Bachelor's project

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

Tore Bergebakken, Jon Åby Bergquist and Helene Yuee Jonson

Dumpster Finder: A mobile app for anonymous, social food waste reduction

How can you allow anonymous use while inhibiting abuse in an app where users can add data freely?

Bachelor's project in Computer Engineering
Supervisor: Donn Morrison
May 2021

NTNU
Kunnskap for en bedre verden

Tore Bergebakken, Jon Åby Bergquist and Helene Yuee Jonson

Dumpster Finder: A mobile app for anonymous, social food waste reduction

How can you allow anonymous use while inhibiting abuse in an app where users can add data freely?

Bachelor's project in Computer Engineering
Supervisor: Donn Morrison
May 2021

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

NTNU

Kunnskap for en bedre verden

Preface

This thesis was written by three students at the Norwegian University of Science and Technology as part of our bachelor's degree in computer engineering. Our task was to make an app for information sharing between dumpster divers.

We chose this assignment because everyone in the group was interested in app development, and dumpster diving seemed like an interesting subject to work on. None of us were involved in the dumpster diving community, but we agreed that it seemed like an interesting topic to learn about. We were also enticed by the idea of potentially providing a service that could be beneficial to our society and the environment.

Working on this project has been both exciting and challenging. Some of the team members had previous experience with app development and wanted more experience in this field. As a team we had no previous experience in programming in the React Native framework, and wanted to use this project as an opportunity to learn it. We had worked with React before, and expected that React Native would not be all that different. Our development process followed the Scrum methodology and our application was developed through different sprints, where new features were added and old ones updated as the process went along.

We would like to thank our supervisor Associate Professor Donn Morrison, the technical staff at IDI for providing a virtual machine for our system, our interview subjects, Jurist Jeanette Bergquist for answering questions about the legal status of dumpster diving, our testers and the people who helped proofread.

Tore Bergebakken

Jon Åby Bergquist

Helene Yuee Jonson

20.05.2021

Date

Assignment

Supermarkets account for a large percentage of food waste in the food supply chain. The goal of this project was to help reduce food waste by creating an application to share information about different dumpster between dumpster divers.

The app was to be written in a portable language/framework such as React Native for platform independence. Software security should be a focus point throughout the development process.

The application should store and share information about:

- Dumpster/supermarket location via GPS
- Information about what kind of food has been thrown away
- Useful information such as best-before dates and the state of the content
- Whether the dumpster is locked or open
- Whether the supermarket views dumpster diving favorably
- Cleanliness of the dumpster
- Dates/times the dumpsters are emptied
- Photos of the dumpster
- Experiences and ratings from visitors to the dumpster

See appendix B.2 for details about the features we actually implemented.

The app should not require user accounts, and no personal data should be stored in the system.

As the project should focus heavily on user security, it would also have to focus on ways to prevent spam from other users and bots that wanted to abuse the system. Together with our product owner/supervisor we discussed the need for moderators and concluded that it would be best if the system was able to work without moderators.

During the development process we held some interviews and together with our supervisor/product owner we agreed to add some proposed features to the application, such as:

- Be able to register visits and to display the number of visits

There were some features in the original assignment that we ended up not prioritizing, in agreement with the client:

- Food recognition from photos could be used via a trained ImageNet model on the server side
- Tor (The Onion Router) support for anonymity

For more detailed information about the assignment, see our requirements documentation in appendix B and our vision document in appendix A.

Summary

Around a third of all food produced in the world is wasted [1]. Food production accounted for 34 percent of global greenhouse gas emissions in 2015 [2]. Grocery stores throw away food that is perfectly edible. Dumpster divers combat this issue by looking through garbage containers for edible food, and eat what others would leave to rot. This activity is looked down upon by a lot of people and considered illegal in many countries ([3] appendix E) , but seems ethically sound in the face of a skewed distribution of food that leaves millions hungry [4].

The goal of this project was to create an app where dumpster divers could share information about dumpsters and the content in them with each other. Security and anonymity were major concerns since dumpster diving is a legal gray area in many countries and those who do it might be looked down upon. Our users might not want to be identified, so we needed to limit the amount of data that could be traced back to them. Having no user accounts, as we originally intended, could be a measure towards this. However, protecting our users' anonymity might make it harder to prevent abuse by users and bots. We wanted to avoid depending on moderators to clear out unwanted data, and saw a user-moderated system as a possible solution. The balance between enabling anonymity and preventing abuse was therefore of great significance to our project, as reflected in our main research question.

We created a mobile app for iOS and Android, with a focus on implementing only what we could consider reasonable when it came to protecting our users' anonymity. We also attempted to make the code maintainable and geared towards being developed as an open source project. Our app had a simple structure, with a map and a list of nearby dumpsters as top-level pages, then things like detailed information about dumpsters, photos of dumpsters, comments and contents as subpages. It was praised for its professional look and intuitive interface during the final user tests.

As part of the process, we conducted interviews with dumpster divers and found some interesting results. Most of those we asked *did* care for privacy and had *environmental* reasons for going dumpster diving. Some suggested features like seeing the number of visitors to a dumpster, and most were satisfied with those we had implemented.

We ended up with a system similar to regular user accounts. Our solution creates an anonymous identifier for each user when they first launch the app, connected to as little data as possible. We avoid tracking our users' location, prevent others from seeing the identifiers through the API, and make data transfers happen through HTTPS only. Some data, like what comments the user has upvoted or downvoted, is only stored in the app itself.

While we did not get to test our result properly, we did end up with a *possible* solution that protects our users' anonymity and preventing abuse. We also found that the need for anonymity does not always conflict with measures for inhibiting abuse.

Contents

List of Figures

Chapter 1

Introduction and relevance

Today's society has been called a throw-away society. A throw-away society is a social concept of consumerism where the development of products outgrows the demand for the products [5]. After the industrial revolution, and particularly in the 20th century, companies started to overproduce goods. To get users to buy more of their products and increase their profits, companies turned to planned obsolescence where products are created with a purposely frail design and destined to fail after some time, so consumers would have to buy new products [6]. Together with single-use items and the development of newer and *marginally* better products, this has led to a situation were people and stores throw away perfectly functional items to get new and *supposedly* better ones. Food is often thrown away while it is still edible, for reasons related to appearance or capacity rather than any actual evaluation of longevity.

Supermarkets and grocery stores accounted for 16% of food waste in Norway in 2018, which is about 75000 tons of food waste a year [7]. The UN estimates that one third of the world's food production goes to waste [8]. According to a report by the Food and Agriculture Organization for the United Nations, around 340 million tonnes of food is wasted worldwide in the consumption phase [9, p. 13]. Stores might throw away food and other products that are perfectly fine, when it is close to or has passed its due date, or when they need to empty their storage to make room for other products. In order to prevent this food from going to waste, some people search for edible food and other useful things in the trash. This activity, called *dumpster diving*, can be done by different people for different reasons. In more developed countries, dumpster diving has become a part of the environmental movement [10], and is thought of as a way to save food from going to waste. It is also a way to save money and is popular among students and young adults [11][12][13]. In less developed countries, dumpster diving has a lower status and is looked upon as something only the poorest do to survive.

As the status of dumpster diving varies around the world, there are also different laws that may or may not prohibit taking things from dumpsters. Therefore, dumpster diving could be considered illegal.It depends on the laws where you live, and the conditions surrounding dumpster diving. Whether it is for environmental, economical or other reasons, people who dumpster dive often want to keep their identity secret from the authorities, store owners and other people. Even if dumpster diving was perfectly legal, it might still be looked down upon.

The goal of this assignment was to create an application that would let dumpster divers find dumpsters in their area and share information about dumpsters with other dumpster divers. The idea was that this would make it easier for dumpster divers to find new places to go dumpster diving and find dumpsters that contain what they want.

Because of the controversial nature of this subject, and its unclear legal status, there was a strong need to guarantee the users' anonymity. Since there was no complete source of information about existing dumpsters at the time our project started, we wanted to create an app where anyone could add data to the system. The dumpster diving community consists for the most part of many small, separate groups of of people that share information among themselves. However, the information is not shared to the broader community through a more widespread system, since there is currently no such system available. The idea was that our app could provide such a system that could be, as much as possible, driven by the community itself.

Our *main* research question was as follows:

> **How can you allow anonymous use while inhibiting abuse in an app where users can add data freely?**

While enabling anonymity and limiting abuse was an important part of our assignment, the focus was on creating the application. Developing a software system entails a lot more than what our main research question focuses on. Therefore, parts of this report will focus more on the development process, our results, and discussions about other topics, than anonymity and abuse. However, we always had a focus on how to balance the need for anonymity against preventing users and bots from abusing the system. Some of the choices made during the development of the app was directly influenced by this need.

Because the main research question did not entirely cover our interviews and some other parts of our report, we had a *secondary* research question:

> **What features do dumpster divers expect in an app designed specifically for them?**

This document is split into six chapters. This chapter is the first, and contains a short summary of the reasons why this project was important and what our problem statement was, in addition to a simple dictionary of terms and acronyms used in the thesis. The second chapter contains information that was relevant for this thesis, including an overview of different ways to ensure anonymity and prevent abuse. The third chapter is about technology and methodology. Here we write about what technologies we used during the project and why we chose them. We also talk about the different methods we used during this project — both during development, and in getting important data for the project. Chapter four gives a detailed description of the results from this thesis. There we talk about what we developed during the thesis, how we worked and what we discovered. In chapter five, we provide a more detailed discussion of our results. Here we discuss what went well and what we could have done better, both in context of the application and in how we worked during the thesis. The last section contains our conclusion(s) with regards to the question stated at the beginning of this document and improvements that can be made to our product at a later stage.

1.1 Vocabulary

API Application Programming Interfaces are interfaces that let different services communicate with each other. In the case of our API, communication happens over the web [14].

CAPTCHA Completely Automated Public Turing test to tell Computers and Humans Apart [15].

CI Continuous Integration is the practice of automated integration of newly developed code [16].

IO-bound IO-bound operations spend the most time waiting for a file read to finish or a network request to complete, as opposed to having their duration depend on how fast the operation can be executed [17, p. 654].

JS JavaScript is a dynamically typed programming language that is widely used on the web, and has been gaining traction in other domains [18].

JSON JavaScript Object Notation is a file format that stores readable text as attribute–value pairs [19].

Middleware software that connects two pieces of software together

Negative network externalities increasing the user base of the service provides a negative effect to a user using the service [20]

NTNU Norges teknisk-naturvitenskapelig universitet / Norwegian University of Science and Technology

ORM An object-relational mapper maps data in a relational database, where data is stored in tables with predetermined columns, to objects in a programming language, and vice versa [21].

Relational database A database that "uses a structure that allows us to identify and access data in relation to another piece of data in the database" [22].

REST Representational state transfer is an architectural style for APIs that deals with resources and a set of standard operations (GET, PUT, POST, etc.) on them, with a few extra criteria like statelessness [23][24].

SQL Structured Query Language is a programming language for managing data in a database [25].

TOR The Onion Router, a software that allows for more anonymous and private internet traffic [26].

TS TypeScript is a programming language that is a superset of JavaScript. It adds static typing, with type checks at compile time only [27].

(G)UI (Graphical) User Interface is the screen/space where users interact with a piece of software [28].

UX User Experience relates to *how* the user interacts with the product/software and their experience with the interaction [29].

VM A Virtual Machine is a virtualization of a computer system that contains the same functionalities as a normal system, but runs inside a host system [30].

Chapter 2

Theory

This chapter goes through the theoretical background for our work — that is, the theories behind our project and the theories behind the technologies that the project depends on.

2.1 Agile methodologies

Agile methodologies are development methods that focus on having an adaptive approach to a project in order to respond quickly to any changes in the process. Examples of agile methods are Extreme programming (XP), Kanban and Scrum. Agile methodologies were popularized by the publication of The Agile Manifesto in 2001. The Agile Manifesto has four main points and aims to change the focus from heavy documentation over to an efficient and adaptive development process:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan [31]

2.1.1 Scrum

The Scrum framework was developed in the early 1990s and " [Scrum] is an iterative, incremental framework for projects and product or application development. It structures development in cycles of work called Sprints." [32].

Scrum is an agile process created to improve communication and cooperation in teams. It enables the team to respond to changes in the requirements, adjust the course quickly if something is amiss, discover problems and obstacles early, and remove obstacles if they get in the way of the development process. The scrum method is based on three pillars:

- Transparency — Feedback loop refers to an open process that gives workers and customer an overview of the project as it develops.

- Inspection — Inspect process towards goal to detect if the project moves away from the goal
- Adaption — Adjust the process if it has deviated outside the limits.

The method splits the process into multiple iterations/sprints. Scrum has three artifacts (product backlog, sprint backlog, and burndown chart). The product backlog is a prioritized list of all desired functionality of the finished product, formulated as user stories. The product backlog is dynamic, and often changes as the project develops. Each sprint backlog contains a subset of the tasks on the product backlog, chosen based on their priority. The burndown chart is a chart that displays the remaining work in the current sprint. It is connected to the sprint backlog and shows the teams progress [33].

After a sprint is finished, the team reviews the sprint in the Sprint Review and the Sprint Retrospective. The Sprint Review is a meeting with the product owner, where they go through the results from the sprint, have a conversation about the product development, get advice, and so forth. The Sprint Retrospective is a way for the team to discuss and adapt regarding the process [32, p. 14]. A way to do this is to write notes in different “columns”: One where the team can write about what went well, one where they can write about what could be done better and one where they can write measures to improve the situation in the next sprint [34, p. 86]. Then the team can take the information and try to adapt it into small changes to do in the next sprint [32, p. 14].

2.2 Universal design

The idea of universal design is based on the concept of designing products that are accessible and usable to people regardless of sex, race, age, disabilities and more. The Center for Universal Design (CUD) has 7 main principles for universal design that will contribute to make the product more inclusive and easier to use. These principles aim to ensure that the product is easy and understandable to use, and that it gives an equal experience to all users regardless of different abilities, preferences, and level of understanding [35].

2.3 Anonymity

Anonymity means that there is no way of identifying the user as an individual outside of the system. This means that whatever service or tool you’ve created, it doesn’t collect any identifiers [36]. This includes direct identifiers such as names, email addresses, phone numbers, and any governmental identification numbers [37]. You also want to avoid collecting indirect identifiers like age, occupation, gender, and anything that could be used to identify the user.

An important step to take in not sharing and storing data that can identify users, is deletion of metadata [38]. Obviously you need to avoid directly collecting data that could identify the user, but you also need to make sure that images or other uploaded files do not contain metadata like GPS location, the resolution of the photo, the creator of the camera, etc.

In order to have a way to connect a real life user to a real individual you need an identifier, but since you don't want any direct or indirect identifiers in your anonymous system, this identifier should be generated on the server-side. Methods to do this could be automatically generated usernames and maybe password, seed phrases, or randomly generated data of sufficient length that are hard to brute force. This is done mostly because humans tend to reuse usernames and passwords on multiple sites [39].

2.4 Preventing abusive content in an anonymous system

Abuse is any content the service defines it to be, typically defined by the terms of service [40]. It would commonly be defined as offensive content: racism, sexism, etc. It would also often include spam, which is unwanted content sent in bulk, mostly known in the context of email [41].

In a system with moderation you could have moderators validating every piece of data added to the server, which would remove all abuse, but this is very costly. The other extreme would be not doing anything, and letting all abuse through. Most solutions lie somewhere in between, utilizing various tools to lower the amount of abuse, but not eradicating it. Moderation teams are a common tool, removing anything they see that breaks the rules. User moderation is also used, but is less effective because this gives the abusers the same tools as anyone else, which means you usually have to limit the power of one individual.

Automated ways to prevent abuse are very useful. Spam filters are commonly used in email systems to separate useful emails from junk or advertisement. The implementation can be everything from simply blocking certain words from entry, to something like a naive Bayes spam filtering that uses statistics and learning with a word list to remove spam. Rate limiting lowers the amount of requests that can be sent from an individual IP address, and is an effective tool against spam. Challenge/response systems like CAPTCHA prevent bots from accessing the system, which greatly lowers the amount of spam [42].

2.5 Anonymous systems

It can often be hard to determine if a system is anonymous or not without looking at its code and testing the system, but there are various systems that at least give the illusion of anonymity.

4Chan and imageboards sites are fairly well-known anonymous services. Though they are known for being riddled with spam, they do have measures to limit the amount. They have administrators and moderators, and do things like temporary RangeBans (blocking IPs within a range of values) and individual IP blacklists.

There are quite a few anonymous social media services. These have often been criticized because of their anonymous nature and the abuse that often follows. Cyberbullying appears to be the biggest issues on sites like these compared to regular social media, because some people feel comfortable posting mean and offensive things when they are anonymous. These

sites are also rarely anonymous, since they require accounts and thus are pseudo anonymous, but they do give people the illusion of anonymity when interacting with others [43].

2.6 Legality of dumpster diving

Items that are thrown away are not *considered* ownerless, but are owned by the store or the company that owns the container [44]. Therefore, an owner could report the incident to the police as theft. In Norway this falls under chapter 27, § 321 “Theft” and § 323 “Smaller theft” [45]. However, there have been no trials for dumpster diving in Norway. If a dumpster diver breaks open a lock, fence, or door, or jumps over a fence to get access to a dumpster, they could also be reported as a burglar – see chapter 27, § 346 “Illegal use etc. of real estate”, chapter 28, § 351 “Damage” and chapter 28, § 352 “Small damage” in [45].

Section 4.3 in *Den enes død, den andres brød* argues that legally speaking, items thrown into a dumpster by a store are owned by the refuse company that will pick it up, since that party has an economic interest in the disposal of waste. However, the text brings up counter-arguments to this claim — that the renovation company will not lose much if some trash is taken, and that socioeconomic factors like the wastefulness of letting e.g. edible food rot, more than make up for the minuscule loss of money. Preventing a society of excessive consumption could be considered more important than supporting a renovation company that is likely to work towards keeping society the way it is [46]. This should make it apparent that the issue is not completely clear-cut, and could be argued for in either way. The laws that relate to dumpster diving do naturally depend on the country where the user resides, and they themself should be aware of the laws in their country.

Chapter 3

Technology and methodology

In this chapter, we will go through the different technologies and methodologies we used in our project. The first part will describe technologies we used and the motivations for our choices. In the second part we will talk about different methods we used during the process — both the different methods we used to collect data from test subjects, and how we as a group worked on this project.

3.1 Technology

We picked quite a few libraries that filled certain needs. However, we limited how many dependencies we had, since we did not want to depend on too many third-party solutions, since a higher amount of dependencies would mean a higher probability that there would be one with a vulnerability.

Most of the project was written in TypeScript, an extension of JavaScript that adds static type checking at compile time. Type checking is a measure against type-related bugs that seem to be more common in dynamically typed languages like JavaScript [47], and a way to have more useful autocompletion in our editors of choice. We used an auto-formatting tool called Prettier to make our code more readable and prevent discontinuities in the code style, and the Jest testing library to write unit tests that could ensure reliability.

3.1.1 Frontend

We chose React Native as our frontend framework, mainly because it allowed easy deployment to both iOS and Android. Another benefit was its ease of use, particularly for a team with some experience with web development in React. It does perform slower than native apps for these platforms, but this should not be an issue for an app that displays data as text, as images, and as markers on a map. Lastly, it is a popular framework with a significant amount of third-party packages available. We wished to limit our reliance on third-party packages, but it was beneficial to use *some* as it would save both time and resources as opposed to having

to create everything ourselves.

Environment

We chose to use Expo with its built-in SDK to simplify the development process, knowing that we could move to a more controlled workflow later. It lets us test changes swiftly since any changes in the code trigger a quick refresh of the app. It also makes showing our app to other people very simple — they can scan a QR code and have the app show up on their phones.

Component library

To avoid spending too much time designing and writing our own widgets from scratch, we chose to use a third-party component library. Our choice fell on React Native Elements at first, but we were not satisfied with it and ended up switching to UI Kitten. The latter was more complete and had a consistent, good-looking style.

Map library

Since the main point of this thesis was *not* to make a map component from scratch, we decided to use a map library. The only two alternatives we found for React Native, were `react-native-maps` and `react-native-mapbox-gl`. The former seemed to use Google Maps and Apple Maps for some of its functionality, which could be problematic in an app with a focus on anonymity, while the latter contained native code and would force us to stop using Expo. We chose to *begin* using the one we would spend *least* effort setting up, and switch to the one that might be more ideal for this anonymity-focused case later.

State management

For managing global state, like the user's preferred theme and cached dumpster data, we used Redux. It focuses on using functions without side effects to create new versions of the application state, instead of mutating it directly. The fact that all objects stored in the state must be serializable, made us unable to store instances of classes directly and contributed to us preferring to use objects specified by interfaces rather than creating classes for every data model.

Other

We wanted to make the app available globally, and needed some framework for handling translations. We chose i18next for its ease of use, its translations are simple JSON files, and because it seemed to be used by a lot of projects. We also found quite a few tools for translators, which should bode well for future work on the project. We used Formik, a popular library for input validation to make sure that users got feedback before anything was sent to the backend, and to prevent unexpected behavior in the app itself.

3.1.2 Backend

We chose to split our backend into three separate services: A relational database that would store metadata about dumpsters, a file server for images, and an API server.

Database

MariaDB, an open source fork of MySQL, was our database solution of choice. We wanted to avoid using a proprietary database system, but still use a popular one. We needed a database with support for geographical data and queries that fetched entries in a given physical area, and MariaDB seemed to match these requirements. We considered using a NoSQL database like MongoDB, but found that our schema relied too much on *relations* between entities to be viable using a simple document structure.

Language

Our file and API servers used the Node.js runtime and the Express framework for writing REST APIs. Node was chosen because we used TypeScript on the frontend and wanted to use the same language in the backend. Additionally, it performs well for IO-bound tasks, which matched with our app not needing CPU-intensive tasks in the API server since that work could be offloaded to the relational database. Our API server would for the most part be an abstraction layer between the client and the database. In the file server, very little work would be required before stashing a file away or fetching a file from storage.

API framework

We chose Express because it is well-known and had a neat system where we could write middleware functions to handle common tasks like validating tokens, and simply add them to the endpoints where their functionality was needed. Our servers have middleware for validating input sent in requests, using the popular library Joi to make the process much more streamlined for us.

Database connection

For handling database calls in the API server, we used Sequelize – a popular ORM that has been around for ten years, with a syntax that is easy to grasp. In 2019, someone discovered a SQL injection vulnerability which was fixed rather quickly after the developers were notified of the problem [48]. While that might sound worrying, the fact that the vulnerability was fixed in a short time seemed like a good sign.

Environment

To isolate our services from the system they were hosted on, we used Docker and Docker Compose. Each service would run in its own Docker container, with just the bare minimum

of ports exposed to the outside network. This made continuously deploying new versions of the server software rather simple, as it was just a matter of rebuilding and restarting a set of containers.

Tokens

The jwt-simple package was used for token support. Tokens are used for user security while barely compromising efficiency compared to constantly authenticating the same user by validating their existence in the database. For more information, see our system documentation in appendix C.

Hashing

Node's built-in Crypto module was used for hashing. In our final build we use PBKDF2 with SHA512 and a random 32 character salt to hash the randomly generated user ID. PBKDF2 is a standard hashing function for passwords because it is deliberately slow [49]. This means that people that try to brute force the system are gonna have a harder time. Because we hash with salt, rainbow tables become essentially useless. Another reason it is useful is because it is hard to reverse.

Word list for usernames

In order to maintain anonymity we decided that it would be ideal to implement a user identification system that generates unique identifiers pseudorandomly. We ended up using a solution similar to the seed phrases used by cryptocurrency wallets. The premise is that x amount of words are randomly chosen from a list of words, this becomes the users identification phrase. The word list we ended up using is the EFF's long list, because it contains a lot of words (7776), and it focuses on not using words that might cause discomfort [50].

3.2 Methodology

3.2.1 User tests

We wanted to test our design as early as possible, to discover any misleading or otherwise problematic parts of our design. Our plan was to test our prototype shortly after finishing it, and to test our product sporadically as we developed it. We would lay out a set of tasks that our participants would accomplish in the prototype (or app), and take detailed notes as they navigated through our user interface.

3.2.2 Interviews

Because our user base might have special concerns regarding our app, and because we had no experience with the subject matter ourselves, we found it necessary to hold interviews with

dumpster divers. The purpose was twofold: To gather information about dumpster diving itself, which we could use to enhance the content of our app, and to test our prototype on people who were more fit to give feedback. In addition to these usual elements, we wanted to know how important privacy is to dumpster divers. If there was no concern about privacy among the people doing it, there would be less reason to have such a high focus on it in our development.

Before each interview, we asked the participant to answer a survey. The interview itself would start with us giving the participant a few tasks to do in our prototype, and end with some open questions that could lead to an open discussion. See appendix E.2 for our interview plan.

3.2.3 Scrum

There are several different ways to do Scrum. Most versions of the methods have a team structure, sprints, and the use of different kinds of documents (called artifacts) to document the process and how far the project has come. These documents are the product backlog, sprint backlog, burndown chart and more.

We decided to use a slightly modified version of the Scrum method. We have worked with Scrum previously and found that it is a good method to use when you are developing a new product. There were some parts of the traditional Scrum method that we did not follow. Normally there is a Scrum master who is responsible for the communication with the product owner, and for supplying the team with the necessary resources. Since we were three team members and our supervisor also had the role as product owner, it would be unnatural for *one* person to take that role. We decided to have a team member who stood for the majority of the communication between the team and the product owner, while the rest of it was down to the team as a whole.

We decided to use Scrum as it is an iterative method that fits well for software development, and we had previous experience with the method. Since the thesis was a relatively large project that would run over a full semester, it was important to have a semi-structured plan throughout the entire project. Thus, it seemed best to use an iterative method. This allowed us to adjust the direction of the project without changing a very detailed plan. This also made us more able to adapt when unexpected problems appeared.

To keep track of the project and our process, we used some of the usual Scrum artifacts. We split the semester into different sprints, where every sprint should have a Scrum board, sprint backlog and burndown chart. For the first months of the semester, we had another mandatory course running in parallel with the thesis. To accommodate for our lectures, assignments, and exam in the course, we decided to run sprints with a slightly lower workload while we attended both courses. We cut each week down to five days on the burndown chart, and planed our workload accordingly. Some of the group members also had electives and other exams during the period of the thesis and we wanted to make sure everyone got all their unrelated work done.

3.3 Work allocation

We decided to distribute some of the responsibilities between different team members. This way we had a system that made sure the administrative part of the project was taken care of.

- **Communication manager:** Responsible for the main communication between supervisor/product owner and the team. *Tore Bergebakken*
- **Meeting organizer:** Responsible for writing and sending notices of meetings. The meeting organizer was also the moderator during meetings. *Tore Bergebakken*
- **Meeting secretary:** Responsible for taking notes during the meetings and writing minutes of meetings. *Helene Yuee Jonson*
- **Interviewer:** Responsible for guiding the interviews and making sure that the interviews stayed on schedule and did not go off track. *Jon Åby Bergquist*
- **interview secretary:** Responsible for taking notes during the interviews. *Tore Bergebakken*

We started this project by discussing what we wanted to get out of this thesis. During this process, we created a labor contract, which you can find in our project manual [51, p 5] This is a document where we wrote down different goals, roles, and terms for the project. Some of the roles listed here do not appear in the labor contract, as they were added later in the process when we found it necessary.

During the development process we had no specific programming roles. We had a sprint backlog where we would select tasks. This made it easier to choose tasks that one felt able to do. As some of the team members had more expertise with *either* frontend or backend work, it seemed likely that there would be a natural split in the team as some worked more on backend or frontend. However, we wanted to make sure that everyone had worked on both parts as this was important in order to have everyone understand how the system worked in its entirety. Other possible roles have been naturally filled when needed as all team members were willing to step up when necessary.

Chapter 4

Results

In this chapter, we present the results of our work during this project. Scientific results include the results we got as part of our research during the project. In systemic results, we write about the application we created as a result of this project. The administrative results describe how the project was carried out and focuses on the development process and distribution of work hours.

4.1 Scientific results

4.1.1 User tests

At the onset of the project we conducted several user tests in order to identify any problems related to the initial design. As shown in appendix D, we found some problematic elements of the design and received a lot of useful feedback. One problem was the positioning of, and choice of icon for the button that would take users to the comment section. This was acted upon and reworked.

Near the end of the project, we tested the app on some of the same users. This final test was informal and thus not properly documented. We discovered a bug with the photo upload. In addition we got feedback on things we were already aware of, that we could improve. The users were quite satisfied with the product and thought it looked professional.

4.1.2 Interviews

We conducted ten interviews with eleven dumpster divers. The participants were asked to fill in a questionnaire before the interview. This was the source for the diagrams shown in this section. We found that people do want privacy, and have various reasons for dumpster diving – the most common being the environmental side of things (fig. 4.1). Some interview subjects suggested adding features that were not part of the initial plan. Among these ideas, there were some that were too ambitious, and others that were doable. The functionality for

registering the number of people visiting a dumpster was one of these ideas. This was implemented, as it did not take too much time to do and seemed quite useful. Several people suggested implementing some sort of notification in case more content appeared in a dumpster the user had marked. We did not consider this doable in the time we had left. A few suggested functionality notifying other users about cases of harmful food found in dumpsters. The participants also talked about the social aspect of dumpster diving, and asked if the app could let people meet up for a dumpster trip — this was far outside the scope of this project, and would conflict with our focus on privacy. We also received suggestions for additional pieces of advice to add to our list of tips, which was put together haphazardly in the very beginning of the project.

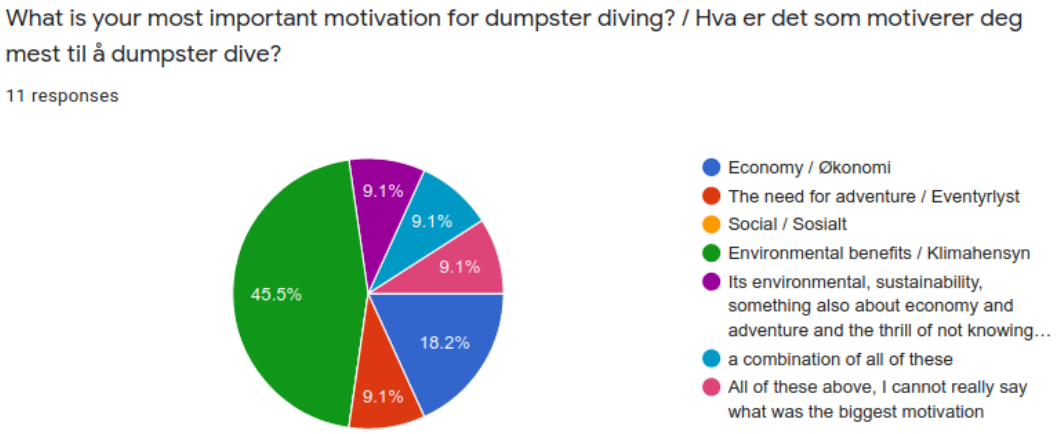


Figure 4.1: Distribution of motivations for dumpster diving

A majority of the respondees (10 out of 11) would like to use the app, though some had criteria that would have to be met before using the app. They commonly thought that there would have to be *enough* information present in the app for them to use it. However, only 27.3% of them stated that they usually do research before diving in a dumpster (fig. 4.2). In some cases, this seemed to stem from their particular situation; they already had a few dumpsters they were familiar with, and felt no need to find others. One person stated the following: “Nah, we just go to a dumpster and look inside it” (appendix E.9.2). They generally thought that this app would be more useful to *new* dumpster divers, rather than veterans who already know where to find things.

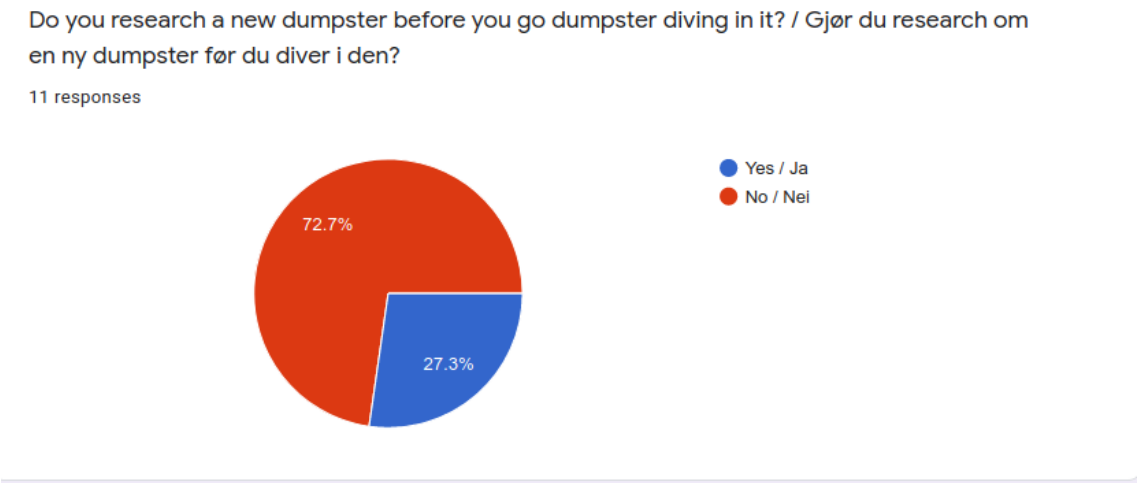


Figure 4.2: Answers to a question about research before diving

In stark contrast to the previous questions, 81.8% were willing to to share information about dumpsters they knew (fig. 4.3).

