NTNU

Kunnskap for en bedre verden

IT2901

LIER01

Team members:

Amund Lunke Røhne
Hjalti Percy Casimis Hjaltason
Lotfi Amin Lazreg
Patrick Moen Allport
Vegard Rognstad Smines

Product owner:

Selamewit Eng

Supervisor:

Serena Lee-Cultura

August 25, 2021

Abstract

This report will introduce the mobile application Bro, describe Bro as a project and document the development process. Bro was made to supplement the introductory program for refugees at Lier municipality. To achieve this, the team developed Bro as a platform to find congregated information and references, as well as interactive learning. This report was written by a team of students, in their third year of a bachelor's degree in informatics, as part of the course IT2901, Informatics Project II. Bro as an idea was formulated by Selamewit Eng, a refugee consultant in Lier municipality, and was after launch maintained by the refugee service at Lier municipality.

Contents

Chapter 1

Introduction

1.1 Background

Integrating into a foreign society is difficult. The refugees have to understand every part of how the community works, how its values are formed and its cultural norms. The refugees also have to understand the various systems, like education, healthcare, housing and recreational activities.

Many refugees have experienced various challenges in their home country and on their way to Norway, creating a situation which often feels chaotic. To stabilize this chaos, Selamewit Eng (a worker in the refugee service in Lier municipality), wished to offer refugees a modern and easy to use tool, to aid in their integration process. To help realize their vision, Eng contacted staff at NTNU and enlisted a group of undergraduate students from the course IT2901, Informatics Project II.

1.2 Objectives

The refugee service provides an introductory program, that aim to help the refugees get settled into their new lives. The introductory program consists of multiple courses. These courses are thought to benefit from a certain degree of digitization, such that the information offered will remain accessible even outside of the classroom.

Following the initial meetings between Selamewit Eng and the developer team, a mission statement which describes the needs for this application was constructed: "A complimentary service to the introductory program, with a wide collection of information and useful references. Additionally, a platform for interactive self-study."

The main goal of this project was to provide the refugee service with a supplementary tool that could aid refugees in learning the various systems in Norwegian society. As a secondary goal it should improve their Norwegian vocabulary through familiarization with text and helpful graphics.

1.3 Product Description

The product is named Bro, translating to bridge in English, which is an allusion to the the Norwegian fairytale *De tre bukkene bruse*.

During the planning stages it was decided that the product would serve as an informational hub and should

be developed for Android and iOS. There would be a breadth of information, and the ability to alter, add, or remove content during the application's lifespan.

There was also a request for the application to work as an educational tool for the refugees. To solve this problem, the team and Selamewit Eng settled on creating an interactive learning platform. This platform uses short cards with text and visual media, such as videos and images, to inform the refugees on various topics related to Norwegian systems and culture. It also aims to present the services offered to them. Everything from how to use ovens in the refugee houses, to how to apply for student loans are important topics for the refugees. After studying the content of the cards, the refugee's knowledge will be tested through the use of a quiz with short and simple multiple choice questions.

Selamewit Eng requested a complete and deployed piece of software. They also desired for the product to be modular enough for refugee services in other regions to be able to use as well. Thus, Bro could potentially be used in the entire sector.

1.4 Stakeholders

Product owner: Selamewit Eng, representing the refugee service in Lier Municipality. As a product owner, Selamewit had the final say on all decisions regarding the product. She did not have any prior experience with software development, but is an experienced worker at the refugee service and also a former refugee herself.

Team: The team members are responsible for the development of the product. The main concern for the team is to communicate effectively with other stakeholders and maintain good routines during the development of the product. The team members all signed a contract (appendix A), which solidified the teams expectations both regarding the general team spirit and the rules each member followed during the development of the product.

Supervisor: Serena Lee-Cultura, a PhD student at NTNU, served as the teams supervisor. The supervisor functioned as an experienced advisor to aid the development team with non-technical problems such as customer communication, good report practices and user testing.

Content creators: The staff at the refugee service would serve as content creators and have access to Bro's content management system (CMS). Through the use of this system the staff would be able to add, edit and delete content in the database. The content creators were not expected to have experience with similar systems and it was therefore important to provide the content creators with an intuitive interface with all the functionality they needed to supplement their introduction program.

End-Users: The end users of Bro were the refugees being resettled in Lier municipality, a very diverse group which due to political factors is an ever changing blend of cultures and people. Their ability and willingness to use Bro was a key factor in evaluating the success of the application. The age, background and educational level of the refugees varies greatly. Their technical aptitude may also span from non-existent to high.

1.5 Measurement of Success

These success metrics (table 1.1 will help in determining whether or not the project serves its purpose after it is deployed. The team therefore formulated a list of goals and metrics that can be used to evaluate if the product meets the stakeholders expectations.

Table 1.1: Application Success metrics, the various goals Bro sought to achieve and how satisfaction is measured

Goal	Metric
Alleviate workload from the refugee service employees.	Increased employee satisfaction and post-release product usage
Provide refugees with concise, useful information that is relevant for their life situation	Product reviews and continued usage
Reduce the need for physical document distribution.	Feedback from Selamewit Eng and content creators.
Engage the user in educational content.	Time spent with the product and course statistics from the CMS system
Potentially unify introductory programs across several municipalities.	Several municipalities make use of the same system.
Make the transition for refugees into Norwegian society easier	Product reviews and time spent with the product
Facilitate for and provide Norwegian language exposure for refugees.	Product usage statistics.

Chapter 2

Product

2.1 Pre-Study

What the team was looking to achieve through the pre-study was to discover what other products have done well and how the team could make a superior product. The team studied other applications strengths and weaknesses. During this process, the team were inspired by many features from the studied applications.

2.1.1 Alternative Solutions

There were a limited number of applications that were made with the purpose to inform and educate refugees. Especially ones that can be adapted for new regions and are simple to maintain. Therefore the team could not find any applications with fully overlapping intentions. At this point it was clear that the team needed to develop a new application from scratch. However, there were several applications with some similar features for the team to draw inspiration from.

2.1.2 Similar Applications

The team studied six different mobile applications. The applications were put into these categories based on their purpose:

1. Language learning - Duolingo
2. Information for immigrants - FindHello, RefugeeBuddy, Settle In
3. General information - Wikipedia, Fandom

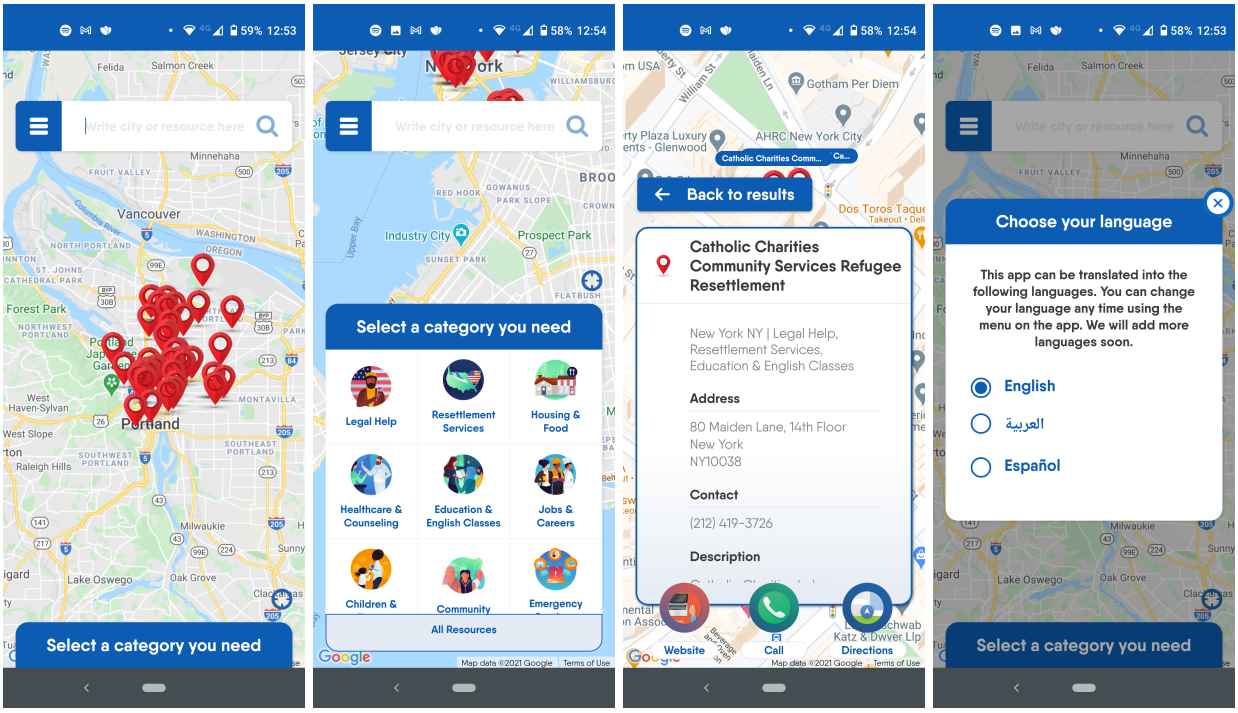
The apps were assessed based on the following criteria and given a score from 0-6.

- **Accessibility** - Refugees generally have a low reading proficiency and Norwegian language knowledge compared to the general public. Bro should therefore have high accessibility when it comes to text-to-speech support and translations. In addition Google Accessibility Scanner was used, which measures content labels, touch target size, clickable items, and text and image contrast
- **Usability** - Some refugees have very low technological proficiency, depending on their previous living conditions and education level. Therefore it is important that Bro needs to guide the refugees by conforming to usability heuristics. Don Normans principles of design [7] will be used to assess the applications.
- **Ease of finding information** - Ease of finding information. Refugees have poor Norwegian comprehension, which leads to them not consuming all the information on the page and only read certain pieces. Therefore the ease of finding information heavily outweighs the quality and depth of the information.

- **User engagement** - Reading through many paragraphs of pure text will often lose the users attention or interest and lead to low knowledge retention. Therefore, user engagement will be assessed.

FindHello

A map-based app aimed at showing immigrants nearby services, providing contact information and a small description for said services. The application has immigrants as a target group, which includes refugees.



