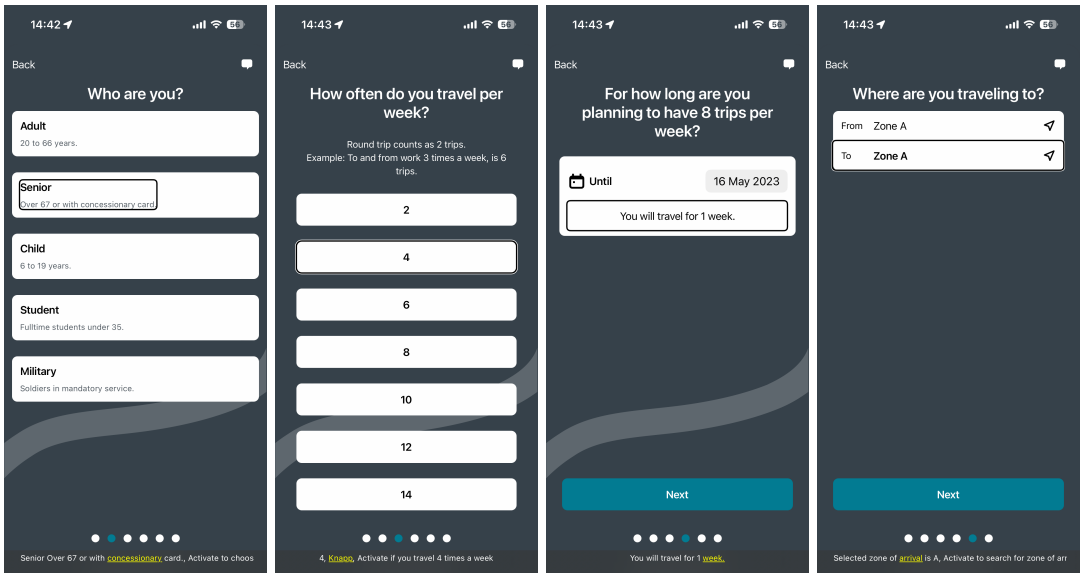


Figure 30: Adjusted screens for screen reader

Furthermore, each component has been supplemented with a more descriptive explanation to facilitate a better understanding of the consequences of different actions.

4.1.5 User tests of final product

Overall, most users found the Ticket Assistant feature easy to locate and access, although some users did not initially notice it. Questions in the questionnaire were generally clear and easy to understand. Suggestions included incorporating night tickets, options for travelling with others, and additional categories like youth. Although the user interface was visually appealing, some users experienced issues with hitboxes and navigation. The overall flow of the questionnaire was logical and user-friendly. The recommended tickets were highly relevant to users' needs, and users understood the next steps after receiving the recommendation. Users provided additional feedback on improving the specific text, navigation, and displaying the price per day.

4.1.6 Ticket combination calculation

The final method for calculating the cheapest combination of tickets is a three-part method consisting of an algorithm for solving unbounded knapsack, a greedy algorithm and a simple calculation for the cost of only using single tickets. Using these three separate algorithms and comparing their results aims to find the cheapest possible combination of tickets (Appendix K).

The algorithm for solving the unbounded knapsack problem uses dynamic programming inspired by the bottom-up manner method described in Cormen et al. 2009. It returns a combination of tickets with a total duration lower or equal to the given duration. Then, it leverages the dynamic programming approach explained in the textbook to build a table of solutions for sub-problems, ultimately computing the solution to the overall problem. By initialising the table with appropriate base cases and iteratively filling the table in a bottom-up manner, the algorithm effectively explores the solution space for the unbounded knapsack problem.

The second part of the solution is a greedy algorithm for finding the cheapest combination of tickets with a total duration of more than or equal to the given time period. It goes through all possible combinations that satisfy the conditions and returns the cheapest.

Then, the cost of only using single tickets is calculated. This is done because there are some cases where single tickets are the cheapest option. Finally, the solution compares the total costs of the different approaches and returns the cheapest combination of tickets.

4.2 The products functional requirements

Together with input from AtB and results from the user survey some functional requirements were designed together with the vision document (Appendix C). The following table displays some of the requirements and whether they were achieved.

Functional requirements	**Achieved**	**Not achieved**	**Comment**
1. Always available from the ticket view	**X**		
2. Tips and information	**X**		
2.1 List view with information	X		
2.2 Emojis	X		
2.3 Expand on click	X		
2.4 Redirect to questionnaire		X	Not implemented as it would need extra maintenance
3. Ticket-advisor	**X**		
3.1 Welcome page	X		
3.2 Navigation buttons	X		
3.3 Close button	X		
3.4 Frequency page, choose the frequency with a slider	X		
3.5 Category picker, choose what kind of traveller you are	X		

Continued on next page

Table 2 – continued from previous page

Functional requirements	Achieved	Not achieved	Comment
3.6 Duration selector, choose the duration of your travel period either with a slider or a date picker	X		
3.7 Zone selector, select the zones you want to travel in	X		
3.8 Recommendation page	X		
3.8.1 Two recommended tickets		X	Decided not to recommend two tickets, but rather compare to buying single tickets
3.8.2 Button for buying ticket	X		
3.8.3 Information about how much you save compared to other tickets		X	Compares only to single tickets
4. Can be navigated using screen-readers	**X**		
5. Displaying the recommended ticket in the ticket page		**X**	Not prioritised

Table 2: Achieved functional requirements

4.3 The products non-functional requirements

Non-functional requirements describe the features or properties of the system. These specifications, which cover issues like dependability, scalability, maintainability, and conformity with industry standards, are crucial in defining the product's quality, effectiveness, and user satisfaction.

4.3.1 Universal design

In an effort to promote inclusion and accessibility for all users, the project successfully incorporated universal design principles. AtB created the design system used in this project, aligning it with universal design principles, including the Web Content Accessibility Guidelines (WCAG) 2.0. A number of evaluations, like contrast checks, were already conducted by the AtB team to ensure that the design system met the relevant accessibility standards.

Assessing Compliance with WCAG Requirements

UUtilsynet's provided assessment form was employed in order to evaluate the solu-

tion's compliance with the WCAG requirements.

UUtilsynet states that public sector-developed apps must adhere to all 47 requirements outlined in the form. The solution falls short of meeting 2 of the applicable requirements for our project. The detailed results can be found in appendix F while the summarised scores are presented below:

Fulfilled	Not Fulfilled	Not Relevant
28	2	16

Table 3: Compliance with WCAG Requirements

The unfulfilled requirements are:

- **1.3.4 - Orientation:** "The intent of this Success Criterion is to ensure that content displays in the orientation (portrait or landscape) preferred by the user. Some websites and applications automatically set and restrict the screen to a particular display orientation and expect that users will respond by rotating their device to match, but this can create problems. Some users have their devices mounted in a fixed orientation (e.g. on the arm of a power wheelchair)." (WAI) 2022b.
- **2.5.3 - Label in Name:** "The intent of this Success Criterion is to ensure that the words which visually label a component are also the words associated with the component programmatically. This helps ensure that people with disabilities can rely on visible labels as a means to interact with the components." (WAI) 2022a.

4.3.2 AtB guidelines

As a well-established company, AtB has provided specific guidelines that must be followed to be able to get the feature into the app.

- **Language requirements** - To match AtB's communication style in both English and Norwegian, a continuous dialogue with the market department has been upheld to clarify the texts used.
- **Feature switch** - To address potential bugs with the new ticket assistant, a feature switch with remote configuration in Firebase is implemented. This allows the feature to be removed from users' devices in real-time if necessary.
- **Minimal maintenance** - The ticket assistant feature has been designed to require minimal maintenance by utilising existing values and data from the app. This means that no further adjustments will be needed if a category is added or a text is changed. By leveraging the app's existing resources, the amount of ongoing maintenance required for the new feature is reduced.

4.3.3 Scalability

The endpoint for obtaining the most affordable ticket option is designed for scalability and adaptability to potential changes in AtB's ticket offerings. For example, if a new

ticket is introduced, the price changes, or a ticket is removed, the solution should still function as expected. Moreover, the solution can adapt to different zones since they don't all have the same types of tickets. The ticket assistant is also designed to be scalable, accommodating changes to categories, zones, and ticket types over time.

This was achieved by ensuring that all ticket information is fetched from external sources. These external sources act like a Single source of truth (SSOT) and allows the endpoint to always have updated and correct information. Furthermore, all filtering of tickets is performed with filters sent from the front-end which in turn are fetched from a SSOT.

The project adheres to AtB's development guidelines, ensuring that the project is easily understandable without prior knowledge. Also, the back-end is well documented and if someone else reads the documentation, they should have no trouble understanding it.