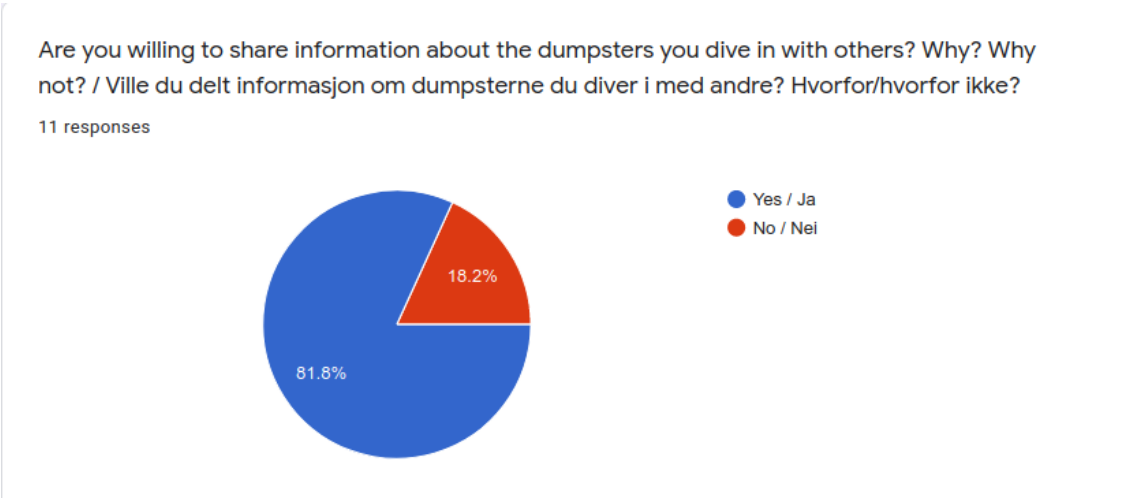


Figure 4.3: The amount of interview participants that would share information with others

We rarely discussed the question of privacy during the interviews, but some thoughts about this matter came forth. According to the answers given in the questionnaire, people *did* consider privacy to be an important concern in relation to dumpster diving (fig. 4.4 and 4.5). The interview subjects had various reasons for wanting to hide the fact that they dumpster dive, including being afraid of letting other people know that they were taking part in such activities.

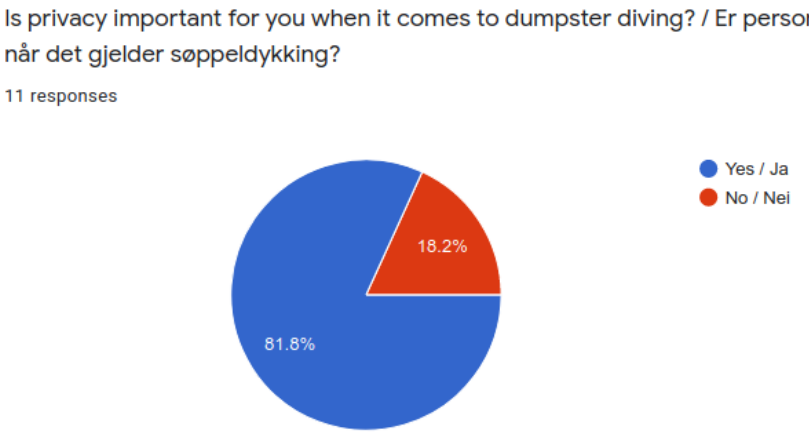


Figure 4.4: The participants' personal care for privacy

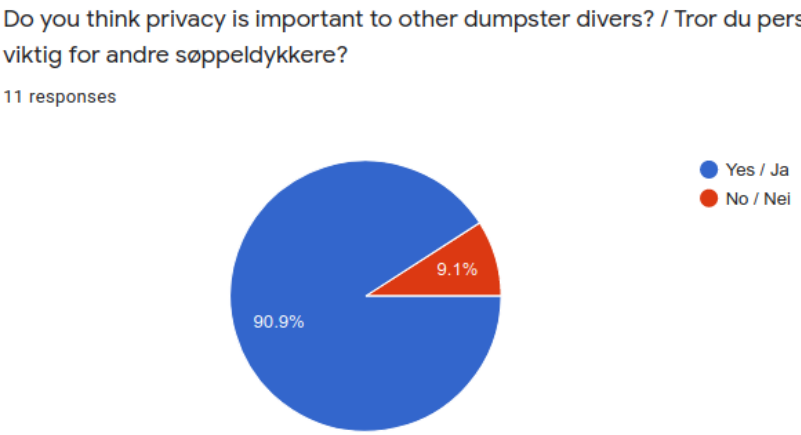


Figure 4.5: How much the participants thought privacy would be important to others

Most of the participants found using the app to be intuitive, though some struggled with specific parts of it. We identified these problems and have made improvements to the UI where necessary. Some users thought they could access a photo gallery by tapping on the picture shown together with dumpster details – we made this possible in the app. Others had trouble accessing the information editor, which we did *not* consider an issue since it was not supposed to be *too* visible, and we used a standard design for the menu one would access it through. A significant amount of our interview subject stated that they would prefer to have to *write* as little as possible in order to add information about something they had found. They would prefer to take a picture or press a button for the kind of content they found. In the end, we did not have time to deal with this issue all that thoroughly, and ended up with a solution where users would be directed to a generic photo gallery if they wanted to upload photos of contents.

Lastly, some users were concerned about the impact of the app — fearing that dumpster diving might be ruined because of popularity and easy access to information.

4.2 Systemic results

4.2.1 Functionality of the application

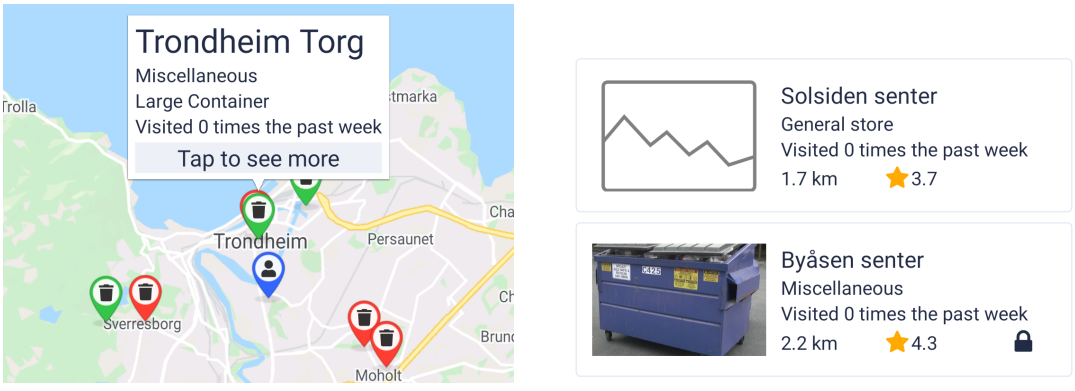
Here, we will go through the functionality of the app as specified by the user stories in appendix B.2. We will base the structure of this chapter on general features rather than going through each user story separately. For our source code, see our GitHub repository or our GitLab repository (only the latter shows CI status).

Map view

A map showing the area around the user's registered position and all dumpsters within a configurable distance from the user's position. The user's position is marked with a blue pin, open dumpsters have green pins while locked dumpsters have red pins. When a pin is selected, the user can see some information about the dumpster — name, store type, dumpster type and the number of people who have visited the dumpster recently. See Figure 4.6.a. We intended to switch to a different map library that did not use Google Maps, but after several hours of attempts, we gave up, as we did not have that much time left. Our supervisor agreed with our choice as well.

List view

A list of dumpsters in the area, shown in Figure 4.6.b. The list view shows the distance to the dumpster from a user's registered position, the rating of the dumpster, a photo of the dumpster, whether it is locked or not, in addition to the information that is shown on the map view.

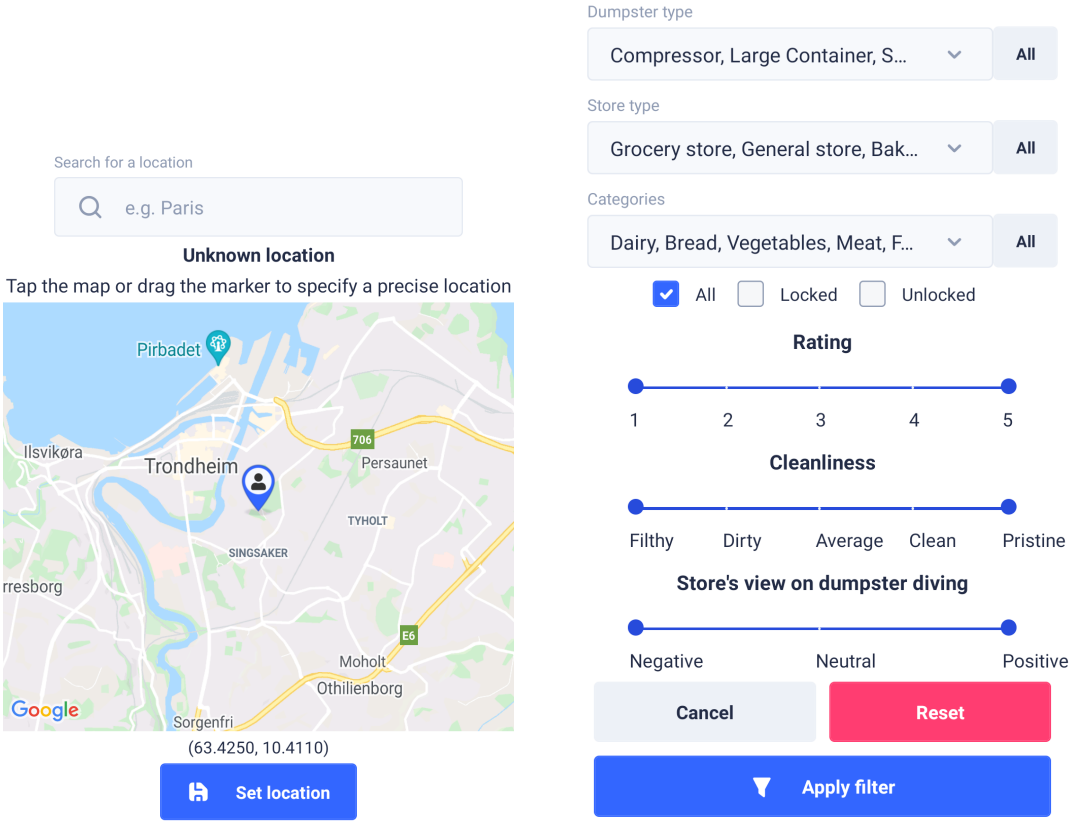


(a) Map (b) List

Figure 4.6: Map view and List view

Filtering

A user can search for dumpsters by name or use the filter to enter specific criteria for the dumpsters. See figure 4.7.b and 4.8.a.



(a) Location (b) Filter

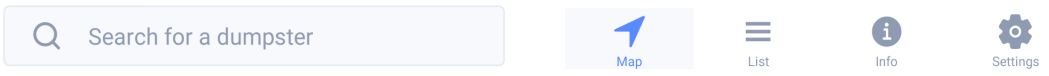
Figure 4.7: Location setter and filter modal

Location

We wanted to avoid using the phone's location data by default, to protect our users' anonymity, and therefore made a manual way to set it the standard. A user can search for a specific location (with suggestions) and/or move the pin to a precise position on the map, in the UI shown in Figure 4.7.a.

Navigation

Users can easily navigate between the different pages of the app. We made the top-level navigator be a tab bar as opposed to a hidden menu that you slide in from the side, since a bar is more immediately visible and will tell the user where they are at all times. See figure 4.8.b.



(a) Search bar (b) Navigation bar

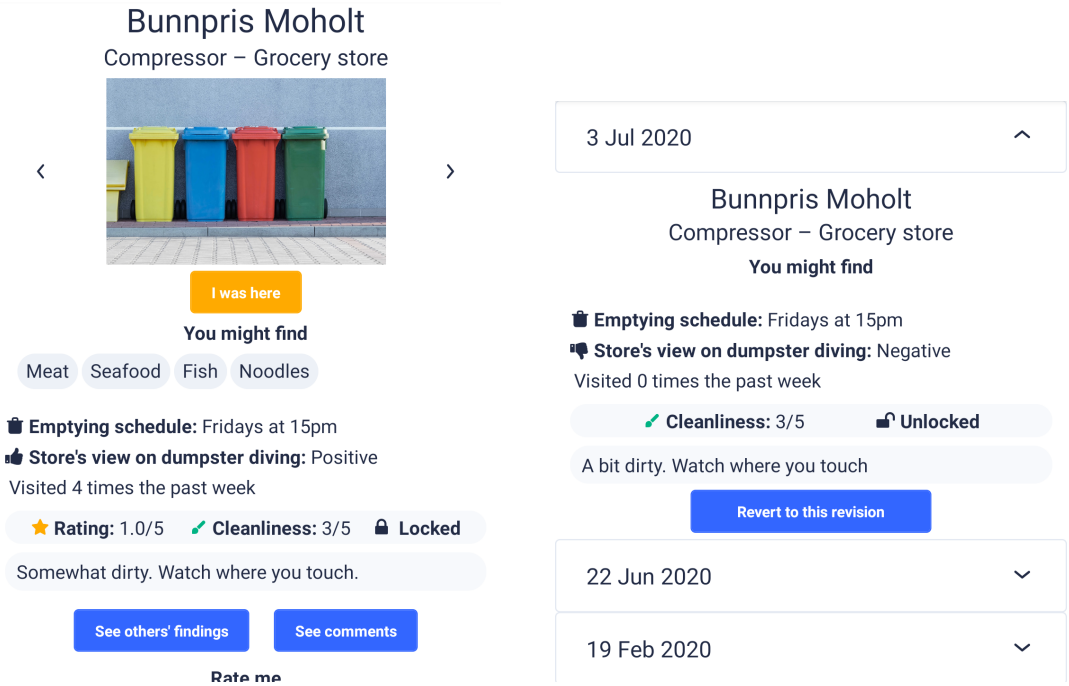
Figure 4.8: Search and navigation bar

Dumpster info

Users can see important information about a dumpster such as if the dumpster is locked, when the dumpster is emptied, how clean the dumpster is, the store's view on dumpster divers, what kind of dumpster it is, the average rating of the dumpster, registered visits to the dumpster, and pictures taken of the dumpster. Users can place a new dumpster at a given location and enter information about that dumpster to add it to the list of dumpsters. Users can also edit a dumpster's data if some of the information is wrong or has changed. See figure 4.9.a. for the dumpster information screen.

Revisions

A user can see previous versions of a dumpster's data and revert to another revision if the current one is wrong. This was another measure against abuse. See figure 4.9.b.



(a) Details (b) Revisions

Figure 4.9: Dumpster details and revisions

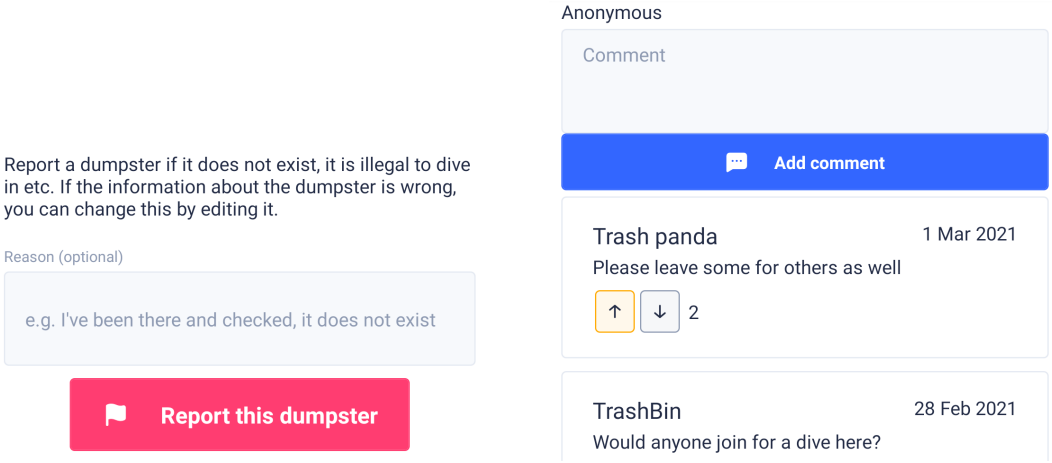
Report nonexistent dumpster

A user can report a dumpster if it does not exist. This handles cases where people or bots have added data with no basis in reality. See figure 4.10.a.

Comments

Users can read other users' comments about a specific dumpster and the date the comment was posted. They can add comments themselves and upvote or downvote comments. They can

choose to hide comments that have a rating lower than -5. See figure 4.10.b. The comments will be registered with a nickname that by itself might not identify the user, and is set to *Anonymous* by default. Nothing prevents users from setting the nickname to one they have used on other platforms if they want to sacrifice some part of their anonymity, but that is ultimately their choice.



(a) Reporting of dumpster

(b) Add a comment and a comment

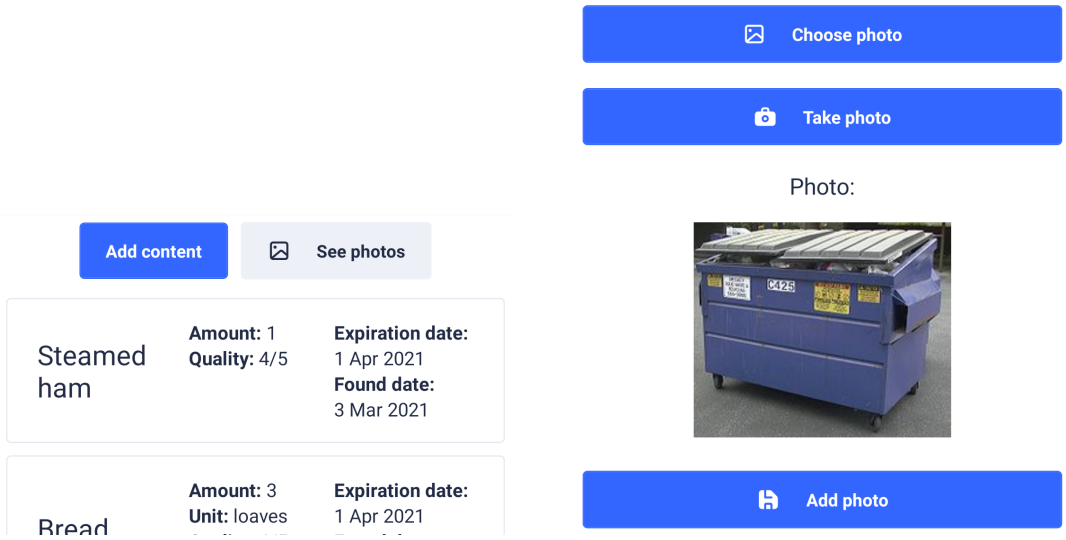
Figure 4.10: Report and comment screen

Dumpster contents

The user can see registered content in a dumpster and important information such as the amount, best-before dates, dates the content was found and what state the content is in. A user can also add different content they have found or take a picture of the contents in the dumpster. See figure 4.11.a.

Pictures

Users can upload pictures of dumpsters and/or their contents, and view them in a gallery — see Figure 4.11.b. We made sure only PNG and JPEG files could be uploaded, and made the API server accept images from *our* photo server *only*.



(a) Content

(b) Add photo

Figure 4.11: Content and Add photo screen

Intro page

An introduction page is shown the first time the app is opened. It lets the user set the language and their given position, explains the user ID, and has some general information about dumpster diving and the application.

User ID

The user ID system played an important part in maintaining the users' anonymity. We could not use a traditional email/username and password system since it might identify the user, so we used a system similar to what crypto wallets use, which is just randomly selected words from a list. We chose to select 6 words.

4.2.2 Non-functional goals

In this section, we will go through the functional goals listed in the vision document (appendix A.6). Since the functionality is already talked about, we will exclude that part.

Usability

> The product shall be easy to comprehend and use for people with various disabilities or lack thereof.

We have followed some common design principles to make our app intuitive and possibly usable by people with some disabilities. For one, we consistently use icons and text in addition to color, to accommodate for color blind users. We have also verified that the app obeys the

device's font size, and that there is good contrast in most components, which helps the visually impaired. There may be other disabilities that we do not properly account for, but for some of them there is a question of whether they would be able to use a smart phone at all.

Reliability

> The backend should have a reasonable uptime (99%, ideally) and be able to avoid crashing under unexpected circumstances (invalid input and the like).

Both the API server and the photo server validate input and are set up to avoid cases where an error causes an actual crash. Setting aside unforeseen bugs, the only exception is if their configuration is invalid, but that *should* cause serious errors, otherwise one might not get to know about the problem.

Performance

We have not performed any stress testing of either part of our application.

> The backend should perform well under expected load, and perform decently under loads that exceed our expectations.

From what we *have* seen, the performance of the backend is atisfactory.

> The frontend — that is, the mobile app — should perform smoothly without being too taxing on the phone's hardware. This might be limited by our choice of framework.

We have experienced delays in some parts of the application, but nothing too concerning. The problems do not seem to be caused by us, rather by certain libraries we used.

Security

> The product shall not store identifying information or other sensitive data that can not be traced back to its users.

The only identifying information would be the user ID, which is stored in a secure way and should not be able to be traced back to the user — unless someone knows a user's ID and has access to the database itself.

> We will take particular care when it comes to transmitting and storing location data.

Location data is only transmitted when the app fetches dumpsters, since the API needs to know what area it should limit its search to, and when dumpsters are added or edited, which is not nearly as directly related to the user's actual position. The server stores the locations of dumpsters, which is absolutely necessary, and nothing more.

> User accounts, if any, shall be protected with the usual measures — passwords shall be stored and transmitted securely.

We did not end up with a typical account system, but we *do* have random identifiers that are hashed and salted when they are stored in the database.

See appendix C.7 for a more detailed explanation of our security measures.

4.3 Administrative results

4.3.1 Development process

The process was split into six sprints that ran for approximately three weeks each. To accommodate for our other courses, the first sprints were a bit shorter than the latter ones. While the Gantt chart (Figure 4.12), sprint backlogs and burndown charts represent some of the work we did, there were tasks we had to work on that could not be placed on the sprint backlog as they did not fit the typical sprint format. Administrative work like meetings, discussions and writing on the reports were not registered in the sprint backlogs as they did not fit in among the other sprint tasks. From the burndown charts, it seems like there was an uneven distribution of work hours throughout the sprints. The reason for these is that we mostly edited down the hours of work only when the task was completely done. The only times we downgraded it before the task was completed, was if the task was almost completely finished and we had to wait for some other tasks before we could finish it completely. For more detailed information about the different sprints and our Gantt chart, shown in Figure 4.12, see our project manual [51].

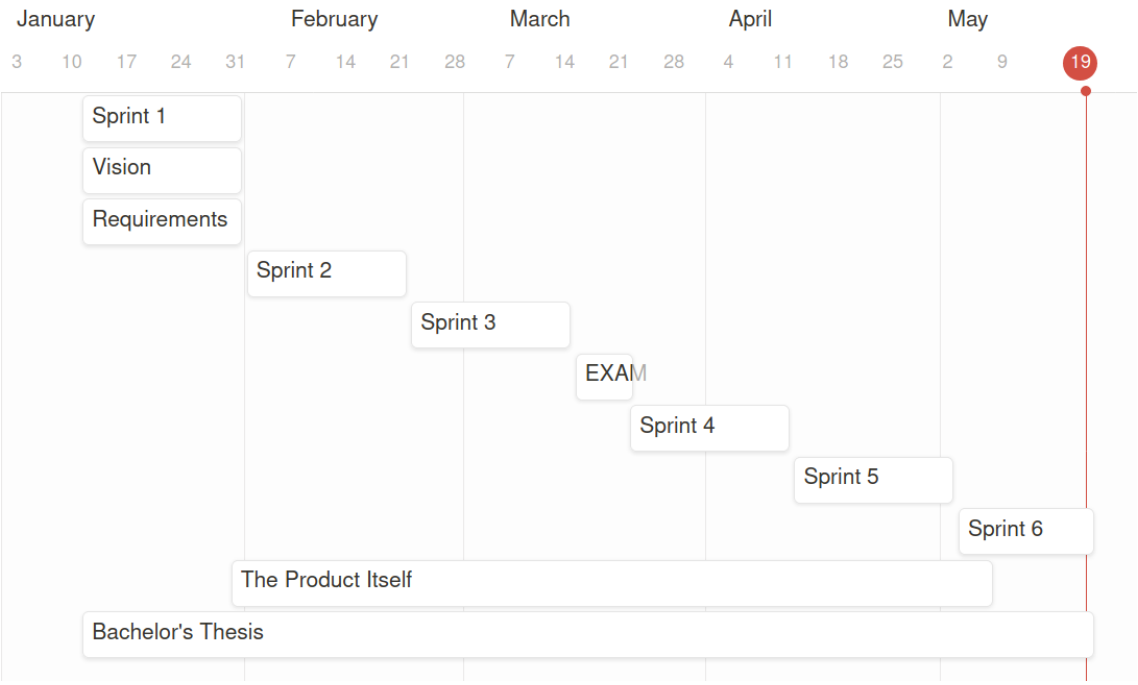


Figure 4.12: The Gantt chart, showing our overarching plan for the project

First sprint

The fist sprint was mainly an organizational sprint as we wanted to make sure that we had a good understanding of our assignment. We wanted to start writing the different documents and finish our first drafts of the vision document and requirement documentation. When they were finished, we discussed them with our product owner in order to clear up any misunderstandings and discuss possible solutions. In GitLab, we created layouts for the different parts of the application so that we could start working without worrying about different setups.

The results from our first retrospective was affected by the fact that we still were in the planning phase. We had not figured out how we wanted to do everything and the best way for the team to work together. This resulted in some miscommunication and uncertainty about what tasks that should be done. We thought that we should do a bit more research on some of our focus topics, but overall we were happy with the sprint and that we worked well together as a team.

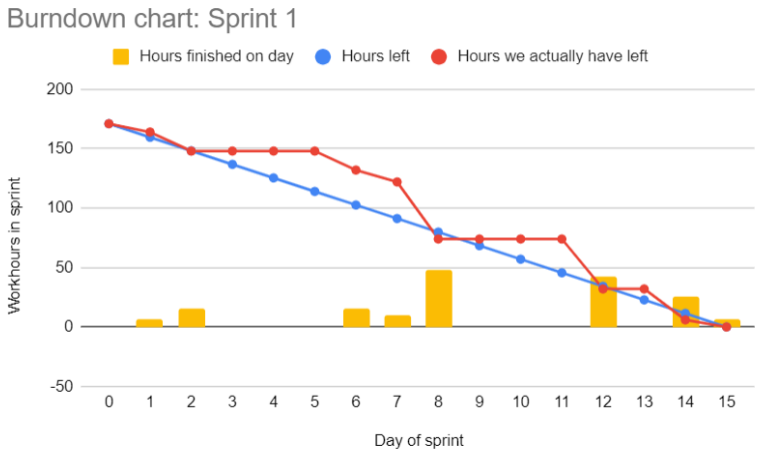


Figure 4.13: Burndown chart from first sprint

Second sprint

Our main goal for sprint two was to get started on the application and implement the most central features. We were able to create a temporary frontend for most of the features, set up the overall layout of the backend, and deploy it to our server. However, we fell a bit behind schedule because the setup of the ORM library took longer than anticipated and we were unable write any DAOs during this sprint, making it impossible to serve data from the database to the client. We also decided to switch component library, since our first choice turned out not to be the ideal library for our project.

The retrospective showed that we had somewhat mixed feelings about the result of this sprint. We were happy with the way the frontend was developing, but we were disappointed with the fact that work on the backend had been delayed and that we were unable to finish all tasks. Some team members also felt that they did not really know what others were doing, but we were happy with the fact that everyone was working and doing their part.

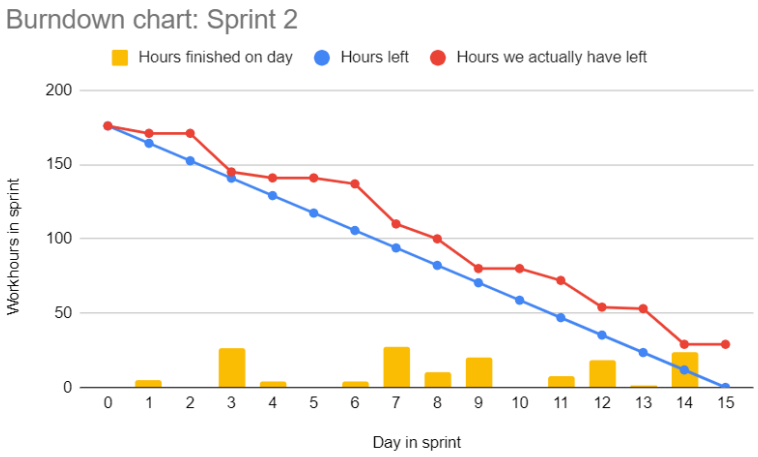


Figure 4.14: Burndown chart from second sprint

Third sprint

During this sprint we were able to start connecting the frontend and the backend. The frontend layout was updated and new features were added. The matching endpoints for these features were created in the backend. At the end of the sprint it was a mostly dynamic application in contrast to the more static proof-of-concept app we had created before. We created a new set of test data and started to create backend tests. One of the team members had an exam and were therefore unable to do much work on the thesis.

The results from the retrospective showed that the team had a positive attitude regarding this sprint. We were still not quite satisfied with the level of communication and planning, as we were unable to finish planning the interviews, and found that we had some misunderstandings. However, we were satisfied that we had been able to connect the client to the server, and felt that we were on schedule.

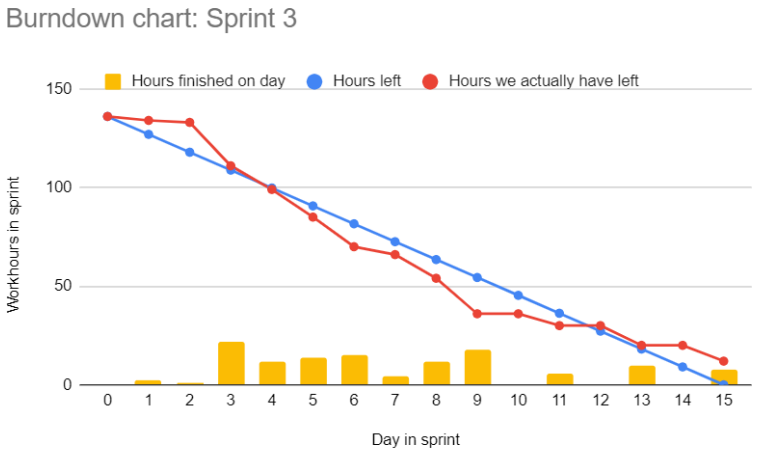


Figure 4.15: Burndown chart from third sprint

Fourth sprint

This sprint started a week after the last sprint as we had an exam that we needed to study for. It was also cut in half by the Easter break, as the team wanted to take some time of. The focus during this sprint was to continue the progress from the third sprint. New features were added to the application and we held interviews with dumpster divers. However, since we knew we would have a break in the middle of the sprint, we did not plan as much work as in the previous sprints. The retrospective showed that we were still struggling a bit with misunderstandings and communication issues. However, we were satisfied with the fact that we were able to finish all the interviews, add input validation and connect almost all frontend features to their endpoints.

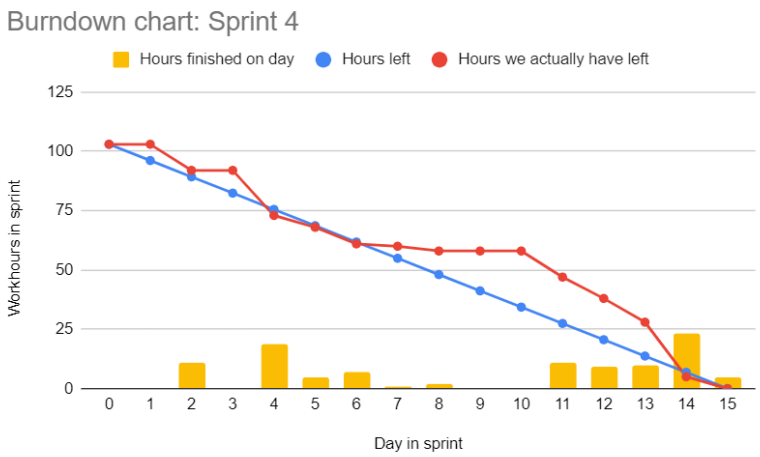


Figure 4.16: Burndown chart from fourth sprint

Fifth sprint

This was our second last sprint and our last sprint before our "feature freeze". Therefore, the main goal was to have a functional application where all features were implemented and working. We also wanted to start writing more on the documentation and these work hours are not in the sprint backlog.

The retrospective showed that overall, the group was satisfied with our progress and the progress plan. We were able to finish all the tasks on the sprint backlog, and our application had all the features we planned it to have for this sprint. The report was starting to take shape, and while some members thought that they could have spent more time on it, we were satisfied with the progress.

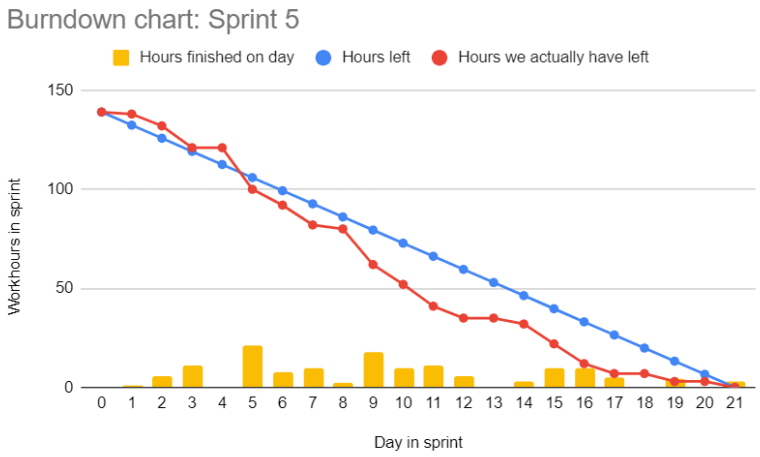


Figure 4.17: Burndown chart from fifth sprint

Sixth sprint

The last sprint was dedicated to writing documentation and fixing some minor bugs. Therefore it was not practical or logical to use a burndown chart.

4.3.2 Work distribution

For time sheets and weekly status reports, see our project manual [51].

Tore Bergebakken

Total work hours: **494.5**

Tore worked almost as much on the frontend as on the backend, and contributed heavily to planning and documenting, as shown in Figure 4.18.