(a) Main view (b) Select category (c) Contact information (d) Choose language

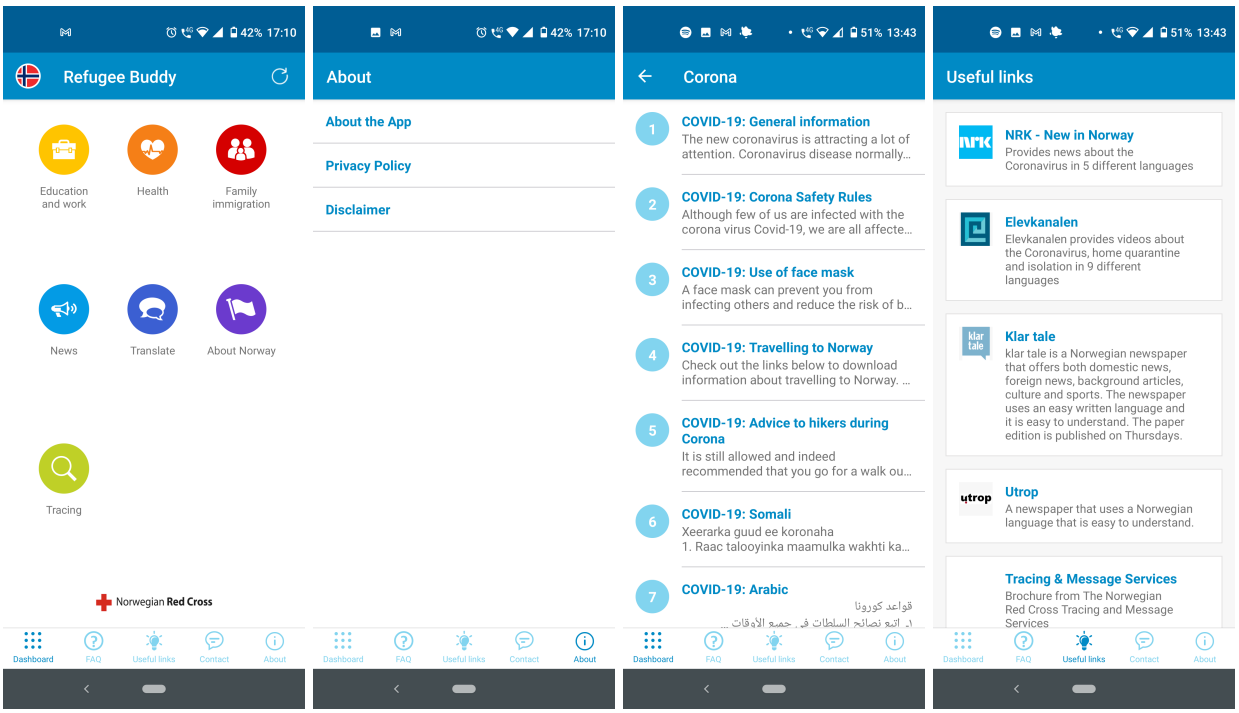
Figure 2.1: The relevant views in the FindHello app

- Accessibility - 5/6 - FindHello provides an option to translate the application to three different languages. This changes the language for navigating the app, but not the language of the content. The app lacks text-to-speech options, but one can use the device operating system (OS) feature for this. Google Accessibility Scanner reported that some text, such as the content in figure 2.1 (c) lacks contrast, which could be an issue for people with bad eyesight.
- Usability - 4/6 - FindHello is similar to a map application, like Google maps or Apple maps. The difference is that FindHello's service is more specific, only displaying services relevant to immigrants. The application design is mostly consistent with existing map based solutions such as Google Maps, which lowers the learning curve. By only including immigrant specific organizations or services, FindHello avoids the typical information overload that immigrants often experience. When clicking items on the map, FindHello provides no instant feedback to the user while waiting to load the information. This could make the user unsure if they clicked correctly, particularly if they have a slow connection. The "Choose language" button is only displayed as text which means it is very hard to find if the user selected the wrong language.
- Ease to find information - 3/6 - By providing a category selection, FindHello makes it easy to find information, given that what you are looking for is within one of the categories. The information about a service within FindHello is limited, but there is always a link to the service website which makes up for this.

- User Engagement - 4/6 - FindHello uses icons and a wide color palette to keep the users attention. The use of a map as the main interactive element is more engaging than searching or browsing through a list of addresses or areas.

Refugee Buddy

Refugee Buddy is a wiki-like app with the goal of giving immigrants access to information gathered in one app. Some information is provided in articles within the app, and some is provided through links to external resources.



(a) Main page (b) About page (c) Corona category page (d) Useful links page

Figure 2.2: Relevant views in the Refugee Buddy app

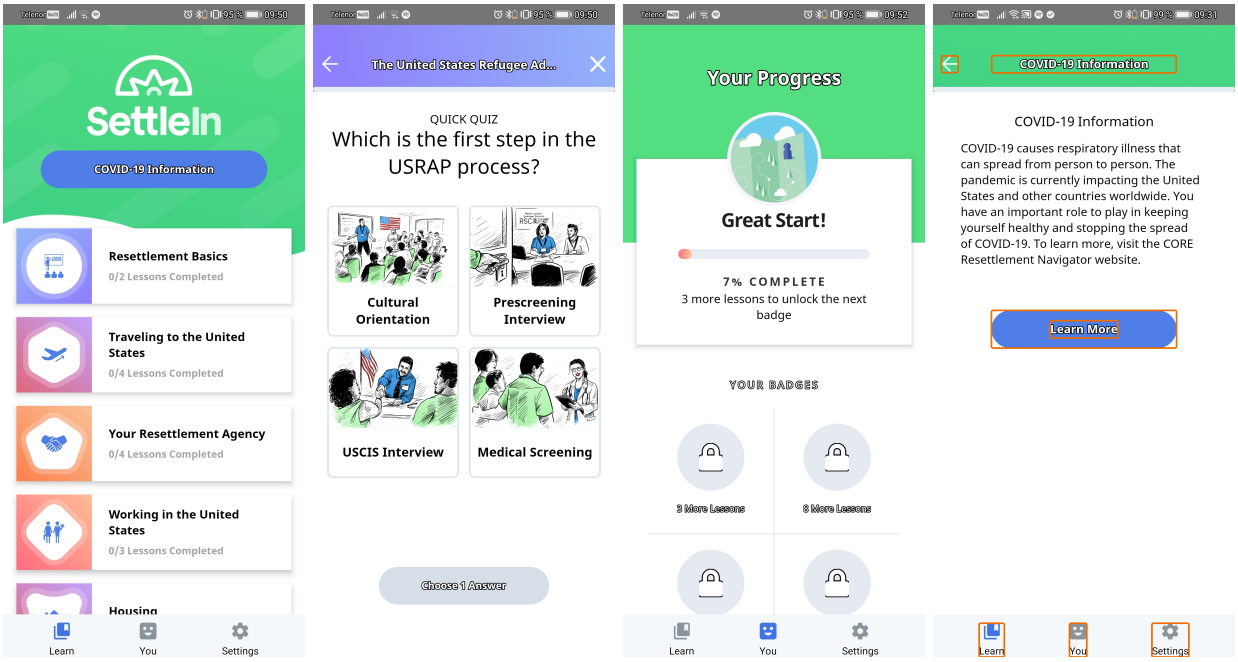
- Accessibility - 5/6. Refugee Buddy can be translated into six languages, translating both navigation buttons and content. The light blue text on white background in the bottom navigation bar is hard to read due to lack of contrast. Google Accessibility Scanner reported that the refresh button and language button at the top in figure 2.2 (a), as well as the back buttons should have a larger clickable area.
- Usability - 2/6 - Refugee Buddy is generally well structured, and it is easy to find specific pieces of information. Its feature set is somewhat lacking, but the existing functionality is at the core of the application. The buttons provide instant feedback and it is clearly indicated to the user when the app is loading. The app is not consistent with conventions for app design as the back buttons (top left, figure 2.2 (c)) always takes the user back to the homepage instead of back to the previous view. If the user clicks a section in the "about" page (figure 2.2 (b)) there is no way to navigate to the other sections without closing and reopening the application. Not all the category views that can be selected in figure 2.2 (a) lead to the same type of page when clicked. Some look like figure 2.2 (c), some look like figure 2.2 (b), and some lead straight to an article.
- Ease to find information - 3/6 - The division into categories is good, but the article pages are overfilled with text. A possible fix is to have indexation at the top of the page, or further sectioning of the content.

The "Useful links" page (figure 2.2 (d)) should have been sectioned as there are far too many links on one page. The Corona category contains articles in 6 different languages despite the user choosing a language when opening the application.

- User Engagement - 1 / 6 - The design is bland, with content delivered as a large block of text. Everything is in the same color except for icons and links. A broader color palette (except for the main page) as well as images, videos and sound could have improved user engagement.

Settle In

Settle In is an app for immigrants to learn about general topics about immigration to the United States, e.g. the laws or the settlement process. It is designed as an interactive learning platform providing small amounts of information at a time in a card-based format using multimedia, and testing the user with a quiz after.



(a) Main page (b) Quiz (c) Profile page (d) Google Accessibility Scan

Figure 2.3: A selection of the relevant views in Settle In. Figure (d) shows an accessibility scan with problems highlighted.

- Accessibility - 4/6 - Settle In has the option to be translated into 7 different languages. The whole app is translated upon choosing a language, including navigation buttons. Settle In has videos and audio lectures as well as text and image based ones, which can be better for users with limited language skills. The video and audio lectures have the option of reading transcript instead of watching or listening. This is important for deaf people. Google Accessibility Scanner reported that some buttons, as shown in figure 2.3 (d), have a very small clickable area which could be a problem for old people or people with motor disorders. A few elements also lacked screen reader labels.

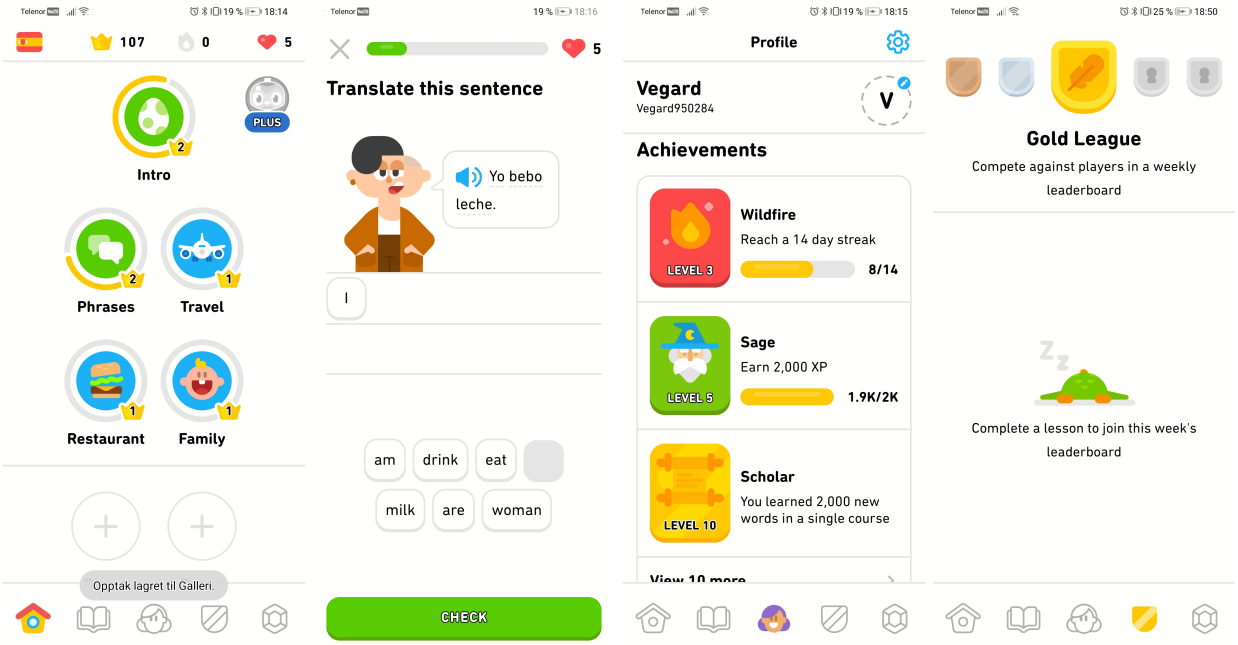
- Usability - 6/6 - Settle In uses large icons and cards with pictures to create a simple and intuitive interface with very little clutter. The lectures are simple without unnecessary text (which feedback from user tests reported as very important when dealing with language barriers). Some clickable elements, such as the cards in figure 2.3 (a), lack instant feedback when clicked, but this is not a major problem as there is no loading time required for the navigation action they perform. Lectures are kept

to a few predefined formats which ensure consistency, with quizzes at the end in the format shown in figure 2.3 (b).