4.4 Administrative results

4.4.1 Progress plan

The planning of the project included a GANTT diagram for creating a rough estimation of when and how much time would be spent on each part of the project (Appendix A). This diagram served as the guideline for planning the sprints which were followed throughout the project. Only small changes were made to the sprints during the project period.

4.4.2 Hour distribution

All team members used Clockify to track hours and log what was worked on. The categories were made using the GANTT diagram as a reference, but some categories were added to provide more context. Estimated and actual time distribution can be found in table 4. The goal was to work evenly throughout the project. From the start of the project to around the end of March the plan was to work two days a week, this was because of another subject (INGT2300). Towards the end of the project, the goal was to work four days a week. See figure 31 for weekly hour distribution.

Category	**Estimated hours**	**Actual hours**
Development	330	426
Lecture	56	56
Main report	600	502
Planning	70	126
Poster	48	24
Prototyping	96	78
Research	96	90
User tests	144	49
Total	1436	1351

Table 4: Estimated and actual hour distribution

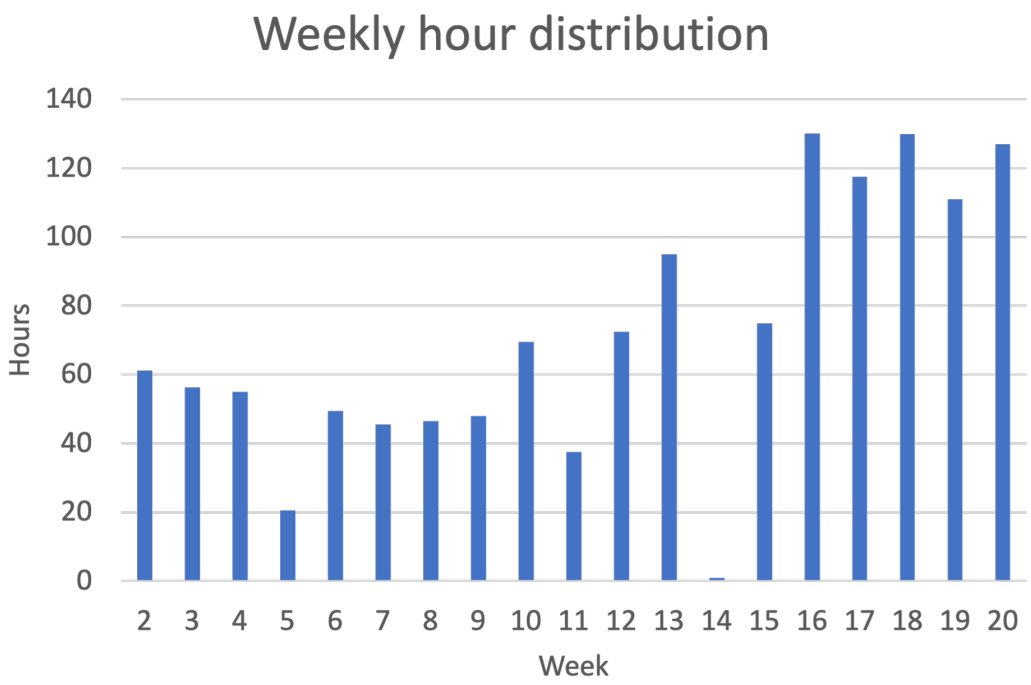


Figure 31: Weekly hour distribution

4.4.3 Development method

During the development process, there were meetings with different departments of AtB to get feedback on the solution. AtB was not present during the Scrum rituals, but there was an ongoing dialogue. Because the team used an agile development method it was easy to implement any changes they wanted. This was especially convenient in the early stages of the project when the direction was still unclear. All Scrum documents can be found in the project handbook. This includes sprint planning, reviews, issue board etc. (Appendix B & G).

The goals for each period or sprint were almost always finished within the planned time frame. An exception was the development of the final product, as it took more time than expected. All sprint goals and whether they were finished were documented in the sprint reviews (Appendix B).

QA testing results

The QA testing process yielded instrumental results, uncovering several minor bugs and areas of improvement. An issue was identified where the assistant was miscalculating savings, which was subsequently fixed to ensure accurate computation. Another challenge emerged related to the tips and information page's incompatibility with enlarged text, potentially affecting accessibility. This was resolved by integrating a scrolling feature, allowing the page to accommodate information overflow irrespective of screen size. A final observation was the lack of a defined order for the tips. Although not a bug, a suggestion was made that structuring the tips would make it more logical and predictable for the team when adding information.

5 Discussion

The purpose of the discussion chapter is to interpret the results, draw connections between the theory and the results provided, and provide context for the overall conclusions.

5.1 User survey

5.1.1 Results from the user survey

With the purpose of mapping user needs, the user survey yielded many valuable insights. Almost half of the participants used the old application, highlighting the need for compelling reasons for users to switch to the new app. This is reflected in the answers from the user survey (Appendix H). In figure 6 many of the answers revolve around features that only exist in the old app or that people didn't see the need to switch. Even though the ticket assistant isn't a feature that was missing from the old app, it could make some people switch. It is important to note that AtB wants users to use the new app rather than the old one.

Figure 5 shows what people want for the ticket purchasing experience to improve. The most common answer included reducing prices or adding discounts, but this is irrelevant to the scope of this project. However, the second most common answer suggested adding some kind of personalised recommendations of what ticket to buy. This supports the creation of a digital ticket assistant that recommends a ticket for the user even though most people were confident that they always bought the cheapest ticket for their use.

In the results chapter, it was mentioned that the median response to the question of whether the user believes it is simple to know which ticket to purchase in every circumstance was five. This suggests that users sometimes need assistance choosing which ticket to buy. In most cases, a user may be unsure about which ticket to purchase for one of two reasons: either they are unsure of which ticket applies in each circumstance, or they are unsure of whether making a different choice would result in cost savings.

Effects of the user survey

The first issue was addressed by introducing the tips and information page (Figure 10). Rather than implementing all edge case scenarios into the ticket assistant, they were relocated to a separate page, which could be accessed from within the ticket assistant (Figure 9). This helped simplify the questionnaire and eliminated the need for specific questions that would be irrelevant to most users. Having it as a standalone page came as a result of the user survey since users wanted answers to edge case scenarios to be more accessible.

To address the second issue, the ticket assistant was developed, which is the most extensive feature of the solution. Its purpose is to find the cheapest ticket based on a user's travel pattern. In response to the question "What should be done differently to make ticket purchasing easier for you?", users expressed that they wanted the app to recommend a ticket without much effort on their part. The solutions suggested by the team were a ticket recommended based on historical data, and a questionnaire

that could recommend a ticket based on a few questions, with the latter being the prioritised solution. As mentioned in the result section, 87% preferred to have a ticket recommended to them, which supports the decision to create a ticket assistant.

Based on the results from the survey the project was scoped down to focus on two things: creating a page for frequently asked questions, and a ticket assistant questionnaire that recommends the cheapest ticket for a user based on their travel habits.

5.1.2 Strengths and weaknesses of the user survey

Construction of survey questions

When designing the user survey, the goal was to avoid guiding participants towards specific answers. Most of the questions succeeded in that regard, resulting in varied and accurate answers. Additionally, the survey provided an understanding of the various users of the AtB application and the factors that needed consideration to ensure the solution catered to all. Another advantage was that the survey was digital and easy to distribute, which led to a lot more answers than anticipated.

Some of the questions were poorly constructed, particularly one that asked about the frequency of bus travel. Whether the question referred to a single trip or a round trip was not specified. Throughout this project, it was assumed that participants considered it a round trip, and specified when there was a lack of clarity.

Another weakness in the questions was the fact that users had to answer with a number and the range of numbers was always odd. This gives the user the possibility to answer a middle value. When having a scale with a true midpoint rating, the midpoint tends to get many answers, because users use it to indicate a lack of opinion. A solution is to remove the midpoint and provide a separate option for those that don't have a strong opinion (Courser and Lavrakas 2012). As a consequence, the answers may not be reliable. Some of these weaknesses could have been addressed by testing the user survey on a selection of people before distributing it to the public. However, it should be noted that the survey was conducted with the purpose of obtaining a general understanding of user needs. Despite some limitations and flaws, the survey was able to fulfil its purpose.

The survey user base

As previously mentioned, a lot of the respondents, almost 46%, were in the age group 18-25. This may have impacted the overall nature of the statistics as young people tend to be more comfortable with using apps and smartphones. However, the amount of respondents increases the credibility of the results.

Furthermore, due to AtB's general bad reputation (*Norges dårligste apper bruker samme teknologi - se listene - Kode24* 2023) and the current pricing strategy that the county municipality (Trøndelag Fylke) employs, a lot of the feedback received was generally unhelpful or not relevant to the project. However, this was expected as these answers were given in the form of text and could therefore be filtered out. They were also often related to the pricing and punctuality of buses and other services.