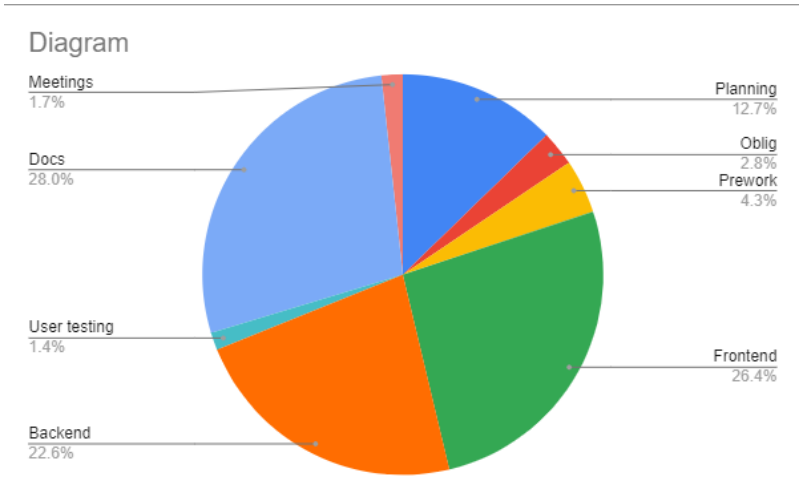


Figure 4.18: Work distribution for Tore

Jon Åby Bergquist

Total work hours: **468**

Jon mostly worked on the backend, and contributed to the planning and documentation, as shown in Figure 4.19.

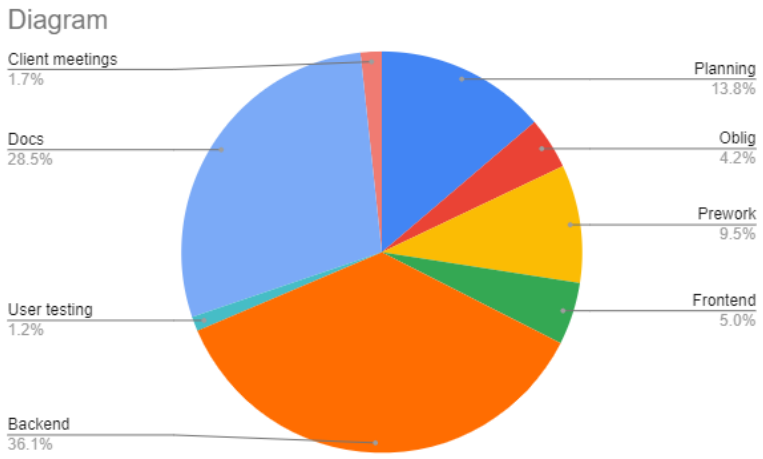


Figure 4.19: Work distribution for Jon

Helene Yuee Jonson

Total work hours: **530.5**

Helene worked mostly on frontend and contributed much to the planing and the documentation, as shown in Figure 4.20.

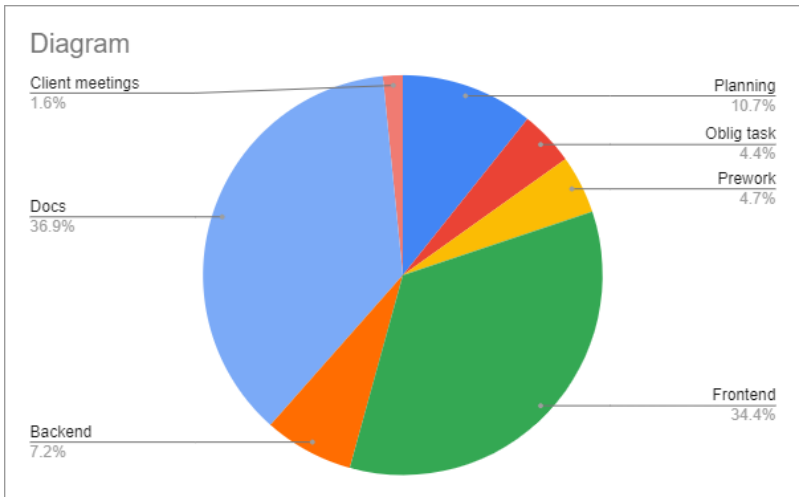


Figure 4.20: Work distribution for Helene

Chapter 5

Discussion

In this section we discuss our results we wrote about in chapter 4. If we are satisfied with the results, we discuss what we think we did right. If we are not satisfied, we discuss what was missing, what we did wrong and what improvements we could have made. We also look at the results relative to our goals at the beginning of the process. If we ended up changing the goals, we discuss why we made those choices, and how it affected the process and the end result.

5.1 Discussion of scientific results

5.1.1 User testing

Organizing user tests early seems to have affected our development process beneficially, since we became aware of problems that otherwise would have taken much longer to discover. However, most of the problems we found were rather easy to fix and could have been taken care of when they eventually were discovered — the main benefit was that the tests steered our design in better directions.

The late test was useful because it confirmed that we had done quite a few things right, and highlighted improvements we were already thinking of making. It also revealed a bug that had appeared a few days before the test, without being discovered.

5.1.2 Interviews

Many of the interviewees would only use the app if there was sufficient information available. They also felt the information they provided should be useful for other people. This requires an active user base. Therefore, our app is affected by the network effect, which states that the benefits of using the app (e.g. getting information) increase as more people use it, and the drawbacks (e.g. not enough information, forced to add more) decrease [52]. However, there are stronger negative network externalities than most apps, because there are not just things

like longer response times that increase the drawbacks with more users; the dumpsters and their contents are in limited supply, which means it does not take many users in a given area before it is almost impossible to actually go dumpster diving, unless you plan very carefully. Given the nature of the supply of trash — it is generally undesirable to produce — the lack of supply will always be likely to be a massive restriction on how many users the app can have within a geographical area.

It is possible that some of the interviewees did not look up information on dumpsters as the information was not available. The only real tool we found in our area was the "Dumpster Diving & Foodsharing Trondheim" group on Facebook, where people shared only a limited of information about specific dumpsters. As highlighted by some of the answers, it seemed like most divers just go to the same dumpsters repeatedly. Eventually, with our product gaining traction and giving dumpster divers access to a large amount of information about nearby dumpsters, this could change, precisely due to the network effect. People could start diving in different dumpsters that they did not know from before.

Most of the dumpster divers we interviewed said that privacy was important to them, and that they thought it was important for other dumpster divers as well. One can imagine that there is still the fear of breaking the law and getting punished. This could be one of the reasons it is important that their identity stays secret, as well as the fact that they might not want others to know that they do it.

A lot of the feedback from the users on new features *could* get incorporated in the future, though due to a lack of time we were unable to handle them in this bachelor project. However, we did implement a visitor registration feature, and the features we *did* have were considered useful by the dumpster divers we interviewed.

5.2 Discussion of systemic results

5.2.1 Functionality of the application

We were quite satisfied with the final product. As our project was to create an application and find solutions while we created it, we focused a lot on developing a working product that later could be published. While there is room for improvements, the application has all the main functionalities that were determined from the original assignment and the meeting with our supervisor. Overall, this has resulted in a functional and working application that is not to far away from the deployment stage. There are some solutions in the application that could be optimized, but overall it works as intended.

Criteria we reached

At the beginning of the process we created user stories with acceptance criteria. The criteria that were reached were the ones that were most essential for the functionality and the users' ability to share information with each other. For all user stories and acceptance criteria, see our requirements documentation in appendix B.

During the creation of the application we started with the criteria and features that would enable the users to receive, add and edit information about dumpsters. As this was the main goal of the app, this felt like the natural place to start. The natural second stage was to implement the features and criteria that would let users add more detailed information and somewhat moderate the app themselves. This was a little tricky since we, as stated by our main research question, had to find a balance between anonymity and the ability to limit system abuse.

Criteria we were unable to complete because we ran out of time

- I can report an image

As any user could upload an image to the system, it should be possible for other users to report images as irrelevant or inappropriate. The image server was created fairly late in the process, and we ran out of time for this report system.

Criteria that could be completed, but were not prioritized as they were not among the main features, but small improvements of existing features

- I can sort the list by rating, food quality and other measurements
- I can change my location temporarily (maybe just in the map view)
- Items that are close to expiring are highlighted/moved up the list
- Items that are long past the expiry date, are moved lower on the list

These goals were not prioritized as they were not important for the overall experience of the app. They could have improved the overall UX, but would not have a significant impact and was therefore not prioritized in favor of more relevant features.

Goals that were dropped because we changed how we wanted the application to work

- Fake dumpsters are removed or archived when enough people report them
- When enough people report it, the image is removed

These goals were set with the intention that the system should operate without moderators. However, after talking with our product owner/supervisor, we agreed that this would not necessarily be the case. As it is, these reports are just stored in the database and some sort of solution should be implemented later. A possible solution would be an admin who looks through the reports and acts on those that are valid. With the present solution, users must have direct access to the database, which is not a very user friendly or secure solution.

User ID

The user ID system was developed because we desired a system that did not contain any direct or indirect identifiers. We wanted to avoid a system with lacking security, so a simple auto-incremented user ID was undesirable on it's own. A UUID might have been more secure, but is not recommended for that purpose and can be difficult to remember. This is why we implemented a solution similar to seed phrases in cryptocurrency wallets. We chose 6 words instead of 12 like crypto wallets normally use. This gives us $7776^6 = 2.21073919721 \cdot 10^{23}$ unique combinations of usernames. The words are combined with a numeric ID during authentication, making brute forcing the system more difficult. We think this is a sufficient level of security. The consequences if your account is breached are less severe with our app than with crypto wallets, and it is likely that users would find 12 words tedious to remember.

Problems

As mentioned in the previous chapter, we ended up *not* switching to a map library that did not use Google Maps. After looking at the library's code, it does not seem to provide information that directly states that *our* app was the reason for the API call, but it does not provide a way to turn off telemetry either. According to Google's privacy policy, they only store location data when the user keeps the option enabled, but when data *is* stored, it will be shared with advertisement companies and Google's affiliations [53]. Therefore, Google Maps seems less than ideal for our use case, but somewhat viable for people who do not care *as* much for privacy, or make sure to disable Google's location tracking.

Another significant issue we experienced, was that we could not get the app to run in standalone mode, only through Expo's development mode. We did not *expect* to run into any issues with this, and thus saw it as completely unnecessary to test it before the final four weeks of the project. A white screen appeared every time we installed and opened the app, without any useful error reports. We spent hours debugging and concluded that we did not have time for this.

5.2.2 Non-functional goals

Usability

As detailed in the results, our product has properties that should accommodate color blind or visually impaired users. These measures should be sufficient, as our app is not likely to be used by people who are unfit for dumpster diving in the first place.

Reliability

We have not tested our product's reliability thoroughly. Issues might crop up in production — that is almost a certainty. However, we're reasonably sure that our servers should not *crash*, but rather report an error whenever something strange happens, since we always make sure to pass errors to a global error handler in our framework.

Performance

Without any large-scale stress testing, we cannot say for sure how performant our server is, though we *do* know that it has not shown signs of being severely slow during our testing. As this is shrouded in uncertainty, it is difficult to conclude with anything here, without further testing of the system.

On the other hand, the mobile app might be a simpler case. It has performed well, without any severe slowdowns or lag, though there seems to be a performance issue with Formik, which validates input, in screens with multiple input fields.

Security

With the exception of missing a CAPTCHA-esque system that could prevent people from making bots to abuse the system, and the unfortunate lack of automated frontend tests, our product seems rather secure. As detailed in appendix C.7, we have taken several measures to ascertain the API's security in particular. The client does not do much besides fetching, displaying and sending data — with the exception of picture uploads from the user's camera roll. Thus, it does not seem to pose a risk to the user's phone.

Since our servers do not store things like IP addresses or the locations of our users, our goal of storing as little data as possible seems to have been reached. However, we may have trod wrong in regards to the visit feature, since we store the ID of the users who have visited a particular dumpster, with a time stamp as well.

5.3 Discussion of administrative results

5.3.1 Work process

The bachelor thesis was our main focus throughout this semester. However, during the first months we also had another subject, which had to be prioritized. Some of us also had electives and other exams during this period. Therefore, we decided to try to meet up and work together at campus two days a week. During the other days we would work from home. As we were only three members, it was relatively easy to communicate well over the internet. By splitting our week, we had enough contact with each other to keep track of each other's work while we had time to prioritize our own workload and focus on the other subjects. Due to the continued outbreak of Covid-19, there were some weeks we were unable to meet up at all because society was mostly closed. However, we were still able to work well together and often discussed the project with each other multiple times a day. During the process we set up regular meetings with our supervisor and all these meetings were digital as well.

5.3.2 Method

Using our modified version of scrum worked well throughout the project. One thing we did not end up doing was regular stand-up meetings at the beginning of each day. As we had different working schedules and lectures to attend to, we often started working at different times of the day. A stand-up meeting at the beginning of the workday would therefore not work. Instead of the stand-up meeting we were available through the Discord system and had small conversations throughout the day as the work progressed. Holding stand-up meetings *might* have helped to avoid some of the misunderstandings we had throughout the project, but we think the solution we had worked well enough. We also tried to use a scrum board but found that it did nothing except take up time. We had to make our scrum board digital, which made the board less accessible. The result was that we often forgot about it and it no longer had the effect of displaying the progress clearly. Instead, the sprint backlog had a more active role. We had created our sprint backlog in the same spreadsheet as our timesheets, and it was directly connected to the burndown chart. Since it was more easily accessible, we selected tasks and registered our names directly *there* instead of doing so on the scrum board as well.

Since we had to split our days between physical meetups and working from home, we had to make sure that we had good ways to communicate with each other. As we were only three members in the team, we were able to keep track of the others' progress and we could easily contact the others throughout the day if we had something we needed to discuss. However, we had some situations where we had some misunderstandings and different team members ended up with different perceptions about the feature we wanted to create. There were also situations where two team members had been discussing something and the third team member had worked on other things in the belief that it was not related to their work, only for this to create a new misunderstanding. In other cases, some team members had written documentation that proposed something different than what other team members thought but was not read through carefully enough to catch the misunderstanding, or the members interpreted the document differently. This led to situations where we created a feature only to realize that other team members had a different perception of how the feature should work. Then we had to discuss it, come to an agreement, and often the feature had to be modified.

As we worked more on the project, we started to learn each other's working habits and schedules. This made us able to better estimate the team's working ability for each week and sprint. After some time, we were better able to estimate our workload. As a result, in the later sprints we finished our planed work and did not have to delay it to a later sprint.

After the interviews were held, we decided to add some new features to the application. We originally intended to go through the results of the interviews together with our supervisor in a sprint review, but he suggested that we should write a summary of the interviews instead. The first feature we developed in response to the interviews, was worked on quite soon after the meeting, without much time for a proper discussion of the interview results between the team members.

5.3.3 Teamwork

The team worked well together. We had all been in the same circle of friends, and some of us had worked together on one or more projects previously. One of our main focus points in the beginning was that we were all equally committed to the thesis and were willing to put enough time and effort into the project. If different team members had different ideas about the end goal, some might have felt betrayed or let down since they would not have been able to accomplish what they wanted, or felt that they had to do all the work. This in turn would have influenced the group dynamic and the end product would have suffered. Since we were aware of this from the beginning, we had an open dialog about the topic and therefore avoided the problem altogether.

The days we were working together at campus gave us the opportunity to discuss more freely with each other and ask for help if someone had an issue. When we worked from home, we had to share our screen and try to direct each other as best as possible. On campus we could just turn our screen to the other person and have them point to it.

The days we were working from home gave us the opportunity to manage our own schedules. As mentioned before, we had some other subjects that we had to focus on during the semester, and this gave us the opportunity to prioritize and work when we had time. We trusted that the other members of the group would work enough hours to reach the end goal within the time limits.

5.4 Anonymity and abuse

Anonymity and abuse do effect each other rather closely. Anonymous services typically have more spam, mostly due to people feeling more comfortable spreading abuse when they aren't identifiable [54]. Another issue is that the normal email and password verification system that sites use block out a lot of potential unwanted users. It is possible our system will avoid some abusers because it's generally not direct interaction with people, instead you interact with data, which could make abusers.

5.4.1 Anonymity

We are satisfied with the anonymous nature of our application. The user ID system ensures that no personal data is collected *directly*. Some things could be improved, like removal of metadata and resizing of images to avoid fingerprinting. It would also be beneficial to change what map library we use. Adding Tor support would be beneficial for those who can deal with its complexity, but is not something we planned on implementing during the bachelor project.

5.4.2 Abuse

It is very difficult to know how our app will actually be used and potentially abused without releasing it and collecting data. We have implemented various measures against abuse like

optional filtering of downvoted comments, our revision system, which allows users to go back to a previous version of a dumpster, and spam limiting functions like rate limiting. This would almost certainly not be sufficient if released to the public. That is why adding a CAPTCHA system should be highly prioritized in the future. We would also change the report functionality depending on what feedback we get. Currently you can report a dumpster, but the only thing that happens is that a database entry is created. An admin would have to remove the dumpster from the database manually, which is not really what we wanted. This system should ideally be changed, we *could* replace it with an upvoting system similar to the comments, or make the system remove a dumpster if a certain amount of reports are made within a given time frame, though the problem with these user moderated solutions is that they can also be abused. There is also the possibility of making a proper moderation system that would *not* require direct access to the database. However, we would prefer a user moderated system for various reasons. One is that dumpsters generally are only relevant in a small geographical area, so finding an expert for each area does not seem feasible to us. It also seems fitting that an app meant for communication between individuals in an area is run by everyone involved and not divided into a hierarchy of administrative power.

Something like a blacklist of banned words or naive Bayesian spam filtering would not be a bad idea, but due to the open source nature of our project, it would be of limited use, because it could be easy to circumvent.

It is possible that our app will be targeted by spammers and abusers of a different nature than most other anonymous services. Some may be dumpster divers that want to keep their territory to themselves, and thus spread misinformation in our app in order to keep others away.

5.5 System perspective

This application has both environmental and economic benefits. The environmental benefit is that the use of this app can reduce food waste. Stores often throw away food that has expired or is about to expire, even if the food is still edible. The process of creating, packaging, distributing and then destroying food that has not been bought is a huge waste of energy and resources. During our research, we found that most of our interview subjects had environmental effects as their primary reason for dumpster diving. By sharing information, our application can help dumpster divers save more food and reduce food waste.

The economic benefits are mostly for those who dumpster dive as they do not have to pay for the food they find. With our app, it would be easier to locate a dumpster with exactly the content you want. If you find all or most of what you need, you don’t have to spend money on it. There might be money to save for the stores as well. If dumpster divers take things from their dumpsters, the trash does not need to be emptied as often, since it takes longer before the dumpsters are full.

A downside to the creation and eventual use of the application is that dumpster diving is more profitable if few people do it. If a lot of people start dumpster diving, it will be less to find for each person. Another issue is the problem that more dumpster divers will inevitably lead

to some dumpster divers that does not clean up after themselves or even break rules. This would lead to a negative attitude towards dumpster divers and could make more stores lock their dumpsters.

5.6 Ethics

Dumpster diving is a legal gray area in many countries. Its status varies based on where you live and the local laws. In Norway, there is a growing focus on dumpster diving and eliminating food waste. In Norway, thrown away goods are owned by the person, store, or company the dumpster belongs to. Therefore, taking the content could be classified as stealing. However, dumpster diving has often been discussed in mainstream media, without threats of prosecution. Since it reduces food waste it is often said that dumpster divers have the ethics on their side, and many consider it to be okay. On the other hand, if a dumpster diver breaks a lock to enter a dumpster or climbs a fence, it is considered forced entry and trespassing, which is more frowned upon and considered illegal.

Taking food that would otherwise be destroyed, does not seem like a particularly bad offense. It is understandable that store owners would want their surroundings clean, and thus want to prevent people from rummaging in the trash without cleaning up after themselves. The question still stands: do these kinds of arguments really weigh up for the extreme wastefulness exercised by most first-world countries today?

Displaying information whether the dumpster is locked or not, and whether the store has a negative view on dumpster divers could be seen as a way of enabling crime. We do tell people not to pick locks on dumpsters that are locked, but this is the extent of our deterrence. There is potential that our app will end up playing a part in somebody effectively breaking the law. However, the users themselves are responsible for their own actions. When an app is published, it is available globally. That does not mean that what the app is meant for is legal in all countries. An example of this is apps where people can rate and find information about the best places to buy cannabis. Smoking cannabis is not illegal everywhere and it is the local laws where the users are located that determine the legality of the action or the distribution of information.

Chapter 6

Conclusion and future work

We started with an idea of a system without any user accounts, where users moderate the content themselves. The end result *has* something very similar to user accounts, but in keeping with the original goal of letting our users be anonymous, the "accounts" consist of random identifiers tied to *some* data like comments and ratings. We found it necessary to prevent users from adding an arbitrary number of ratings of a dumpster, in order to prevent them from inflating the average rating in one direction or the other. We also decided to prevent users from deleting other people's comments, since the comments' rating would determine if they would show up at all. Lastly, each registered visit to a dumpster is tied to the anonymous identifier. It is *theoretically possible* that these connections could be used to identify the user, though only with a data set that already contains some location data. Some data is only stored locally, like the last time the user visited a dumpster, the user's rating of a dumpster, and whether the user has upvoted a particular comment.

Regarding abuse control, we saved edits of dumpsters in a revision history and made it easy to switch to different versions of dumpster data. Comments were moderated with up- and downvotes, like Reddit, and we allowed users to *delete* their own comments. We made dumpster contents more freely moderated, with the option to delete any content entry, since this section was expected to change *very* often. To handle situations where a dumpster that *did not even exist* was added, we made it possible to report dumpsters. Because we did not end up adding a CAPTCHA, we had to leave some things up to admins for the time being, like deletion of dumpsters after a certain number of people have reported them, which we intended to be automatic. Additionally, admins need access to the database in order to perform their tasks, which is not a user-friendly or secure solution. We did not prioritize creating an admin interface when this was not part of what we intended.

Some of our interview subjects suggested additional features, but all of them seemed to find those we had implemented to be useful. Among the suggested features were a system that registered the amount of visitors to a dumpster in a given time frame — this was implemented. We did not get around to implementing features like notifications of new contents in marked dumpster, warnings about possible food poisoning or any way of helping people meet up to go dumpster diving.

All things considered, we *do* have a solution that seems viable for protecting anonymity and preventing some abuse, and we are satisfied with our product. Since we did not get the opportunity to test our system in a real-world community, any conclusions we draw might turn out to be slightly different from reality — and there will be a need for further research. We do not have a clear-cut answer to the main research question, but we *have* outlined a few ways to handle the balance it refers to. Additionally, while there *are* cases where measures for preventing abuse may conflict with protecting the users' anonymity, these concerns do not seem to be mutually exclusive.

6.1 Future work

If this application should be developed further, the natural step would be to make it more secure. Particularly in the area of privacy and anonymity, and in preventing bots and users from abusing the system. The two following paragraphs deal with some features directly related to this. Additionally, in order to improve user privacy and anonymity, the map library used in the app should be changed to a different one that does not rely on Google Maps.

A CAPTCHA system was planned, but not implemented. The most likely candidate was hCaptcha. If further developments are made, adding a CAPTCHA should be prioritized. A CAPTCHA solution's primary purpose would be to prevent bots from accessing our endpoints and spamming info like dumpsters or comments. It is probable that the CAPTCHA check would be implemented before the user authentication endpoint if it's an instance based authentication, if it's a time based authentication where you do it once and it lasts X amount of time, it would likely be added to all post requests.

Tor support was an optional idea in the original task list written by our client. It would be a big leap in the direction of anonymity. The idea is to let the server be a Tor hidden service. This would not affect the users' experience with the application. The only thing is that the app might be a bit slower, but not by much, and the added anonymity would outweigh that.

There currently is no solution for reporting and removing images posted by other users. Currently users can only remove their own images. The ability to report images would be an important step towards preventing spam, and should be prioritized for future work.

Currently, an image is just verified as valid. Uploaded images are not altered in any way. All metadata of an image should ideally be deleted when it is uploaded, because it contains information that could potentially be used to identify the user and trace their steps. Automatically resizing images to a standard size would also be beneficial, the image size could be used to identify which phone is used, and standardization could potentially make things look neater and load faster.

As our focus was creating and adding features to the application, the system has not been properly stress tested. While we have not found any issues during our small scale testing, and have tried to create a system that would handle a significant load, it would be smart to do large scale stress tests to reveal any potential performance problems.

If we take a less technical approach, it would be interesting to see how the application is used

in the real world, and create a plan for possible new features and updates based on feedback from our users.

During our interviews we got some feedback on features that users might want to have in the application. One additional feature could be the ability to mark a dumpster as a favorite and then have it appear on top of the list of dumpsters, and/or in a list of its own. Another feature that was suggested, was push notifications for new contents in the user's favorite dumpsters and other important events.

Bibliography

[1] Food and Agriculture Organization of the United Nations. *Food Loss and Food Waste*. URL: http://www.fao.org/food-loss-and-food-waste/flw-data). (retrieved 20.05.2021).

[2] United Nations. *Food systems account for over one-third of global greenhouse gas emissions*. URL: https://news.un.org/en/story/2021/03/1086822. (retrieved 20.05.2021).

[3] Wikipedia writers. *Dumpster diving*. URL: https://en.wikipedia.org/wiki/Dumpster_diving#Legal_status. (retrieved 20.05.2021).

[4] United Nations. *Food*. URL: https://www.un.org/en/global-issues/food. (retrieved 20.05.2021).

[5] Wikipedia. *Throw-away society*. URL: https://en.wikipedia.org/wiki/Throw-away_society. (retrieved 20.05.2021).

[6] Wikipedia. *Consumerism*. URL: https://en.wikipedia.org/wiki/Consumerism. (retrieved 20.05.2021).

[7] Svein Inge Meland. *Butikker kaster enorme mengder mat. To studenter mener nye strekkoder kan sette en stopper for alt matsvinnet*. URL: https://forskning.no/mat-ntnu-partner/butikker-kaster-enorme-mengder-mat-to-studenter-mener-nye-strekkoder-kan-sette-en-stopper-for-alt-matsvinnet/1299118. (retrieved 16.05.2021).

[8] Matsentralen Norge. *Matsentralen Norge – Hva er matsvinn?* URL: https://www.matsentralen.no/matsvinn. (retrieved 19.05.2021).

[9] Food and Agriculture Organization of the United Nations. *Food wastage footprint. Impacts on natural resources.* URL: http://www.fao.org/3/i3347e/i3347e.pdf. (retrieved 19.05.2021).

[10] Turjenter.no. "Lev mer miljøvennlig – utnytt maten!" Norsk. In: (Nov. 29, 2019). URL: https://turjenter.no/lev-mer-miljovennlig-utnytt-maten/. (retrieved 20.05.2021).

[11] Synne Mauseth and Line Pevik. "Vilde (23) bruker bare 3600 kroner i året på mat - resten finner hun i søpla." Norsk. In: (Jan. 18, 2016). URL: https://trd.by/livsstil/2016/01/14/Vilde-23-bruker-bare-3600-kroner-i-%C3%A5ret-p%C3%A5-mat-resten-finner-hun-i-s%C3%B8pla-12015600.ece. (retrieved 20.05.2021).

[12] Synne Mauseth and Line Pevik. “Slik klarte Une (28) å ikke bruke ei eneste krone på mat i løpet av én måned.” Norsk. In: (Sept. 4, 2020). URL: https://trd.by/livsstil/2019/10/11/Slik-klarte-Une-28-%C3%A5-ikke-bruke-ei-eneste-krone-p%C3%A5-mat-i-l%C3%B8pet-av-%C3%A9n-m%C3%A5ned-20132001.ece. (retrieved 20.05.2021).

[13] Vebjørn Kallelid. “Kveldsdukkert i matsøpla.” Norsk. In: (Nov. 5, 2020). URL: https://underdusken.no/bendik-stenberg-bergekultur-dumpster-diving/kveldsdukkert-i-matsopla/279559. (retrieved 20.05.2021).

[14] Cambridge Dictionary. *API*. URL: https://dictionary.cambridge.org/dictionary/english/api. (retrieved 19.05.2021).

[15] Carnegie Mellon University. *CAPTCHA: Telling Humans and Computers Apart Automatically*. URL: http://captcha.net/. (retrieved 19.05.2021).

[16] MAX REHKOPF. *What is Continuous Integration?* URL: https://www.atlassian.com/continuous-delivery/continuous-integration#:~:text=Continuous%20integration%20(CI)%20is%20the,builds%20and%20tests%20then%20run.. (retrieved 20.05.2021).

[17] Thomas Anderson. *Operating systems : principles and practice*. eng. 2nd ed. West Lake Hills, TX: Recursive Books, 2014. ISBN: 978-0-9856735-2-9.

[18] Mozilla staff. *JavaScript*. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript. (retrieved 20.05.2021).

[19] ECMA international. *Introducing JSON*. URL: https://www.json.org/json-en.html. (retrieved 20.05.2021).

[20] Mike Moffatt. *What Are Network Externalities?* URL: https://www.thoughtco.com/introduction-to-network-externalities-1146145#:~:text=Negative%20network%20externalities%20exist%20if,are%20using%20it%20as%20well.. (retrieved 20.05.2021).

[21] Full Stack Python. *Object-relational Mappers (ORMs)*. URL: https://www.fullstackpython.com/object-relational-mappers-orms.html. (retrieved 19.05.2021).

[22] Codecademy staff. *What is a Relational Database Management System?* URL: https://www.codecademy.com/articles/what-is-rdbms-sql#:~:text=A%20relational%20database%20is%20a,database%20is%20organized%20into%20tables.. (retrieved 04.05.2021).

[23] restfulapi.net. *What is REST*. URL: https://restfulapi.net/. (retrieved 19.05.2021).

[24] Wikipedia. *Representational state transfer*. URL: https://en.wikipedia.org/wiki/Representational_state_transfer. (retrieved 19.05.2021).

[25] Michelle Knight. *What Is Structured Query Language (SQL)?* URL: https://www.dataversity.net/structured-query-language-sql/#:~:text=Structured%20Query%20Language%20or%20SQL,or%20attributes%2C%20categorized%20into%20columns%2C. (retrieved 20.05.2021).

[26] Tor staff. *Browse Privately. Explore Freely.* URL: https://www.torproject.org/about/history/. (retrieved 20.05.2021).

[27] Microsoft. *TypeScript for the New Programmer*. URL: https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html. (retrieved 19.05.2021).

[28] Oxford Learner's Dictionaries staff. *user interface*. URL: https://www.oxfordlearnersdictionaries.com/definition/english/user-interface#:~:text=(computing),%2Dto%2Duse%20user%20interface.. (retrieved 20.05.2021).

[29] Oxford Learner's Dictionaries staff. *user experience*. URL: https://www.oxfordlearnersdictionaries.com/definition/english/user-experience#:~:text=%2F%CB%8Cju%CB%90z%C9%99r%20%C9%AAk%CB%88sp%C9%AAri%C9%99ns%2F,pleasant%20it%20is%20to%20use. (retrieved 20.05.2021).

[30] Oxford Learner's Dictionaries staff. *virtual machine*. URL: https://www.oxfordlearnersdictionaries.com/definition/english/virtual-machine#:~:text=%2F%CB%8Cv%C9%9C%CB%90t%CA%83u%C9%99l%20m%C9%99%CB%88%CA%83i%CB%90n%2F,operating%20environmentTopics%20Computersc2. (retrieved 20.05.2021).

[31] Jim Highsmith. *The Agile Manifesto*. URL: https://agilemanifesto.org/. (retrieved 12.05.2021).

[32] Pete Deemer et al. *The scrum primer*. 2009.

[33] Grethe Sandstrak. "SCRUM." University Lecture. 2019.

[34] Henrik Kniberg. *Scrum and XP from the trenches : how we do scrum*. eng. InfoQ enterprise software development series. S.l.: C4Media, 2007. ISBN: 9781430322641.

[35] Majid Rouhani. "Universell utforming." University Lecture. 2019.

[36] Seattle University. *Anonymity, Confidentiality, & Privacy*. URL: https://www.seattleu.edu/irb/guidance--faq/anonymity-privacy--confidentiality/. (retrieved 14.05.2021).

[37] Berkley Univeristy of California Office for Protection of Human Subjects. *HIPAA PHI: Definition of PHI and List of 18 Identifiers*. URL: https://cphs.berkeley.edu/hipaa/hipaa18.html. (retrieved 13.05.2021).

[38] J.M. Porup. *8 steps to being (almost) completely anonymous online*. URL: https://www.csoonline.com/article/2975193/9-steps-completely-anonymous-online.html. (retrieved 14.05.2021).

[39] Michael Greene. *The password reuse problem is a ticking time bomb*. URL: https://www.helpnetsecurity.com/2019/11/12/password-reuse-problem/. (retrieved 20.05.2021).

[40] Sjoerd Pijnappel. *How to Manage Abusive Content in Your Online Community*. URL: https://www.getopensocial.com/blog/community-management/manage-abusive-content/. (retrieved 20.05.2021).

[41] University of Cambridge Faculty of Mathematics. *Spam and how to deal with it*. URL: https://www.maths.cam.ac.uk/computing/email/junk/intro. (retrieved 20.05.2021).

[42] Wikipedia writers. *Anti-spam techniques*. URL: https://en.wikipedia.org/wiki/Anti-spam_techniques. (retrieved 14.05.2021).

[43] Wikipedia users. *Anonymous social media*. URL: https://en.wikipedia.org/wiki/Anonymous_social_media. (retrieved 18.05.2021).

[44] Jeanette Bergquist. Private communication. Jurist. Apr. 30, 2021.

[45] Justis- og beredskapsdepartementet. *Lov om straff (straffeloven)*. URL: https://lovdata.no/dokument/NL/lov/2005-05-20-28/*#*. (retrieved 29.04.2021).

[46] Daniella Slabinski. “Dereliksjonslæren: den enes død, den andres brød. Om retten til å ta ting andre har kastet.” Norwegian. MA thesis. Universitetet i Bergen, June 1, 2015. URL: https://bora.uib.no/bora-xmlui/handle/1956/11501. (retrieved 05.05.2021).