- Ease to find information - 3/6 - The information is locked behind lectures and if the user is in the middle of a lecture program there is no way to go back to the start until finished. This app works as a quiz tool with quick explanations and is not well suited for a user who wants to find the answer to a specific question, or learn more about a very specific topic. An improvement could be an FAQ and potentially wiki-pages to find specific information.
- User Engagement 5/6 - Settle In uses pictures, video and audio frequently to ensure user engagement. There is also a simple progression system, shown in figure 2.3 (c), with awards for completing more lectures. This is a good way to implement game features in a learning platform, but could have been expanded upon similar to the way it is done in Duolingo.

Duolingo

Duolingo is a quiz-based language learning app. It functions more like a game than a traditional learning platform, using interactive features and reward systems, as well as having the user progress through "levels".



(a) Main page in a language course (b) Level in a course (c) Profile page (d) Ranking page

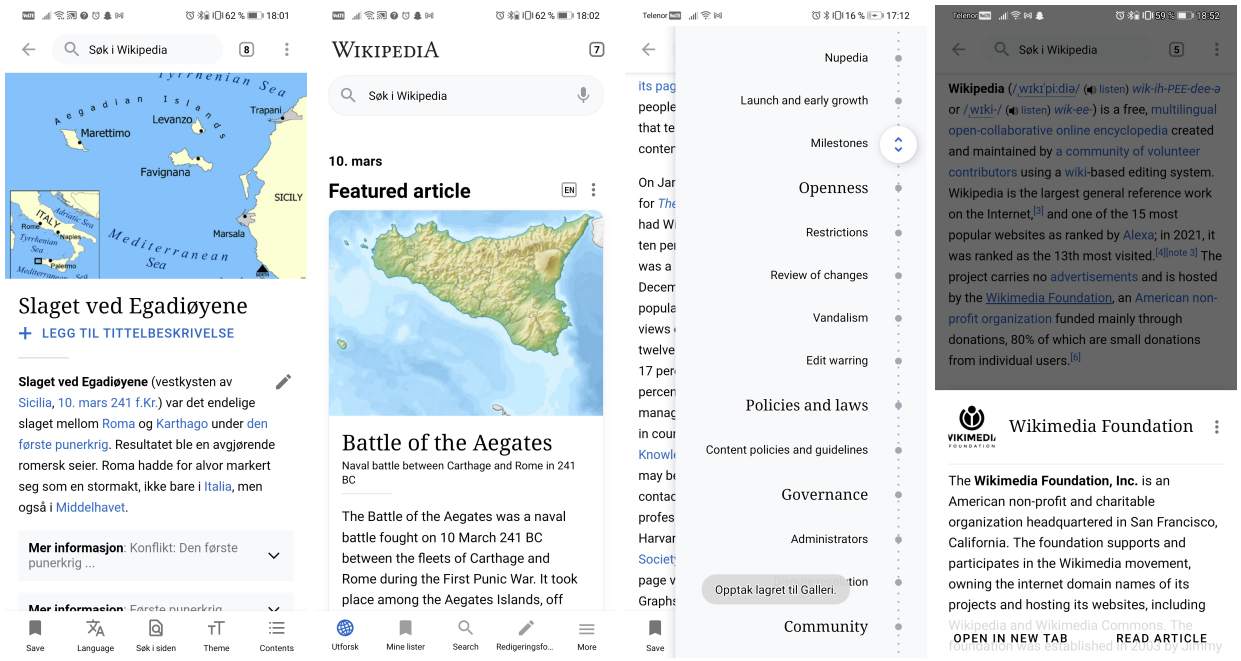
Figure 2.4: Relevant views in the spanish course of the Duolingo app

- Accessibility - 5/6 - Duolingo is a language learning app, so the accessibility for users with different languages is naturally very high. The Google Accessibility Scanner found some elements lacking labels for screen readers. There were also a few buttons which were recommended to be bigger, but overall no major problems were found.
- Usability - 5/6 - Duolingo is generally very intuitive and consistently uses icons either combined with or replacing text for navigation, such as the bottom navigation bar seen in figure 2.4 (a). The app also gives instant feedback on user inputs when pressing buttons, and indicates to the user when the application is loading.

- Ease to find information - 2/6 - The app is not designed for looking up information. The user must complete the levels that are available and can not access most courses until enough progress has been made. If the user is looking for specific information Duolingo is therefore not very useful. If the user wants to know more during a level it is possible to click the text in the speech bubble (figure 2.4 (b)) to see translations for the words.
- User engagement - 6/6 - Duolingo makes use of several game-like features to keep the user engaged. The user has a progress bar and they lose a heart each time they make a mistake as seen on the top of figure 2.4 (b). The app also has a progression system where the user can reach higher levels by completing enough levels or by proving their skill. A reward system with achievements figure 2.4 (c) and "crowns" is also used, as well as a ranking system (figure 2.4 (d)) which motivates the user to complete more levels each week. A social aspect has also been added through comments on tasks.

Wikipedia - the Mobile Application

Wikipedia is a wiki-based app for general information. The main purpose of Wikipedia is to function as an encyclopedia where it is extremely easy to find any piece of information.



(a) Homepage (b) Article view (c) Article contents tab (d) Article link pop up view

Figure 2.5: Relevant views in the Wikipedia app

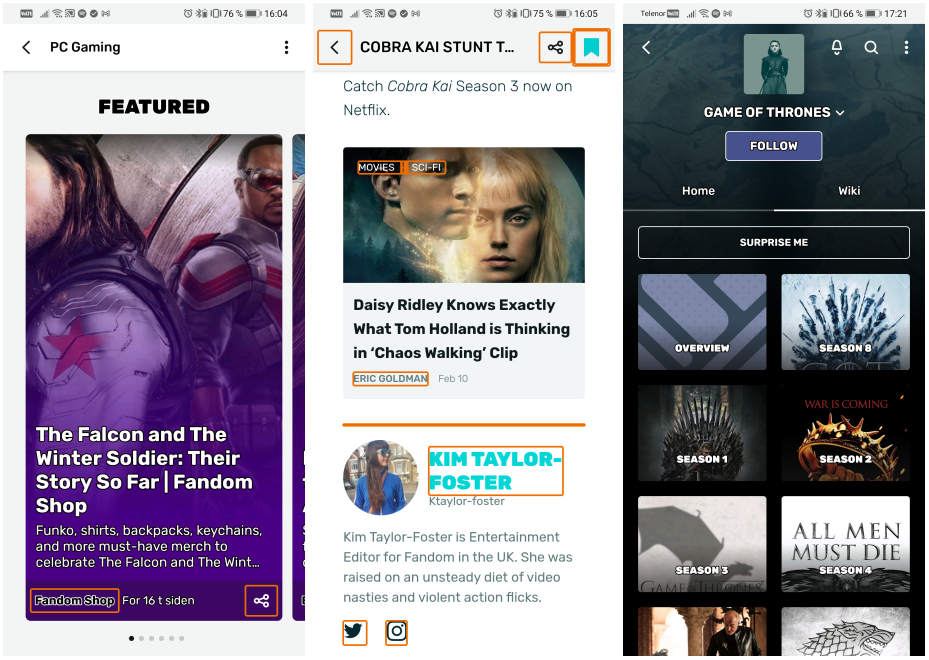
- Accessibility - 6/6 - Wikipedia has implemented language selection in multiple ways, making it easy to switch between languages within an article. The app also translates most of the text for buttons and settings depending on the phone's system language. Despite this some elements seem to always be in English, such as how the bottom bar in figure 2.5 (a) does not use the same language as the search bar on the same page. Scanning Wikipedia with Google Accessibility Scanner yielded no meaningful accessibility problems.
- Usability - 5/6 - Wikipedia gives instant feedback on all user inputs. The layout is simple and intuitive, following established conventions for app design. The "contents" tab shown in figure 2.5 (c) gives the user a quick way to browse the contents of an article with an intuitive mapping (mapping as defined in Don Normans principles of design [7]). When the user is reading an article the bottom navigation bar

changes, as seen in figure 2.5 (a) compared to figure 2.5 (b), which slightly impacts the consistency of the application, but gives the user more relevant features. If the user clicks on multiple articles, using the blue links, there is no button which navigates directly to the home page (figure 2.5 (b)). The user would either have to click the back button multiple times or open a new tab if they wanted to navigate back.

- Ease to find information - 6/6 - The content on Wikipedia has a completely flat structure with no hierarchy of categories to find content. This works well because of the large amount of available, user generated articles, which would make a navigation tree hard to implement. As a result of this the only practical way to find content on Wikipedia is to search for it. The search system is well made and finding the needed information is usually very simple. The references between articles, as well the pop-up feature (figure 2.5 (d)) when clicking on an article link makes it easy to find related information.
- User Engagement - 3/6 - Wikipedia uses featured articles and "Top read" on the front page to grab the users attention when the app is opened. The articles link to related articles and further reading which keeps the user interested. Apart from these features Wikipedia does not have a design optimized for user engagement, as it has no features that gives the user incentive to use it more actively. The app is mostly designed similar to Google, to be used when the user has a specific question or topic they want to research.

Fandom

Fandom is a wiki-based app for information about popular culture, such as games and TV-shows.



(a) Main page in category (b) Article page (c) Main page for a fandom

Figure 2.6: (a) and (b) are automatically generated by Google Accessibility Scanner. Elements highlighted in orange have accessibility problems.

- Accessibility - 4/6 - Fandom mostly uses large cards with easily readable text, with some exceptions. The content within a genre (such as (c) in figure 2.6) can be filtered by language, although this feature is a bit hard to find. Google Accessibility Scanner found multiple elements in the Fandom app without screen reader labels. Some buttons were also smaller than recommended as shown in image (b) in figure

2.6. Although most text is easily readable, some text, such as “KIM TAYLOR-FOSTER” and “ERIC GOLDMAN” in image (b) does not have as much contrast as recommended by Google Accessibility Scanner.

- Usability - 5/6 - Fandom provides good feedback for all user inputs. The layout is simple, intuitive and efficient to use. The use of image cards such as in (a) and (c), figure 2.6, allows for a highly aesthetic and minimalist design. Some options, such as the language filter mentioned under accessibility, are hard to find and the app does not provide the user with help and documentation. The app also lacks some consistency in the design as not all “Fandoms” (such as figure 2.6 (c)) have the same structure. Some have the same as (c) and others are just a list of articles.
- Ease to find information - 6/6 - Fandom has very good categories with sub-categories and indices on the pages, where you click on what you specifically want to read. The well structured card views and the intuitive hierarchy of the content makes it easy to find information despite being an application with large amounts of data.
- User Engagement - 5/6 - Fandom has a good front page with featured articles and videos. The frequent use of images and videos throughout the app, as well as a vibrant color scheme keeps the users attention. Interaction between users and frequently updated content gives the user incentive to keep returning to the application frequently.

Complete Table

Table 2.1: Table summarizing all scores, as well as providing an average “Overall Score”

App	FindHello	Refugee-buddy	Settle In	Duolingo	Fandom	Wikipedia
Accessibility	5	5	4	5	4	6
Usability	4	2	6	5	5	5
Ease to find information	3	3	3	2	6	6
User engagement	4	1	5	6	5	3
Overall Score	4	2.8	4.5	5.3	5	4.8

2.1.3 Inspirations

Each time a team member found a promising feature, the team looked at the feature and decided if this was something that would benefit Bro. If it was deemed beneficial, the team made the feature into a user story and included it in the user story backlog.