Despite the sources of error, the user survey provided valuable feedback from various users of public transport in Trondheim. It provided a useful indicator of what the users thought the present solution was missing and what they needed assistance with within

the application.

5.2 Prototype user test

5.2.1 Results from the prototype user tests

Overall, participants had a positive perception of the prototype and provided valuable input. Common observations included unclear headings on some pages, with some participants finding the wording too serious or formal. Additionally, users expressed a desire to see the savings achieved by choosing a particular ticket type on the summary page (Figure 29). Participants generally found the app's flow to be natural and intuitive.

Usability

In terms of usability, most users found the solution to be intuitive and easy to use. Few had problems navigating through the ticket assistant questionnaire, but there were some areas where people got confused. The pages that received an overall good review were the category picker page, the frequency page, and the summary page (Figure 17). Most users felt they were easy to use and generally didn't need help understanding what they needed to do. One of the main focuses when creating the pages was making the questions easy to understand and answer, which is probably why many users found most of the pages intuitive. Also, the Q&A page (Figure 20) received mostly good feedback and was seen as a very useful and needed addition to the AtB app.

As expected, users found the period picker page hard to understand and use, and the team found it challenging as well (Figure 13). Most users took some time to figure out what was asked of them because the heading was unclear. When they finally understood what to do, some complained that there wasn't an alternative that matched the duration they wanted. There was a date picker on the screen, but many felt that it disappeared on the page. This could be because the date picker was close to the header and separated from the quick alternatives. People also requested confirmation of their selection.

Positive feedback

Features appreciated by users were the more descriptive information on the category picker (Figure 12). The normal descriptions of travellers in the AtB app are short and only provide the absolute minimum necessary information. Furthermore, users enjoyed the addition of emojis, as this made the app feel more welcoming, playful and less strict. The slider component was a well-met feature, people found it easy to use and if they were unsure of an exact number, the slider made it easier to give an approximation. It also created variation within the questionnaire, ensuring that users stay focused on the content.

Issues

The biggest issue uncovered from the user tests was users experiencing difficulty locating the ticket assistant (Figure 8). Even when being on the ticket overview page, most users didn't find the assistant and rather pressed the period ticket tile. This could be because the assistant tile blended too much into the page. It was also located on

the bottom of the page and therefore required scrolling to find. The assistant shouldn't be too visible on the tickets overview page, because most of the time, users don't need to see it. It would probably be annoying for users if the assistant was forced upon them. Still, it should be easy to find if you are wondering about what ticket you should buy. Figuring out how to balance this was one of the biggest challenges in the design process.

Main findings and consequences

Based on user input, the questionnaire was redesigned to incorporate swipeable pages for simpler navigation. The tile on the ticket overview page also received a redesign to improve clarity and it was also decided to extend one category by default in the category picker. Additionally, suggested tickets should display prices, savings, and better descriptions (Figure 15). This was one of the most requested features when conducting the tests. Lastly, the FAQ page should be separate from the ticket assistant for a more streamlined experience.

5.2.2 Strengths and weaknesses of the prototype user tests

The prototype test was the first time the solution was tested on actual users, and not just the team members. The goal of the user tests was to collect feedback on the prototype and use that to build a foundation for the implemented solution. For that purpose, the user tests worked. This is because the tests were designed to avoid any leading questions that might influence the test subjects' responses. The tests should start with the user getting almost no help, and allow them to explore the solution as they find it naturally. Then after completing the questionnaire, they would be asked what their immediate thoughts were, and finally some direct questions about the solution. By letting the user explore freely, the tests provided genuine feedback from users exploring the solution for the first time.

User base

It's crucial to evaluate a diverse range of the application's user base when conducting user tests, enabling the application to accommodate a variety of needs. Thus, a diverse group of test subjects was used for the prototype test (Figure 7). The thoughts or issues of the senior were given more weight when deciding what should be changed in the prototype compared to how a young adult felt. This is because a young adult with more knowledge using mobile applications is typically better at adapting to new solutions compared to a senior with less experience. The aim was to get as broad of a user base as possible with 3 subjects from each group. However, it was challenging finding enough seniors to serve as test subjects. Still, the selection should be broad enough to provide insight and help improve the prototype.

The choice to expedite the recruitment process by engaging individuals easily accessible may have introduced a selection bias. Several of the subjects were employees from AtB, and therefore have greater knowledge of the ticketing system than the general public and a more accepting view of the app and its features. However, it could also be advantageous to carry out tests on these individuals since their understanding of the ticketing system might provide more insightful feedback.

A/B testing

Using A/B testing at the end of the user test made users realise that some elements could be different from what they were initially presented with. It gave the team a direct response to what works, and what doesn't. Tracking how many users wanted each alternative made it easy to choose what alternative to continue working on (Table 1). It also made the test more varied, combining free exploration and instinctive thoughts with more direct questions. It is important to note that the free exploration came before the A/B testing, hence the user was not affected by anyone other than their own opinions.

The prototype user tests provided invaluable feedback on the solution, despite having some sources of error. What seemed intuitive to the team might not be intuitive for the majority of people. Conducting a user test like this prior to implementing the solution made the process of development easier, compared to what it would have been if the user tests were only carried out post-development.

Accessibility Testing

One weakness of the prototype user tests was the lack of testing with individuals who may require the use of accessibility features. By not including users who rely on screen readers or other assistive technologies, the assessment of how well the accessibility solutions actually worked could not be accurately determined. This limitation in the testing process may have prevented the identification of potential issues and improvements in the solution's universal design. In the future, it is essential to involve users with diverse needs and abilities in the testing process to ensure that the application caters to everyone and provides an inclusive experience for all users.

5.3 Final product

The final user interface was showcased in the results section, and this section will discuss them. Both the design choices that were made and the limitations of the solution will be explored.

5.3.1 Integrating the solution into the AtB app

When it comes to the ticket overview page (Figure 22), the goal was to make the ticket assistant always available, and at the same time not intrusive. Finding that balance proved to be difficult because hiding the feature too much made users not find it even when being asked to look for it. The solution that got the best feedback was making the tile wider, and therefore not looking too similar to a ticket. Also, adding a different icon to the tile and giving it some slightly different colours to the tickets made it look a bit different. Still, after the last user tests some people didn't find the assistant, but it was much better than the previous solutions. The beta tag informs the user that it is an experimental feature.

5.3.2 Ticket assistant questionnaire

Questions

The questions in the questionnaire remained mostly the same throughout the project

(Figure 21). The wording saw changes, but the core remained the same. However, the flow saw some changes, like having the frequency and duration questions after one another as they are closely related. Also having the category question at the beginning rather than after frequency felt more natural to the users.

Response formats

One of the most crucial elements of the ticket assistant flow is how the user can answer each question. Because if the user has to use too much time answering each question the flow will be worse. On the other hand, the user should be able to answer each question extensively and not feel like they are restricted from providing enough information. This issue was present when designing the duration selector page (Figure 27). Some users wanted to be able to select a duration quickly, but others wanted to be able to select a specific date. There were also complaints when the user couldn't find the quick option for the duration they wanted. A solution was creating a slider replacing the quick options, this makes it easier for the user to select an approximation. Also, a date picker for selecting an exact date made it possible for the user to do both. On the other pages, users found it simple to answer the questions and didn't have many complaints about the design.

Adjustments for screen reader

One could argue that if the UI needs to be changed when a screen reader is active, the design itself might require reconsideration. However, after conducting tests with prototypes, it was observed that sliders performed well and serve as a quick and efficient method for users to input their preferences (Figure 30). At the same time, sliders provide numerous options to accommodate diverse user needs.

It is important to recognise that screen readers provide a different way of interacting with visual interfaces. Consequently, it seems logical for the UI to adapt and optimise its design for screen reader usage, ensuring that the user experience remains intuitive and accessible across different input methods. In this context, the UI adaptation for screen readers does not necessarily indicate a fundamental design flaw but rather highlights the importance of accommodating diverse user needs and interaction styles.

Main findings and consequences

In conclusion, the development of the ticket assistant system focused on achieving a balance between user accessibility and providing a non-intrusive experience. Design choices were guided by user feedback, with iterative improvements made to achieve an intuitive user interface. While some challenges persisted, such as finding the optimal balance between quick selections and more detailed options, the team developed solutions that considered different user preferences.

5.3.3 Ticket recommendation

There were many challenges when developing the methods that calculate the cheapest ticket. First of all, it had to be decided if the algorithm should calculate one recommended ticket or a combination of tickets to cover the duration the user set. The team decided to go for the latter, mainly because this makes room for interesting future additions to the ticket assistant. As of today, the algorithm calculates the cheapest combination of tickets, but the front-end of the application only gives the user one

of the tickets in the combination (the one with the longest duration). In the future, a possible expansion is that the app can remember the recommended ticket combination and recommend the next ticket in the combination when the current one expires.