[47] Stefan Hanenberg et al. “An empirical study on the impact of static typing on software maintainability.” eng. In: *Empirical software engineering : an international journal* 19.5 (2014), pp. 1335–1382. ISSN: 1382-3256.

[48] Liran Tal. *Sequelize ORM npm library found vulnerable to SQL Injection attacks*. URL: https://snyk.io/blog/sequelize-orm-npm-library-found-vulnerable-to-sql-injection-attacks/. (retrieved 04.05.2021).

[49] Svetlin Nakov. *PBKDF2*. URL: https://cryptobook.nakov.com/mac-and-key-derivation/pbkdf2. (retrieved 20.05.2021).

[50] Joseph Bonneau. *Deep Dive: EFF’s New Wordlists for Random Passphrases*. URL: https://www.eff.org/deeplinks/2016/07/new-wordlists-random-passphrases. (retrieved 20.05.2021).

[51] Tore Bergebakken, Jon Åby Bergquist, and Helene Yuee Jonson. *Project manual - Dumpster Fire*. 2021.

[52] Tim Stobierski. *WHAT ARE NETWORK EFFECTS?* URL: https://online.hbs.edu/blog/post/what-are-network-effects. (retrieved 04.05.2021).

[53] Google. *Privacy Policy – Privacy & Terms – Google*. URL: https://policies.google.com/privacy. (retrieved 11.05.2021).

[54] Joe Dawson. *Who Is That? The Study of Anonymity and Behavior*. URL: https://www.psychologicalscience.org/observer/who-is-that-the-study-of-anonymity-and-behavior#:~:text=Behavioral%20studies%20on%20the%20role,values%20to%20guide%20individual%20behavior.. (retrieved 19.05.2021).

[55] Electronic Frontier Foundation. *Certbot*. URL: https://certbot.eff.org/. (retrieved 10.05.2021).

[56] Mozilla. *Mozilla SSL Configuration Generator*. URL: https://ssl-config.mozilla.org/#server=nginx&version=1.17.7&config=intermediate&openssl=1.1.1d&guideline=5.6. (retrieved 10.05.2021).

[57] Sequelize community. *Sequelize’s function for escaping values in SQL queries*. URL: https://github.com/sequelize/sequelize/blob/main/lib/sql-string.js. (retrieved 08.05.2021).

Appendices

Appendix A

Vision document

Audit history

Date	Version	Description	Author(s)
12.01.2021	0.1	Creation of document	Helene Y. Jonson
13.01.2021	0.2	First draft	Tore Bergebakken, Jon Åby Bergquist and Helene Y. Jonson
18.01.2021	0.5	Rough draft finished	Tore Bergebakken, Jon Åby Bergquist and Helene Y. Jonson
19.01.2021	1.0	Finishing touches	Tore Bergebakken and Jon Åby Bergquist
25.01.2021	1.1	Small changes	Tore Bergebakken and Helene Y. Jonson
18.05.2021	1.2	Grammar and layout fixes	Tore Bergebakken

A.1 Introduction

The purpose of this project is to create an app that makes dumpster diving easier, and this document outlines what we intend to achieve. The primary goal of the project is an app that allows communication between dumpster divers about dumpsters, while ensuring that their privacy is maintained.

A.2 Summary of problem and product

A.2.1 Problem summary

There is no current system that this project is building on, however there are some solutions to similar problems. Some dumpster divers use social medias like Facebook groups (link) for communicating with other dumpster divers, which does not focus on privacy and is not a specialized tool. This project aims to be a simpler and more efficient solution that also focuses more on security and privacy.

Problem with	distributing information about local garbage bins in an area; currently done through various divergent means
involves	dumpster divers.
As a result of this,	there is no centralized source of information on dumpsters and their related metadata.
A successful solution will	let divers share information (location, types of trash etc.) in a more organized and private fashion.

A.2.2 Product summary

Made for	dumpster divers
that	want to be able to see what's in dumpsters without visiting them all, and tell others what's in them.
The product	is a mobile app
that	allows people to see and add information on dumpsters so they can only search the ones they find relevant.
As opposed to	things like Facebook groups
our product	focuses on security, will not store personal data and is purely practical, not a social media

A.3 Overall description of stakeholders and users

A.3.1 Summary of stakeholders

Name	Description	Role during development
Donn Morrison	Lecturer that wants this app made	Product owner
Tore Bergebakken, Jon Åby Bergquist, Helene Yuee Jonson	Those who design the system and write the code – they get to keep the fruits of their labour	Developers

A.3.2 Summary of users

Name	Description	Role during development	Represented by
Dumpster divers	People who search dumpsters for food and other salvageable items	Interview subjects, end-user representatives	Donn Morrison and potential interview subjects

A.3.3 User environment

The product is an app that should be available for both android and iOS, and optionally for desktop computers. It will be designed to be used in whichever context the user desires, so the *physical* user environment is very broadly defined.

A.3.4 Summary of user needs

Need	Priority	Concerns	Today's solution	Proposed solution
Find dumpsters in the local area	High	Dumpster divers	Various means of localized information sharing	Let users enter their location, find and display dumpsters in an area around that position – on a map or as a list
See dumpster information/metadata	High	Dumpster divers	Gather information from various sources	Let the app display useful information about a dumpster
Share the location of a dumpster	High	Dumpster divers	Various means of giving directions	Let (registered) users place a marker on a map where a dumpster is
Add dumpster metadata	High	Dumpster divers	Again, all sorts of information sharing	Have (editable) fields for telling whether the dumpster is locked, when it tends to be emptied, etc. – and let users add things like photos, tags and food quality ratings
Review or post experiences about specific dumpsters	High	Dumpster divers	Posting experiences on miscellaneous social media	Let (registered users or any users) add reviews to a specific dumpster's entry
Share photos of dumpsters and their contents	High	Dumpster divers	Posting pictures to image sharing sites or other social media	Add or take pictures of dumpsters through the app
Protect the users' privacy	High	Dumpster divers	Depends on the method used for sharing information	Store and transmit as little identifiable information as possible, ensure no malicious third party can access *all* data in our database, etc.
Prevent spam and misinformation	Medium	Dumpster divers and maintainers	Same as above	Have (local) admins clean up content AND/OR let users rate dumpsters by factuality/quality and thus make the system handle the issue dynamically AND/OR some kind of rating for users, possibly based on non-infringing information like the device number for the phone

A.3.5 Alternatives to our product

Too Good to Go is an app that lets shops arrange controlled hand-outs of food that would otherwise be thrown away. This does *not* take care of food that actually ends up in dumpsters, and is not a viable solution for the problem our product aims to solve.

There are various Facebook groups like "Dumpster Diving & Foodsharing Trondheim" that focus on dumpster diving, where people may post information about dumpsters, but that does not seem to be the main purpose. Those groups are more about food sharing.

A.4 Product overview

A.4.1 Its role in the user environment

It will be a tool to make dumpster diving more efficient and less time consuming, while making sure that the users' privacy is not disturbed.

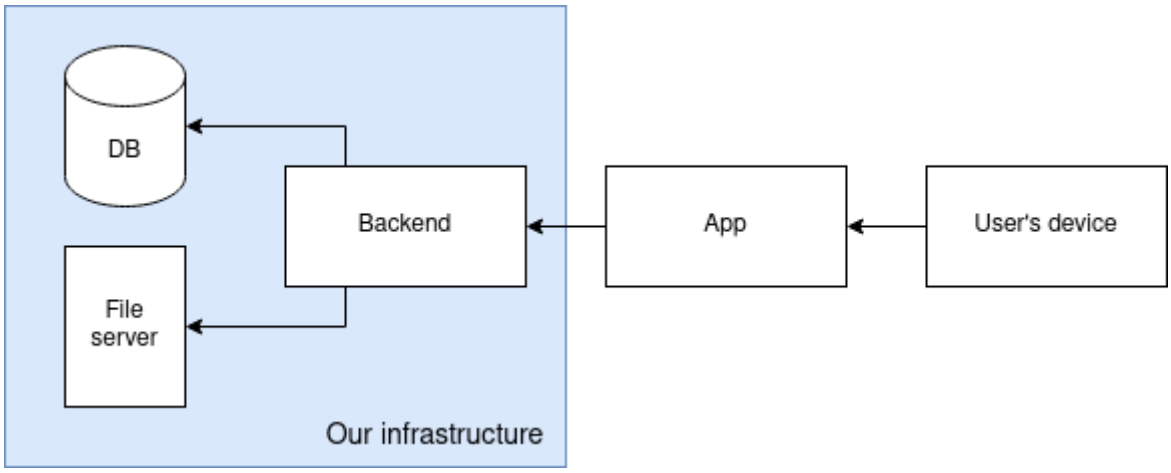


Figure A.1: Illustration of the product's architectural relation to a user's device

A.4.2 Assumptions and dependencies

We assume the following:

- There is an issue with perfectly edible food being thrown away
- Dumpster divers typically have access to smart phones
- There is no good solution that satisfies our criteria on the market today
- We will implement a solution that does not require a user system

Our product depends on these conditions:

- That future updates to either IOS or Android don't conflict with our code, if such a conflict were to arise, we would need a maintainer updating the app
- That the 3 layers (app, backend and database) communicate with each other without issues
- The only external dependency the app utilizes is OpenStreetMap (link)

A.5 Functional properties

1. The ability of for the user to add their location, will give suggestions (cities) based on letters written
2. Display a list with all dumpsters in the given area
3. Show the list as nodes on a map (position is part of the data)
4. Be able to choose any individual dumpster, either from the list view or from the map, and go to a page with its data
5. See the data for a specific dumpster. This will include information about whether it is locked or not, if it is a compressor, how dirty it is, its position, its emptying schedule and more.
6. Add which items are located inside the dumpster, and useful info like expiration dates and amounts
7. Upload images related to the dumpster or its contents
8. Add a new dumpster on the map, including any data
9. Update a specific dumpster's information
10. Update the products in a dumpster, change the amounts of the product that is available
11. Move the map to a new location to see the dumpsters in that area
12. The ability to choose whether to only see recent additions (following the emptying schedules or some predefined constant) to dumpsters or see all information.
13. Search for the products you want and only show the relevant dumpsters
14. Rate the information about a dumpster, so that others know if it's useful or not
15. Rate a user's reliability so that the information they post in the future can be blocked
16. Be able to filter out information from poorly rated users
17. Be able to filter out dumpsters that don't meet your criteria – filter by food quality and the like

A.6 Non-functional properties and other requirements

This section uses the *FURPS* model for software quality. (link)

A.6.1 Functionality

The product shall be usable for all the intended purposes, as described in sections A.5 and A.3.4.

A.6.2 Usability

The product shall be easy to comprehend and use for people with various disabilities or lack thereof.

A.6.3 Reliability

The backend should have a reasonable uptime (99%, ideally) and be able to avoid crashing under unexpected circumstances (invalid input and the like).

A.6.4 Performance

The backend should perform well under expected load, and perform decently under loads that exceed our expectations.

The frontend – that is, the mobile app – should be performant and run smoothly without being too taxing on the phone's hardware. This might be limited by our choice of framework.

A.6.5 Security

The product shall not store identifying information or other sensitive data that could be traced back to its users.

We will take particular care when it comes to transmitting and storing location data.

User accounts, if any, shall be protected with the usual measures – passwords shall be stored and transmitted securely.

Appendix B

Requirements documentation

Audit history

Date	Version	Description	Author
12.01.2021	0.1	Creation of document	Helene Y. Jonson
13.01.2021	0.2	First draft of user stories	Tore Bergebakken, Helene Y. Jonson
28.01.2021	1.0	Finish first draft	Tore Bergebakken, Jon Å. Bergquist, Helene Y. Jonson
03.05.2021	1.1	Update user stories and wireframes	Tore Bergebakken

B.1 Introduction

This document was made in connection with our bachelor thesis in Computer Engineering at the Norwegian University of Science and Technology. Our assignment was to create an application where dumpster divers can publish and view information about different dumpsters, with focus on security and anonymity. The purpose of this document is to describe the different technical and functional requirements for this application. The document contains different user stories created to document the intended use and functionality of the application. The domain model attempts to explain the problem area and how the application's data should be organized, while the prototypes show a starting point for our user interface.

B.2 User stories

B.2.1 Map View

As a user
I want to get an overview of dumpsters in a given area
so that I can see if there are any dumpsters near my location

Acceptance criteria:

- ☑ I can see dumpsters as markers on a map
- ☑ I can see my own position on the map
- ☑ I can easily change my location by dragging the map (disputable)
- ☑ I can apply filters to e.g. only see dumpsters that are *not* locked
- ☑ I can tap a dumpster marker to see some information about it

B.2.2 List View

As a user
I want to see a list of dumpsters in my area
so that I can see if there are dumpsters close to me that have contents I desire

Acceptance criteria:

- ☑ I can see a list of nearby dumpsters
- ☑ I can easily change my location
- ☐ I can sort the list by rating, food quality and other measurements
- ☑ I can apply filters to e.g. only see dumpsters that are *not* locked

B.2.3 Filtering

As a user
I want to be able to filter out dumpsters
so that I only see dumpsters that meet my requirements

Acceptance criteria:

- ☑ I can filter out locked dumpsters
- ☑ I can filter dumpsters by type
- ☑ I can apply a cleanliness threshold
- ☑ I can filter dumpsters by content
- ☑ I can choose store types
- ☑ I can filter by rating

B.2.4 Location

As a user
I want to be able to change my location
so that I can see dumpsters in other areas

Acceptance criteria:

- ☑ I can change my location permanently
- ☐ I can change my location temporarily (just in the map view, perhaps)

B.2.5 Dumpster Info

As a user
I want to get information about different dumpsters
so that I can see if going to a particular dumpster is worth it

Acceptance criteria:

- ☑ I can view details about a dumpster (from the map or list view)
- ☑ I can easily go back to the previous view
- ☑ I can easily get to other parts of the UI with different kinds of information from this view

As a user
I want to be able to add a new dumpster to the system
so that other dumpster divers get to know about the dumpster and can collect from it

Acceptance criteria:

- ☑ I can add a new dumpster and register information about it
- ☑ I can easily mark the dumpster's position on the map

B.2.6 Revisions

As a user
I want to see the edit history of a dumpster
so that I can check if there has been any malicious changes to the information

Acceptance criteria:

- ☑ I can view all information related to each version of the dumpster
- ☑ The most recent version is displayed first
- ☑ I do not see everything immediately and have to select a revision to see its information

As a user
I want to be able to revert to an older version of a dumpster
so that other users can see information that is more correct than what it was changed to

Acceptance criteria:

- ☑ I can switch to a different version of a dumpster quite easily
- ☑ After reverting, I am taken back to the details view, now with the information from the

version I chose

B.2.7 Locked Dumpsters

As a user
I want to know if (and when?) a dumpster is locked
so that I can know if I can access the contents of the dumpster *easily*

Acceptance criteria:

- ☑ I can easily tell if the dumpster I view is locked by an icon and/or some text
- ☑ This information is easily visible on the map, by color-coding

As a user
I want to be able to update if a dumpster is locked
so that others can be updated about any changes to the dumpster

Acceptance criteria:

- ☑ I can set this when creating a dumpster
- ☑ I can toggle this information

B.2.8 Ratings

As a user
I want to look at the rating of a dumpster
so that I can know about other peoples' experiences going dumpster diving in that dumpster

Acceptance criteria:

- ☑ The rating for this dumpster is displayed in an understandable manner

As a user
I want to be able to add a rating to a dumpster
so that I can tell other dumpster divers about my experience with that dumpster on a given scale

Acceptance criteria:

- ☑ I can rate a selected dumpster
- ☑ The displayed rating changes because of my rating

B.2.9 Pictures

As a user
I want to view pictures of a dumpster
so that I can see what it contains and what the state of the container is

Acceptance criteria:

- ☑ I can see various pictures of a dumpster

- ☑ If there are no submitted pictures, I see a placeholder image
- ☑ The images are zoomed in when placed in smaller views (e.g. the list view)

As a user
I want to be able to share a picture of a dumpster
so that others can see what the dumpster contains and what state the dumpster is in

Acceptance criteria:

- ☑ I can upload a picture I've taken before
- ☑ I can take a picture on the spot and upload it
- ☑ I get feedback if the image is too large
- ☑ I get feedback if the image format is unsupported or invalid
- ☑ I get to see a preview of the image in the app
- ☑ The image appears among the other images immediately after upload

B.2.10 Emptying of Dumpsters

As a user
I want to know when a dumpster is emptied
so that I know when the dumpster is likely to have the most content

Acceptance criteria:

- ☑ I can see on what days the dumpster is emptied if this is known
- ☑ If the schedule is unknown, that should be stated.

As a user
I want to update the emptying schedule of a dumpster
so that others can dive in the dumpster when it is most likely to be full

Acceptance criteria:

- ☑ It is possible to edit when a dumpster is emptied if the schedule changes
- ☑ The schedule is entered in a user-friendly way

B.2.11 Best-Before Dates

As a user
I want to see the best-before dates of the products in a dumpster
so that I know if it is worth going to the dumpster to collect the contents

Acceptance criteria:

- ☑ The dates are displayed in an intuitive way
- ☐ Items that are close to expiring are highlighted (moved up on list?)
- ☐ Items that are long past the expiry date, are moved lower on the list

As a user
I want to be able to add best-before dates to the registered contents of a dumpster
so that other dumpster divers can get some of the content before it expires

Acceptance criteria:

- ☑ This information is treated as optional
- ☑ Dates are entered in a sensible way

B.2.12 Compressors

As a user
I want to know if a dumpster is a compressor or a different type of dumpster
so that I know if it is likely that I will find items that are intact

Acceptance criteria:

- ☑ It is clearly stated what kind of container the selected dumpster is

As a user
I want to be able to edit what kind of container a dumpster is
so that the data is updated and other dumpster divers can get the information

Acceptance criteria:

- ☑ It is possible to edit what kind of container a dumpster is

B.2.13 Dumpster Contents

As a user
I want to see a list of *types of* contents in a dumpster
so that I can see if the dumpster contains what I want

Acceptance criteria:

- ☑ It displays a list of the types of content that others have found in the dumpster
- ☑ Optional data is not shown

As a user
I want to update/add to a list of what a dumpster contains
so that others can see if what they need is in that dumpster

Acceptance criteria:

- ☑ Items can be added to the list if new items are found in the container
- ☑ Optional data is treated correctly
- ☑ Items can be removed from the list if that sort of item is not found in the container anymore

B.2.14 Favorability

As a user
I want to know if the supermarket, shop or store views dumpster diving favorably
so that I am aware of what I can expect if I end up stumbling upon a staff member

Acceptance criteria:

- ☑ Whether the store views dumpster diving favorably or not, is displayed
- ☑ What this *favorability thing* means, is clearly stated

As a user
I want to be able to edit whether the supermarket, store or shop views dumpster diving favorably or not
so that it can reflect changes in the store's opinion on dumpster diving

Acceptance criteria:

- ☑ I can change this if the store changes its view on dumpster diving
- ☑ I cannot enter an invalid value

B.2.15 Cleanliness

As a user
I want to check how clean a dumpster is deemed to be
so that I can make sure to dress appropriately for the job (and possibly reevaluate my decision about going diving in that dumpster)

Acceptance criteria:

- ☑ I can see how clean a dumpster is, on a scale from impeccably clean to utterly filthy

As a user
I want to be able to edit a dumpster's cleanliness rating
so that others can know what state the dumpster is in

Acceptance criteria:

- ☑ I can edit how clean the dumpster is said to be
- ☑ I cannot enter an invalid value

B.2.16 Visits

As a user
I want to see how many people have been to a dumpster in the past few days
so that I can avoid going to a dumpster that has been stripped clean by other people

Acceptance criteria:

- ☑ I can see how many people have visited a dumpster within a certain time frame
- ☑ I can change what time frame I would like to see this count for, to account for the amount of activity in my area

As a user
I want to state that I have visited a dumpster recently
so that others can know that people have been there, and make more informed decisions

Acceptance criteria:

- ☑ I can register my visit with the click of a button
- ☑ I cannot register a visit if less than x hours have passed since my last visit, to prevent

spam

B.2.17 Navigation

As a user
I want to be able to easily navigate between the different views of dumpsters
so that I can use what is best in any situation without loosing data

Acceptance criteria:

- ☑ Users find it easy to navigate between different pages in the app
- ☑ The app can store your position locally so it can show dumpsters close to you
- ☑ No information is lost (in unpredictable ways) when you navigate between the screens

B.2.18 Intro page

As a user
I want to be shown an intro page the first time I launch the app
so that I can get introduced to the app and set my position the first time I am using the app

Acceptance criteria:

- ☑ When a user first opens the app, they will be shown an introduction page
- ☑ I can set my position here, before I enter the main functionality of the app
- ☑ The user ID is displayed and explained
- ☑ I see a brief introduction to what this app is
- ☑ I can read some tips about dumpster diving before I start using the app

As a user
I want to get some tips about dumpster diving
so that I know some tips and tricks as a novice dumpster diver

Acceptance criteria:

- ☑ I see some tips when I first launch the app
- ☑ I can read these tips again, if I want

B.2.19 Report misinformation

As a user
I want to be able to report dumpsters that don’t exist
so that non-existent dumpsters don’t fill the lists

Acceptance criteria:

- ☑ I can report a dumpster that doesn’t exist
- ☐ Fake dumpsters are removed or archived when enough people report them

B.2.20 Report unwanted images

As a user
I want to be able to report images that aren't relevant for the dumpster
so that offensive and irrelevant images can be removed

Acceptance criteria:

- ☐ I can report an image
- ☐ When enough people report it, the image is removed

B.2.21 Read comments

As a user
I want to be able to read comments other users have written
so that I can learn peoples experience with the dumpster

Acceptance criteria:

- ☑ I can see comments from other users

B.2.22 Write comments

As a user
I want to be able to write a comment about a dumpster
so that share my experience with others

Acceptance criteria:

- ☑ I can write and add a comment
- ☑ I cannot add an empty comment
- ☑ My comments are visible in future viewings

B.2.23 Downvote/upvote comments

As a user
I want to be able to upvote and downvote comments
so that My view of a comment can be shared with others

Acceptance criteria:

- ☑ I can press the upvote or downvote button and the rating changes as expected
- ☑ If I press the same button again, the vote is negated
- ☑ If I press the other button after I have voted, the vote is changed
- ☑ I cannot vote on my own comments

B.2.24 Hide negatively rated comments

As a user
I want to be able to filter out negatively rated comments
so that I don’t have to see comments other users have deemed negative

Acceptance criteria:

- ☑ Negative comments don’t appear when i look at comments
- ☑ I can choose to see negatively rated comments

B.3 Domain model

Dumpsters are the primary entities in the system. The other entities *mainly* represent different features a dumpster can have. If an uploaded photo is inappropriate or a dumpster isn't there, users can report it. Dumpsters can have contents of various ContentTypes, recorded as DumpsterContents. Users can give a Dumpster a Rating, and leave a Comment to tell others about their experiences. To keep the information in the application relevant, we will only use the most recent ratings for the specific dumpster. Ratings and Photos have keys. These will be used to keep track of the user that posted the image or rating, so they can change a rating or delete an image they posted. In the case of ratings, this prevents one person from adding multiple ratings for the same dumpster to the system. It is possible that the system will be altered over time if a better solution is found.

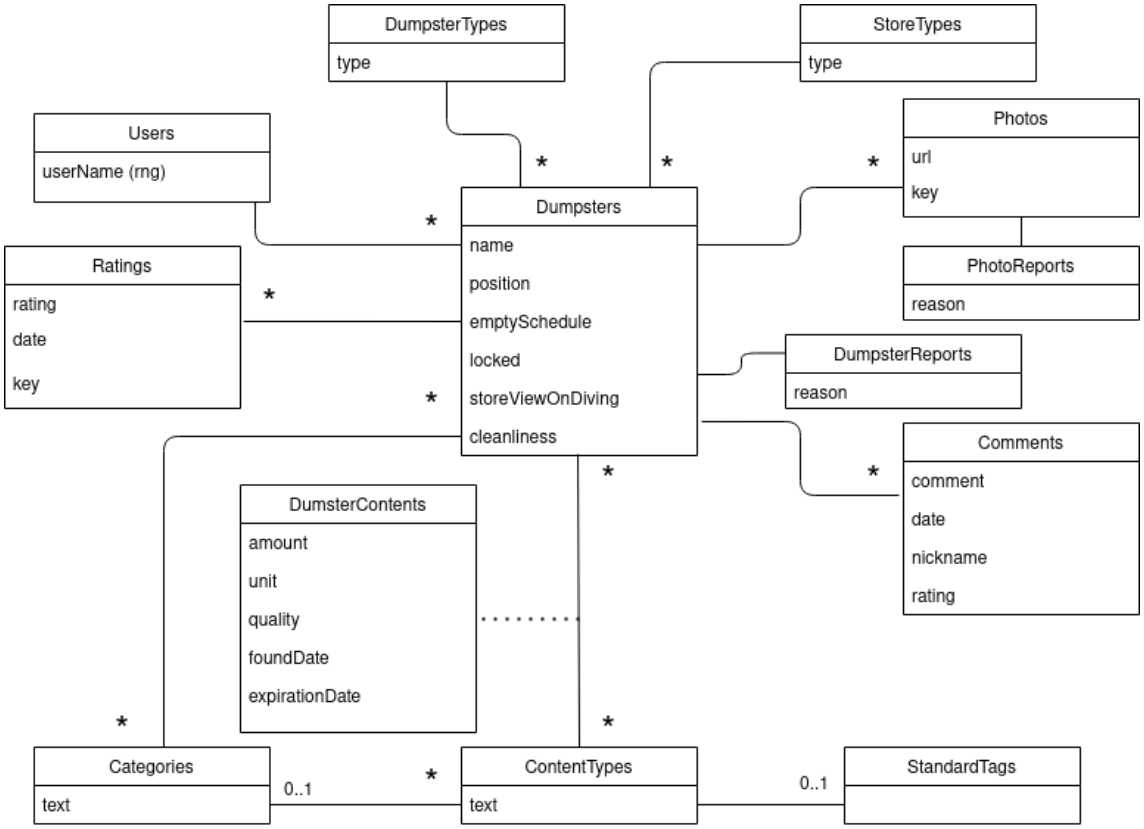


Figure B.1: Domain model of the product

B.4 Prototypes

B.4.1 Wireframes

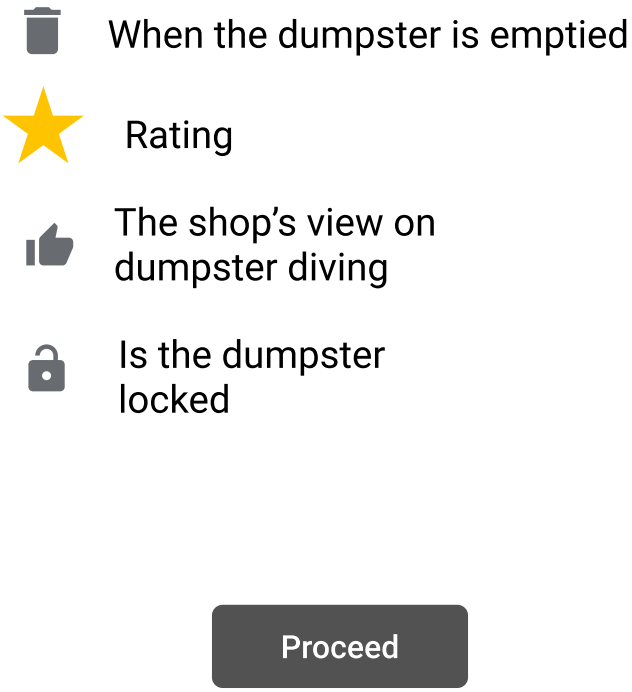
We made some wireframes using Figma, a popular prototyping tool. View an interactive version of the wireframe here (it really is a better experience than wading through the following pages).

Dumpster Finder

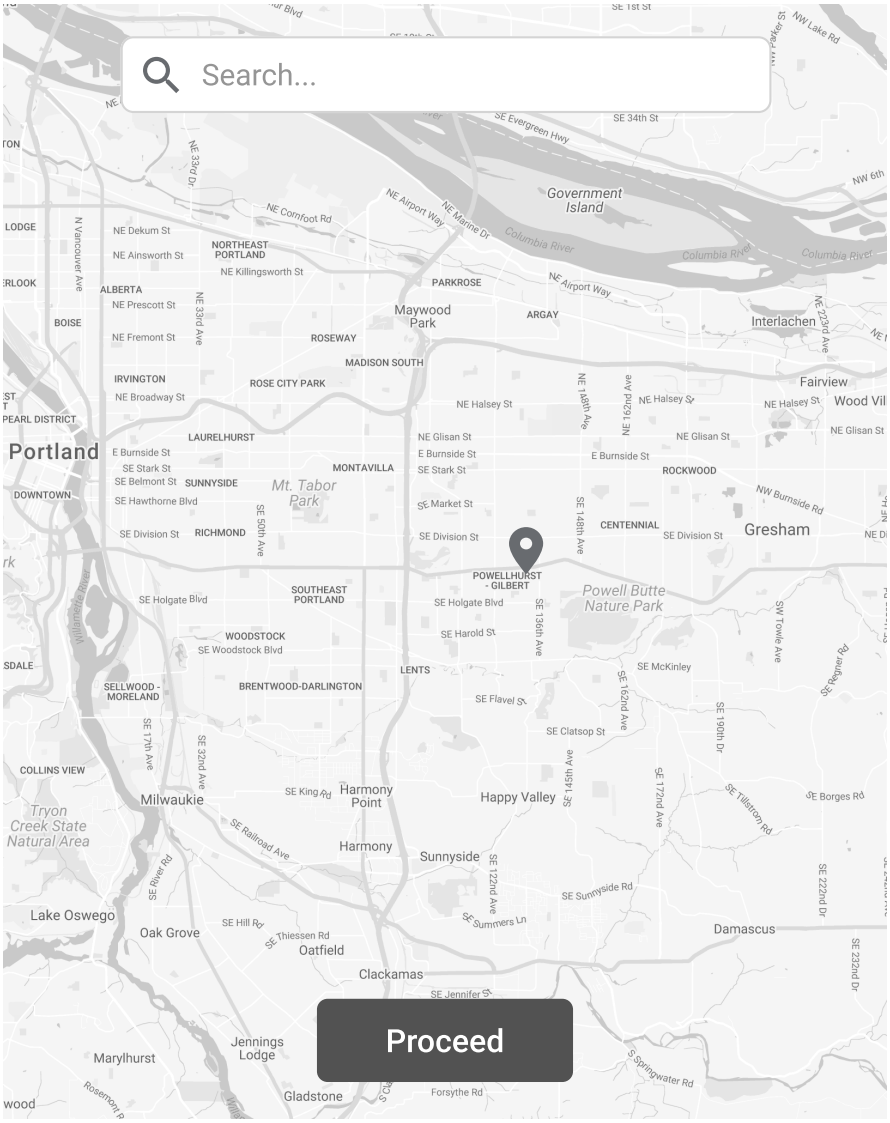
Dumpster diving advice

1. Don’t go diving in locked dumpsters
2. Don’t go in dumpsters marked private
3. Don’t go dumpster diving alone
4. Dress accordingly

Proceed


Dumpster
Finder
When the dumpster is emptied
Rating
The shop's view on
dumpster diving
Is the dumpster
locked
Proceed