Fandom

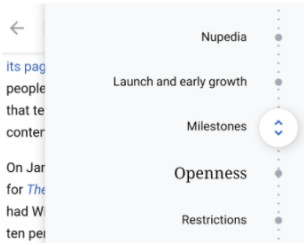
The Fandom app (figure 2.7) had several good solutions for navigation and displayed a lot of information in a clear and concise way. The team decided to take inspiration from the large category icons and expandable subheaders.

(a) Large pictures or icons for categories (b) Expandable subheaders within a page

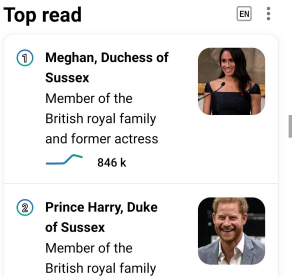
Figure 2.7: Fandom inspirations

Wikipedia

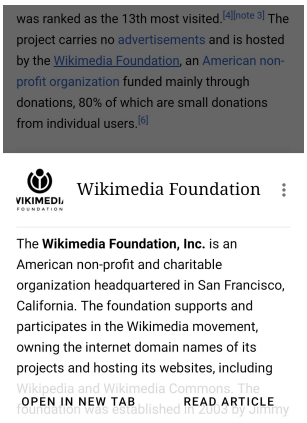
The Wikipedia app has very efficient navigation and is fairly engaging despite being an encyclopedia. The team decided to draw inspiration from a few key features (figure 2.8) they believe are responsible for this.



(a) Index in article



(b) Frontpage with top read articles

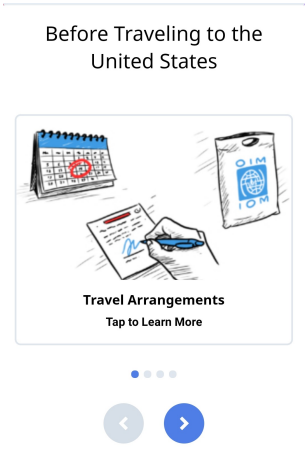


(c) Option to click on concepts and have a popup with explanation

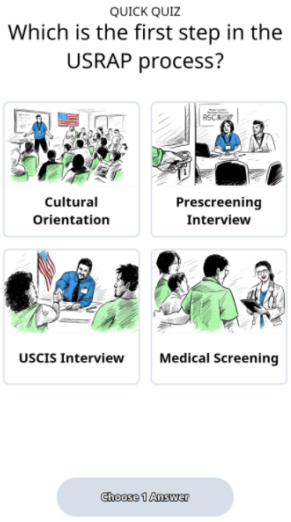
Figure 2.8: Wikipedia inspirations

Settle In

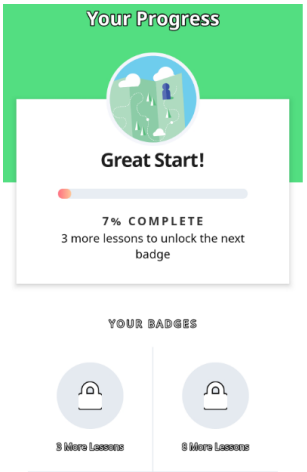
The Settle In app (figure 2.9) has a very intuitive and engaging course and quiz format which would be well suited for the Bro app. It also uses a progression system the team believes would increase user engagement.



(a) Course format



(b) Quiz format



(c) Gamification

Figure 2.9: Settle In inspirations

2.2 Requirements

The team had several meetings with Selamewit Eng to determine the scope of the project and exactly what features the Bro should contain, as well as what nonfunctional requirements are important for the user base. After deciding on the general purpose and features Bro, together with Selamewit Eng, the team converted them to specific and concrete nonfunctional and functional requirements.

2.2.1 Nonfunctional Requirements

The nonfunctional requirements were chosen and prioritized based on Selamewit's description of the user base, and her vision of what Bro should be. The users were described as having different levels of technical competence and Norwegian language skills. Therefore, requirements related to usability are of high priority. Reliability is also a high priority because Lier municipality does not have a development team that can maintain Bro after it's deployment,

Table 2.2: The non-functional requirements

ID	Description
NFR1	The product should be user friendly and intuitive to use
NFR2	The product should use recognizable/universal icons and symbols
NFR3	The product should be built with a focus on reliability
NFR4	The product should be built to support easy administration
NFR5	The product should be scalable to multiple municipalities
NFR6	The product API should have a predictable response time
NFR7	The product should make use of existing code and frameworks where it is appropriate

2.2.2 Functional Requirements

The functional requirements were grouped in the following order: main purpose (FR1-FR3), application content (FR4-FR9), administrative functionality (FR10-FR12) and security and legal compliance (FR13-FR18). Within each group the functional requirements are sorted according to their importance.

Table 2.3: Bro functional requirements

ID	Description
FR1	The product should include guidance for immigrants in several subjects selected by the customer
FR2	The product should include content in the form of articles, and interactive courses
FR3	The product should include game-like aspects for course content
FR4	The product should have support for embedded images, video, sound and documents in articles and courses
FR5	The product should have support for multiple localized versions of all content
FR6	The product should have a settings panel for user customization
FR7	The product should have restrictions for embedded files, i.e. size, duration, format, quality, etc.
FR8	The product should have support for TTS (text-to-speech) in articles and courses
FR9	User data should be used to create a personalized user experience
FR10	The product should enable administrators to add, delete and edit content
FR11	The product should be configured for easy deployment regardless of platform
FR12	The product should track anonymized user engagement statistics of courses
FR13	The product should comply with GDPR regulations
FR14	The product should retrieve all content from an external server
FR15	The product should satisfy the requirements of both the Google Play Store and Apple App Store
FR16	All user data should be stored locally, and never be transferred or processed externally
FR17	The product should make use of SSL encryption for all traffic
FR18	The product should be published with an open-source license

2.3 User Stories

An essential part of agile software development is a user-centered development process. To create meaningful agile tasks one needs to describe the need of the user and what has to happen in order to satisfy that need. One useful way of doing this is to create user stories.

2.3.1 User Story Creation

The team took the project concept and the functional/non-functional requirements to define a set of user stories (table 2.4). To ensure consistent user stories, the team used the Connextra format [1] in the user stories.

The user stories were rated using story points [8] with a range of 1 to 60 points. The points did not signify any value of time, however, they did represent the team's perceived difficulty of a task. This decision was made by the team as they were still inexperienced with the technologies being used. Thus, making accurate time predictions would be difficult during the early stages of development. As "percieved difficulty" was less abstract, using story points was much more useful to define the scope of a task.

To calculate story points, the team played a round of planning poker. One of the members would briefly present the user story, which was followed by every team member suggesting a story point rating. If the range of answers were quite wide, the team would discuss the user story in more detail. This was to consolidate a common understanding of what the user story actually meant. If the wide range was caused due to a misunderstanding, the team would play another round of planning poker. Afterwards, the team would set the story point value of that user story to the average of all the suggested values. If a user story got assigned a higher value than the agreed upon max (60), the team would break the story into smaller parts.

Tasks

As user stories may be comprehensive, the user stories were broken down into individual tasks. The requirements for such a task is that it has to have a small, concrete goal. To properly identify tasks in the user stories, the team discussed the user stories in the sprint planning meeting. This was to use the entire wealth of experience within the team to identify objectives and potential pitfalls. This would heavily assist the developmental process, as an overwhelmingly large user story may be hard to break down by one person.

Table 2.4: User stories

Jira ID	User story description	Story points
BL-10	As a user, I want to have a consistent article design, so that I can easily find information within an article.	100
BL-12	As a developer I want to have a development environment, so that I can start working	70
BL-79	As a developer, I want to have an outline the design of the product, so that I can efficiently design new views.	100
BL-1	As a user, I want courses with a engaging and consistent design, so that I can get a better learning experience	15
BL-11	As an admin, I want to be able to add, delete and edit content in a simple manner, on a regular basis, so that I can keep the content up-to-date.	40
BL-1	As a user,I want courses with a engaging and consistent design, so that I can get a better learning experience	15
BL-49	As a user, I want to be greeted by a home page when I first open the app, so that I can see what is available to me.	10
BL-95	As a user, I want to be able to view course cards with information, so that I can make my learning experience more interactive.	40
BL-117	As a user, I want to have an introductory text in the home page, so that I can get an understanding of what the application is.	15
BL-137	As a user, I want to be able to click on course info cards and reveal more information, so that I can learn more about the topic.	25
BL-138	As a user, I want to be able to navigate through the course cards using arrow buttons, so that I can navigate if i am unfamilliar with swiping.	20
BL-122	As a user, I want to see recommended courses on the home page, so that I can view the most important courses to me first	35
BL-142	As a user, I want to have a course list view, so that I can easily find courses that interest me.	40
BL-97	As a user, I want to be quizzed at the end of each course, so that I can check my knowledge.	55
BL-151	As a user, I want to be able to navigate between the homepage, course, article and setting page, quickly and easily, so that I can find the content I want to see.	50
BL-92	As a user, I want a carousel card view of main categories, so that I can easily find the type of articles i want to read	30
BL-94	As a user, I want an article list view, so that I can find the article I want to read within a category	20
BL-90	As a user, I want to have a view for reading an article, so that I can read the content available	25
BL-22	As a user, I want to be able to view embedded videos in articles and courses, so that I can watch more useful and engaging content.	15
BL-25	As an admin, I want to be able to add localized versions of the content, so that I can let the users can have it in different languages.	20
BL-71	As a user, I want to be able to see content in my preferred language, or Norwegian if not available, so that I can use the app in my preferred language.	35
BL-89	As a user, I want to have a cover photo in each article, so that I can get a context for the content.	10
BL-90	As a user, I want to have a view for reading an article, so that I can read the content available	25
BL-91	As a user, I want to have attached static files to the articles, so that I can read more on the topic.	15
BL-119	As a user, I want to see recommended articles on the home page, so that I can read the most important information first.	15
BL-157	As a user, I want to push a navbar button to return to the main page in the fuctionality, so I can quickly reset my navigation tree	25
BL-158	As a user, I want my media files to be persistent, so that I don't have to re-upload them	10
BL-113	As a user, I want to be able to view videos and images in courses and articles, so that I can get a more engaging learning experience	20
BL-50	As a user, I want to be tutored in how to use the app so that I can see how the application works	30
BL-120	As a user, I want to see course stats on the home page, so that I can see my progress.	NA*
BL-28	As a user, I want to have content read up for me through TTS, so that I can understand the content without reading the text	NA*
BL-84	As a user, I want to select a term in an article, and get information about it, so that I can understand what the term means.	NA*
BL-21	As an admin, I want to receive anonymized data from courses, so that I can see which topics are more difficult to understand	NA*
BL-38	As a user, I want to be able to search for information by it's content title, so that I can browse more efficiently.	NA*
BL-83	As an admin, i want to be able to add videos and images to courses, so that I can make more engaging courses	NA*
BL-98	As a user, I want articles to be grouped by my chosen preferred language, so that I can first find the articles I can easily read.	NA*
BL-104	As a user, I want to be able to see images and videos in articles so that I can learn in different ways.	NA*
BL-116	As a user, I want an intuitive design of the settings page, so that I can easily configure and personalize the app.	NA*
BL-93	As a user, I want to see my history of articles read, so that I can read a previously read article again.	NA*
BL-86	As a user, I want to be able to filter articles based on municipality, so that I can find relevant articles quickly.	NA*
BL-85	As a user, I want to favorite an article, so that I can easily read the article again.	NA*
BL-155	As a user, I want all text to be translated, so that i can easily navigate the app.	NA*
BL-156	As a user, i want a push notification when new content is added, so that i can know when new content is released.	NA*
BL-166	As a user, I want to see what category each course is in, so that I can easier find what I'm looking for.	NA*

NA - Not Assigned. User stories not yet included in a sprint planning and therefore has no story points assigned.*

2.4 Use Cases

Use cases in this context describes the various ways the different groups of users interact with the systems.