After deciding that a combination of tickets is what the algorithm should calculate, the problem was figuring out how. The solution was, as mentioned previously, a combination of an unbounded knapsack problem and a greedy algorithm. One of the biggest issues with this solution was that the single ticket doesn't fit into either of the algorithms. Given that the number of days is the weight of the items in the knapsack, a single ticket that only has a weight of 1.5 to 3 hours doesn't make sense (equal to 0 days). Initially, the solution was to calculate a frequency of travel threshold for when single tickets are cheaper than period tickets. The threshold was calculated based on how many single tickets you could buy in a month which would still be cheaper than the monthly ticket. Later on, this was dropped, and now the algorithm always estimates how much just purchasing single tickets would cost and compares it to the outcomes of the greedy algorithm and the unbounded knapsack. It was difficult to calculate an accurate threshold, and always calculating the cost of single tickets is simpler. This is why the threshold was scrapped.

The algorithm is also designed to function as expected even when new ticket types are introduced, prices change, or some ticket types are missing. This flexibility is particularly important for travel between different zones, as not all zones have the same ticket types. When travelling across multiple zones, some ticket types may be unavailable. As a result, the algorithm has been built to adapt to various ticket types and should continue to work effectively if any changes are made.

5.3.4 Limitations of the final product

Extensive catergory descriptions

Even though users found the more descriptive texts on the categories a nice addition, they were not incorporated as it would increase the maintenance of the feature. Therefore the current solution uses the default descriptions (Figure 25).

Ferry tickets

Although adaptable and scalable, the algorithm has its limitations. Ferry tickets are currently not taken into account by the algorithm. This implies that a user who has to travel between areas where taking the ferry is cheaper than taking the bus would still only be advised to buy bus tickets. When a ferry is available, the journey distance and time are often reduced, and the cost is also reduced. Ferry tickets are excluded because they differ slightly from bus tickets and because they cannot be purchased through the new AtB app. Therefore, recommending a ticket that a consumer is unable to purchase wouldn't make sense. Adding this support is considered a potential extension of the solution but was not given priority in the project's scope.

Night tickets

Another limitation is that the solution does not take night tickets into consideration. Because regular single tickets and adult period tickets aren't valid on the night buses, which usually run Friday and Saturday after midnight, some consumers asked for

support for night tickets. Night tickets were not highly prioritised, as the primary objective of this tool was to facilitate ticket acquisition for users' future travel patterns. Given that the majority of individuals do not travel during nighttime hours, the focus was placed on addressing more prevalent travel patterns to better serve the needs of the user base. A potential solution to address the issue of night ticket availability could involve incorporating an additional question within the ticket questionnaire, specifically inquiring whether the user intends to travel during nighttime hours. Should this be the case, the system would then include night ticket options in the recommendations. This enhancement could improve the overall user experience and cater to a wider range of travel patterns.

Simpler user experience

Having a simpler questionnaire with fewer tickets and questions can greatly improve the user experience. By reducing the number of options, users can navigate the system more easily and make decisions quickly. Simplicity also promotes clarity and understanding, reducing the chances of misunderstandings. A streamlined questionnaire enhances usability and could increase user engagement. However, it's important to maintain a balance between simplicity and inclusivity.

5.3.5 User tests of the final product

Mostly, the user test responses were positive. This is likely due to thorough research of user needs that was done before producing prototypes which focused on what users wanted. The feedback from users was always considered greatly when developing, therefore, it makes sense that people were satisfied with the solution. The biggest issue unveiled in the final tests was that the slider hit-box was small on Android devices. This was immediately addressed and now the hit-boxes should be large enough on all devices.

Test subjects in this round were primarily people that hadn't seen the feature before. This was intended because then they would begin the test with a fresh mind. The user tests may have been flawed because there were no tests on seniors. However, older individuals had been tested in earlier stages of development and their needs have been considered. Additionally, user testing of people with disabilities was not conducted, and such testing is necessary to ensure that the accessibility measures incorporated serve their intended purpose. It's important to note that this must be done in the future to ensure that the feature is operable by everyone.

5.4 Functional requirements

Most of the functional requirements that were set in the vision document were achieved (Table 2). The ones that were not achieved were not prioritised because of time, this includes requirements 3.8.1, 3.8.3 and 5 (Table 2). For the feature to make it to production everything had to be finished in April or early May. Together with AtB, the most important features were defined and for example, storing the recommended ticket in the ticket overview page was not prioritised. The same goes for recommending different types of tickets and comparing the cost of these. These functional requirements were not seen as essential for the features to work as expected.

The requirement of always having the solution available in the ticket overview has been achieved but not quite as expected. Late in the project, it was decided to label the feature as beta, because it is quite an extensive feature and if the user does not agree with the recommended ticket AtB wouldn't get as much criticism for it. Also, the redirect from the tips and information page (Figure 23) to the questionnaire (Figure 24) was removed late in the project due to extra maintenance for AtB. All the other texts on the tips and information page could be dynamically altered from Firestore, while the one with the button could not.

All the other achieved features have been completed as expected, and no major deviations happened. It is important to note that AtB has continuously been involved in directing the solution and they are very satisfied with the solution that the team has come up with.

5.5 Non-functional requirements

The non-functional requirements include universal design, guidelines from AtB and scalability.

5.5.1 Universal design

AtB AS, as the primary app for purchasing and locating travel options in Trøndelag, serves around 150,000 users, making WCAG compliance crucial. Ensuring accessibility for all users, including those with disabilities, provides equal access to information and services, increases user autonomy, and enhances the overall user experience. Additionally, it helps AtB comply with legal requirements and demonstrates a commitment to social responsibility. By adhering to WCAG guidelines, AtB enables independent travel and supports the diverse needs of its users, fostering a more inclusive and equitable society.

This also makes it important for the solution to comply with WCAG guidelines, as it will be integrated into the AtB app. By ensuring that the assistant is accessible and user-friendly, the overall user experience can be improved. Additionally, prioritising accessibility in the solution demonstrates a commitment to inclusivity and social responsibility, aligning with ATB's values and goals.

Consequences of not following WCAG requirements

By not adhering to WCAG requirements, the AtB app risks alienating a segment of its user base, leading to dissatisfaction, negative user experiences, and loss of users. Addressing these unfulfilled requirements is crucial to ensure equal access and an optimal user experience for all users, regardless of their abilities. By not following 1.3.4 - Orientation, people with mounted phones might not be able to operate the app. By not following 2.5.3 - Label in name, users that rely on speech input or other assistive technologies using the accessibility labels are not able to use the solution at all.

In conclusion, the solution's implementation of universal design principles effectively promotes inclusion and accessibility for a diverse range of users. While it does not fully comply with all WCAG requirements, the vast majority of these standards have

been met, and the remaining areas of non-compliance are partly due to limitations in the underlying AtB app. Overall, the positive aspects of the design far outweigh the limitations, contributing to a more accessible and inclusive user experience. However, addressing the unmet requirements will further enhance the overall accessibility and usability of the app for all users.

5.5.2 AtB guidelines

All team members made sure that the implementation closely adhered to AtB's guidelines because getting the feature into production was one of the team's main objectives. The first feature the team developed and got merged into the master branch was the implementation of a feature switch that allowed for remotely disabling the feature on users' devices. This was one of the most crucial requirements. The ticket assistant also needed to require little maintenance, which was another crucial requirement. This makes it easier for AtB to adopt the solution. Every decision made during development was centred on the goal of ensuring that the solution was as modular as possible.

5.5.3 Scalability

An important aspect of the solution was the ability for it to be scalable as well as adaptable to new tickets. Throughout the implementation period, a great deal of effort was put towards ensuring the solution fetches all the data it needs from external sources that are continuously kept up to date. Doing this ensured that the team did not create separate data sources that the AtB team would need to update.

Furthermore, creating the solution this way, allows for it to be adaptable if AtB decides to implement other types of tickets or change the period of specific tickets. The solution's use of external single sources of truth that the rest of the app also uses, allows for easy maintenance of the entire app and keeps the ticket assistant low maintenance.

5.6 Administrative results

5.6.1 Progress plan

The GANTT diagram that was created at the beginning of the project was not used very much. It functioned as a base for the sprints that were developed and followed. The sprints were created in GitHub and were followed throughout the whole project with only a small change because the Easter holiday was not accounted for. Also, getting the code reviewed and merged into the master branch took more time than expected, which delayed writing the main report. Compared to the GANTT diagram, implementing the solution started later than planned, this is because prototyping was more extensive than anticipated and the other subject INGT2300 also required more time than initially thought.