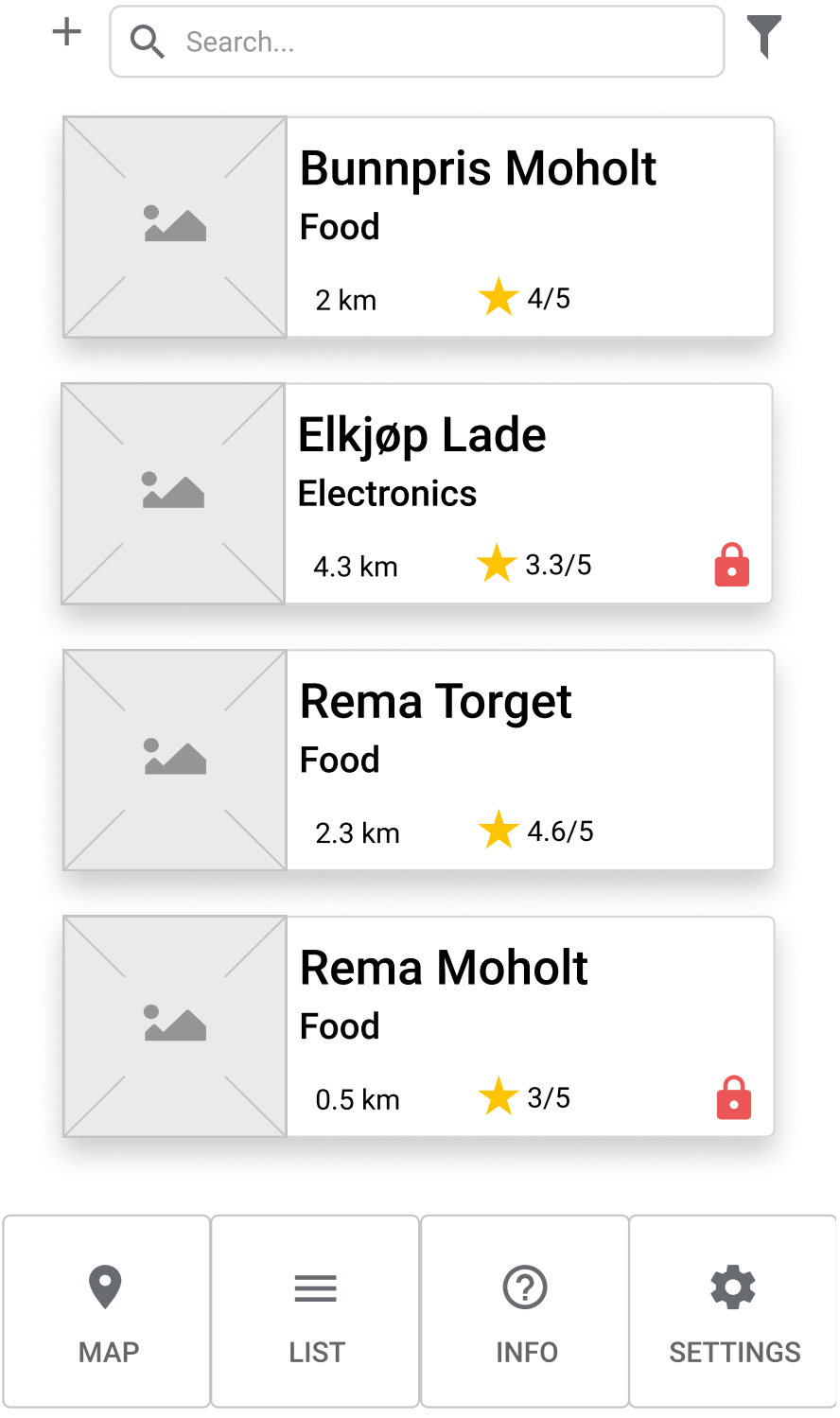
Set your location

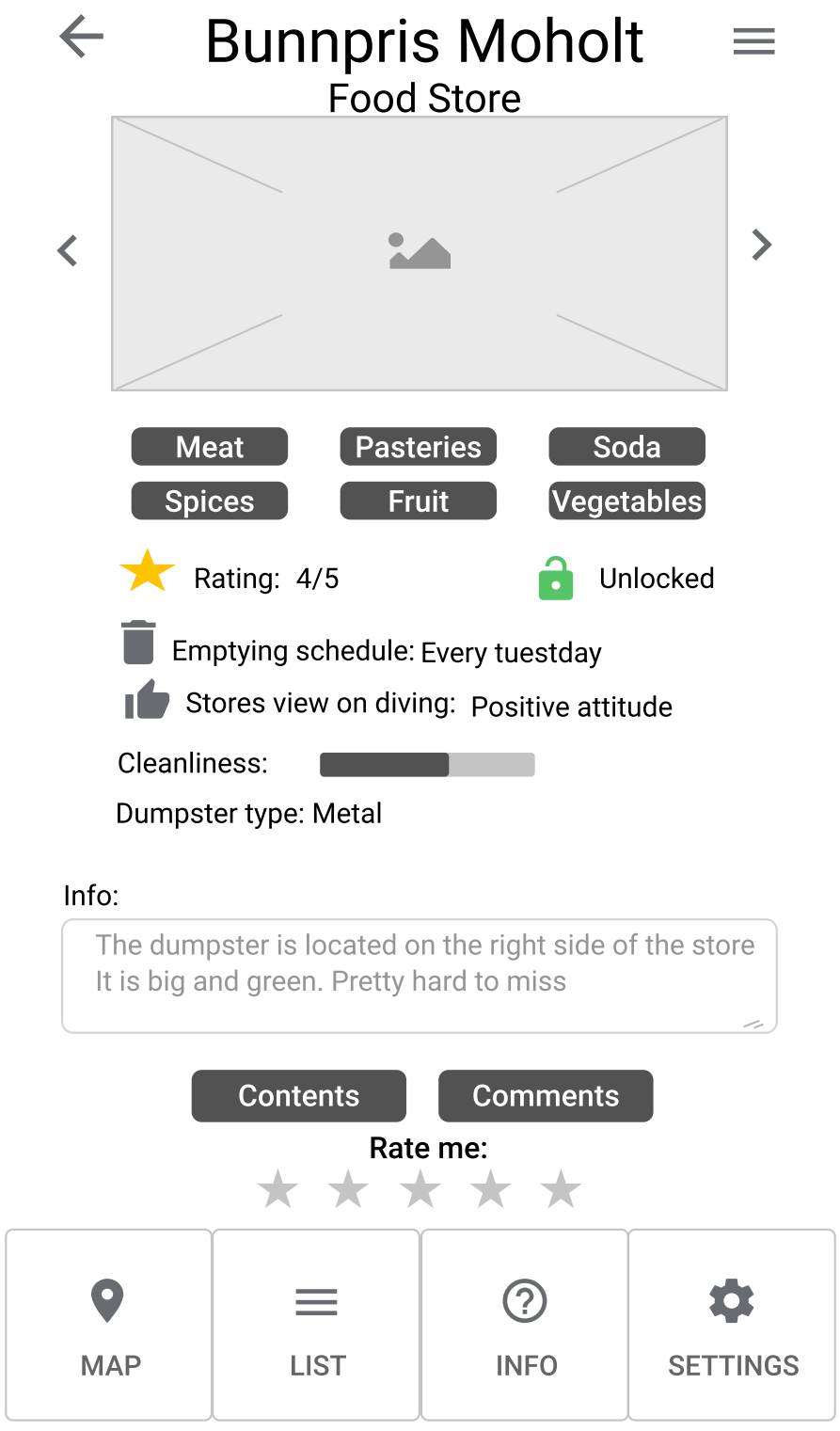


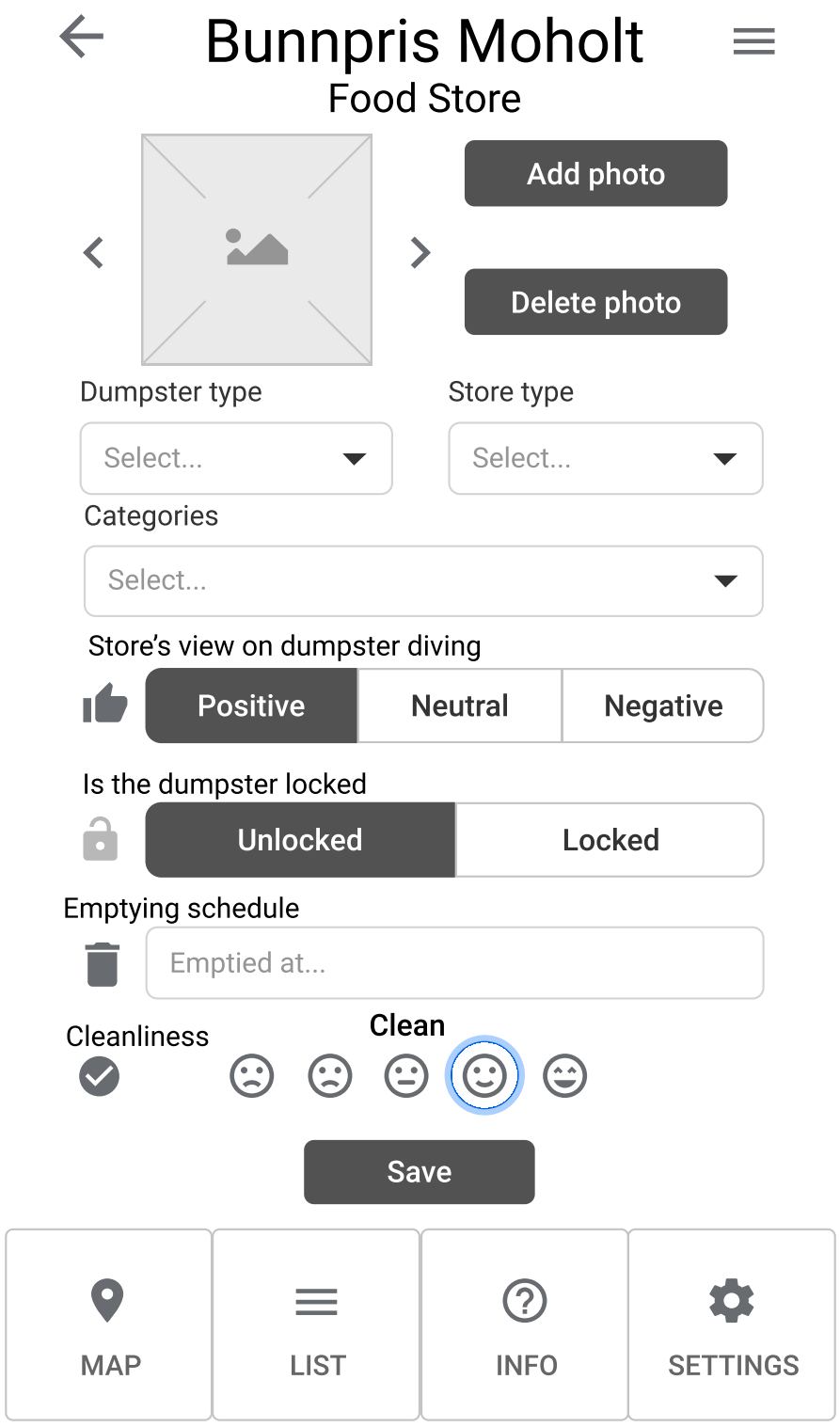
(23.1258685, 33.10461086)

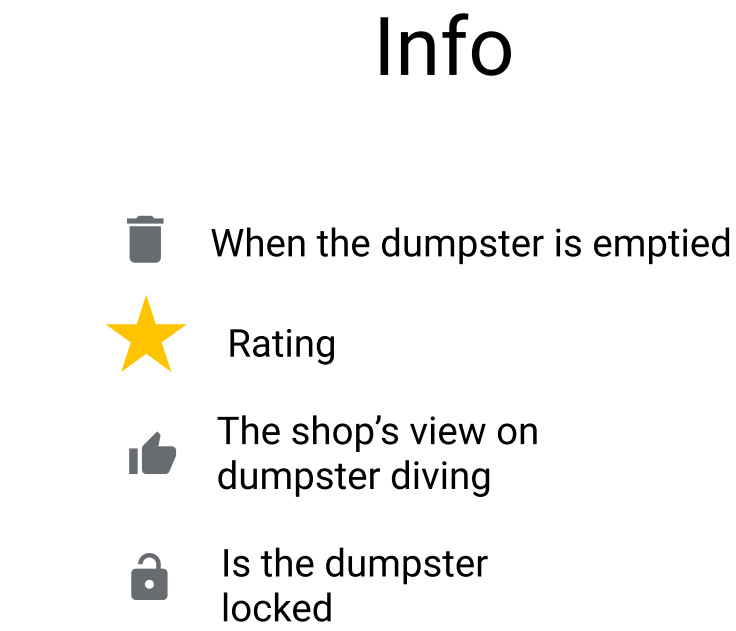
Your location is only stored locally on your device. It will be sent to our server when dumpster data is fetched.
It is never shared with any third-party application.

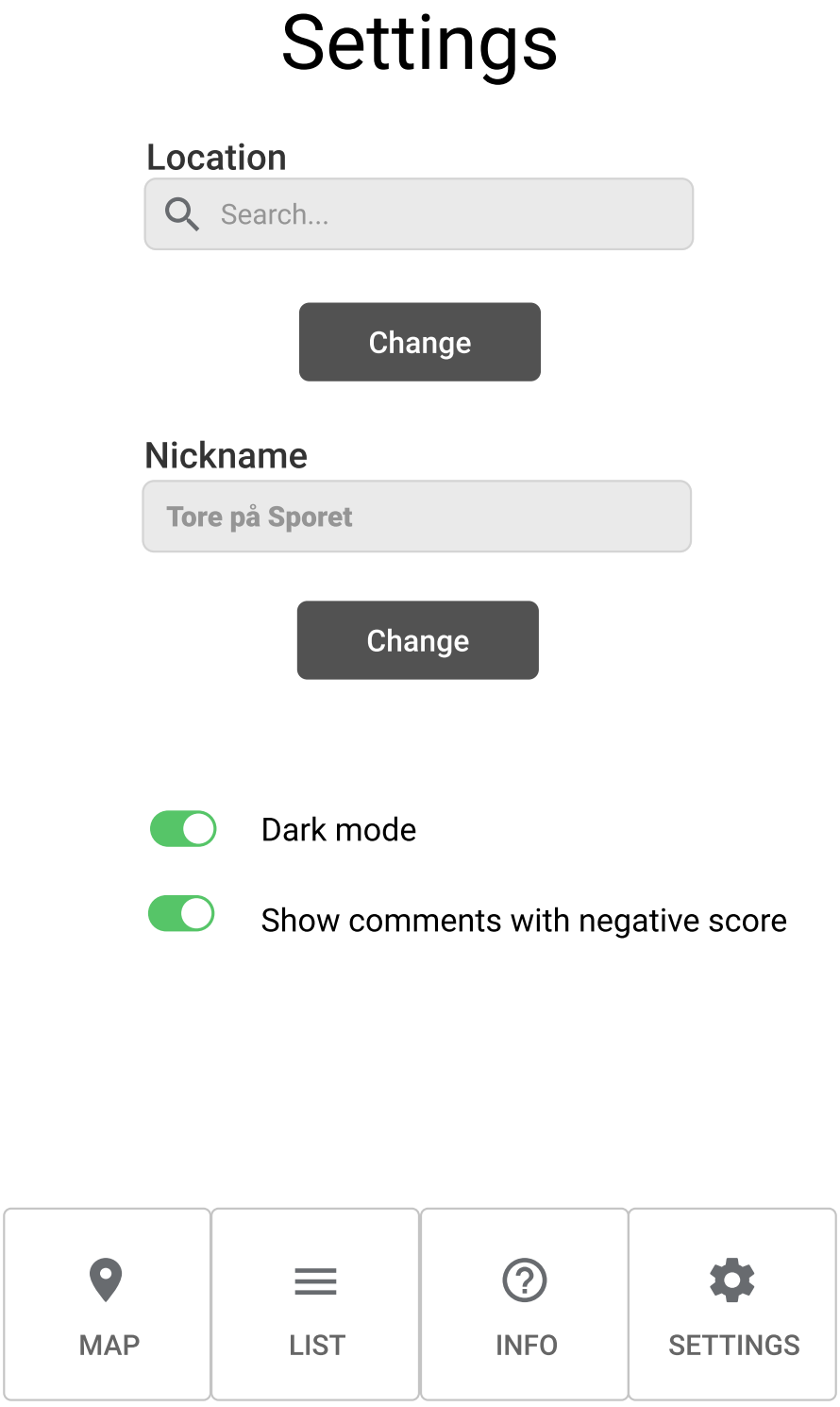

Search...
WASHINGTON
OREGON
Columbia River
Government Island
Cascade Park West
Cascade Park East
Maywood Park
Interlachen
Fairview
Gresham
Mt. Tabor Park
Powell Butte Nature Park
Milwaukie
Happy Valley
Harmony
Sunnyside
Damascus
Oak Grove
Oatfield
Clackamas
Jennings Lodge
Marylhurst
Gladstone
MAP
LIST
INFO
SETTINGS

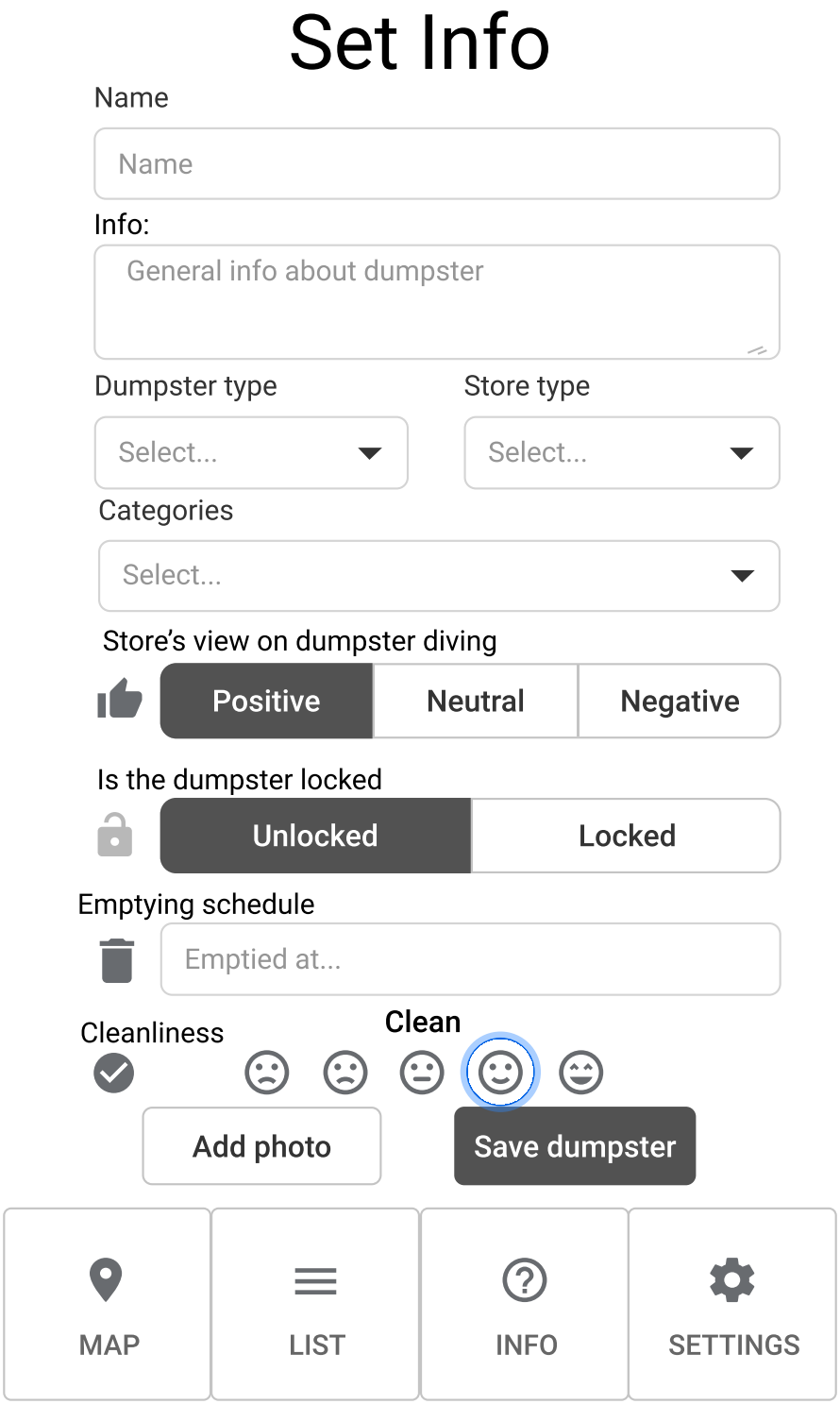

Search...
Bunnpris Moholt
Food
2 km
4/5
Elkjøp Lade
Electronics
4.3 km
3.3/5
Rema Torget
Food
2.3 km
4.6/5
Rema Moholt
Food
0.5 km
3/5
MAP
LIST
INFO
SETTINGS


Bunnpris Moholt
Food Store
Meat
Pasteries
Soda
Spices
Fruit
Vegetables
Rating: 4/5
Unlocked
Emptying schedule: Every tuestday
Stores view on diving: Positive attitude
Cleanliness:
Dumpster type: Metal
Info:
The dumpster is located on the right side of the store
It is big and green. Pretty hard to miss
Contents
Comments
Rate me:
MAP
LIST
INFO
SETTINGS

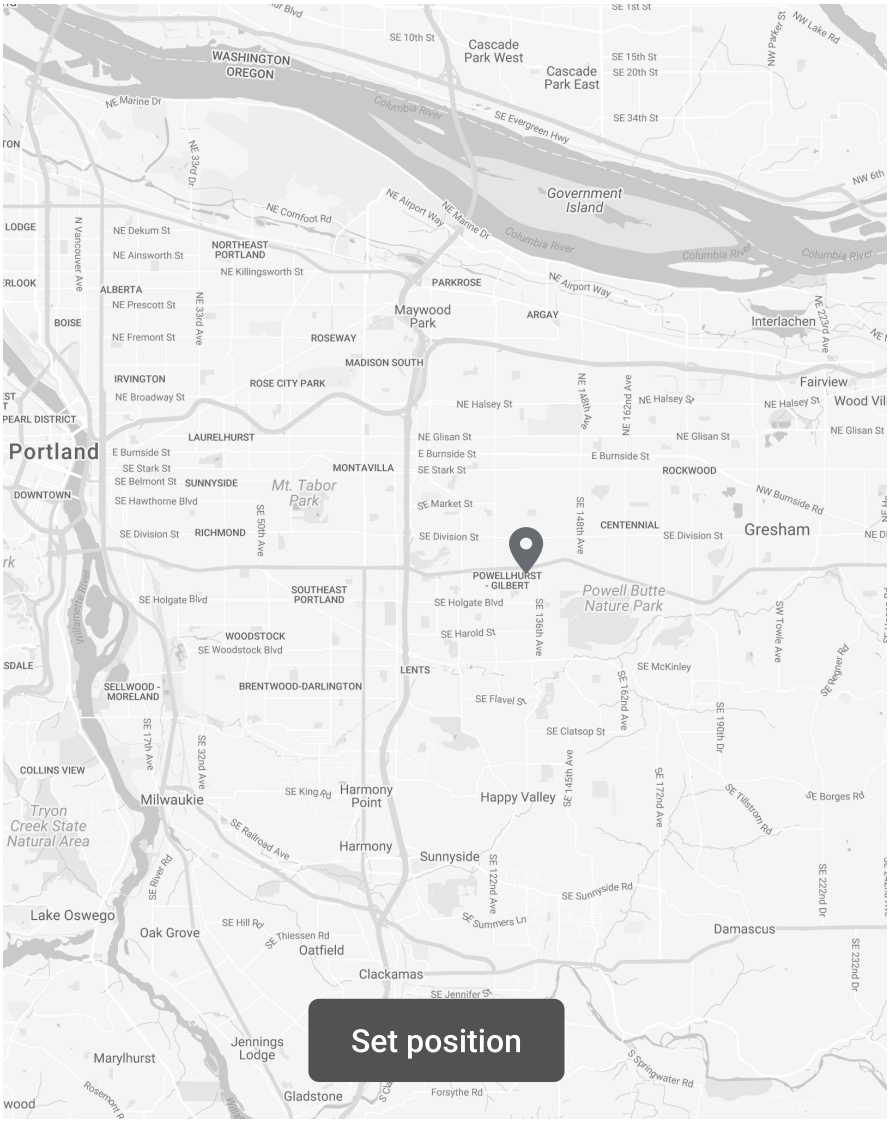

Bunnpris Moholt
Food Store
Add photo
Delete photo
Dumpster type
Select...
Store type
Select...
Categories
Select...
Store's view on dumpster diving
Positive
Neutral
Negative
Is the dumpster locked
Unlocked
Locked
Emptying schedule
Emptied at...
Cleanliness
Clean
Save
MAP
LIST
INFO
SETTINGS


Info
When the dumpster is emptied
Rating
The shop's view on dumpster diving
Is the dumpster locked
Dumpster diving advice
1. Don't go diving in locked dumpsters
2. Don't go in dumpsters marked private
3. Don't go dumpster diving alone
4. Dress accordingly
MAP
LIST
INFO
SETTINGS


Settings
Location
Search...
Change
Nickname
Tore på Sporet
Change
Dark mode
Show comments with negative score
MAP
LIST
INFO
SETTINGS

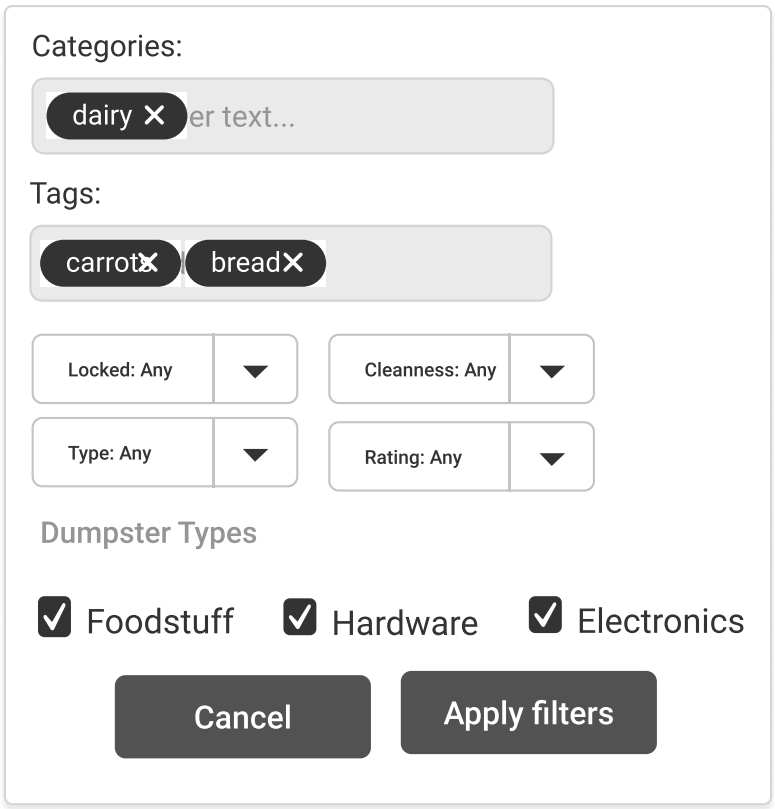

Set Info
Name
Name
Info:
General info about dumpster
Dumpster type
Select...
Store type
Select...
Categories
Select...
Store's view on dumpster diving
Positive
Neutral
Negative
Is the dumpster locked
Unlocked
Locked
Emptying schedule
Emptied at...
Cleanliness
Clean
Add photo
Save dumpster
MAP
LIST
INFO
SETTINGS

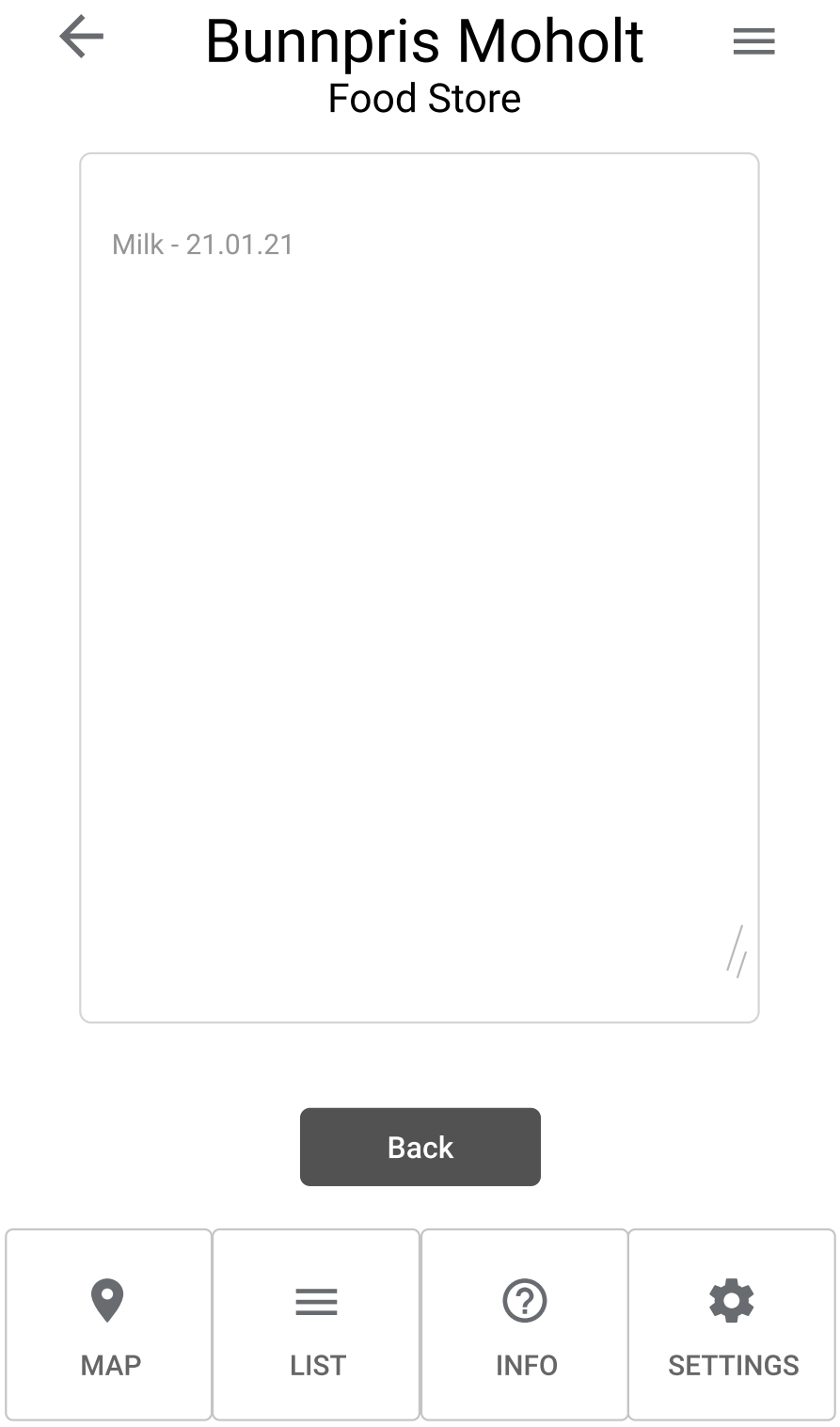
Set Position

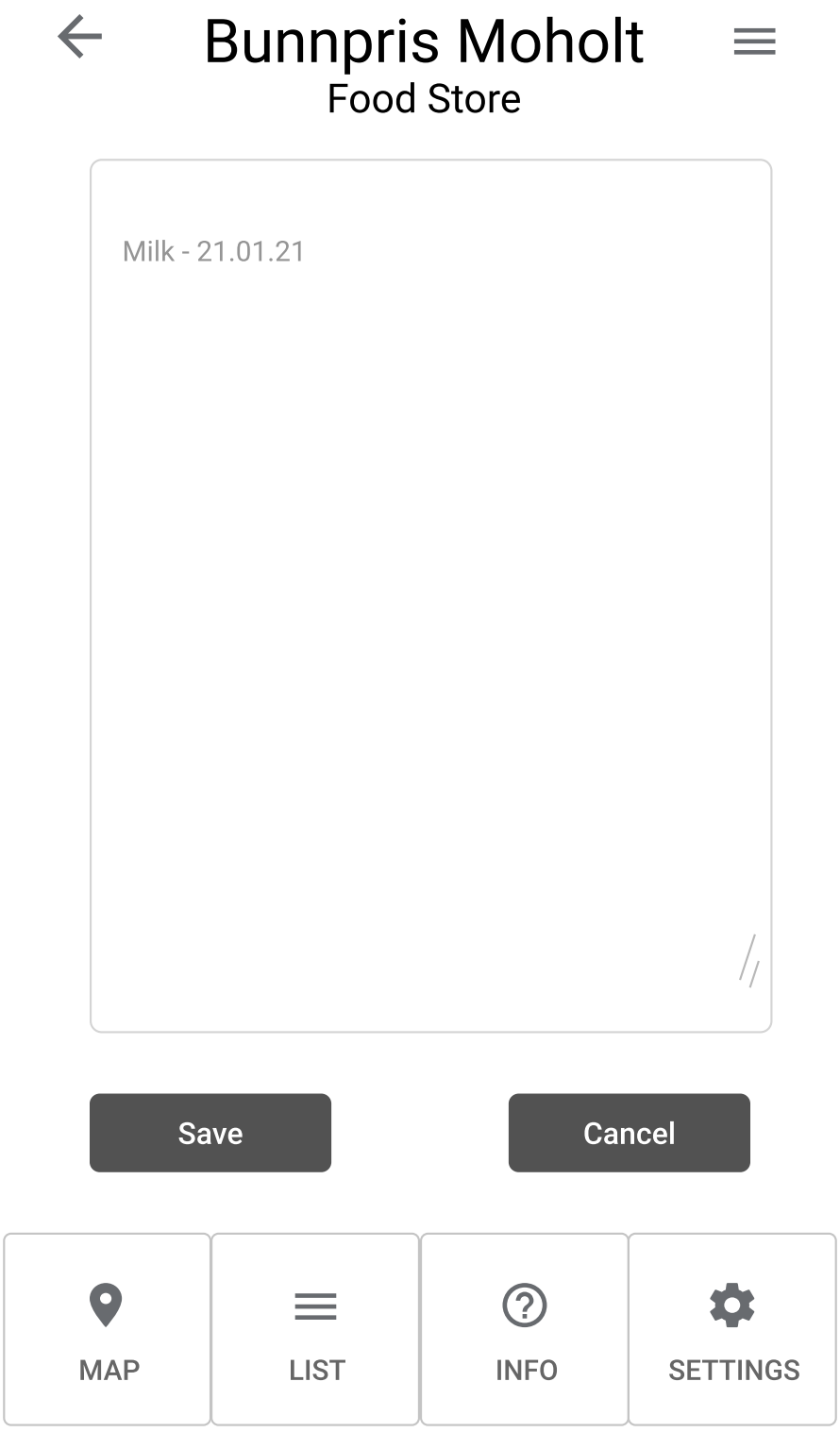


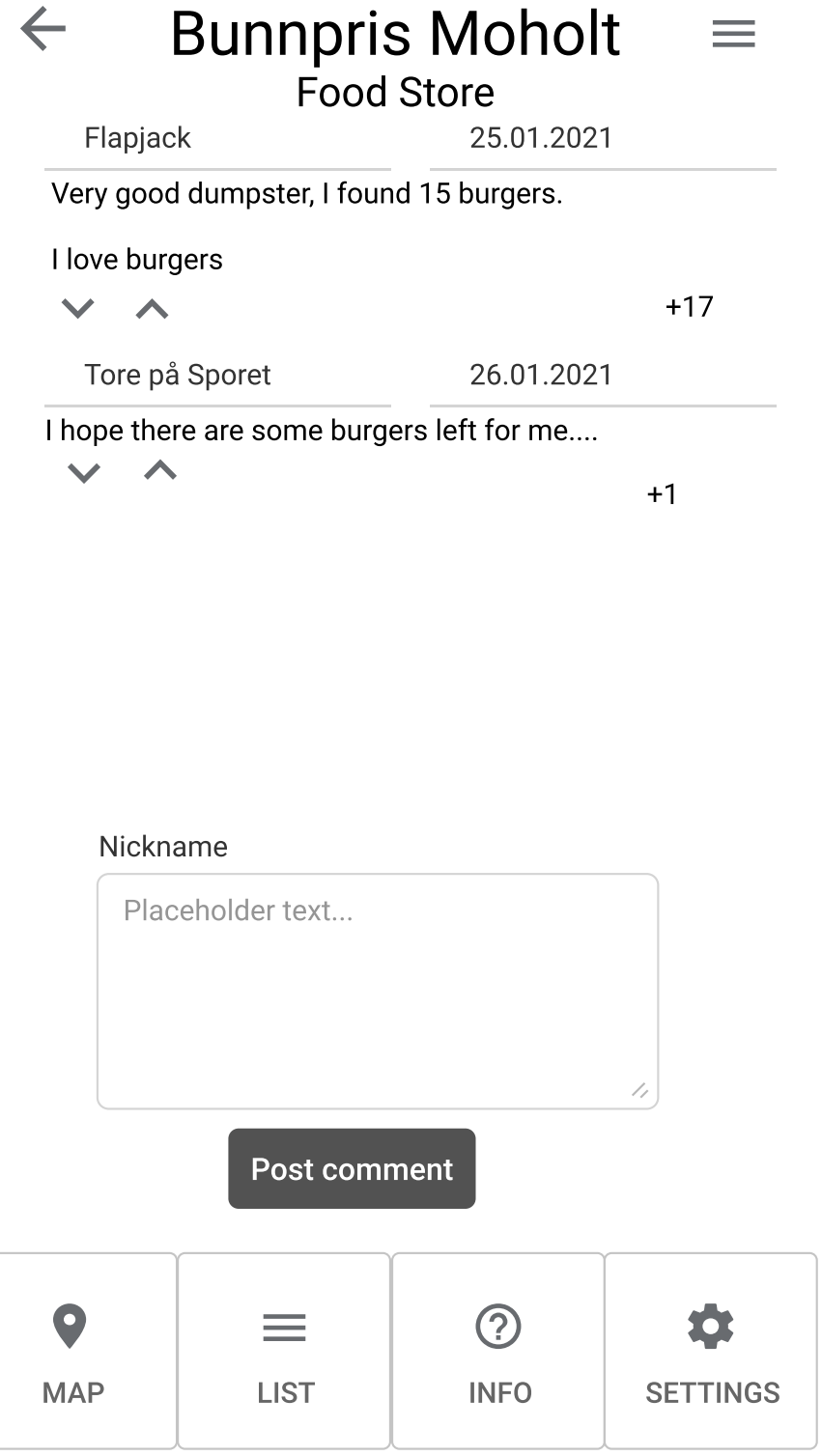
(23.1258685, 33.10461086)