2.4.1 Admin

The admin panel was made available to certain staff members at the refugee service, referred to as the content creators. Their main use of Bro would be interaction with the Strapi interface. In Strapi the content creators are able to delete, edit or write articles, article categories and courses.

The diagram below (figure 2.10) displays the process where a content creator creates a category in Strapi and then creates an article, potentially using the new category. The content creator also has the option of marking the article as recommended while creating the article.

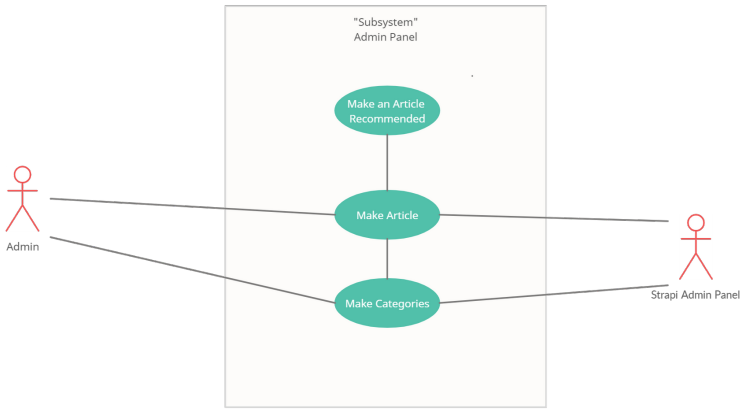


Figure 2.10: Content creator interacting with admin panel in Strapi

2.4.2 Article

One of the key features of Bro is the article system. The articles consist of helpful resources for individuals in their particular position, as well as context for these resources.

Figure 2.11 shows the process of how a user would find a particular article from the *article* view. The *article* view is the starting point after tapping the articles button. From the *article* view the refugee has the possibility of swiping between different categories, before pressing a button and being sent to the sub-category. Inside this sub-category certain filters kick in, filtering out articles specific for different regions. By tapping on an article the refugee is able to read it.

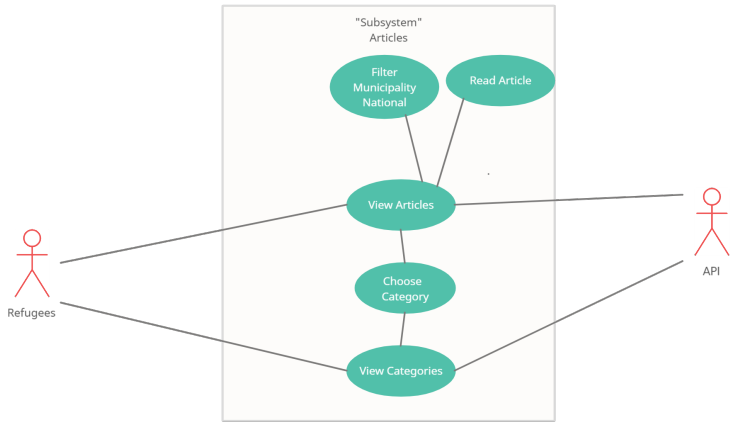


Figure 2.11: End user interacting with article system

2.4.3 Course

The course functionality was created to be a more engaging way for the refugee to learn about a topic and get exposed to the Norwegian language. In this part of Bro the refugee will browse through a short deck of cards and answer a few questions afterwards.

Figure 2.12 describes the process of taking a course from the *course-list* view. The *course-list* view being the overview page of all available courses. When the refugee enters the courses section of Bro, filters will remove courses from other municipalities and display recommended courses on top. The refugee is able to tap on a course and play through it.

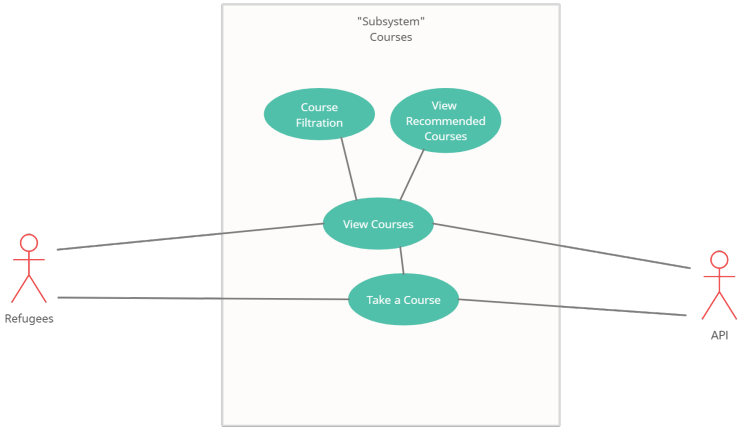


Figure 2.12: End user interacting with course system

2.4.4 Profile

The refugees can have many different backgrounds and it is therefore important to give the refugees some options to personalize the app. The most important part of this would simply be to change preferred language on articles, such that articles with their preferred language will be displayed.

Figure 2.13 describes the processes the refugee can go through in the *profile* view. The refugee can change

their personal information, mainly being their main language and current home region. The refugee will also be able to read the Terms of Service on this page.

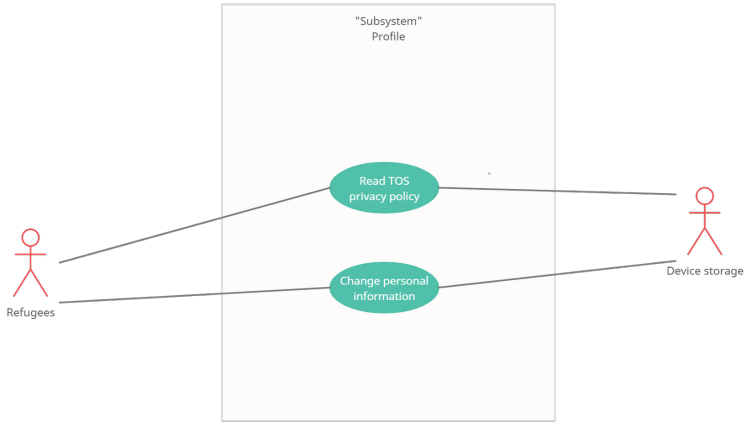


Figure 2.13: End user viewing profile settings

2.5 Architecture

Bro's architecture is represented by five types of views, following the 4+1 view model [6]: the logic, process, development and physical views, as well as the use case views covered in section 2.4. The views illustrate the complete structure of the system, and how the different packages and devices interact with each other. Additionally, it serves as good a resource that the team can reference during development.

2.5.1 Architecturally Significant Requirements (ASR)

The architecturally significant requirements, also known as architectural drivers, are the main features that most architectural decisions are based on. Their absence would result in an architecture without any justifications.

FR7: Embedded media

Bro should have support for embedded images, video, sound and documents in both articles and courses. This will require an architecture with a static file server, as redeploying the application every time new media is added is highly impractical.

FR10: Administrate content

Bro should enable administrators to add, change and delete content. As redeploying the app to implement changes is impractical, a CMS is required in the architecture.

FR11: Deployment

Bro's CMS system should be easy to redeploy to any cloud hosting provider. It is therefore important that the system is built as containerized microservices.

FR14: All content should be from an external server

As Bro does not collect any data from its users, all personalized features are dependent on a robust query system between Bro's front-end module and the CMS. Strong utilization of the BLoC (business logic component) pattern [3] enabled the development team to accomplish this.

FR16: Storage of user data

Bro should only store user data locally. This requires that all user based customization of content is handled client side, through the local storage of the device.

NFR3: Reliability

As Bro will not be transferred to another development team for maintenance in the foreseeable future, an architecture which guarantees a high degree of reliability is required.

NFR5: Scalability

Bro should be built with the intention of scaling to all municipalities in mind. This will affect the architecture of both the CMS and Bro's front-end module, as they will require a higher degree of modularity.

NFR6: Predictable response time

As Bro has a big focus on usability, it is crucial for the front-end module to be responsive and have the least amount of delays and stuttering possible. The quality of the architecture has a significant impact on this, and a conscious decision was made by the development team to utilize a client/server pattern.

2.5.2 Architectural patterns

The Bro system makes use of several established architectural patterns of varying complexity.

Client/Server pattern

Bro is dependent on having a single source of truth for all content in the service. For this reason, it is beneficial to make use of a client/server pattern as opposed to a peer-to-peer pattern. It guarantees higher availability and better reliability, in exchange for a higher price tag.

Microservice architecture

It is desirable for Lier municipality to have the option to change hosting providers at any time. By utilizing a microservice architecture it becomes possible to relocate certain components without having it affect the rest of the system. For instance, if Lier municipality were to find a cheaper hosting solution for the database, they could easily change the provider, and have the service up and running again with minimal tweaks to the CMS.

BLoC pattern

The BLoC pattern is developed by the BLoC Community [3], which consists of several Flutter and Dart developers. It is a version of the traditional Model-View-Controller (MVC) pattern, modified to work better with the Dart environment. MVC with it's model, view and controller modules have data, BLoC, and presentation module counterparts in BLoC. Due to the team's inexperience with Flutter it was difficult to implement architectures of their own. Using BLoC proved to be a valuable asset in understanding the thought process of Flutter and writing good maintainable code.

2.5.3 Views

These views are illustrated using the UML standard, and their purpose is to convey significant architectural attributes in a format that is easier to digest than pure written text.

Logical View

The logical view in figure 2.14 describes the main application classes of Bro's front-end module. It follows the BLoC Community's [3] specifications and is divided into three packages, *Data*, *BLoC* and *Presentation*.

- The Data package is responsible for all data transfers between the front-end module, the server, and the device local storage through *data provider* classes. Additionally, it contains repository classes that define what data to retrieve externally.
- The BLoC package contains all of the front-end module's business logic. It includes three different types of classes, *event*, *state* and *BLoC* classes. The BLoC classes makes use of the event and state classes to translate the information retrieved from the data repositories into a format that can be used in the presentation package.
- The Presentation package contains all classes that composes the rendered app. It is rendered based upon the states defined in the BLoC package, and also handles user input and application lifecycle events.