5.6.2 Hour distribution

The use of Clockify and creating labels for different types of work made it easy to track the hour distribution of each area of work. When looking at the GANTT diagram (Appendix A) and comparing it to the final hour distribution (Table 4) there are some deviations. First of all, some of the categories were worked more spread out throughout the project. For example, the planning tag was used throughout the project for sprint planning, and not only in the beginning as shown in the GANTT diagram. Development was estimated to take around 330 hours. Code reviews contributed to the fact that it took more time than expected. Also, keeping the code up to the standards at AtB and following their guidelines was time-consuming.

The team, despite lacking prior experience with projects of this scale, had experience with similar types of projects, just on a smaller scale. Consequently, determining the appropriate allocation of time for each component of the project, which spanned the entirety of the semester, proved difficult. However, the team benefited from creating time estimates, even if not always precise, as these helped establish a structured approach to managing various tasks. This process was further enhanced by utilising predetermined sprints within GitHub, which proved crucial in structuring the work.

5.6.3 Development method

Discussion of agile development method results

As previously mentioned the chosen form of development was Scrum with two-week sprints. Because the team kept in constant communication and everyone was for the most part aware and up to date with what everyone else was doing at any particular stage, the need for daily Scrum was not present. Because of this daily Scrum was done on a per-need basis, often in periods where the nature of the work changed frequently, like while writing the report. In the early stages of development, the design was still being finalised and the use of Scrum helped as an iterative approach allowed for iteratively improving the design.

In terms of the Scrum process, the team decided to not have a Scrum master. This was because the team is small and therefore the responsibility for the Scrum process was shared between all team members. In hindsight, the Scrum process could have been more structured and more strictly followed if there was a designated Scrum master. However, due to the size and restrictions of the project, a less strict Scrum adaption was more beneficial. The team feels that the development methodology employed was a success even though there is room for improvement.

A product owner was also not selected. Though a product owner might have streamlined backlog management, the absence of the role in such a small team wasn't necessarily a misstep. Shared ownership is both feasible and beneficial in a three-person team. Everyone contributed to decisions, reducing the need for a dedicated product owner. This compact structure fostered versatility and a more participative approach, which led to a more well-rounded and collaborative team.

Working with AtB

The initial plan was to have code reviews with reviewers from AtB on a regular basis, but these were pushed to the end of the project period. The thought was that the

team didn't want AtB to review code that wasn't necessarily finished or that wouldn't be included in the final product as this would be a waste of time. However, having continuous reviews from AtB would have eased the workload at the end of the project. It was not anticipated that making the changes suggested by AtB would take as much time as it did. Therefore, it may have been wiser to have more frequent pull requests and code reviews from AtB to spread the workload more evenly throughout the project.

To ensure that code quality was upheld throughout the project, the team made sure that everything implemented mirrored existing patterns and paradigms in the existing codebase. Direct dialogues with developers at AtB and comprehensive meetings with designers helped satisfy these goals. Furthermore, developers from AtB were also helpful when the team had questions regarding a specific pattern or implementation.

QA testing

The results from the quality assurance (QA) testing offered significant insights into the development of the software. It shows that QA testing is a critical aspect of the development lifecycle, not only for identifying bugs but also for enhancing user experience and accessibility. For instance, the adjustments made to the savings calculation and the tips and information page's text enlargement compatibility illustrate the necessity of QA testing. The addition of a scrolling feature demonstrates how QA testing can lead to innovative solutions that improve usability. The suggestion for a structured order for the tips, although not a defect, highlights how QA testing can provide valuable feedback for software improvements. This underlines the importance of a robust QA process in not only ensuring functional correctness but also in identifying opportunities for better design and user experience.

5.6.4 Team Collaboration and Communication

One of the most significant strengths of the team was their ability to collaborate effectively, stemming from their prior experience working together. Each team member brought unique skills and expertise to the table, allowing for an efficient distribution of work that capitalised on individual strengths. Open communication and an environment that encouraged input and suggestions from all members further bolstered the collaborative atmosphere.

The team aimed to maintain a consistent workload throughout the duration of the project, which was successfully achieved. Writing sections of the main report as various project components were completed enabled the team to capture their insights and observations while they were still fresh in memory. This approach greatly facilitated the writing phase at later stages, as much of the content had already been drafted.

Throughout the project, the team adhered to the collaboration agreement, and any disputes or disagreements were resolved through open dialogue and mutual decision-making. This strong collaborative foundation contributed significantly to the success of the project.

6 Conclusion and further work

6.1 Conclusion

The goal of the project was to create a digital assistant in the new AtB app, with the purpose of helping users choose the correct ticket for their use. Whether the user has an edge case scenario like travelling with a dog or bicycle, or if they want to find out what ticket is cheapest for them based on their travelling pattern.

A digital survey was used to map user needs and to define the scope of the assistant. Even though some of the questions were poorly constructed, they gave a good indication of the user's needs. This led to a two parted solution; an assistant for choosing the cheapest ticket based on a user's habits, and a tips and information page where answers to edge case scenarios could be found.

After the user survey, a more precise research question was formalised:

> *How can we help users quickly and accurately select the right ticket based on their needs, while keeping consistency with the app's current design language?*

Given the research question, one major weakness is that the ticket assistant does not support ferry or nightly travel. AtB covers many areas where ferries are a common transportation method. Furthermore, a large part of the users are students or young adults who often travel at night. Not being able to cover these travellers could be seen as a weakness. However, most use cases are covered and the solution has received positive feedback from both people within AtB and tested users.

To answer the last part of the research question and find a design that fits user and AtB expectations, user tests on prototypes were conducted iteratively throughout the project. Utilising user tests as a prominent tool and focusing on creating a design that users appreciate has ensured that users find the feature intuitive and easy to use. Continuous testing of the solution has validated the assumption that users consider the feature useful. It is reasonable to conclude that the digital assistant feels like a natural extension of the already familiar AtB application.

Whether the feature will help users pick the right ticket, and ultimately answer the research question, is difficult to predict. In order to determine whether the feature is being utilised and fulfils its purpose, AtB needs to examine reviews and feedback, as well as assess if customer support receives fewer inquiries about the appropriate ticket to purchase. Also, assuming that the feature is going to be released into the app, it would be easier to assess its success if AtB had analysing tools within the application to determine whether the feature is actually being used. However, AtB currently doesn't have this kind of functionality, and given the project's scope, it's not something the team could affect. Based on the thorough user tests conducted and the response from AtB it will most likely be appreciated by AtB's users.

At the time of writing, the feature is undergoing the final phase of QA testing in the staging environment. It is anticipated that, upon successful completion of this process, it will be included in the next public release.

6.2 Recommendations for future work or improvements

AtB is satisfied with the current solution, but there is room for further improvements and also extensions to the overall ticket assistant. Many of these improvements or additions are already in the backlog and have been discussed by the team.

One of the most important additions would be the inclusion of analytic tools to track user behaviour in the app. This would help determine what features work and don't in the application, which would be especially helpful for the ticket assistant features. It would also help track user patterns within the application and identify potential problems with the layout or flow of the app.

6.2.1 Historical data

When starting the project, one of the goals for the assistant was to recommend a ticket based on historical data. As said previously, this was not prioritised as it would be difficult to know what type of period ticket the user would need. A solution could be to notify the user when they should start considering a period ticket and then recommend they use the ticket assistant feature to determine what kind of period ticket to choose. Using historical data could also determine whether the user should buy night tickets.

6.2.2 Support for more ticket types

The most obvious improvement for the ticket assistant is adding support for more ticket types, like night tickets, ferry tickets and youth tickets. Many of the zones that AtB cover in Trøndelag have ferries, and it doesn't make sense for people to travel by bus if there is a possibility to take the ferry. However, as of writing, the ferry ticket and youth ticket are not available for purchase in the new AtB app. Hence, it has not been prioritised in this project. The same applies to night tickets, as they were not available for purchase in the new app at the beginning of the project. Including the night tickets in the ticket assistant could be done by asking if the user is going to use night tickets.

6.2.3 Ticket combination

In the current solution, the cheapest combination of tickets is calculated. This is not utilised on the front-end, but it could be implemented by displaying the next ticket in the combination on the ticket overview page. As of now, the solution recommends the user go through the questionnaire again when their recommended ticket expires. Storing the combination could save the user from having to answer the questions again.

7 Social impact

This chapter will examine the social and sustainability impacts of the bachelor thesis. Understanding these impacts is critical to understanding the broader implications of the project. Both positive and negative impacts of the results will be explored and discussed.