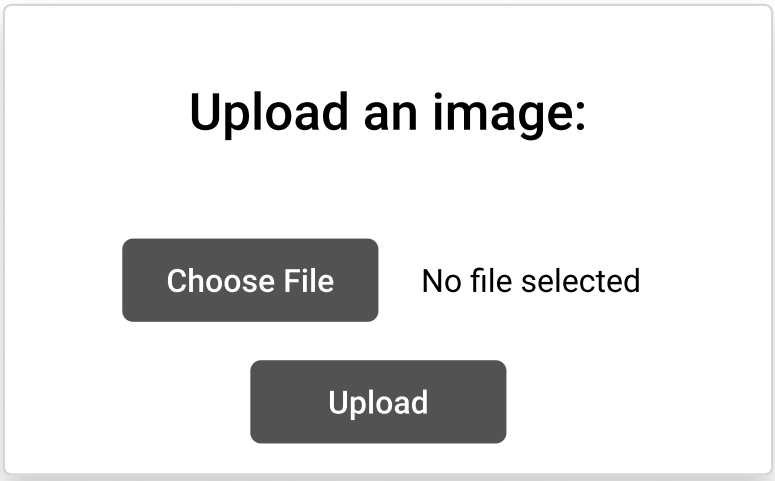
MAP LIST INFO SETTINGS

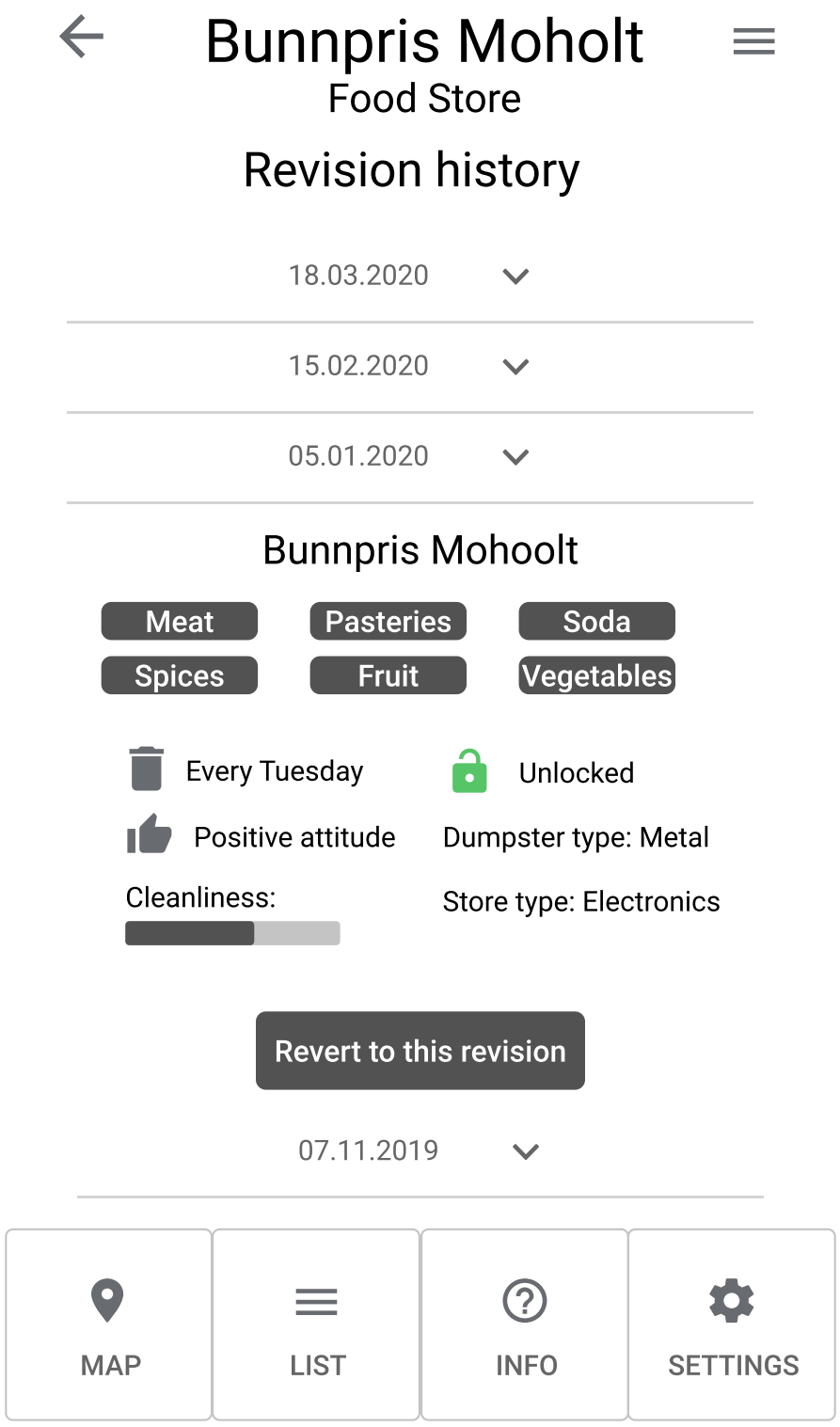

Categories:
dairy
er text...
Tags:
carrot
bread
Locked: Any
Cleanness: Any
Type: Any
Rating: Any
Dumpster Types
Foodstuff
Hardware
Electronics
Cancel
Apply filters

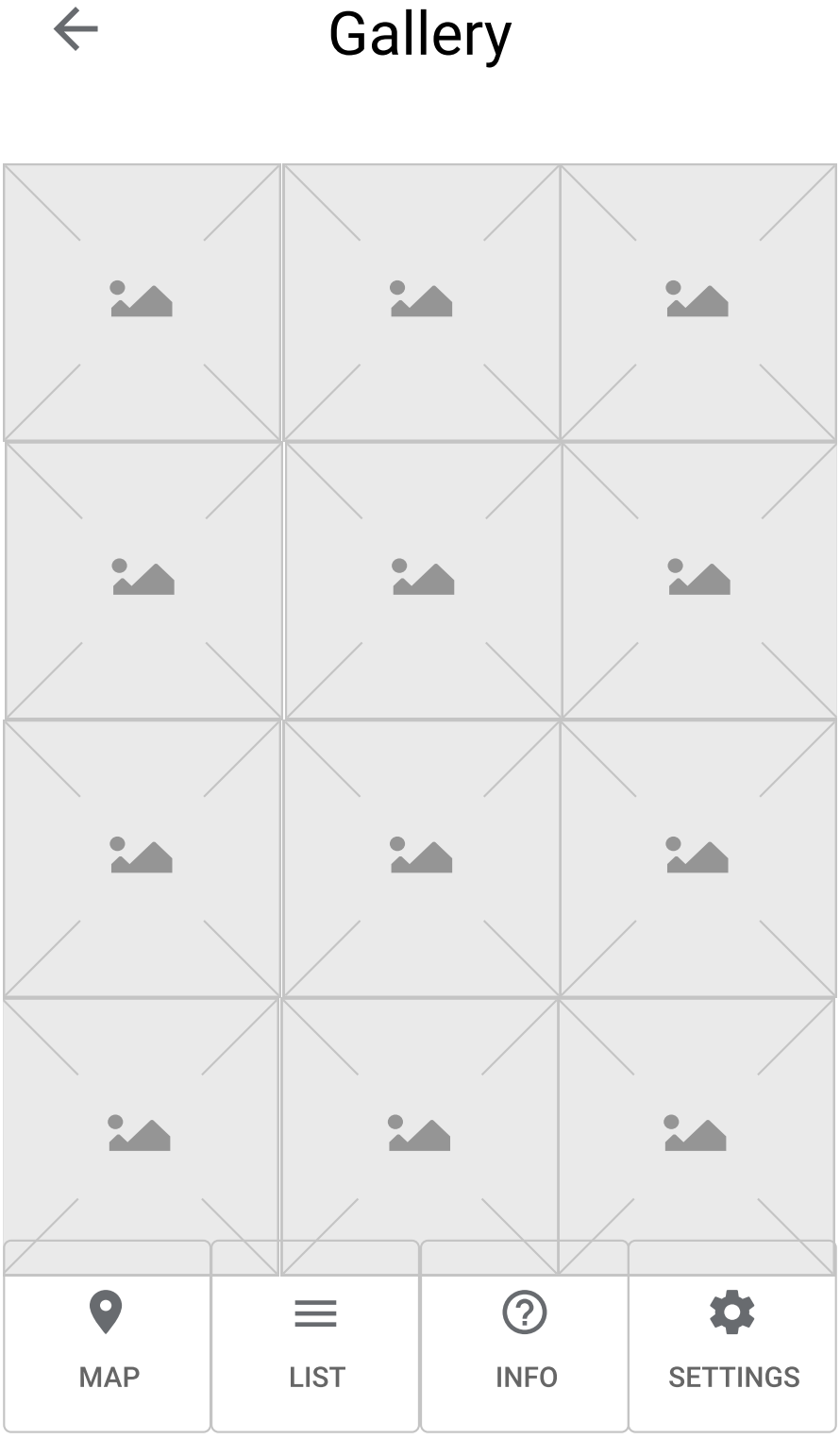

Bunnpris Moholt
Food Store
Milk - 21.01.21
Back
MAP
LIST
INFO
SETTINGS


Bunnpris Moholt
Food Store
Milk - 21.01.21
Save
Cancel
MAP
LIST
INFO
SETTINGS


Bunnpris Moholt
Food Store
Flapjack
25.01.2021
Very good dumpster, I found 15 burgers.
I love burgers
+17
Tore på Sporet
26.01.2021
I hope there are some burgers left for me....
+1
Nickname
Placeholder text...
Post comment
MAP
LIST
INFO
SETTINGS


Upload an image:
Choose File
No file selected
Upload


Bunnpris Moholt
Food Store
Revision history
18.03.2020
15.02.2020
05.01.2020
Bunnpris Mohoolt
Meat
Pasteries
Soda
Spices
Fruit
Vegetables
Every Tuesday
Unlocked
Positive attitude
Dumpster type: Metal
Cleanliness:
Store type: Electronics
Revert to this revision
07.11.2019
MAP
LIST
INFO
SETTINGS


Gallery
MAP
LIST
INFO
SETTINGS

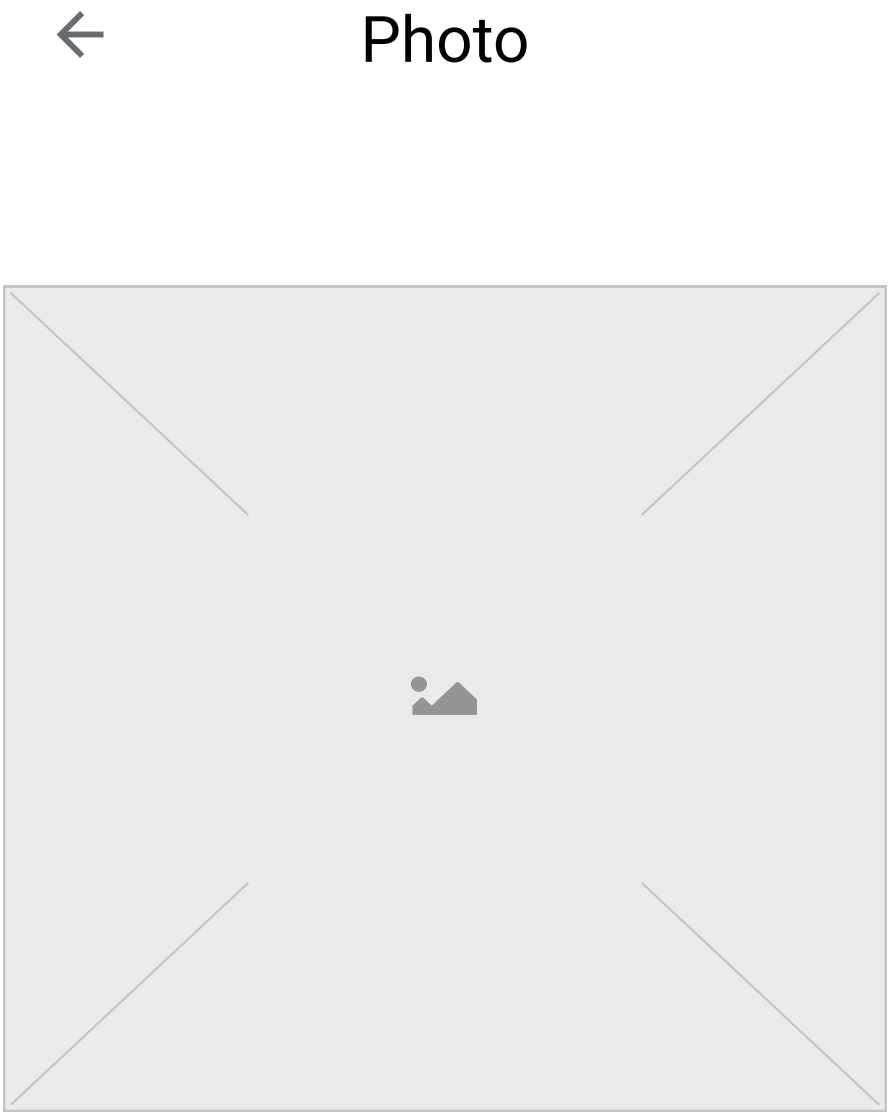
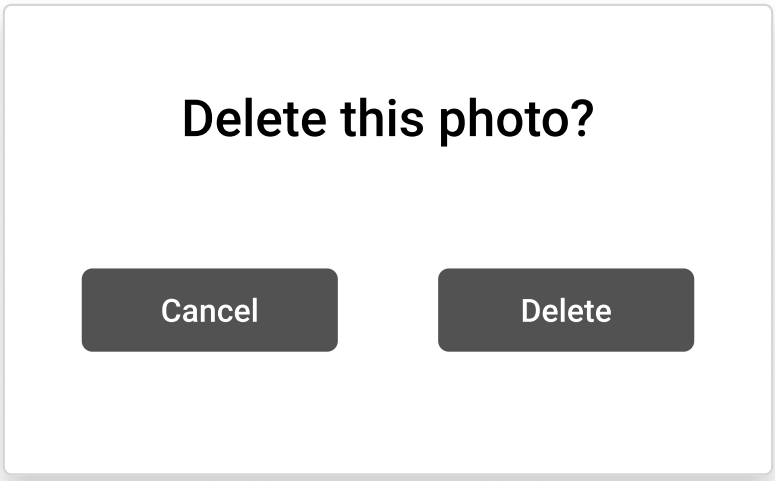
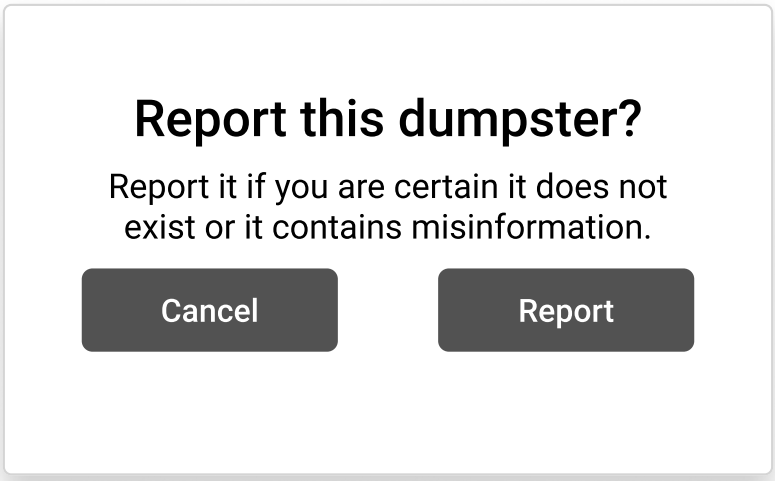


Photo
MAP
LIST
INFO
SETTINGS


Delete this photo?
Cancel
Delete


Report this dumpster?
Report it if you are certain it does not exist or it contains misinformation.
Cancel
Report

Appendix C

System documentation

Audit history

Date	Version	Description	Author
12.01.2021	0.1	Creation of document	Helene Y. Jonson
02.05.2021	0.2	WIP draft	Tore Bergebakken
10.05.2021	1.0	First complete draft	Tore Bergebakken, Jon Åby Bergquist
18.05.2021	1.1	Final version	Tore Bergebakken, Jon Åby Bergquist
20.05.2021	1.2	Final version, now with class diagrams	Tore Bergebakken

C.1 Introduction

This document was written by three students at the Norwegian University of Science and Technology as part of our bachelor thesis in computer engineering. Our assignment was to create an application where dumpster divers can share information about different dumpsters with other dumpster divers. The purpose of this document is to explain how the system was structured and other aspects of its design. It will explain the project's architecture, how its files are structured, detail the database schema, explain the different endpoints in the API, go through the different means of security enhancements, provide instructions for installing and running the project, explain where the source code documentation can be found and how to generate it, and go through the continuous integration pipeline.

C.2 Architecture

As Figure C.1 shows, our system consists of a mobile app and a server split into several containerized services: an API server, a file server and a database, connected to the outside world through a reverse proxy.

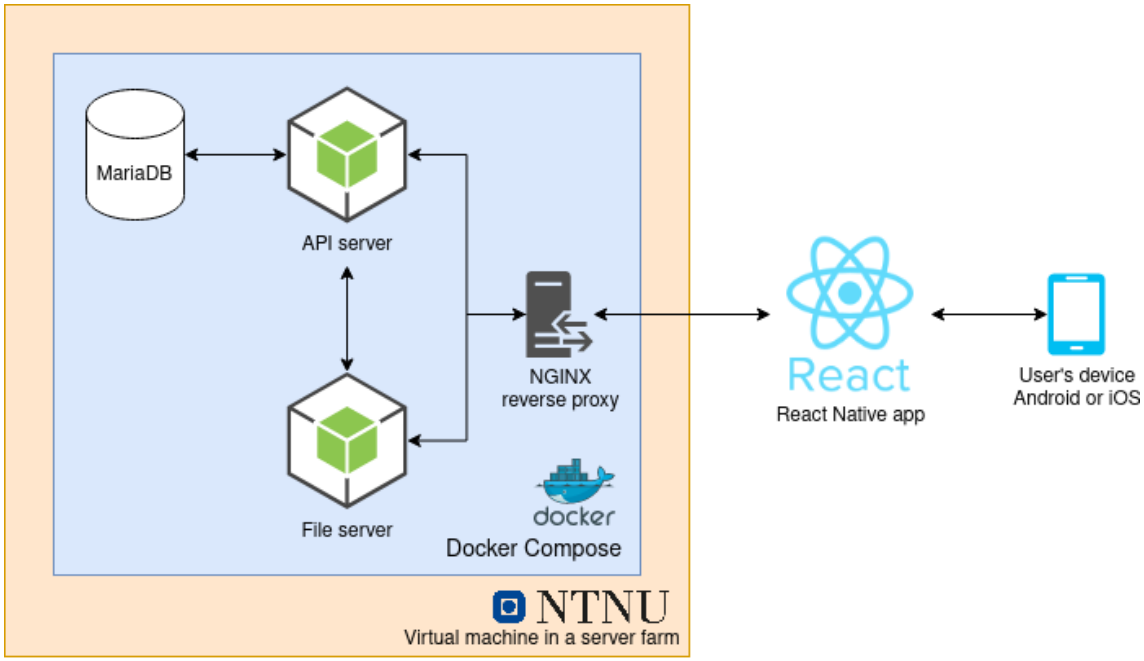


Figure C.1: Architecture diagram

C.3 Project structure

The root folder contains a README and some useful things.

C.3.1 backend

Here, all components of our backend are configured.

- `docker-compose.yml`: Defines the set of containers that the backend consists of, and any links between them
- `renew_certs.sh`: Script for renewing SSL certificates for the sake of HTTPS support
- `dumpster.service`: systemd unit file
- `Makefile`: Contains scripts that set up the database and more
- `tables.sh`: Related to the Makefile

api

- `@types`: Some extra TypeScript definitions
- `config`: Files that extract environment variables and set up some object (like our Sequelize instance and our logger), and a file that exports a function for test setup
- `daos`: Data Access Objects that serve as wrappers around Sequelize cards, with some transactional logic and error reporting
  - `__tests__`: Unit tests for these DAOs
- `middleware`: Express middleware functions that handle things like tokens, errors and rate limiting
- `models`: Sequelize models that correspond to tables in the database
- `node_modules`: The folder where all our third-party dependencies end up
- `routes`: API routes, exported as Express routers that are connected to their respective paths in `server.ts`
- `types`: TypeScript interfaces for objects that the API server sends and receives
- `utils`: Utilities for hashing and generating user IDs and handling tokens
- `validators`: Joi validators that check that data the API receives fits with a certain schema
- `index.ts`: An entry point that imports and starts the Express server
- `server.ts`: Connects all routes to their paths, as described in section C.6
- `package.json`: The usual file where our dependencies and commands are listed

- `.env`: Contains *secret* environment variables necessary for letting the API know things like which port it should listen on
- `Dockerfile`: Defines how the API server's container should be set up

`db`

Database scripts.

- `init.sql`: Defines the structure of the database and a necessary function
- `constants.sql`: Adds categories and types of stores and dumpsters
- `data.sql`: Inserts test data that is not meant to be used in production
- `Dockerfile`: Defines how the database container should be set up
- `Makefile`: For quick setup and refresh of the database while developing
- `setup.sh`: Related to the Makefile

`nginx`

Configuration files for the NGINX proxy, which is responsible for routing requests to the API or photo server depending on the path, and enabling HTTPS.

- `web`: Web content
- `nginx.conf`: The configuration file
- `Dockerfile`: Defines the NGINX container, and replaces `example.com` in the config file with the actual domain name

`pics`

Picture server, the place where pictures uploaded through the app are sent, and where clients will receive images from. Roughly the same as `api`, but without any Sequelize models. No need to elaborate.

- `@types`
- `config`
- `controllers`: Inappropriately named; contains a file with some functions used when a photo is uploaded
- `middleware`
- `node_modules`
- `routes`

- `types`
- `validators`
- `index.ts`
- `server.ts`
- `package.json`
- `.env`
- `Dockerfile`

C.3.2 `frontend`

The files that make up the React Native client for our backend.

- `@types`: Extra TypeScript definitions
- `assets`
  - `fonts`: Extra non-system fonts
  - `images`: Map marker icons and placeholder images
- `components`: React components (or widgets) that are *not* entire screens themselves
  - `basicComponents`: Simple components
  - `cards`: Components with a card at the top level – for displaying information about a single entity
  - `compoundComponents`: More complex components
  - `dumpsterInfo`: Components that display information about a dumpster
  - `map`: Components related to maps
  - `modals`: Components that are used as modals (pop-ups)
  - `selects`: Various kinds of dropdown selection components
  - `settings`: Components used on the settings screen
  - `textComponents`: Components that display some static text
- `constants`: Various constant values
- `coverage`: Produced by Jest on test runs
- `docs`: Documentation generated by typedoc
- `hooks`: Custom hooks – functions called in React components that set up some functionality
- `models`: Interfaces and classes that describe our data models

- `navigation`: Files that set up the navigation structure used by React Navigation – mostly from the template we used
- `redux`: Handles global application state
  - `slices`: Redux slices that take care of specific parts of the application state
  - `store.ts`: Combines the state of each slice into a whole
  - `tokenInterceptors.ts`: Axios interceptors that handle tokens in requests – placed here because they use Redux state
- `node_modules`: The folder where all our third-party dependencies end up
- `screens`: Entire screens in the application
  - `addDumpster`: Screens that are part of the process of adding a dumpster
  - `dumpsterInfo`: Screens that display and/or edit info about a dumpster
  - `main`: Top-level screens — the four you get to through the icons in the tab bar
  - `photo`: Screens that display, take or upload photos
- `services`: Each service handles calls to a specific part of the API
- `translations`: JSON files with translations for all text displayed in the application
- `utils`: Miscellaneous utilities used in some parts of the application – date formatting, distance calculation, dumpster filtering and error reporting
- `app.config.js`: Expo configuration
- `App.tsx`: Contains the top-level component of the app. Deals with (re)setting state when the app loads, and several other important tasks.
- `i18n.tsx`: Sets up i18next, which handles translations
- `package.json`: The usual file where our dependencies and commands are listed
- `types.tsx`: Types used by React Navigation
- `.env`: Environment variables like the URL to the API and photo server

C.4 Class diagram

Our code base contains very few classes. We have service classes, *some* classes for data models and classes for models in our ORM. In this section, we provide diagrams to visualize some of them.

The data models in the frontend are shown in Figure C.2. Very few of them have any methods at all, and their reason for being classes in the first place was to translate strings with dates to actual date objects. Each of them have a constructor that takes data from the API as raw JavaScript objects converted directly from JSON, which does not support the Date type.

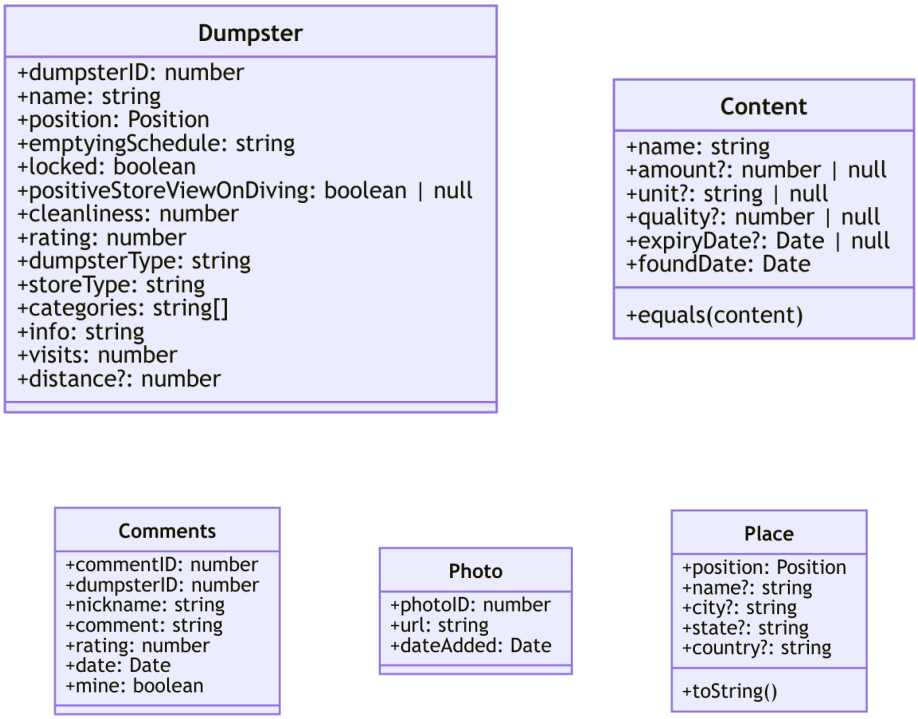


Figure C.2: Frontend data classes

The frontend's service classes, which send requests to the API, are shown in Figure C.3. Since the only thing they have in common is that their constructor takes an instance of an object that performs HTTP requests, it did not seem necessary to have them inherit an abstract class.

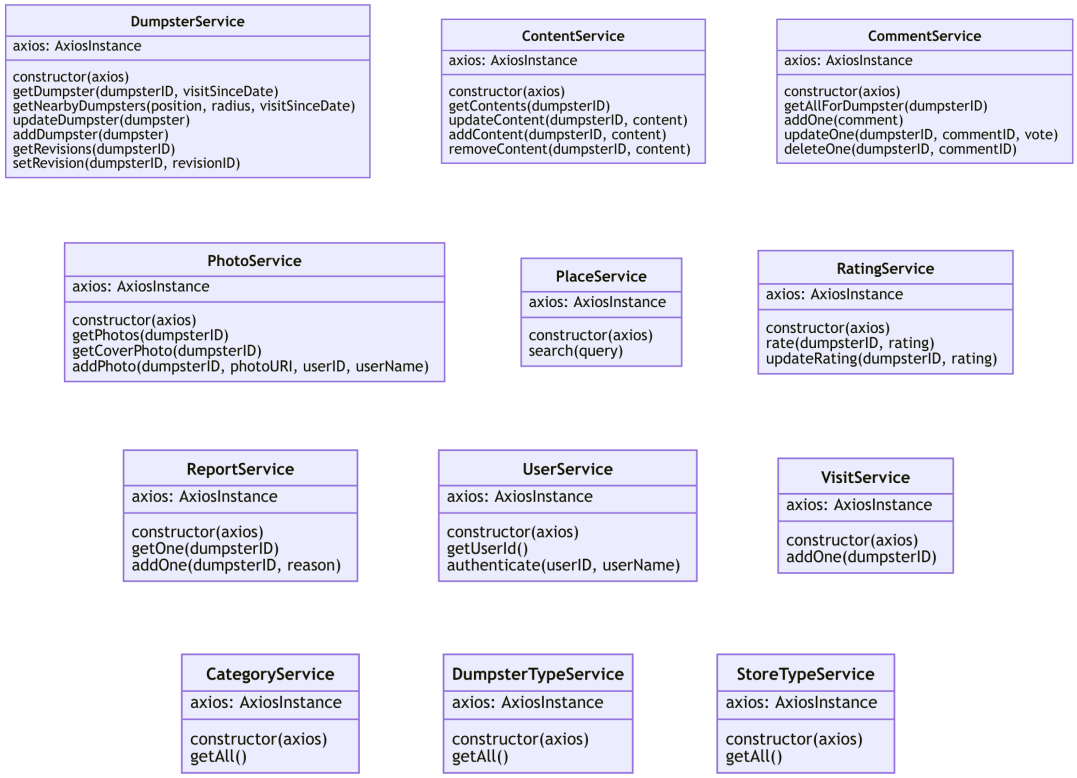


Figure C.3: Service classes

Sequelize, our ORM of choice, requires classes that act as definitions of tables in the database. Since most of the Sequelize models just reflect the database model (Figure C.5), we only show a few of the model classes in Figure C.4. Each model inherits Sequelize's Model class and implements an interface with attributes. The latter is necessary for correct type hints with TypeScript.

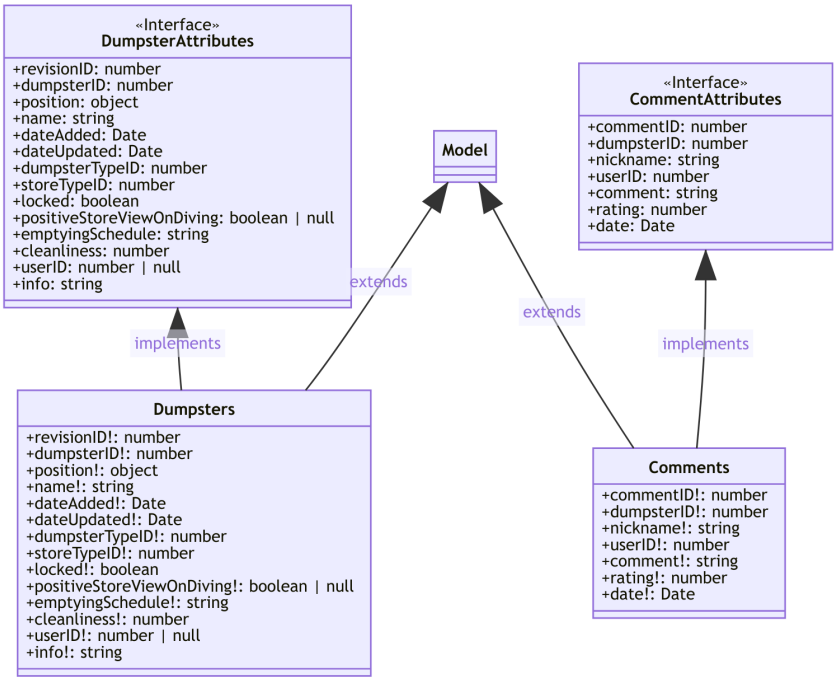


Figure C.4: Sequelize models

C.5 Database model

Figure C.5 shows our database model. We visualize complex junction tables for many-to-many relations with a line going down to the relation it describes, but make exceptions in cases where it makes more sense to treat the table as its own, separate entity — and in those cases, the entities have custom primary keys.