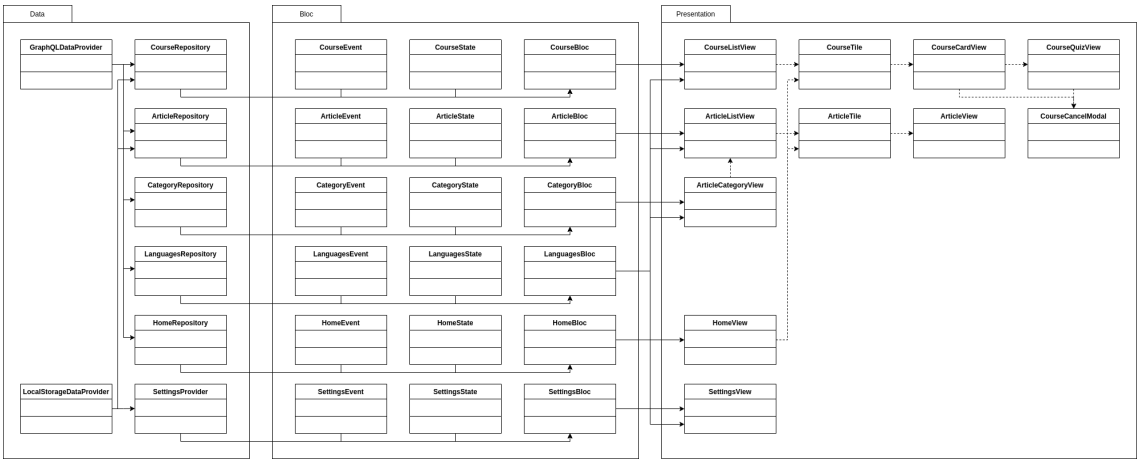


Figure 2.14: Logical view of Bro's front-end module

Process View

The view state machine pictured in figure 2.15 displays the flow of Bro's front-end module. Using this diagram, one can get an idea of which views leads where, and the available views in the application as a whole. The entry point for the application is illustrated with a solid black circle, and any views that lead outside of Bro uses a solid black circle with a white outline.

The most complex part of the view state machine belongs to the *course system*. Some courses can be accessed from both the *home* view and the *course-list* view, meaning that Bro will have to keep track of what views the user has visited in order for them to be able to return to these views. The solution to this is a *cancel-course* modal facilitating the cancellation of a course, and returning the user to where they came from.

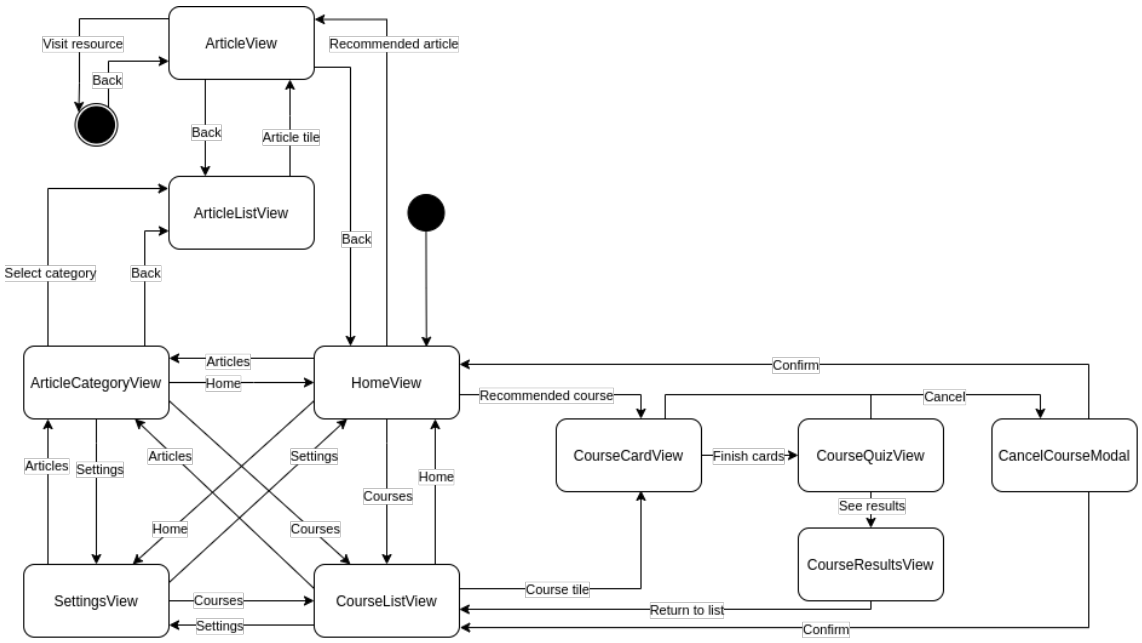


Figure 2.15: View state machine of Bro's front-end module

Physical View

Bro's physical view depicted in figure 2.16 shows three device groups. The *Heroku Cloud* device grouping includes the PostgreSQL database, the Strapi CMS, and Bucketeer for static file storage. Both PostgreSQL and Bucketeer has an interaface to the Strapi CMS, while all external communication goes through the Strapi CMS. The admin and user clients that communicates with the *Heroku Cloud* are represented with an Android, iOS and a browser device.

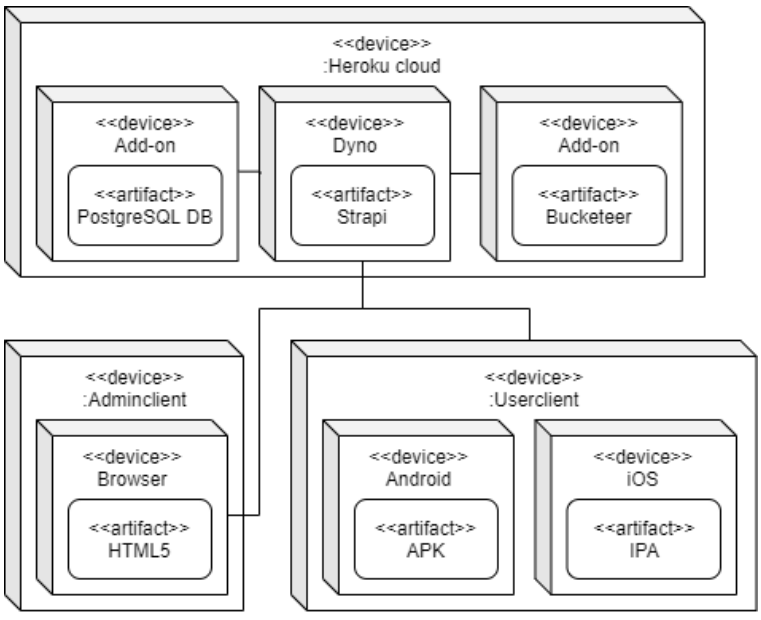
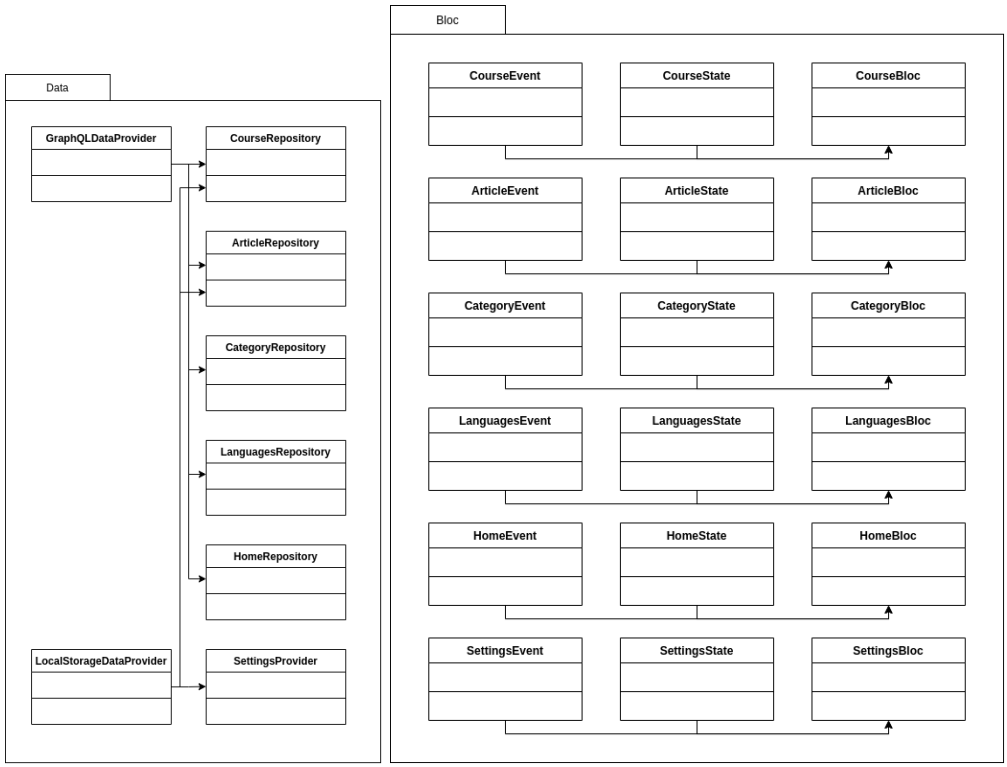


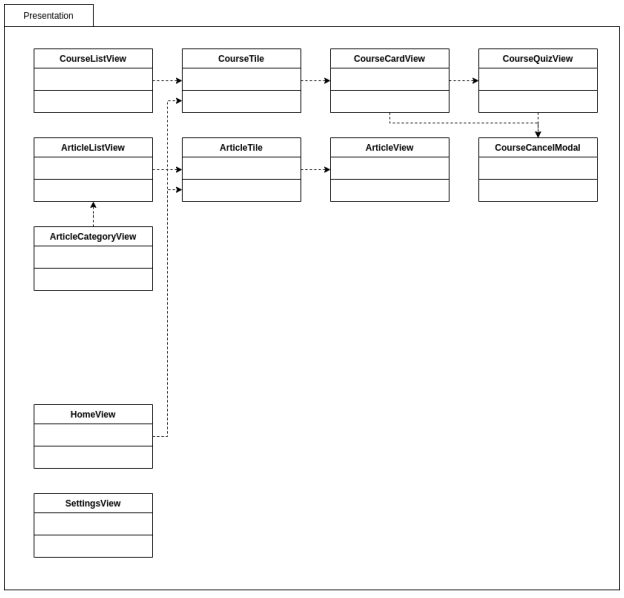
Figure 2.16: Physical view of Bro

Development View

Figure 2.17 depicts the initial development view of the packages in Bro's front-end module. It served as a guideline for the project structure and content, but not as a strict unchangeable plan.



(a) Data (b) BLoC



(c) Presentation

Figure 2.17: The three packges in Bro's front-end module.

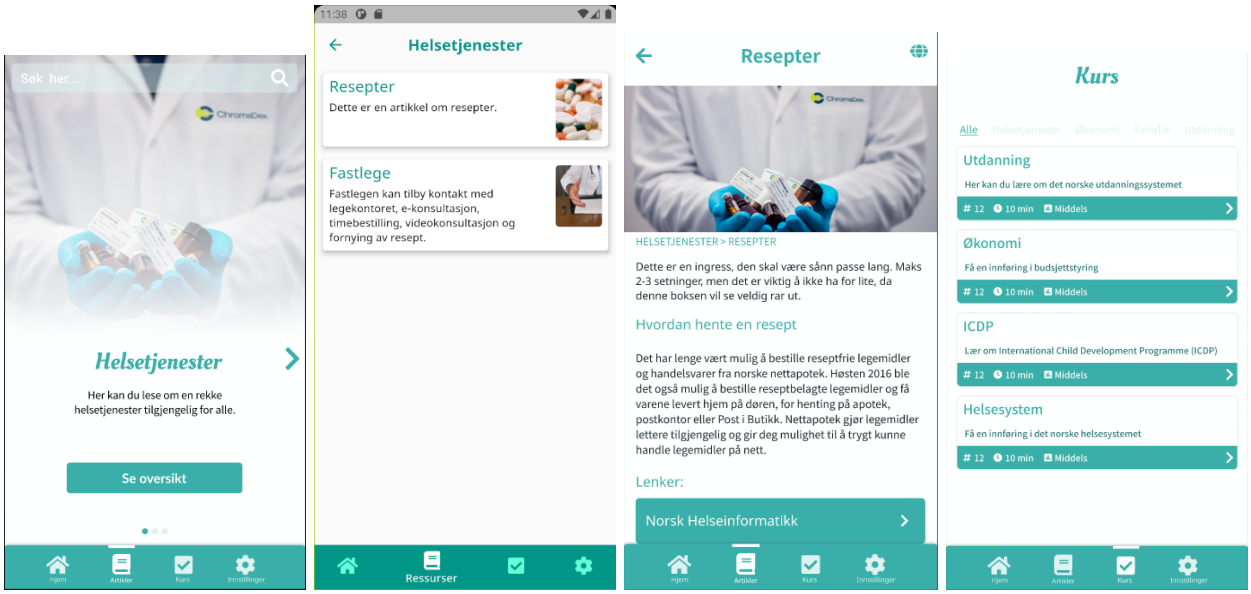
2.6 First Design Iteration

Design is an essential part of app development and Bro is no exception. The design process was especially important since refugees have varying degrees of technical and Norwegian-language fluency. When developing an application, it is important to go through several design iterations to improve the end-product. The team went through two major design iterations during the development of Bro. The first design iteration relied heavily on the pre-study findings and the use case diagrams (2.4), while the second design iteration used the findings from the usability test.