People in general have less spending power than they used to (Økland 2023), whether as a result of large electricity bills, rising food prices, or inflation. At the same time, the cost of public transport shows no signs of being reduced in the foreseeable future. The ticket recommendation feature helps users find the most cost-effective ticket for their specific needs, ultimately saving them money. Lowering the cost of travel allows people to spend more on other areas leading to a healthier economy for individuals.

There is a chance that people may misunderstand the questions because the developers have a different view than the actual users. Inaccurate user input may lead to a user spending more than they would have if they didn't use the recommendation feature. This highlights the importance of universally understandable texts within the feature.

Before, figuring out what type of ticket to buy took a lot of time by having to research the different ticket types, calculating the prices and comparing them. Also, it took unnecessarily long to figure out what ticket to buy in edge-case scenarios, like when travelling with a dog. Earlier, this information was not available directly in the app but was instead hidden away on the website. Users can save time and effort by simplifying the ticket selection process, leading to a more efficient and enjoyable travel experience.

7.1 Profession ethical questions

Engineers have a responsibility and are required not to exclude or discriminate when creating solutions or products. When creating a feature for a public transport system which has around 150 000 users, this is especially important. The feature should work for everyone that utilises public transport, which is every part of society. This means that nothing should limit a person from using the feature created in this project, and this is regulated through *Lov om likestilling og forbud mot diskriminering, Kapittel 3, § 17 & § 18* 2017. Hence, there has been an emphasis on accessibility options to make the feature as accessible as possible. The use of screen readers and support for English as a language are measures taken to ensure this.

By providing a ticket recommendation feature, the AtB app becomes more accessible and user-friendly. Especially for those who have difficulties navigating complex fare structures or understanding ticket options. It contributes to a more inclusive society by helping those with cognitive difficulties. *Lov om likestilling og forbud mot diskriminering, Kapittel 3, § 17 & § 18* 2017 applies here as well and it is also one of FN's sustainability goals (Goal 10.2 *Mindre ulikhet* 2023). Having transparency and clarity about the different ticket rules is important for people to trust AtB as a company. Also, people will probably have a better understanding of the different tickets, or know where to find the necessary information. This will ease the workload of AtB's customer support and make it easier for AtB to guide users who need more critical

help.

In order to use the ticket recommendation tool, users may be required to submit personal information, such as travel interests and habits, which could cause privacy concerns. However, this data is not stored anywhere and is only used for one-time calculations. This is not specified to the users directly in the app and is a concern that should be addressed.

The solution must not discriminate or have a bias based on location or travel habits. Right now there is no support, in the recommendation feature or the AtB app itself, for ferry tickets. This can be seen as a bias towards people living in the city because it doesn't facilitate people who only have access to ferries as public transportation.

7.2 Sustainability

By improving the overall user experience of the AtB app and making it easier for users to find the right ticket, this project may encourage more people to use public transportation, thereby reducing the reliance on private vehicles. A study from 2020 showed that in the UK a conventionally fuelled car emitted almost double the amount of what conventionally fuelled buses did per person (Logan et al. 2020). Also, AtB uses both hybrid and fully electrical buses. Conventionally fuelled buses have 9 and 36 times higher emission numbers than hybrid and electric buses respectively. Therefore, making more people switch from driving a car to taking public transport will greatly help with achieving the environmental goals for the upcoming future. FN's sustainability goal 11.6, aims to reduce cities and local communities' effects on air pollution. Goal 11.2 also aims to have easier access to sustainable modes of transport, and by making it easier to buy tickets, the solution in this thesis could contribute to this goal (*Bærekraftige byer og lokalsamfunn* 2023).

While the enhancement of the AtB app is expected to encourage a shift from private vehicles to public transit, there are potential environmental implications that need consideration. One unintended outcome could be the discouragement of active transport modes like walking or cycling. Additionally, the increased reliance on digital services, such as the ticket recommendation feature, escalates the demand for data centres and associated infrastructure, contributing to a higher environmental footprint (Ferreira et al. 2018). Furthermore, the convenience and ease of use brought by the app might unintentionally motivate more frequent travel, leading to elevated energy usage and emissions.

Sustainability does not only include environmental sustainability. Economical sustainability involves economic growth while at the same time reducing climate gas emissions. The digital assistant promotes the use of public transport and aims to prevent people from spending more than necessary. Therefore, it encourages economic growth by enabling people to have more money to spend elsewhere while also reducing greenhouse gas emissions by promoting public transport over private vehicles.

Bibliography

(WAI), Web Accessibility Initiative (Dec. 2022a). *Label in Name (Level A)*. URL: https://www.w3.org/WAI/WCAG21/Understanding/label-in-name.html (visited on 8th May 2023).

— (Dec. 2022b). *Orientation (Level AA)*. URL: https://www.w3.org/WAI/WCAG21/Understanding/orientation.html (visited on 8th May 2023).

About - Git (2023). URL: https://git-scm.com/about (visited on 10th May 2023).

An introduction to screen readers | AbilityNet (2021). URL: https://abilitynet.org.uk/factsheets/introduction-screen-readers.

Bærekraftige byer og lokalsamfunn (2023). URL: https:https://www.fn.no/om-fn/fns-baerekraftsmaal/baerekraftige-byer-og-lokalsamfunn (visited on 19th May 2023).

Blac, Paul E. (July 2021a). 'Knapsack problem'. In: *Dictionary of Algorithms and Data Structures [online]*. URL: https://www.nist.gov/dads/HTML/knapsackProblem.html (visited on 9th Mar. 2023).

— (July 2021b). 'Unbounded Knapsack Problem'. In: *Dictionary of Algorithms and Data Structures [online]*. URL: https://www.nist.gov/dads/HTML/unboundedKnapsack.html (visited on 13th Mar. 2023).

Chart, Lucid (2023). *Lucidchart*. URL: https://www.lucidchart.com/pages/product (visited on 22nd Feb. 2023).

Chumburidze, Manana, Irakli Basheleishvili and Anano Khetsuriani (Feb. 2019). 'Dynamic Programming and Greedy Algorithm Strategy for Solving Several Classes of Graph Optimization Problems'. In: *BRAIN. Broad Research in Artificial Intelligence and Neuroscience* 10.1, pp. 101–107. ISSN: 2067-3957. URL: https://edusoft.ro/brain/index.php/brain/article/view/886.

Clockify (2023). URL: https://clockify.me/ (visited on 23rd Feb. 2023).

Cloudflare (2023). *What is BaaS? | Backend-as-a-Service vs. serverless*. URL: https://www.cloudflare.com/learning/serverless/glossary/backend-as-a-service-baas/ (visited on 9th Mar. 2023).

Cormen, Thomas H. et al. (2009). *Introduction to Algorithms*. 3rd ed. MIT Press.

Courser, Matthew W. and Paul J. Lavrakas (Dec. 2012). 'Item-Nonresponse and the 10-Point Response Scale in Telephone Surveys'. In: *Survey Practice* 5.4, pp. 1–5. ISSN: 2168-0094. DOI: 10.29115/SP-2012-0021. URL: https://doi.org/10.29115/SP-2012-0021.

Denzin, Norman K. (1978). 'Triangulation: A Case for Methodological Evaluation and Combination'. In: *Sociological Methods*.

Discord | Your Place to Talk and Hang Out (2023). URL: https://discord.com/ (visited on 10th May 2023).

Docs, GitHub (2023). *Collaborating with pull requests*. URL: https://docs.github.com/en/pull-requests/collaborating-with-pull-requests (visited on 4th May 2023).

Electronic ticket (2023). URL: https://en.wikipedia.org/wiki/Electronic_ticket#cite_note-1 (visited on 8th Feb. 2023).

Features | GitHub (2023). URL: https://github.com/features (visited on 10th May 2023).

Features | Google Drive (2023). URL: https://www.google.com/drive/#features (visited on 10th May 2023).

Features | Slack (2023). URL: https://slack.com/intl/en-au/features (visited on 10th May 2023).

Feldman, Stuart (Feb. 2005). 'Quality Assurance: Much More than Testing'. In: *Queue* 3.1, pp. 26–29. ISSN: 15427749. DOI: 10.1145/1046931.1046943. URL: https://dl.acm.org/doi/10.1145/1046931.1046943.

Ferreira, Joao et al. (2018). 'Estimating the Environmental Impact of Data Centers'. In: pp. 1–4. DOI: 10.1109/NCA.2018.8548326.

Figma (2023). URL: https://www.figma.com/prototyping/ (visited on 22nd Feb. 2023).

Jick, Todd D. (Dec. 1979). 'Mixing Qualitative and Quantitative Methods: Triangulation in Action'. In: *Administrative Science Quarterly* 24.4, p. 602. ISSN: 00018392. DOI: 10.2307/2392366.