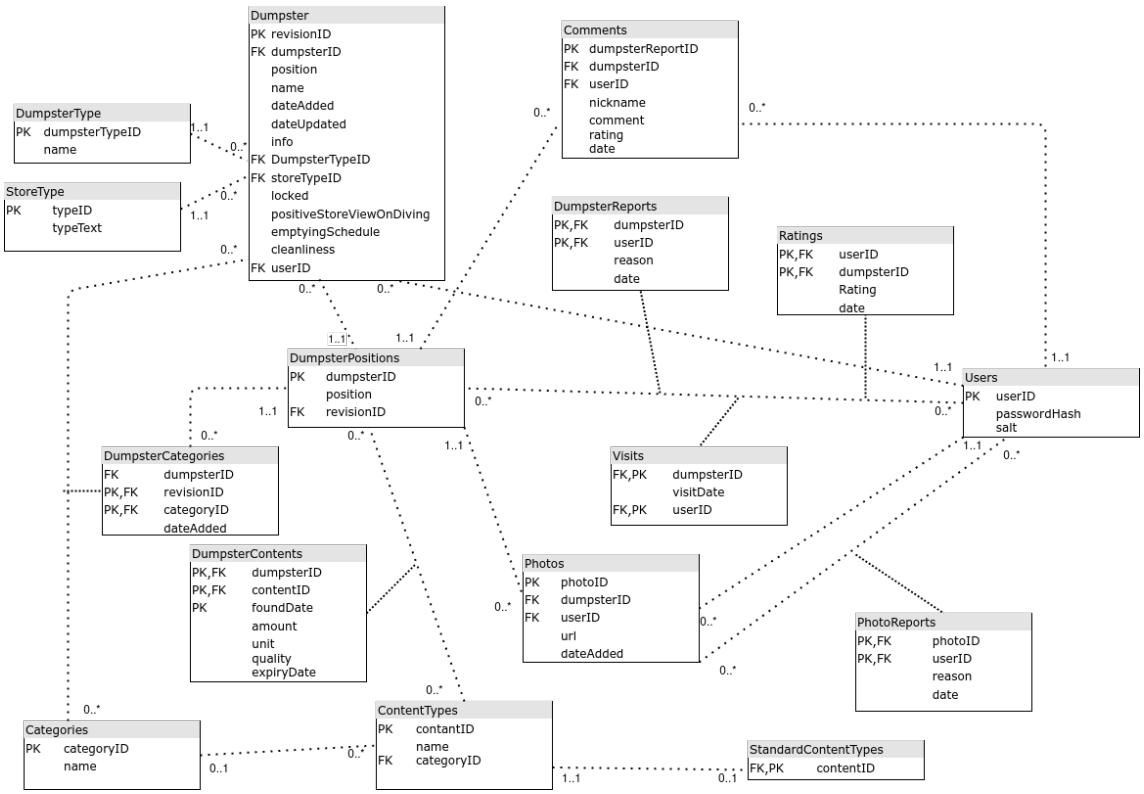


Figure C.5: ER diagram showing our product's database schema

C.6 Server services

This section contains a brief summary of the resources provided by our REST servers. We recommend reading the OpenAPI documentation found at `/api/spec` and `/pic/spec` on (development) instances of our backend.

C.6.1 API server

The API server handles storing, modification and retrieval of regular information, excluding actual image data. Its endpoint layout is shown in figures C.6 to C.8. All paths are prefixed with `/api`.

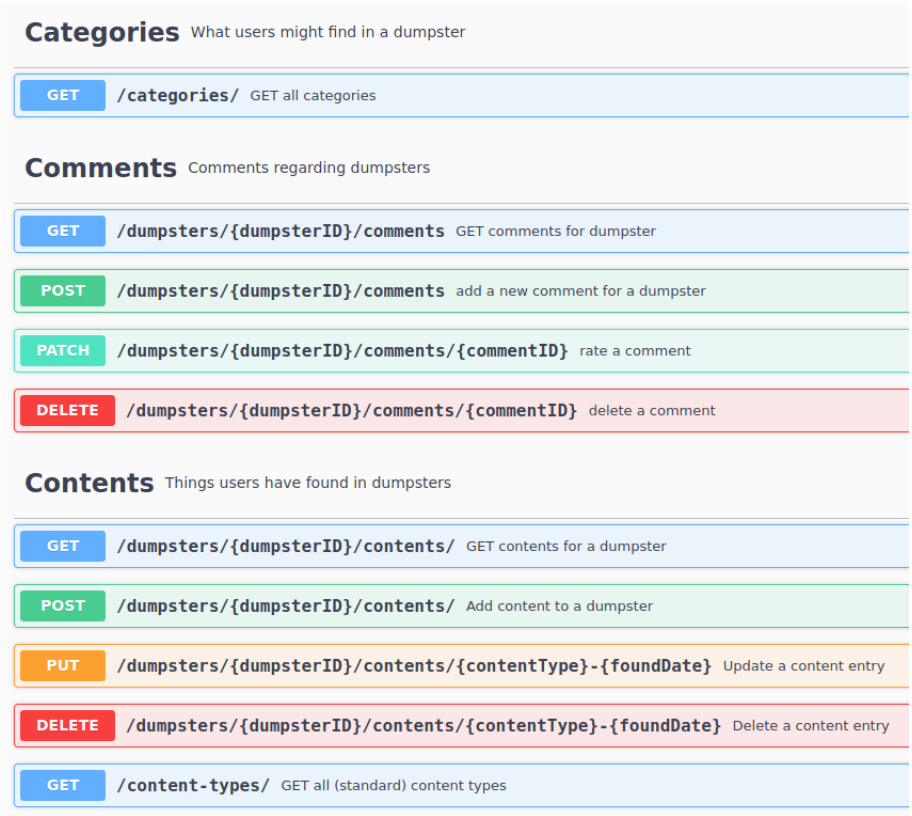


Figure C.6: API endpoints, part 1

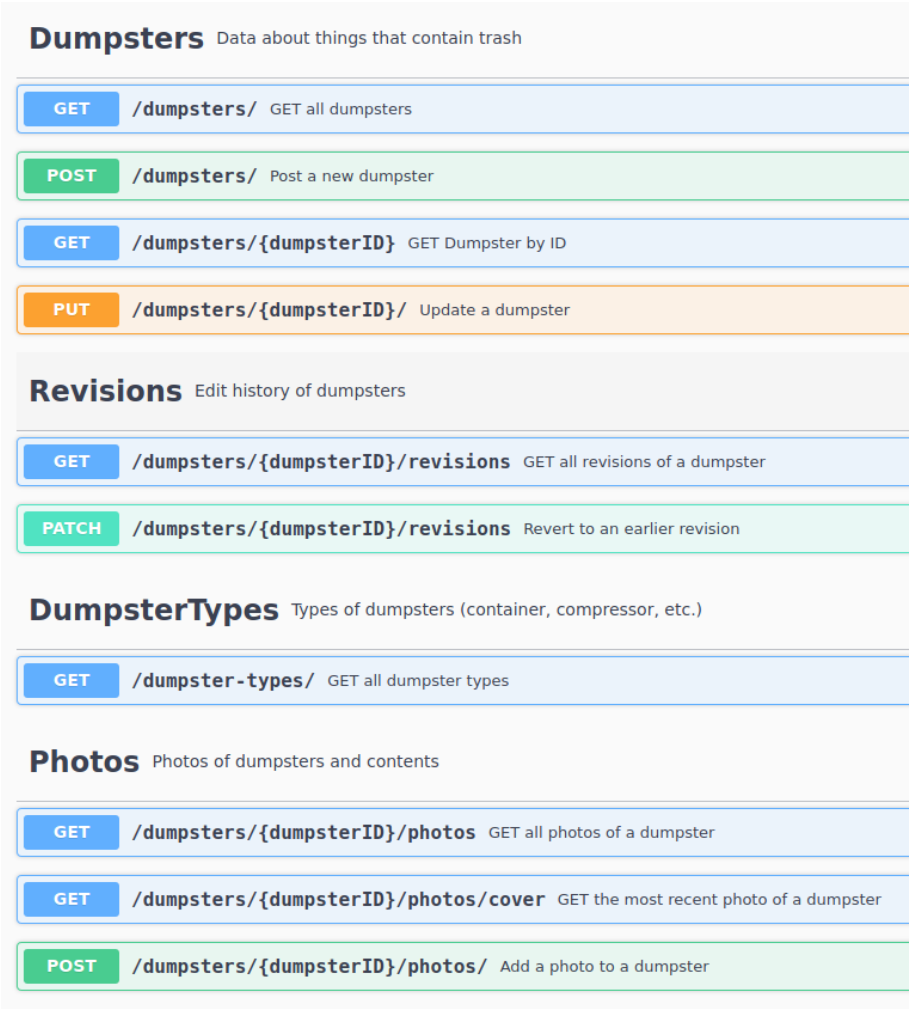


Figure C.7: API endpoints, part 2

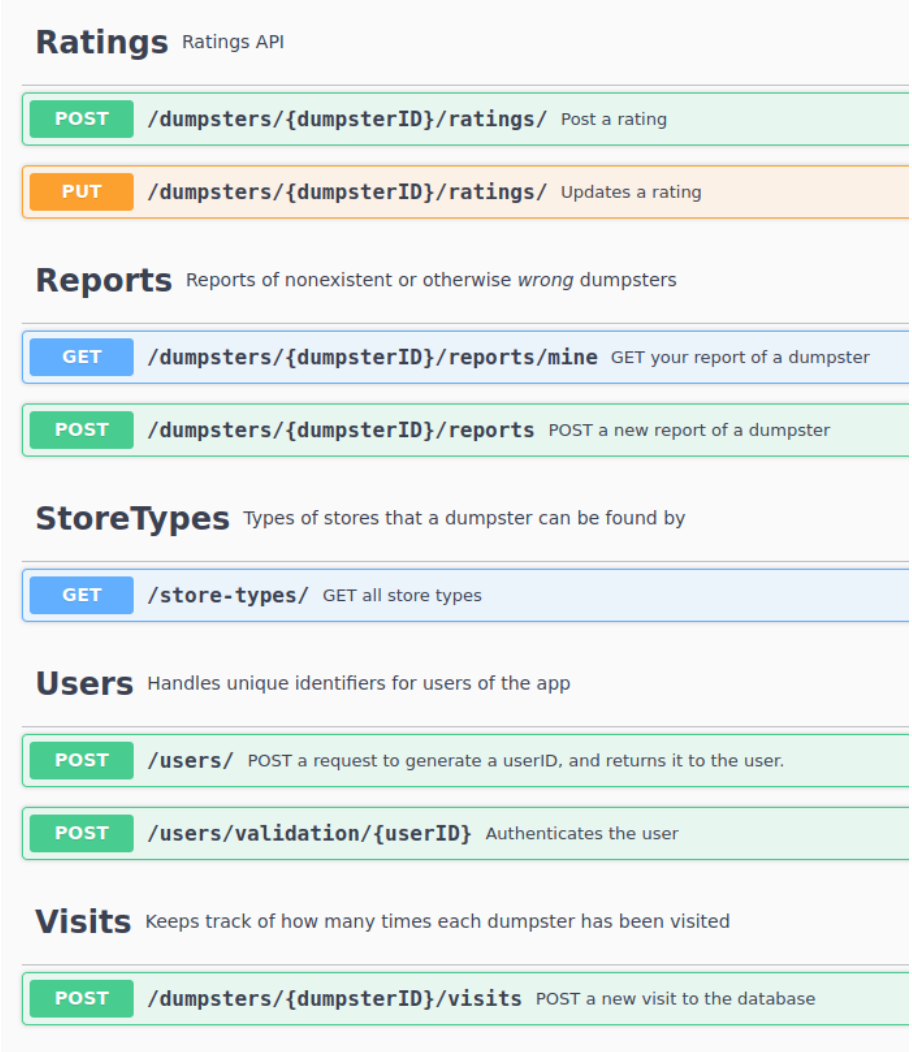


Figure C.8: API endpoints, part 3

`/categories/` All known categories, GET only

`/content-types/` Standard types of contents, GET only

`/dumpster-types/` Types of dumpsters, GET only

`/store-types/` Store types, GET only

`/dumpsters/` Information about specific dumpsters

`/dumpsters/xx/comments/` Thoughts about a dumpster

`/dumpsters/xx/contents/` Things found in a dumpster

`/dumpsters/xx/photos/` Photos of a dumpster or its contents

`/dumpsters/xx/photos/cover/` Cover photo to display if there's only room for one image

`/dumpsters/xx/reports/` Reports of a dumpster – it might not exist

`/dumpsters/xx/revisions/` Revision history for a dumpster

`/dumpsters/xx/visits/` Accepts POST requests to register visits to a dumpster

`/users/` POST to get your own user ID

`/users/validation/xx/` POST to authenticate and receive a token

C.6.2 Photo server

The API is shown in Figure C.9. It stores and distributes actual image files. Relatively uncomplicated, with only *two* endpoints.

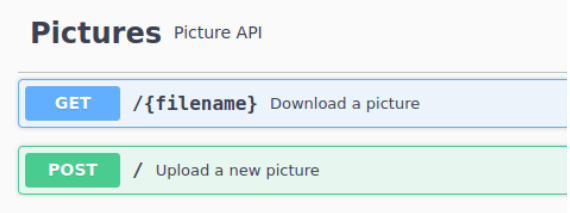


Figure C.9: Photo API

`/pic/` POST to add a photo

`/pic/xx.[jpg|png]` GET to download a photo

C.7 Security

C.7.1 Encrypted data transfer

The app communicates with our server over HTTPS, which ensures that no data except the host name or IP is transmitted in plaintext form — it ensures some degree of confidentiality. Our reverse proxy uses Certbot to get signed SSL certificates from Let's Encrypt [55]. We used Mozilla's suggested set of ciphers for moderately broad compatibility [56] and got a good rating in a SSL-Labs scan, see Figure C.10.

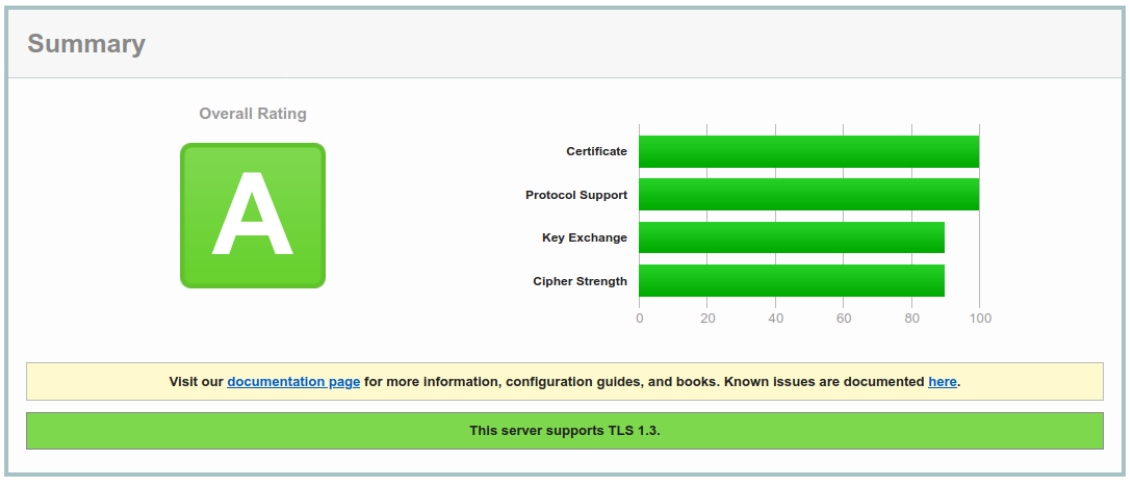


Figure C.10: SSL-Labs' rating of our server

C.7.2 Handling of user IDs

Because we didn't want to store any user information, we needed a way to identify users that wasn't email/username and password. The problem with using an email or username is that it can be used to identify the user, because people reuse usernames, or use usernames that might contain information that could potentially identify the user. We also wanted a system where you could potentially change device and still keep your old account. Something like a UUID would be difficult for the user to remember and tedious to write down. A simple numeric auto-increment ID on its own would be useful for identifying a user, but offers very little in terms of security, since it's fairly easy to guess. We ended up using a system similar to the seed phrases that cryptocurrency wallets use, which generates (in our case) 6 words from a list of 7776 words. Crypto wallets normally use 12 words, but the need for security isn't as important in our app, since all you really can do with your userID is delete images and comments that were made by your user. There is also the possibility of being able to track the user's past locations, since some information like comments, pictures and visits could potentially get accessed — however, even if the user was compromised it would be quite difficult to know who the user belongs to, since everything is generated pseudo-randomly. So we felt that 6 words struck a nice balance between being relatively easy to remember/write down, and still providing security.

We originally made it so that the first 4 words would be hashed naively to find the correct entry in the database and the whole 6 words was hashed with a random salt. This solution was more than enough in terms how many users could exist ($7776^4 = 3.65615844006 \cdot 10^{13}$), but left the security lacking since the naive hashes are vulnerable to rainbow tables, and if the hash was solved, you would only need $7776^2 = 60466176$ attempts to guarantee access to the user. So we removed this 4-word feature, and replaced it with a numeric ID with auto-increment to identify the row in the table as well as the 6 words as input to validate the user. There is currently no way to transfer the userID from an old phone to a new one, but it is one of the future features that are planned.

C.7.3 Tokens

A simple JWT (JSON Web Token) system was implemented to save time on authenticating users for every endpoint call that required validation. It is just a simple system that checks the database if the 6 word + userID combination is valid and creates a token based on the userID + the time it expires using a secret. The server then receives this token for all post requests, update requests or other requests where userID is relevant. The token is validated using middleware. There is a 30 minute active period as well as a one hour grace period where a new token is generated automatically and sent back to the user. Naturally it checks if the secret is correct when decoding the token.

The token does not really add much in terms of security, it just saves time that would have been used when authenticating the 6 words + userID. Tokens do not compromise the security much either, there is a risk of it being caught and used, and theoretically there is a risk of it being reused with the grace period constantly, however HTTPS does scramble the token so this is unlikely. There is no real risk of the secret being guessed, as it is a 128 character long random string, so people generating their own token is not really feasible.

C.7.4 SQL injection

Our API server uses Sequelize to access the database, a library which handles escaping of raw values in SQL queries well [57]. We ran some quick tests with sqlmap, a tool for testing how vulnerable a server is to SQL injections. No vulnerabilities were discovered.

C.7.5 Input validation

We made sure to validate all input the API receives through its endpoints, and validate the user's input in the app before it is sent to the API. Instead of writing our own functions for it, and potentially running into bugs, we used Joi, which simplified the process a lot.

C.7.6 Rate limiting

Another important concern in security is *availability*. To prevent our server from being overloaded by requests, we added rate limiting functionality, so that users (or bots) can only send

a limited number of requests from a given IP within a given timeframe.

C.7.7 Path traversal

This *could* have been an issue in the file server — however, each request to the GET endpoint has to have a filename that matches `[a-zA-Z0-9]+[.](jpg|png)`. There is simply *no way* for an attacker to traverse the server's folder structure when this restriction is in place. The only feasible way would be if that part of the Express library did not work properly.

C.7.8 Sensitive data exposure

The only piece of *potentially* sensitive information in this application, would be the anonymous identifier that could *theoretically* be used to identify a person, though only together with the data connected to it and a significant amount of external location data. This identifier connects users and data like ratings, comments and dumpster visits. However, the API *should not* reveal other people's identifier *in any case*.

C.7.9 Server security

We configured our firewall to prevent access to any ports other than those for SSH and the NGINX server itself. Password login was disabled, requiring your SSH keys to match those on the server in order to access it.

C.8 Installation and running

C.8.1 Running the backend as a developer

Prerequisites

For all parts of the system except the database, you need a (recent) version of npm installed. It is required for installing packages for Node.js and running the scripts that start the server and the like.

For running the database, you need to install MariaDB, e.g. by running `apt install mariadb-server` on a Debian-based Linux distro.

Database

The following instructions assume you are standing in the `backend/db` folder.

To build the image and start it:

```
make maria
```

Alternatively, to start a local MariaDB server on WSL:

```
make wsl
```

To run the setup script against the database, which creates the necessary tables and procedures:

```
make tables
```

To fill the tables with test data:

```
make data
```

NB: You might need to wait for a few seconds before executing this command. Additionally, *make sure* to have a `.env` file in the backend folder.

To clean up the containers:

```
make clean
```

Note for Windows users, or others who run into trouble: If these instructions do not work, try cloning the repo with LF line endings instead of Windows' default CRLF – or just look at the `Makefile` and setup script (`setup.sh`) and adapt the commands to your needs.

API server

The following instructions assume you are standing in the `backend/api` folder.

Create a `.env` file containing something like the following:

```
API_HOST=127.0.0.1
API_PORT=3000
DB_HOST=127.0.0.1
# and so on
```

`...` as specified in the `.env.template` file.

Note that you *must* use `127.0.0.1` and not `localhost` as the database host if you run it in a Docker container.

Depending on where you run the app, you may need to change the `PHOTO_URL` variable to match the IP the app makes contact with.

Run `npm install` to install dependencies, then run `npm start` to start the API server. *Make sure that your database is running*, otherwise the API server will crash after 10 unsuccessful connection attempts.

Photo server

The following instructions assume you are standing in the `backend/pics` folder.

Create a `.env` file containing something like the following:

```
PIC_PORT=3000
PIC_HOST=localhost
API_URL=http://localhost:3000/api/
# and so on
```

`...` as specified in the `.env.template` file.

Run `npm install` to install dependencies, then run `npm start` to start the picture server.

C.8.2 Backend deployment

Acquire a server with a recent Linux distro. Install rsync, Docker and Docker Compose, e.g.:

```
apt install rsync docker.io docker-compose
```

You should *set up SSH keys and disable password-based authentication, see section C.8.2.*

Create a user (e.g. `dumpster`) without administrative privileges, but in the `docker` group:

```
useradd --create-home dumpster
groupadd docker
usermod -aG docker dumpster
```

Let $SERVER_IP be your server's IP in the following snippets.

Transfer the contents of your repository to an appropriate folder in the dumpster user's home directory: (this assumes you stand in the root directory of the repo)

```
rsync --archive
      --exclude='.git'
      --exclude='node_modules'
      --exclude='.env'
      backend/ "dumpster@$SERVER_IP:dumpster"
```

Copy your API's .env template file and make a dynamic link to it:

```
scp backend/api/.env.template "dumpster@$SERVER_IP:dumpster/api"
ssh dumpster@$SERVER_IP ln -s dumpster/api/.env dumpster/.env
```

Then tweak it to fit the following pattern:

```
HTTPS=true
PROJECT_PATH=/home/dumpster/dumpster
# API server:
API_PORT=3000
API_HOST="<your server's domain or IP>"
NODE_ENV=production
TOKEN_SECRET="<some random, long string>"
# Photo server:
PHOTO_URL="https://<your server's domain or IP>/pic/"
# Database:
DB_NAME=dumpster
DB_USER=root
DB_PASSWORD="<your database password>"
DB_HOST=db
DB_PORT=3306
DB_DIALECT=mariadb
# Certbot:
EMAIL="<email of whoever wants to sign this>"
DOMAIN_NAME="<your server's domain or IP>"
```

Copy over the photo server's .env template

```
scp backend/pics/.env.template "dumpster@$SERVER_IP:dumpster/pics"
```

and tweak it a little as well – it should look like this:

```
PIC_PORT=3000
PIC_HOST=pics
PIC_URL="https://<your server's domain or IP>/pic/"
API_URL=http://api:3000/api/
PIC_FOLDER=/var/uploads/
PIC_MAX_SIZE=10000000
```

Ideally, you'd set up HTTPS for some extra security here, see section C.8.2 for instructions.

Copy over the `systemd` unit, reload the daemon and start the service:

```
scp backend/dumpster.service \
    "dumpster@$SERVER_IP:.config/systemd/user/"
ssh dumpster@$SERVER_IP systemctl daemon-reload
ssh dumpster@$SERVER_IP systemctl --user start dumpster
```

Wait a few seconds, then create the database tables:

```
ssh dumpster@$SERVER_IP "cd dumpster && make tables"
```

After this, the `.gitlab-ci.yml` file should make GitLab CI perform updates automatically after changes to `develop`. The server should be up and running, accessible from port 443.

SSH hardening

Using SSH keys and disabling password authentication are important security measures you may want to take. Specifically, generate an SSH key for your computer, add the public key to `.ssh/authorized_keys`, make sure you can log in without a password, and finally disable the `PasswordAuthentication` option in the SSH config (and perhaps disable `PermitRootLogin` as well).

You can also install `fail2ban` and run it with a basic configuration like this:
(in `/etc/fail2ban/jail.local`)

```
[DEFAULT]
; a rather strict penalty
bantime = 1d

[sshd]
enabled = true
; since we use ufw, make fail2ban use it too
banaction = ufw
; (this could be a bad idea)
ignoreip = <your server's IP>
```

SSL certificates

The HTTPS setup detailed in this section was influenced by a Digital Ocean tutorial.

In order to secure the connection between the app and your instance of the server, you should acquire an SSL certificate and enable HTTPS. Our setup uses `certbot` to get certificates signed by Let's Encrypt, for no cost at all. You *do* need a server that is available to the outside world, otherwise the servers of Let's Encrypt won't be able to access your server.

Create a Diffie-Hellman key in the `backend` folder:

```
mkdir dhparam
openssl dhparam -out dhparam/dhparam-2048.pem 2048
```

Comment out the second server block in the NGINX config, and uncomment the parts of the first server block that are indicated by other comments. Since you do not yet have a certificate, everything must happen through HTTP. Let's Encrypt needs to be able to query for your challenge file.

```
http {
    # (...)

    server {
        # (...)

        # (uncomment when first acquiring SSL cert)
        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;

        location ~ /.well-known/acme-challenge {
            allow all;
            root /var/www/html;
        }

        # (uncomment when first acquiring SSL cert)
        location / {
            allow all;
        }

        # (...)
    }

    # (comment out when first acquiring SSL cert)
    # server {
    # (...)
    # }
}
```

Start the service and check the logs with `docker-compose logs certbot`. If no issues crop up, proceed.

Now that you *do* have a certificate, revert your changes to `nginx.conf` and restart the service. It should be possible to access your server through a normal web browser. Confirm that your connection is encrypted.

To renew your certificate automatically, add an entry in your crontab (with `crontab -e`):

```
0 6 * * * PROJECT_PATH=/home/user/dumpster-finder /home/user/dumpster-finder/renew_certs.sh >>
```

(the `PROJECT_PATH` is required to let the script navigate into the correct folder)

C.8.3 Running the app

To run the app with connection to an instance of our backend, you need to specify the address to the server. **Make sure the server is running.**

Create a `.env` file with variables like those set in `.env.template` – it might look like this:

```
API_URL=http://xxx.yy.zz:3000/api/
PIC_URL=http://xxx.yy.zz:3001/pic/
```

Or like this, if you have a domain name and are running a proper instance with HTTPS:

```
API_URL=https://your.domain/api/
PIC_URL=https://your.domain/pic/
```

Then start the app in development mode:

```
npm start
```

Now you should be able to connect an emulator or a device to the Expo server.

C.8.4 Publishing the app

Create an Expo account and run `expo build:android` or `expo build:ios` to build and publish the app. On subsequent deployments, you should just need to do `expo publish` to update the JS bundle, since the native code should stay more or less the same.

C.9 Documentation of source code

C.9.1 Source code documentation

We used Typedoc to generate documentation for our TypeScript files, based on types and TSDoc comments. During development, you would for the most part use your IDE's way of viewing documentation. Documentation can be generated for each part of the project by entering its corresponding folder (`backend/api`, `backend/pics`, and `frontend`) and running `npm run docs` (assuming you've run `npm install` first). This command should generate a folder called `docs`, in which you'll find the generated documentation. Open `docs/index.html` in a web browser to view it.

The most recent documentation is also available at GitLab Pages:

- API server docs
- Photo server docs
- Frontend docs

For the actual repository with all our code, see

- NTNU's GitLab (with CI)
- GitHub (without CI, publicly available)

C.9.2 API documentation

We used Swagger to document our API, which entailed writing comments with OpenAPI specifications for each endpoint. The API documentation is generated each time you run the API or photo server, and is available at `http://<your domain or IP + port>/api/spec` and `http://<your domain or IP + port>/pic/spec`, respectively.

C.10 Continuous integration and testing

C.10.1 CI pipeline

We use GitLab CI and run our CI jobs inside the official Node.js and MariaDB docker containers.

Figure C.11 shows a successful run of our pipeline. The amount of jobs differs depending on the branch a commit was pushed to. On pushes to master or develop, documentation is generated and the current version of the system is deployed to our test server and published through Expo. On other branches, those stages are dropped. In all cases, the dependency, test and audit stages run. Dependencies are only cached for the API server, since the app has dependencies that are too large to fit in the cache, and the photo server is not tested. The test stage runs unit tests and GitLab's check for exposed secrets (passwords, API keys, etc.). The audit stage uses Auditjs to scan our project's dependencies for known vulnerabilities.

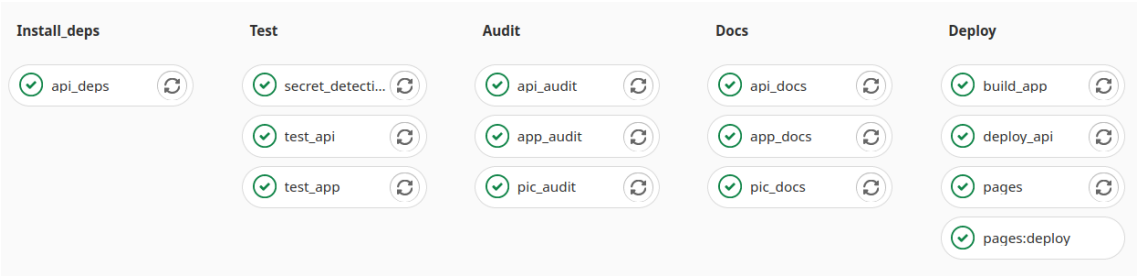


Figure C.11: Screenshot of a successful pipeline run

C.10.2 Tests

To check for possible regressions or malfunctioning additions, we run some tests on each push. We run unit tests of the DAOs in the API server, and some functionality in the app. However, we do not run any tests of the app's UI components. Unfortunately, we did not have time to debug our problems with testing React components in the frontend. This is an issue we've faced before, but it seemed to be even more challenging with React Native and our choice of component library.

In total, we had around 95% coverage of the files we *did* test.

To run the unit tests yourself, stand in either `frontend/` or `backend/api/` and run `npm test`, after having installed dependencies and set up your environment as specified in C.8.3 and C.8.1.

The easiest way to test the API's functionality is to open the Swagger documentation in your web browser and go through each endpoint. We decided to avoid writing tests for the API itself, since most of the functionality stems from the DAOs and Swagger makes manual testing easy.

Appendix D

User tests

D.1 Plan

1. Sett posisjonen din
2. Hvor finner du infoen igjen?
3. Filtrer vekk dumpstere som er kompressorer
4. Se informasjon om en dumpster
5. Legg til en kommentar
6. La dumpsteren være låst
7. Legg til et bilde
8. Fjern et bilde
9. Rediger posisjonen din
10. Hvor tenker du at du ville gått for å lage en ny dumpster?

D.2 Brukertest #1

Subject: En student som ikke er videre interessert i å begynne med dumpster diving

- Gode tips

1. Sett posisjonen din
    - Skrevet i location, søkt opp, trykt
    - Måtte si fra om select
2. Hvor finner du infoen igjen?

- Ville tenkt kanskje info, da
- Litt symbolforklaring, positivt

3. Filtrer vekk dumpstere som er kompressorer
   - Går til settings
   - Spesifiserer akkurat nå, ikke permanent
   - Finner filter
   - Velger type bare dumpster
   - Satt ut av at uiet ikke endrer seg
   - Trykker på filter for å gå ut
4. Se informasjon om en dumpster
   - Trykker på den umiddelbart
   - Tanker om info:
     - Detaljert
     - Rangering, åpen, når den tømmes, hvor ren den er, innhold etc.
     - Contents-*boksen* var litt rar, utydelig om det var utløpsdato eller funndato
5. Legg til en kommentar
   - Hovra over endringsknappen før han tok rett
   - trykka rett inne på kommentarfeltet
6. La dumpsteren være låst
   - går tilbake til info, til edit, trykker på togglen
7. Legg til et bilde
   - trykker på rett knapp osv.
8. Fjern et bilde
   - Rett knapp naturligvis
   - spurte om skifting av bilde → fint med piler
9. Rediger posisjonen din
   - går til kart, trykker på den posisjonsknappen
   - sier at det er et annet sted → innstillinger
   - mest intuitivt å gå til kartet
10. Hvor tenker du at du ville gått for å lage en ny dumpster?
    - ville tenkt list, nei, ville gått på innstillinger (?)

    - plusstegn på kartet? kunne vært det også

- oversiktlig, ikke for mye informasjon, greit at det er enkelt og uten for mye
- mulig å ta den i bruk hvis jeg noen gang dumpster diver
- andre kommentarer: kompressorer annet ikon (det har vi tenkt på! yay!)