2.6.1 Outline

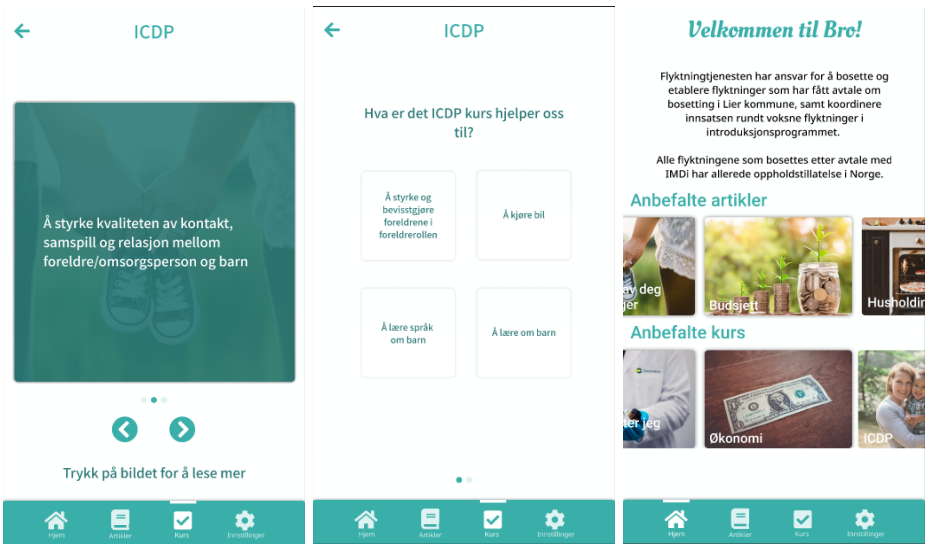
Bro will consist of two main parts: an *article-list* view, and a course-list view. The concept of article is defined by as page displaying explanatory text and links to other websites for more information. Article is where a user should look **to find information** quickly. Bro will also have a *home* view which will be the first view when the application is started.

Articles are separated in different categories shown in *category* view (figure 2.18(a)). The articles are shown in a list view. (figure 2.18(b)). Articles consists of bread-text and references to other sites and documents (figure 2.18(c)). The available courses are displayed in a *course-list* view (figure 2.18(d)). A course consists of multiple information cards (figure 2.19(a)) with bite-sized information, images and videos. After the information cards, the user will be prompted with quiz-cards (figure 2.19(b)). A course is where a user should look to **learn** something. The home view will contain recommended articles, recommended courses and an introductory text (2.19(c)).



(a) Category-view (b) Article-list view (c) Article (d) Course-list view

Figure 2.18: The category-view, the article-list view, the article view and the course-list view



(a) Information card (b) Quiz card (c) Home view

Figure 2.19: The information card view, the quiz card view and the home view

2.6.2 Usability Test *11.03.2021*

During sprint 2, the team was given the opportunity by Selmawit Eng, to have a usability test. Because of Bro having a specific target group, it was preferable to have a stratified sample, with the most important factor being testing on users involved in the introductory program for refugees. Therefore the usability test was performed on four test users that were enrolled in the introductory program for refugees. In preparation, the team made a Figma design prototype which included the functionality that would be tested.

Design Hypothesis

The goal of the usability test was to find potential improvements in the design. This usability test focused on the course section of Bro, since it was going to be developed first. The team had a lot of discussion around the design prior to the usability test and the test would serve as a proof of concept for various parts of Bro. The team was interested in seeing how the users were able to navigate through the product. The team did not know if the users had poor Norwegian comprehension and would find the icons intuitive. The team was also interested in which pages the users would gravitate towards when the team used phrases such as: "You want to learn..." and "You want to find more information about...". The question was whether the users would go to the article section, or the course section. The team's mental model was that if a user wanted to learn something they would take a course, and to find specific information they would read an article. The team hoped that the users would share this mental model. If this could not be effectively communicated, then changes would have had to be made.

2.6.3 Planning

The team had a usability test via screen share with Microsoft teams. The test-team had screen share of a mobile device with the Figma design prototype. One team member served as the test leader and was the only member of the team talking with the test users. One team member functioned as a navigator of the Figma design prototype, moving between slides at the test leaders directions. The rest of the team members filled out an observer form and took general notes. During each test three people would be in the Microsoft teams call: the interpreter, the test user and the test leader. The test leader would go through these points:

1. Introduction of the test leader and the development team.

2. Description the purpose of the test. As well as making it clear that it was the product being evaluated, not the subject.
3. Informing the participant that they can cancel at any time. Thereby, creating security and a sense of control.
4. Description of the equipment in the room and the limitations of the prototype.
5. Explain how to "think aloud", and gain insight into the user's mental model.
6. Explain that the test-leader cannot offer help during the test.
7. Example of the tasks and introduction of the product.
8. Ask if there are any questions before beginning the test.
9. Going through the user test tasks.
10. Ask pre-decided follow up questions
11. End the test by having a conversation about the subject's experience.

An example of one of the tasks is: "You want to take an education. Therefore you want to explore education options in Norway. How would you do this through the app?"

The test task section of the test would be followed by a semi-structured interview, where the test leader would pose the following questions, followed by probing for additional information:

1. What did you like about the product?
2. What did you find confusing?
3. What did you not like?

All the test users filled out a consent form before the test (Appendix B) After the test, a System Usability Scale (SUS) was sent to the users as a Google Form. The SUS will give feedback on the general usability of the product.

		Strongly disagree				Strongly agree
1.	I think that I would like to use Bro frequently.	1	2	3	4	5
2.	I found Bro unnecessarily complex.	1	2	3	4	5
3.	I thought Bro was easy to use.	1	2	3	4	5
4.	I think that I would need the support of a technical person to be able to use Bro.	1	2	3	4	5
5.	I found the various functions in Bro were well integrated.	1	2	3	4	5
6.	I thought there was too much inconsistency in Bro.	1	2	3	4	5
7.	I would imagine that most people would learn to use Bro very quickly.	1	2	3	4	5
8.	I found Bro very cumbersome (awkward) to use.	1	2	3	4	5
9.	I felt very confident using Bro.	1	2	3	4	5
10.	I needed to learn a lot of things before I could get going with Bro.	1	2	3	4	5

Figure 2.20: System Usability Scale

2.6.4 Execution

The team had a bit of a unique situation considering the test users did not speak Norwegian, which meant the user test was conducted with the help of an interpreter in Arabic. Using an interpreter was a new experience, which took some time getting used to for the test leader. Even though the test users were thinking aloud, the interpreter had to interpret which caused delay and confusion. The team spoke to four test users. The first test went slowly. The test leader tried to let the user talk and avoid helping, but the team experienced complete breakdowns (user not understanding how to move forward), and several misunderstandings. Some of the breakdowns could have possibly been avoided by having a longer introduction of how the test would be conducted. As time passed the test leader became more accustomed to the situation and understood that the test leader could say the key points to the interpreter, who would then rephrase and explain in a detailed manner what was said to the test user. This understanding made the test more effective and the last tests were by far the most useful. Another very important factor was that the test leader possessed, although limited, Arabic language knowledge. This made it possible for the test leader to some degree follow the conversation in Arabic, and correct the interpreter multiple times. In the discussions after, it became clear that the test leader had picked up on a lot of subtleties and provided additional insights.

2.6.5 Data Collected

During the user test, audio and our screen showing the Figma slides was recorded. The observing team members collected: A log of breakdowns (Breakdown, Cause, Solution), loose observations (Appendix C). Answers to questions from a Google Form was also collected (Appendix D).

2.6.6 Data Usage

After conducting the user tests, the team congregated a lot of data. This included: a log of breakdowns (when users made a mistake, or did not know what to do next), free observations and eventual clarification

questions between test leader and test user. The team analyzed the breakdowns, to find the root causes. The root causes were studied to see what changes were possible to make. The SUS-score was calculated by a formula [10] based on the answers on the Google Form (Appendix D). The three follow up questions aided the team in analyzing the breakdowns, and provided valuable insight on what the prototype did well. Additionally, through probing the team got feedback on specific elements of the design that would need further development.

2.6.7 Data Analysis

The design prototype had a low SUS-score of 56.3 in comparison to the average of 68 [10], which indicates that the design prototype was poor. What needs to be kept in mind is that the test users did not hold a phone in their hands; they looked at a screen share of a mobile screen. It is plausible that the users would feel more confident, and naturally be able to navigate more smoothly if they held a phone with the Bro front-end module installed.

Breakdown 1

- **What happened**
  User clicked on the home button (figure 2.21) in the navigation bar, when the user was already in the *home view*.
- **Causes**
  The current view is indicated by a white line over the icon in the navigation bar. The user did not understand this concept. The reason might be that the user was not familiar with this type of indication, or that the color of the background and the white bar was too similar, which lead to colors blending in each other.
- **Solutions**
  Changing the color of either the background or the white bar to a more distinctive color, like black.

Figure 2.21: Breakdown 1

Breakdown 2

- **What happened**
  User navigated to the *article* view, via the button labeled article on the navigation bar (figure 2.22), instead of the *course* view when they were given the task of finding a course.
- **Causes**
  The Arabic words for course and article are similar in meaning. The test users did not have a clear understanding of the Bro's definitions of the terms article and course. In other words, their mental models differed on these definitions.
- **Solutions**
  Renaming the concepts of article and course, into something more distinctive for the users. Choosing other icons in the navigation bar is something to be considered as well.

Figure 2.22: Breakdown 2

Breakdown 3

- **What happened**
  User wanted to click on see overview, instead of using the arrow buttons to navigate through the categories in the *category* view (figure 2.23).

- **Causes**
  The user might have understood the "see overview" button as meaning "see overview of all articles", or of "all categories". The dots indicating that one out of three categories is shown, might be too small and hard to notice. The arrow button might also be hard to notice, or the users might not familiar with the concept of moving through pages with an arrow button.

- **Solutions**
  Making the dots larger and brighter will make them easier to notice and is a possible solution. Changing the color and size of the arrow button is another possible solution. If the concept of navigating with such arrow buttons is too alien for the users, a change of navigation structure is also possible. Changing the text from see overview to see article overview, is also worth considering.