Klabnik, Steve and Carol Nichols (2023). 'The Rust Programming Language'. In: Rust 1.67.1. The Rust Foundation. Chap. What Is Ownership? URL: https://doc.rust-lang.org/book/ch04-01-what-is-ownership.html (visited on 26th Apr. 2023).

Knapsack problem (2023). URL: https://en.wikipedia.org/wiki/Knapsack_problem (visited on 7th Mar. 2023).

Logan, Kathryn G., John D. Nelson and Astley Hastings (Aug. 2020). 'Electric and hydrogen buses: Shifting from conventionally fuelled cars in the UK'. In: *Transportation Research Part D: Transport and Environment* 85, p. 102350. ISSN: 1361-9209. DOI: 10.1016/J.TRD.2020.102350.

Lov om likestilling og forbud mot diskriminering, Kapittel 3, § 17 & § 18 (June 2017). URL: https://lovdata.no/dokument/NL/lov/2017-06-16-51/KAPITTEL_3#%5C%C2%5C%A717.

Lowhorn, Greg (2007). *Qualitative and Quantitative Research: How to Choose the Best Design*. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2235986.

Mindre ulikhet (2023). URL: https://www.fn.no/om-fn/fns-baerekraftsmaal/mindre-ulikhet (visited on 19th May 2023).

Moran, Kate (Dec. 2019). *Usability Testing 101*. URL: https://www.nngroup.com/articles/usability-testing-101/ (visited on 14th Feb. 2023).

Nettskjema (2023). URL: https://i.ntnu.no/wiki/-/wiki/Norsk/nettskjema (visited on 2nd Feb. 2023).

Nielsen, Jakob (Mar. 2000). *Why You Only Need to Test with 5 Users*. URL: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/ (visited on 14th Feb. 2023).

— (Aug. 2005). *Putting A/B Testing in Its Place*. URL: https://www.nngroup.com/articles/putting-ab-testing-in-its-place/ (visited on 14th Feb. 2023).

Norges dårligste apper bruker samme teknologi - se listene - Kode24 (Mar. 2023). URL: https://www.kode24.no/artikkel/norges-darligste-apper-bruker-samme-teknologi-se-listene/78845369 (visited on 20th May 2023).

Økland, Trym Kristian (Feb. 2023). *KPI opp 7,0 prosent siste tolv måneder*. URL: https://www.ssb.no/priser-og-prisindekser/konsumpriser/statistikk/konsumprisindeksen/artikler/kpi-opp-7-0-prosent-siste-tolv-maneder (visited on 20th May 2023).

Overleaf documentation (2023). URL: https://www.overleaf.com/learn (visited on 22nd Feb. 2023).

Pauly, Spencer (Aug. 2020). *Why I Switched Away From Google Firestore – and don't regret it*. URL: https://spencerpauly.com/tech/why-i-switched-away-from-google-firestore/ (visited on 9th Mar. 2023).

Radigan, Dan (2023). *What is Kanban?* URL: https://www.atlassian.com/agile/kanban#:~:text=What%5C%20is%5C%20kanban%5C%3F,of%5C%20work%5C%20at%5C%20any%5C%20time. (visited on 19th May 2023).

Rehkopf, Max (2023). *Kanban vs. scrum: which agile are you?* URL: https://www.atlassian.com/agile/kanban/kanban-vs-scrum (visited on 8th Feb. 2023).

Schmitz, Andy (2012). *Inductive or Deductive? Two Different Approaches*. URL: https://saylordotorg.github.io/text_principles-of-sociological-inquiry-qualitative-and-quantitative-methods/s05-03-inductive-or-deductive-two-dif.html.

Schwaber, Ken and Jeff Sutherland (2020). *Scrum Guide*. URL: https://scrumguides.org/scrum-guide.html#scrum-team (visited on 19th May 2023).

The Importance of Prototypes in Product Design (Aug. 2019). URL: https://wp.nyu.edu/dispatch/2019/08/30/the-importance-of-prototypes-in-product-design/#:~:text=Prototypes%5C%20allow%5C%20for%5C%20user%5C%20testing,can%5C%20make%5C%20changes%5C%20early%5C%20on.&text=Creating%5C%20any%5C%20product%5C%20requires%5C%20a,and%5C%20concepts%5C%20with%5C%20one%5C%20another. (visited on 2nd Feb. 2023).

UUtilsynet (2023). URL: https://www.uutilsynet.no/ (visited on 23rd Feb. 2023).

Web Content Accessibility Guidelines (WCAG) 2.1 (June 2018). Tech. rep. W3C (World Wide Web Consortium).

Welcome | Actix (2023). URL: https://actix.rs/docs (visited on 26th Apr. 2023).

What is cross-platform mobile development? (2023). URL: https://kotlinlang.org/docs/cross-platform-mobile-development.html (visited on 26th Jan. 2023).

What is Scrum? (2023). URL: https://www.scrum.org/learning-series/what-is-scrum (visited on 19th May 2023).

Zolkifli, N. N., A. Ngah and A. Deraman (2023). *Version Control System: A Review*. URL: https://www.sciencedirect.com/science/article/pii/S1877050918314819?via%5C%3Dihub (visited on 26th Jan. 2023).

Appendix

A Pre-project plan

B Project handbook

C Vision document

D Requirements documentation

E System documentation

F WCAG checklist

Sist oppdatert: 21.09.2022	Oversikt over status for universell utforming i én enkelt nettløsning. Her ser du WCAG-kravene som nettsteder skal oppfylle, sortert på de fire prinsippene i WCAG-standarden. For hver av retningslinjene finnes beskrivelser, lenker til mer informasjon, m.m., samt en nedtrekksmeny der du kan notere status pr i dag. Du kan laste ned arket og bruke det som arbeidsliste, men følg alltid med på uutilsynet.no for oppdatert kravliste.

uutilsynet

	Retningslinje	Beskrivelse av retningslinje	Suksesskriterium	Følges kravet?	Svaret som skal avgis
Prinsipp 1: Mulig å	1.1 Tekstalternativer	*Gi tekstalternativer til alt ikke-tekstlig innhold, slik at*			
			1.1.1 Ikke-tekstlig innhold	Vi har ikke denne typen innhold	
	1.2 Tidsbaserte medier (lyd og video)	*Gi alternativer til tidsbaserte medier.*			
			1.2.1 Bare lyd og video (forhåndsinnspilt)	Vi har ikke denne typen innhold	
			1.2.2 Teksting (forhåndsinnspilt)	Vi har ikke denne typen innhold	
	1.3 Mulig å tilpasse	*Lag innhold som kan presenteres på forskjellige måter uten at informasjon*			
			1.3.1 Informasjon og relasjoner	Ja	
			1.3.2 Meningsfylt rekkefølge	Ja	
			1.3.3 Sensoriske egenskaper	Ja	
			1.3.4 Visningsretning	Nei	
			1.3.5 Identifiser formål med inndata	Vi har ikke denne typen innhold	

oppfatte	1.4 Mulig å skille fra hverandre	*Gjør det enklere for brukerne å se og høre innhold, blant annet ved å*		
			1.4.1 Bruk av farge	Ja
			1.4.2. Styring av lyd	typen innhold
			1.4.3 Kontrast (minimum)	Ja
			1.4.4 Endring av tekststørrelse	Ja
			1.4.5 Bilder av tekst	Ja
			1.4.10 Dynamisk tilpasning (Reflow)	Vi har ikke denne typen innhold
			1.4.11 Kontrast for ikke-tekstlig innhold	Ja
			1.4.12 Tekstavstand	Ikke sjekket
			1.4.13 Pekerfølsomt innhold eller innhold ved tastaturfokus	Vi har ikke denne typen innhold
	2.1 Tilgjengelig med tastatur	*Gjør all funksjonaliteten tilgjengelig med tastatur.*		
			2.1.1 Tastatur	Ja
			2.1.2 Ingen tastaturfelle	Ja
			2.1.4 Hurtigtaster som består av ett tegn	Vi har ikke denne typen innhold
	2.2 Nok tid	*Gi brukerne nok tid til å lese og bruke innhold.*		
			2.2.1 Justerbar hastighet	Vi har ikke denne typen innhold
			2.2.2 Pause, stopp, skjul	Vi har ikke denne typen innhold
	2.3 Anfall og andre fysiske reaksjoner	*Ikke utform innhold på en måte som er kjent for å*		