D.3 Brukertest #2

Subject: En student som selv har gjort brukertesting før

1. Sett posisjonen din
    - location og search → her søker du, og så knappen.
    - ser at det er en del spots → forrvirrende prototype, litt
    - hva er pin'ene? dumpsters som er tilgjengelige, vet ikke om det betyr at de har noe ferskt, eller om de bare er åpne for tømming
2. Hvor finner du infoen igjen?
    - info, jada, kjenner det igjen
3. Filtrer vekk dumpstere som er kompressorer
    - går til map, filterknappen, trykke på kategori, så ser han type, ønsker ikke kompressor → dumpster, den andre mulige (ops, knappen lukker ikke overlayet)
4. Se informasjon om en dumpster
    - vilkårlig
    - trykker på en pin, ser at det er info
    - "Moholt til og med i USA"
    - tanker om info:
        - tags for type mat eller avfall
        - står contents, med dato på når det ble lagt inn eller går ut (!!!)
        - rating, antatt brukergjort
        - åpen, greit å vite
        - hver tirsdag → når den tømmes
        - aggregering av brukeres rating
        - cleanness → er det brukerrangering eller butikk?
        - ser ut som det er mulig å se flere bilder ELLER se andre dumpstere
        - ser også at jeg kan flagge → favorittmarkering
5. Rapportere dumpster

- lurer på om det er *det* flagget er til → rett til rett sted

6. Legg til en kommentar
   - noe som ser ut som lag en ny kommentar → går inn på edit
   - tekstfelt han kan redigere → contents → ooooops → trykker save
   - du har ikke lagt til kommentar, sier jeg fra om
   - litt usikker på om han kan trykke på comments siden den er gråa ut
   - *her* ser han at han kan legge inn kommentarer
   - – trykke på post-knappen
   - "15 burgere, den var fin"
7. La dumpsteren være låst
   - open er en padlock, han er inne på details view → feil
   - går inn på listevisning, ser også et padlock-symbol, antar det er feil igjen
   - prøver å gå til settings, eh, hm, kanskje map igjen og så tilbake til dumpster og så rediger
   - glidebryter → sette til locked
   - hva er positive attitude? personlig markering av at du selv liker den → ooppps
8. Legg til et bilde
   - add photo → choose file → velge bilde → upload
9. Fjern et bilde
   - delete, naturlig → delete photo, usikker på om det er bildet han har valgt eller hva det er, antar til slutt at det er det som er i bakgrunnen nå
   - hvordan bla? piler.
10. Rediger posisjonen din
    - tilbake, tilbake, ser det er en sånn posisjonsknapp på kartet
    - antar at det kommer opp en markør eller noe
    - alternativ måte, hvor er den? settings → location → change, yes
11. Hvor tenker du at du ville gått for å lage en ny dumpster?
    - enten map eller list, kanskje velge posisjon der du kan velge å opprette dumpster
    - på list ser han ikke noen mulighet for det
    - trykke på noe ekstra, altså

- virker nice, generelt intuitivt kartbasert app
- intuitiv plassering av dumpsters som POIs

- savner symbol for forskjellige dumpstere
- filter var jo greit, kommer ikke på noe anna her
- listevisninga er grei
- få opp de som er i nærheten i lista
  - men det gjør den jo
- ha en sånn mulighet for å favorite dumpstere
  - hvor ville lista vært?
  - filtrere på det i map og list
  - usikker på om han ville hatt egen fane med favoritter
  - egentlig fint som default i map og list, eller sette om det skal være default i settings
    - min tanke: defaultfilter av alle slag i settings, kanskje?
- på noen av punktene med info er det utydelig
- både utløpsdato og innleggingsdato
- tags er fint
- positive attitude er uklart
- cleanness → bruker eller butikk-info?
  - idé fra meg: det kan klarlegges på starten, I guess
- funksjonalitet for å markere det du har tatt ut, for å oppdatere infoen
  - skal det være tatt ut eller det som er igjen?
- han her tror han kjenner en del som har drevet med dumpster diving
- enda en liten ting: nei, det var ikke noe
- design fint
- vil du ha mer fargerikt? ja, hadde vært fint med konsekvent fargepalett
- mye er åpenbart responsivt design
- animasjoner, nevner han (drag-and-drop i listevisning gir ikke mening, men ok)
- i *too good to go* er det greit at det er i nærheten, på dumpster diving er det greit med større radius
- overrasket over at vi har fått til så mye bare i løpet av januar

Appendix E

Interviews

E.1 Disclaimer

This document is written in a mix of English and Norwegian. That was a natural consequence of doing interviews with both English- and Norwegian-speaking dumpster divers. Additionally, we have made no effort to clean this document up any more than what was absolutely necessary — our supervisor said we did not need to.

E.2 Tidsplan

2 min: Fortelle litt om oss og forklare hva poenget med appen er.

7 min: vise dem app/wireframes

5 min: annet/ fri spørsmål

E.3 Spørsmål i undersøkelsen

Do you research a new dumpster before you go dumpster diving in it? / Gjør du research om en ny dumpster før du går dumpster diving i den? (kvantifiserbart)

What sorts of information do you need about a dumpster before you go diving in it? / Hva slags informasjon trenger du om en søppeldunk før du dra og dykker etter søppel?

What information is important for you when you're diving? What would you like to know?/ Hva er viktig info når man skal dive? Hva ønsker du å vite?

What are the most important qualities of a dumpster? / Hva er de viktigste kvalitetene til en dumpster?

Are you willing to share information about the dumpsters you dive in with others? Why? Why not? / Ville du delt informasjon om dumpsterne du diver i med andre? Hvorfor/hvorfor ikke?

Is privacy important for you when it comes to dumpster diving? / Er personvern viktig for deg når det gjelder søppeldykking?

Do you think privacy is important to other dumpster divers? / Tror du personvern vil være viktig for andre søppeldykkere?

Link to survey: *https://forms.gle/5PtADojkQKhQKVTi9*

Link to wireframes: *https://www.figma.com/proto/(...)*

E.4 Test #0

E.4.1 Oppgaver: (stopp når tida er ute)

1. *Sett posisjonen din (guiden dukker opp, nb)*
2. *Se informasjon om en dumpster*
3. *Gi den en rating*
4. *Legg til en kommentar*
5. *Rediger dumpsteren (typ sett den til å være låst eller noe)*
6. *Gå tilbake til en tidligere versjon av dumpsteren*
7. *Legg til en ny dumpster*

   A little hard to find, plus button bottom right
8. *Legg til et bilde til denne dumpsteren (funker ikke i appen)*
9. *Endre posisjonen din*

   Clicked the target icon on the map

E.4.2 Spørsmål til samtalen etterpå:

What do you think of the concept for this app? Would you use it? / Hva syns du om konseptet vårt? Ville du brukt denne appen? (burde dekke spesifikke ting de (mis)liker)

Yeah probably, if it had information

What would you like to be able to do in a dumpster diving app? Is there anything we might have missed? / Hva slags ting ville du hatt mulighet til å gjøre i en app som denne, som ikke allerede er en del av den?

If i am a user, i would wonder if i was able to be notified if a dumpster has been altered.

Is there anything in the app that you don't think is relevant? / er det noe i appen du føler ikke er relevant? (the information we do have is already shown)

Info screen confusing, it's sort of a legend, the icons in particular

If you could change one thing about the design, what would it be? Would you use this app if that change was made? / Hvis du kunne endre ett aspekt av designet, hva ville det vært? Ville du brukt appen hvis denne endringen ble gjort?

Use a more up to date UI, Material so it looks like other apps

Slide menu that's always visible
Bar to add functionality

A lot of information in the dumpster page.

E.5 Test #1

Technical difficulties strike again... or not

E.5.1 Oppgaver: (stopp når tida er ute)

1. *Sett posisjonen din (guiden dukker opp, nb)*
2. *Se informasjon om en dumpster*
3. *Gi den en rating*
4. *Legg til en kommentar*
5. *Rediger dumpsteren (typ sett den til å være låst eller noe)*
6. *Gå tilbake til en tidligere versjon av dumpsteren*
7. *Legg til en ny dumpster*
8. *Legg til et bilde til denne dumpsteren (funker ikke i appen)*
9. *Endre posisjonen din*

E.5.2 Spørsmål til samtalen etterpå:

What do you think of the concept for this app? Would you use it? / Hva syns du om konseptet vårt? Ville du brukt denne appen? (burde dekke spesifikke ting de (mis)liker)

What would you like to be able to do in a dumpster diving app? Is there anything we might have missed? / Hva slags ting ville du hatt mulighet til å gjøre i en app som denne, som ikke allerede er en del av den?

Is there anything in the app that you don't think is relevant? / er det noe i appen du føler ikke er relevant? (the information we do have is already shown)

Kun ha **litt mer spesifikasjn for hva slags dumpsterType**

Tidspunkt er nyttig, og fjerne gamle kommentarer siden de

If you could change one thing about the design, what would it be? Would you use this app if that change was made? / Hvis du kunne endre ett aspekt av designet, hva ville det vært? Ville du brukt appen hvis denne endringen ble gjort?
Virker greit, siden det er informasjonside

One thing as a concept is that people share a lot in [REDACTED], and social groups. No longer really a thing in 2016. Dumpster diving facebook groups

Dumpster diver google map, nobody renewed it.

People should respect the lock

Also nice that we have store view on diving.

Different in Berlin, no longer

At night check what type of store, see if it's open. And see maybe a new picture

Questions behind finance.

Recommend not to dive into a compressor

Cleanliness should be taken care of by the divers. They should leave it as clean as they find it

Noen steder organiserer seg bedre

Student

Mulig å ha en **checkin** function for å se hvor mange som har vært der.

Hover over funktionalitet

Mulig å dele inn butikker etter fordelingstasjon, siden lignende butikker har samme distrubitør med samme kastedatoer

Ikke skriv ting for kontents, Foretrekker buttons med hvor lenge de har igjen, eller lignende.

Alt du finner i container er i eget ansvar i info siden

Visibility *of dumpsters* som mulig variabel, noen vil ikke bli sett av andre

Eller at de så gjemt at noen kan være gjemt

Vanntett konteiner, eller ikke fordi de kan bli fylt med vann over tid

Kanskje ta kontakt med **folkekjøkken**, som har direkte kontakt med butikker

Territorielle er som regel ikke et stort problem i Norge, kulturelt kan det være store forskjellier fra land og kulturer på hvordan man skal behandle det. Shadowsociety, generelt små

grupper.

E.6 Test #2

E.6.1 Oppgaver: (stopp når tida er ute)

1. *Sett posisjonen din (guiden dukker opp, nb)*

   Clicks in the bloody input field, of course

2. *Se informasjon om en dumpster*

   Goes straight to it

3. *Gi den en rating*

   Clicks on the star things

4. *Legg til en kommentar*

   Comment on the info box... oh, there's comments below there

5. *Rediger dumpsteren (typ sett den til å være låst eller noe)*

   Goes to da menu → edit, clicks around

6. *Gå tilbake til en tidligere versjon av dumpsteren*

   Revision history, yes

7. *Legg til en ny dumpster*

   Go to the map, sees the + icon, sets pos, sets info

8. *Legg til et bilde til denne dumpsteren (funker ikke i appen)*

   Click on the picture itself... Or edit dumpster → add photo

9. *Endre posisjonen din*

   Goes to the map, clicks the position icon there (of course)

E.6.2 Spørsmål til samtalen etterpå:

What do you think of the concept for this app? Would you use it? / Hva syns du om konseptet vårt? Ville du brukt denne appen? (burde dekke spesifikke ting de (mis)liker)

Has a lot of potential, especially where they live w/o access to dumpsters in [REDACTED] except in the weekends, will be good to see crowdsourced knowledge about dumpsters

What would you like to be able to do in a dumpster diving app? Is there anything we might have missed? / Hva slags ting ville du hatt mulighet til å gjøre i en app som denne, som ikke allerede er en del av den?

Maybe like listings on Finn.no, like a list of hot items, e.g. found 10 boxes of coconut oil & want to shout it out. Then sounds positive to our implementation. An **indication of whether anyone has claimed it**---

Is there anything in the app that you don't think is relevant? / er det noe i appen du føler ikke er relevant? (the information we do have is already shown)

Kun ha **litt mer spesifikasjn for hva slags dumpsterType**

If you could change one thing about the design, what would it be? Would you use this app if that change was made? / Hvis du kunne endre ett aspekt av designet, hva ville det vært? Ville du brukt appen hvis denne endringen ble gjort?
(nothing specific)

E.7 Test #3

E.7.1 Oppgaver: (stopp når tida er ute)

1. *Sett posisjonen din (guiden dukker opp, nb)*

   Reagerte på "ikke gå alene"

   Går til settings, ignorerte at det sto i guiden

2. *Se informasjon om en dumpster*

   Gikk rett til details

3. *Gi den en rating*

   Trykker rett på <stjerne> Rating

   Så ser hen stjernene der nede

   (hen så ikke andre del så godt)

4. *Legg til en kommentar*

   Legger merke til det info-feltet as usual

   Hen har lyst til å skrolle ned her

   Så går hen til kommentarfeltet

5. *Rediger dumpsteren (typ sett den til å være låst eller noe)*

   Redigert den? Eeeeeh. . . Virker litt i tvil

   Trykker på contents, går tilbake

   Hva kan man redigere? (osv.)

   Måtte lede hen til burgermenyen

   Hen så det som mer naturlig å trykke på noe mer nede/i midten

   "Det blir jo på en måte rating, da, men med flere mulige ratinger"

6. *Gå tilbake til en tidligere versjon av dumpsteren*

   Da trykker hen seg inn på menyen (siden vi nevnte den)

   Så hadde hen sett historikk, jadda

7. *Legg til en ny dumpster*

   Går til kart, liste, ser plusstegnet, kommer dit

   “unlocked , locked” Noen er **av og til åpen, av og til lukket**

   Forslag: Hvor mange ganger åpen, hvor mange ganger lukket

8. *Legg til et bilde til denne dumpsteren (funker ikke i appen)*

   Enda en person som trykker direkte på bildet!

   Eeeeeller gå på edit og trykke der

9. *Endre posisjonen din*

E.7.2 Spørsmål til samtalen etterpå:

What do you think of the concept for this app? Would you use it? / Hva syns du om konseptet vårt? Ville du brukt denne appen? (burde dekke spesifikke ting de (mis)liker)

Ja, hvis hen skulle på en ny plass der hen ikke hadde vært før.

Vet ikke om hen ville brukt den på et sted hen har vært før

Er ikke typen til å logge ting

Føles ikke naturlig å logge ting på telefonen

Har dokk vært med i den dumpsterdiving-gruppa, der har noen lagd et sånt kart

Hadde vært nyttig, men vanskelig å holde det oppdatert

What would you like to be able to do in a dumpster diving app? Is there anything we might have missed? / Hva slags ting ville du hatt mulighet til å gjøre i en app som denne, som ikke allerede er en del av den?

Hvor mange som har vært der (gjerne graf over når det er mange folk der, eller bare en enkel popularitetssatistikk)

Is there anything in the app that you don’t think is relevant? / er det noe i appen du føler ikke er relevant? (the information we do have is already shown)

Infoen er ikke så nyttig, og det står ikke overskrift.... Tror hen hadde skjønt det. (gjelder ikonforklaringen)

Heller ha info om hvordan du legger til en ny o.l.

If you could change one thing about the design, what would it be? Would you use this app if that change was made? / Hvis du kunne endre ett aspekt av designet, hva ville det vært? Ville du brukt appen hvis denne endringen ble gjort?
Teksten på kartet er ganske liten (eeeh, det er wireframen)

Kjempekjekt

Har ikke noen tanker som hen kommer på

Åpner sorteringsboksen

Ser at vi kan sortere på andre ting enn mat også (nevner ulovligheten igjen)

Strengt tatt er jo all dumpsterdiving ulovlig (andres eiendom), men ingen blir tatt for det

Forslag til råd: Enkelt og greit: **på papiret tyveri, men det går til en god sak**

E.8 Test #4

E.8.1 Oppgaver: (stopp når tida er ute)

1. *Sett posisjonen din (guiden dukker opp, nb)*

   Virker litt ustødig

   Jesus, hvorfor valgte jeg denne oppgaven? Det er min feil.

   Vi måtte forklare at dette bare er startsiden, og (kanskje viktigere) hva som menes med posisjon... Ops.

   Osv.: ville satt lokasjon til X, men skal jeg snakke om dumpstere?

   Ops ops ops ops ops ops ops

2. *Se informasjon om en dumpster*

   Ville jo sjekka ut dumpsteren, og hvis andre dumpstere kunne dele hva de hadde oppdaga tidligere på dagen (ville sett det), ville delt det selv (hvis det var mulig). Ville sjekka ut alt i nærområdet (hvis noe var av interesse?).

   Trykker på en markør på kartet, naturligvis

   Ser jo egentlig veldig greit ut ift. at det står oppført kjøtt, krydder, frukt...

   Nevner godteri, snacks (vi har vel det som kategorier? >_>)

   Ville sett på eventuelle nylige kommentarer

   Pga. frukt og grønnsaker ville hen valgt å ta turen ned

   Påpeker "The shop's view on dumpster diving" som en nyttig informasjonsbit

   Fan av dark mode (yay)

3. *Gi den en rating*

   "Da tenke æ ... asså, æ e jo ikke så teknisk anlagt og har ikke peiling på sånt"

   Da er det snakk om dumpsteren den dagen, eller generelt? (→ kanskje vi bør tenke over dette...)

   Trykker på det innlysende stedet, ville gitt den en treer

4. *Legg til en kommentar*

   Trykker på comments, trykker på skrivefeltet, jadda!

5. *Rediger dumpsteren (typ sett den til å være låst eller noe)*

   Contents, kanskje? (nei, hvis du ville redigert *innholdet*, da hadde du gått dit)

   (Vi spesifiserer hva som skal endres)

   Hamburger time! → edit, korrekt

6. *Gå tilbake til en tidligere versjon av dumpsteren*

   Hamburger → revision history → Vi skipper, ops

   Spm. om struktur: Ville ikke kommet på like mye bra selv, men: Skulle ønske en liten notis til nye divere om at de vennligst ikke skal rote og ikke plage butikkeierne (**gå etter stengetid**) altså. Dagtid er iffy.

   Eventuelt også legge inn om en har en dårlig erfaring (eks.: det er ei koko dame ved Bunnpris som har et hat mot dumpster divere)

   Blomstre, pyntegjenstander, vaskemiddel, andre husholdningsgjenstander, øl osv.

7. *Legg til en ny dumpster*

   Går tilbake til kartet, trykker på filtreringsknappen, neivel, trykker på + endelig (satt fast på set position)

   Konkluderte med at det var intuitivt og at hen ville funnet fram

8. *Legg til et bilde til denne dumpsteren (funker ikke i appen)*

9. *Endre posisjonen din*

E.8.2 Spørsmål til samtalen etterpå:

What do you think of the concept for this app? Would you use it? / Hva syns du om konseptet vårt? Ville du brukt denne appen? (burde dekke spesifikke ting de (mis)liker)

Ville absolutt brukt den. Litt sånn at ikke alle vil dele hvis de snubler over ei gullgruve, men er også mange som vil spre godene (og kunnskapen).

What would you like to be able to do in a dumpster diving app? Is there anything we might have missed? / Hva slags ting ville du hatt mulighet til å gjøre i en app som denne, som ikke allerede er en del av den?

Nei, en annen ting: **Vegetariske og veganske produkter**, oversikt over det → flere kategorier/tags for det, som veganer ville hen bare sagt fra at det lå masse ribbe der.

Is there anything in the app that you don't think is relevant? / er det noe i appen du føler ikke er relevant? (the information we do have is already shown)

Komplimenterer tipsene på infosida

Kommer ikke på noe hen ville endret

If you could change one thing about the design, what would it be? Would you use this app if that change was made? / Hvis du kunne endre ett aspekt av designet, hva ville det vært? Ville du brukt appen hvis denne endringen ble gjort?
Tror ikke det, er ikke noe tech-savvy, men ser ikke ut som man må være særlig god for å få det til.

Andre kommentarer?

Eh, hm, øh, nei

Vil gjerne være med på å teste.

Hadde det vært mulig å få opp **notification** hvis noe man har markert (en konteiner, noen innholdspunkter) er funnet (i)?

E.9 Test #5

E.9.1 Oppgaver: (stopp når tida er ute)

1. *Sett posisjonen din (guiden dukker opp, nb)*

   (we see them slide the cursor above allll the text)

   Yeah, you can't type anything

2. *Se informasjon om en dumpster*

   Clicks on the damn marker, sees information

   This looks very simple, I believe there's supposed to be an image here and what people have found here

3. *Gi den en rating*

   Goes down to the rating thing

4. *Legg til en kommentar*

   Clicks on the correct burton

   (notices the contents)

5. *Rediger dumpsteren (typ sett den til å være låst eller noe)*

   Press emptying schedule, then press hamburger → edit dumpster

   Quite intuitive actually

   If we'd add anything... We'd add ***when the shop closes*** so that we can avoid going there when it is open

6. *Gå tilbake til en tidligere versjon av dumpsteren*

   Goes back to details, we have to explain what we mean, then they go to revision history, notice the revert button

7. *Legg til en ny dumpster*

   OMG they find the plus icon immediately

   Set position → information, yes

8. *Legg til et bilde til denne dumpsteren (funker ikke i appen)*

   CLICKS ON THE PICTURE of course

   Then goes to ze hamburger

   Anecdote about shop

9. *Endre posisjonen din*

   Clicks search on the map screen (whoooooooooooooops), then goes to settings, sees the location thingy

E.9.2 Spørsmål til samtalen etterpå:

What do you think of the concept for this app? Would you use it? / Hva syns du om konseptet vårt? Ville du brukt denne appen? (burde dekke spesifikke ting de (mis)liker)

As a new dumpster diver, *yes*. But for veteran divers... We just go to some dumpsters we already know. Also not into it as a lifestyle, won't contribute too much to it. They do meet up with folks and like the social aspect.

They know somebody who *would* be excited to use this, though.

...poison in the food, jesus christ... →

Regarding the survey, some questions were too limited:

What source of information do you need b4 diving? **Nah, we just go to a dumpster and look inside it.**

Important qualities – not entirely sure what this question means. We explain. They wrote that it was important that it had food, shape is irrelevant, smell is... bad?

Importance of privacy – people are different, it depends. Oh, they thought it would be related to meeting people... Whoops. Explanation, then: They wouldn't like people to know that. They'd answer honestly, but wouldn't share publicly anywhere. Also something about shop owners.

What would you like to be able to do in a dumpster diving app? Is there anything we might have missed? / Hva slags ting ville du hatt mulighet til å gjøre i en app som denne, som ikke allerede er en del av den?

No, I think it's very good. This is exactly what we'd want to share.

They'd like to add a rule: Keep the place around clean. (is this covered by our current advice?)

Is there anything in the app that you don't think is relevant? / er det noe i appen du føler ikke er relevant? (the information we do have is already shown)

No, I wouldn't add or remove anything.

Rating of dumpsters is a good thing, in particular.

If you could change one thing about the design, what would it be? Would you use this app if that change was made? / Hvis du kunne endre ett aspekt av designet, hva ville det vært? Ville du brukt appen hvis denne endringen ble gjort?
If I'd use it a little longer, maybe, but it's quite clean.

(mentions cleaning the place again)

(tells about a sneaky bastard; if dumpster locked, toothpick inside lock;)

(legality: we won't get charged for it, there's nothing of value in there (ironically))

(when will it be ready? Publish it!)

(will be available for questions after 5 on weekdays)

(**might know more people that could actually test this thing**)

E.10 Test #6

E.10.1 Oppgaver: (stopp når tida er ute)

1. *Sett posisjonen din (guiden dukker opp, nb)*

   (well yes, it happened)

2. *Se informasjon om en dumpster*

   Straight to the point

   There is supposed to be some images here?

3. *Gi den en rating*

   I can rate this dumpster, I guess, and there's an emptying schedule

4. *Legg til en kommentar*

   Goes to the comment screen by pressing the correct button, looks around

5. *Rediger dumpsteren (typ sett den til å være låst eller noe)*

   Menu → edit

   Is it going to be so that everyone can edit information?

   →→ one feedback would be, **I would like to see when it was last edited**

6. *Gå tilbake til en tidligere versjon av dumpsteren*

   (it happened)

7. *Legg til en ny dumpster*

   Looks at the map, wanders around all the buttons, clicks on the plus button in list view **IT LEADS TO THE INFO SETTING THING, NOT TO THE POSITION SETTER NB NB NB** so they were kinda confused about where you'd set that

8. *Legg til et bilde til denne dumpsteren (funker ikke i appen)*

   Pressed the add button

9. *Endre posisjonen din*

   Thinks the position button the map would show dumpsters around you, right?

   Discussed location services and so on. (more should've been written here)

E.10.2 Spørsmål til samtalen etterpå:

What do you think of the concept for this app? Would you use it? / Hva syns du om konseptet vårt? Ville du brukt denne appen? (burde dekke spesifikke ting de (mis)liker)

Personally, great for exploration. But usually goes to fixed set of places. Good if it is often updated, but that depends on the community.

What would you like to be able to do in a dumpster diving app? Is there anything we might have missed? / Hva slags ting ville du hatt mulighet til å gjøre i en app som denne, som ikke allerede er en del av den?

Whoops, the filter selection for dumpster type actually shows up in the editor

There are large and smaller containers, some you might have to climb into

Maybe some comments here (store view is positive, but only after you come after it's closed)

→ this is in info, kinda

Are these

It's a bit confusing for me, the difference between these tags and the contents.

Confusing because the categories are some kind of

. . . so maybe they are the things that are there now? (regarding the categories, actually)

I think it would be much more practical to add the contents as a picture (NB)

Adding a photo seems separate from contents (**this is an important question right here**)

General comments can be useful, (various things about possible merging, I brought up the idea)

For this person it'd be cumbersome to write about what it contained, wants to post a picture instead

Is there anything in the app that you don't think is relevant? / er det noe i appen du føler ikke er relevant? (the information we do have is already shown)

(see above)

If you could change one thing about the design, what would it be? Would you use this app if that change was made? / Hvis du kunne endre ett aspekt av designet, hva ville det vært? Ville du brukt appen hvis denne endringen ble gjort?

(see above)

Anything else?

No, I don't think so.

So with this app, there would be like a community around it?

Sometimes, notifications of e.g. harmful food (**NEW IDEA**) could be incorporated.

Also, there are free fridges in [REDACTED], perhaps add information about food sharing places.

Would like to test.

E.11 Test #7

E.11.1 Oppgaver: (stopp når tida er ute)

1. *Sett posisjonen din (guiden dukker opp, nb)*

   "Dumpster finder", okay

   Tries to type in the field after proceeding, everything's okay

   (then reaches for the position icon)

2. *Se informasjon om en dumpster*

   Clicks on the damn marker

   "So the idea is that the person who goes to the dumpster can say what's in it (lists things)"

3. *Gi den en rating*

   (yes)

4. *Legg til en kommentar*

   Goes to comments

   Also looks at the contents screen

   User driven? Yes.

   (asks about categories, if they are completely custom)

5. *Rediger dumpsteren (typ sett den til å være låst eller noe)*

   (skipped)

6. *Gå tilbake til en tidligere versjon av dumpsteren*

   (eh)

7. *Legg til en ny dumpster*

   Goes through all the things, clicks the dropdowns, yada yada

8. *Legg til et bilde til denne dumpsteren (funker ikke i appen)*

9. *Endre posisjonen din*

E.11.2 Spørsmål til samtalen etterpå:

What do you think of the concept for this app? Would you use it? / Hva syns du om konseptet vårt? Ville du brukt denne appen? (burde dekke spesifikke ting de (mis)liker)

I think it's really cool, definitely would use it. Would probably check it before I go dumpster diving. I don't know how likely people are to update it – **you may not want to let others know about the dumpsters you know of** (in part because lots of others coming over and making a mess). Would be very useful (in conclusion). But it's in a way, like, I see I can get all these things, then don't update it, then people will still see it like it is there.

Maybe good idea to add these free fridges.

What would you like to be able to do in a dumpster diving app? Is there anything we might have missed? / Hva slags ting ville du hatt mulighet til å gjøre i en app som denne, som ikke allerede er en del av den?

Is there anything in the app that you don't think is relevant? / er det noe i appen du føler ikke er relevant? (the information we do have is already shown)

The rating? Is it a rating of the things you've found, or the store, or. . . ?

(we might need to lessen the ambiguity of this!)

If you could change one thing about the design, what would it be? Would you use this app if that change was made? / Hvis du kunne endre ett aspekt av designet, hva ville det vært? Ville du brukt appen hvis denne endringen ble gjort?

Um. . . No, not really. I like it. Simple, clean.

Other things?

Is this, like, are you thinking of having this as free to download on App Store, or is this supposed to just be circulated in the communities? (out of curiosity)

Testing the app: We told them

E.12 Test #8

E.12.1 Oppgaver: (stopp når tida er ute)

1. *Sett posisjonen din (guiden dukker opp, nb)*

   Jajaja

2. *Se informasjon om en dumpster*

   Virrer rundt, vi har ikke gitt en oppgave enda

   Går inn på details via listevisning

   "Contents and comments, okay"

   Går rundt på diverse av sidene, vet ikke hvordan jeg skal oppsummere alt dette.

   "Hehe, Tore på sporet"

3. *Gi den en rating*

4. *Legg til en kommentar*

5. *Rediger dumpsteren (typ sett den til å være låst eller noe)*

   Hvor ville jeg trykket? → trykket direkte på kategoriboks, direkte på rating.

   Jon forteller hvor hen skal trykke, da går det jo.

6. *Gå tilbake til en tidligere versjon av dumpsteren*

   Så når du går inne (...)

Vi forklarte diverse ting ift. denne sida, hen tenkte at det ville være sånn at folk *skrev* at *dette* var det som var i dumpsteren i dag osv.

7. *Legg til en ny dumpster*
8. *Legg til et bilde til denne dumpsteren (funker ikke i appen)*
9. *Endre posisjonen din*

E.12.2 Spørsmål til samtalen etterpå:

What do you think of the concept for this app? Would you use it? / Hva syns du om konseptet vårt? Ville du brukt denne appen? (burde dekke spesifikke ting de (mis)liker)

Vet ikke om det er mange nok til at det blir nok informasjon, eller *om* folk vil gjøre det.

Hadde brukt den. Nevner også Facebook-sida, men hen liker ikke å bruke Facebook, så denne appen hadde vært aktuell.

What would you like to be able to do in a dumpster diving app? Is there anything we might have missed? / Hva slags ting ville du hatt mulighet til å gjøre i en app som denne, som ikke allerede er en del av den?

Location, hvor du finner den, hvis du ikke kan se... (idk.)

Dekker litt, eh, aller meste.

Is there anything in the app that you don't think is relevant? / er det noe i appen du føler ikke er relevant? (the information we do have is already shown)

Skjønte ikke helt hva du gjør med disse [kategoriene] – er det for å vise hva som er inni der?

Går inn i edit dumpster, hvordan legger du til [kategorier]?

Tenkt på det meste, jeg tenker ikke på noe annet, men kanskje det kunne vært interessant for folk å koble sammen for å gå sammen om divinga.

If you could change one thing about the design, what would it be? Would you use this app if that change was made? / Hvis du kunne endre ett aspekt av designet, hva ville det vært? Ville du brukt appen hvis denne endringen ble gjort?

Kanskje dere kan **fargekode etter noen av disse kategoriene** (viser på mappet)

Kanskje **rating kan gjenspeiles** (rød verst, grønn best)

Villig til å teste? Ja, gjerne, bare send den over

Foreslår at vi kan spørre i Facebook-gruppene etter folk som

E.13 Test #9

E.13.1 Oppgaver: (stopp når tida er ute)

1. *Sett posisjonen din (guiden dukker opp, nb)*

Clicks on the input field, proceeds

2. *Se informasjon om en dumpster*

   Asks if the markers are dumpsters, then clicks

   TUESTDAY

   I would like some hierarchy changes: For me, the most important is when it is emptied (to know when it is emptied) and where *exactly is it*. You don't know *exactly* what is in it → maybe som kind of naming on the categories thingy.

   Most important is the schedule, maybe the positive attitude is also good.

   First thing is schedule, could be whole week, you could have a calendar thing.

3. *Gi den en rating*

4. *Legg til en kommentar*

5. *Rediger dumpsteren (typ sett den til å være låst eller noe)*

   I don't know how to edit it.

   Jon instructs them to open the hamburge menu (wellllll)

   What is the difference between dumpster type and categories (whoops, just wrong thing in categories again. . .)

   Would appreciate that the visual look is kept hacker-style. Would be cool to (...)

   Shows us some poster with stereotypical hacker colors

   Has a background in design!

6. *Gå tilbake til en tidligere versjon av dumpsteren*

7. *Legg til en ny dumpster*

   Jon just states that exists

8. *Legg til et bilde til denne dumpsteren (funker ikke i appen)*

9. *Endre posisjonen din*

E.13.2 Spørsmål til samtalen etterpå:

What do you think of the concept for this app? Would you use it? / Hva syns du om konseptet vårt? Ville du brukt denne appen? (burde dekke spesifikke ting de (mis)liker)

Yeah, maybe. One more thing, it's nice to track, like, if people have been there recently.

Let's say that this app gains traction, what happens to the dumpsters? (much activity, perhaps?)

(yes, we will have a visitation thing, but only as a simple click of a button)

Would there be some dumpsters that would become popular and/or never show up in the app.

We need to address the questions like **what happens if this becomes popular**.

Design is usually important, like, understand the *impact* of what you're creating.

But other than that... Yeah, cool.

What would you like to be able to do in a dumpster diving app? Is there anything we might have missed? / Hva slags ting ville du hatt mulighet til å gjøre i en app som denne, som ikke allerede er en del av den?

Is there anything in the app that you don't think is relevant? / er det noe i appen du føler ikke er relevant? (the information we do have is already shown)

If you could change one thing about the design, what would it be? Would you use this app if that change was made? / Hvis du kunne endre ett aspekt av designet, hva ville det vært? Ville du brukt appen hvis denne endringen ble gjort?

Testing: Yeah, sure, just haven't done dumpster diving for a while (year and a half ago)

Can give feedback on interface.

Bergebakken, Bergquist, Jonson

Dumpster Finder: A mobile app for anonymous, social food waste reduction

NTNU
Kunnskap for en bedre verden