Prinsipp 2: Mulig å betjene			2.3.1 Terskelverdi på maksimalt tre glimt	Ja
	2.4 Navigerbar	*Gjør det mulig for brukerne å navigere, finne innhold og vite hvor de befinner seg.*		
			2.4.1 Hoppe over blokker	Vi har ikke denne typen innhold
			2.4.2 Sidetitler	Ja
			2.4.3 Fokusrekkefølge	Ja
			2.4.4 Formål med lenke (i kontekst)	Vi har ikke denne typen innhold
			2.4.5 Flere måter	Ja
			2.4.6 Overskrifter og ledetekster	Ja
			2.4.7 Synlig fokus	Ja
	2.5 Inndatametoder	*Gjør det enklere for brukerne å betjene funksjonalitet med andre*		
			2.5.1 Pekerbevegelse	Ja
			2.5.2 Pekeravbrytelse	Ja
			2.5.3 Ledetekst i navn	Nei
			2.5.4 Bevegelsesaktivering	Vi har ikke denne typen innhold
	3.1 Leselig	*Det må være mulig å forstå informasjon og betjening av*		
			3.1.1 Språk på siden	Ja
			3.1.2 Språk på deler av innhold	Vi har ikke denne typen innhold
	3.2 Forutsigbar	*Sørg for at websider presenteres og fungerer på*		
			3.2.1 Fokus	Ja

Prinsipp 3: Forståelig			3.2.2 Inndata	Ja
			3.2.3 Konsekvent navigering	Ja
			3.2.4 Konsekvent identifikasjon	Ja
	3.3 Inndatahjelp	*Hjelp brukere med å unngå feil og å rette opp feil.*		
			3.3.1 Identifikasjon av feil	Ja
			3.3.2 Ledetekster eller instruksjoner	Ja
			3.3.3 Forslag ved feil	Ja
			3.3.4 Forhindring av feil (juridiske feil, økonomiske feil, datafeil)	Vi har ikke denne typen innhold
Prinsipp 4: Robust	4.1 Kompatibel	*Sørg for best mulig kompatibilitet med aktuelle og fremtidige brukeragenter, inkludert*		
			4.1.1 Parsing (oppdeling)	Ja
			4.1.2 Navn, rolle, verdi	Ja
			4.1.3 Statusbeskjeder	typen innhold

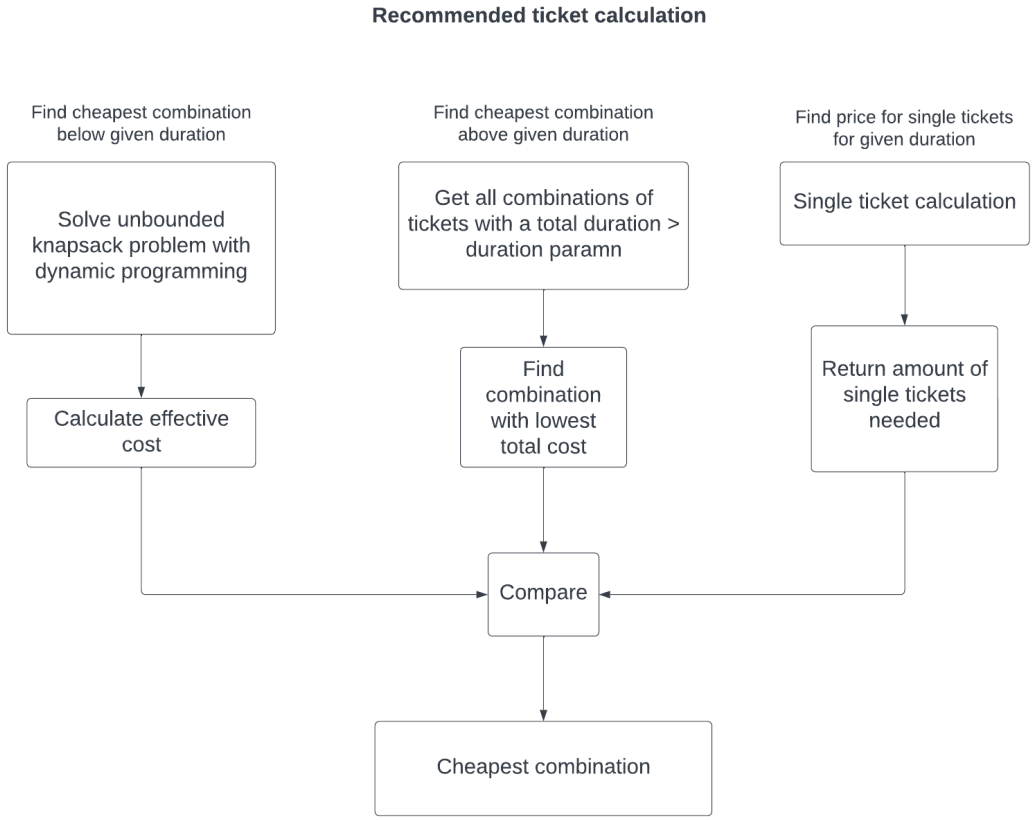
G Issue board

H User-survey report

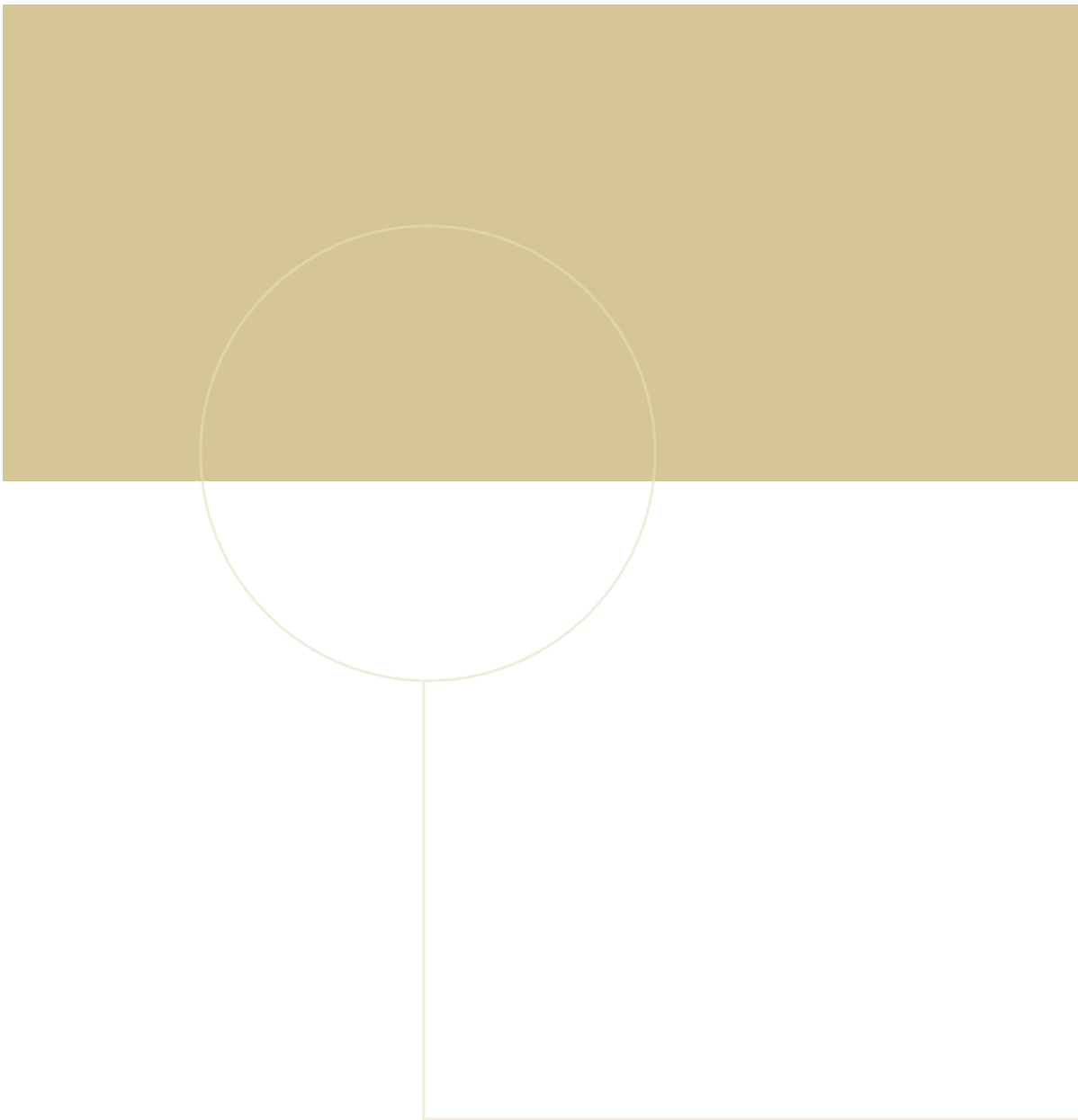
I Prototype user test results

J User test of final product results

K Ticket calculation algorithm flow chart



L Source code



NTNU

Norwegian University of
Science and Technology