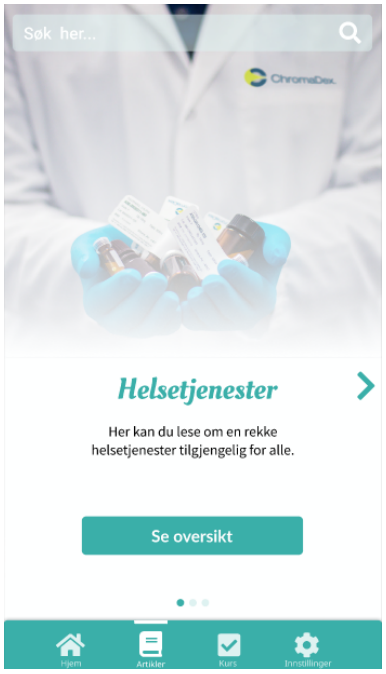


Figure 2.23: Breakdown 3

Breakdown 4

- **What happened**
  User did not understand that it was possible to scroll in the *category* view (figure 2.24).

- **Causes**
  There was nothing that helped the user understand that scrolling was possible. It looked like a margin at the bottom was missing and that the page ended with the turquoise button. The format of the user test is also a possible cause, as the user told us that if he held the phone in his hand he would naturally try to scroll the page.

- **Solutions**
  Some form of showing that the page is scroll-able. For example a scroll bar on the right hand side, or an arrow that shows you can scroll down.

Figure 2.24: Breakdown 4

Breakdown 5

- **What happened**
  Multiple users were confused and got stuck interacting with the recommended articles and courses on the *home* view (figure 2.25).
- **Causes**
  Because there was a lot of content immediately shown to the users, they struggled to understand that there were other views.
- **Solutions**
  Only showing the most important content and hiding the rest behind accordions and dropdown-buttons.

Figure 2.25: Breakdown 5

2.6.8 General Feedback on the Prototype

What Users Liked:

The users expressed that the design of Bro looked professional. The users liked the icons and felt they were intuitive, with the exception of the article and course icons being a bit too similar. The team confirmed that users would use the product, even if most of the content ended up being in Norwegian.

The most important thing the team learned from the test was that users with low reading comprehension would skip through text-filled parts of the site. Therefore, all redundant text will be removed. Only essential text should be shown on the screen at any time, which means that additional text should be hidden and accessed through buttons or links.

What Users Did Not Like:

The users expressed a lack of distinction between courses and articles. The navigation through a course demanded patience, and if someone was impatient they would lose engagement.

2.6.9 Conclusion

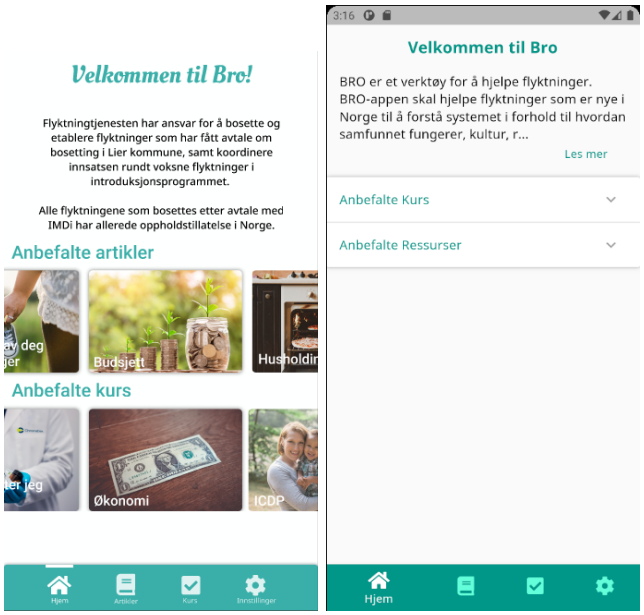
Having a usability test with test users helped us improve the product's prototype by providing us with a better understanding of the technical limitations of the team's end users. The test gave us insight as to how users with low reading comprehension navigate apps, which helped us avoid pitfalls. Based on what the users expressed during the tests, a lot of the design was user friendly which was a focus point in the development process.

2.7 Second Design Iteration

During the second design iteration the team altered the design based on the findings from the usability test. Overall the changes made were mostly to improve visibility. From the first usability test the team got a much better idea of the technical aptitude of the target group. Additionally it was discovered that people with low reading fluency have a tendency to skip longer paragraphs of text, therefore we locked away content behind "view more" buttons and accordions.

2.7.1 Major Changes

Content on the *home* view was hidden behind accordions and a "view more" button in order to help with visibility as a result of general feedback on the design prototype and lessons learned from **Breakdown 5** (figure 2.26).



(a) Before (b) After

Figure 2.26: Increased visibility

The navigation bar function of indicating which view is currently showing was changed to increase visibility as a result of **Breakdown 1** (figure 2.27).

(a) Before (b) After

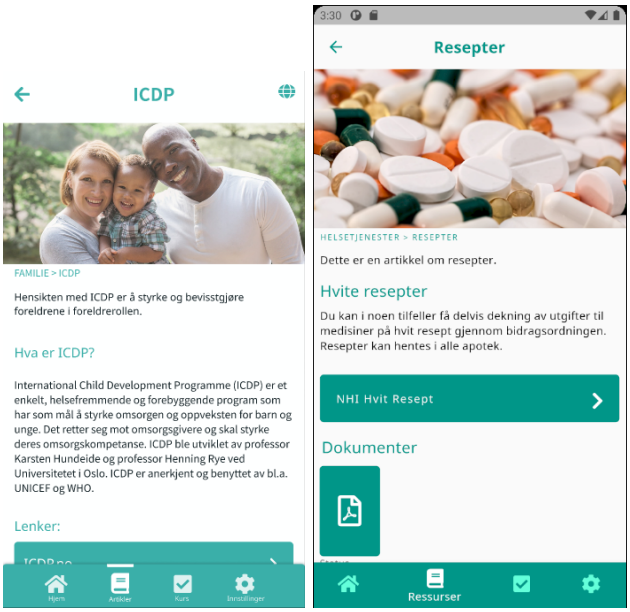
Figure 2.27: Changing navbar to increase visibility

The term "Article" was exchanged for "Resource", as a result from analysis of **Breakdown 2** (figure 2.28).

(a) Before (b) After

Figure 2.28: Rename of *Article* to *Resource*

"Resources" pivoted from having more content generated by the Lier municipality staff, to having less in-house generated content and rather refer to other sources. This change was a result of feedback from staff in Lier municipality, as they did no want to produce as much content as originally planned (figure 2.29).



(a) Before (b) After

Figure 2.29: Changes to Resource format

Borders were added to the question alternatives, so that they could more easily be separated from the background as a result of general feedback on the design prototype (figure 2.30).

(a) Before (b) After

Figure 2.30: Changes to quiz border

2.7.2 Second User Test

The team was planning a second user test, where the priority would be to test an application prototype on mobile devices. This would prove or disapprove the team's former hypothesis; the low SUS-score was a consequence of users not having a phone in their hands while testing. Considering the test would occur so late in the teams development process (between the second to last and the last sprint), the results would not affect the end-product in a high degree. The team would only be able to fix minor- bugs and design changes. Any substantial changes would not be possible with the team's remaining time-frame, but would be valuable information for any further development. Another important aspect of this user test would be to test an almost finished application with actual content. This way the staff at the refugee service would get an idea of what kind of content users liked, which would help them in producing content. Due to unforeseen circumstances (Sprint diary 5: section 3.5.7), the second user test was not executed.

2.8 Implementation

For the development process to go as smooth as possible, it is important to have properly defined routines and planned out technologies.

2.8.1 Definition of Done

After breaking down the stories into tasks, the team needed to have a consistent definition of done. The criteria set for the tasks were as follows:

- Acceptance criteria met
- CI/CD (continuous integration and continuous deployment) passed
- Unit tests passed
- Widget test passed
- Linting
- Code review passed
- Non-functional requirements met

When completing the set of tasks belonging to a user story, the user story also had to pass the following criteria:

- Accepted by the Product Owner
- End-to-End tests have to be written and passed
- A manual functionality test has to be performed

2.8.2 Tech Stack

Bro makes use of a variety of powerful and modern technologies that excel in agile development environments.

Front-End Application - Flutter

Flutter is a cross-platform UI development toolkit that utilizes the Dart programming language. Bro's front-end module has a focus on usability and Flutter provides the team with the tools required to make an application with consistent looks and features, regardless of OS version or platform, with the least amount of unnecessary work.

Having the same look and feel is much easier achieved using Flutter, in comparison to other solutions such as React Native. As React Native sometimes require platform specific native code, it also requires more testing. Considering the time constraints on this project, it is beneficial to have more capacity for actual development.

Back-End Content Management System - Strapi Headless CMS

Strapi is a headless CMS, and is usually used for systems such as blogs and newspapers. It is designed in such a way that content creators who are not technically adept can still utilize the service efficiently. Strapi is composed of *content-types* that define the structure and details of the content. Additionally, it generates API endpoints when new *content-types* are created and has a pre-built user interface. This massively cuts down on development time, which is better spent improving the usability of Bro's front-end. It also features administrator login, with different roles. For the free version of Strapi, these are *Super-Admin*, *Editor* and *Creator*. This enables the content creators from Lier municipality to adapt a publisher and creator workflow, which in turn should improve the overall quality of Bro's content. The *Super-Admin* role is meant for the development and possible maintenance of the product, and permits the administrator to make changes to *content-types*. When Bro is deployed to the public, this role will most likely not see any more use.

Strapi has three advantages that made it suitable for the Bro project. The first being that it is self hosted, meaning that Lier municipality has the ability to change hosting provider in the future. Secondly, the project can be completed with the free tier of Strapi. Thirdly, Strapi has content internationalization, which is very useful as Bro supports different languages.

There are several headless CMS products that are more mature and feature rich than Strapi on the market. Sanity.io and GraphCMS, being two of the more popular alternatives. However, the cost and forced hosting solutions provided by these services, made them less suitable for Bro.

Deployment - Heroku

As Lier municipality did not wish to host the CMS solution at their servers, the development team was tasked with finding an alternative. After considering a couple of hosting providers, Heroku was pitched and approved by Selamewit Eng.

Heroku has coined the term dynos, which consists of containerized applications using Docker. Developing with Heroku is free, and enabled the team to start development with the actual production environment from the beginning. Additionally, Heroku provides a series of "add-ons" that expand on the regular dyno services. Bro makes heavy use of images, videos and other static files which in turn means that the product

will require static file storage in addition to a database. The Bucketeer and Heroku Postgres "add-ons" will be used for this.

Among Heroku's competitors one can find tech giants Microsoft and Amazon, with their respective services Azure and AWS. Both of these services have an abundance of features and complexity, which become unnecessary for a project of Bro's size. While Azure and AWS has very competitive pricing for their more performance heavy tiers, Heroku makes up for it with their relatively cheap standard plan.

2.8.3 Coding Conventions

Google supplies style guidelines for the Dart language that they call the "Effective Dart: Style" [2]. It consists of general guidelines for how Dart code should and should not be written, and ensured a consistent code look and quality throughout the development of Bro's front-end module. To uphold this adopted style the team made use of a linter, that would analyze and report any inconsistencies as a part of the CI process. In addition to this, the CI would run all tests to ensure that functionality would not break upon merging new code to the master branch. The CI would run once for every new commit to a pull request on the master branch.

After the CI would pass, the next step in integrating the new code was code review. Whenever a team member created a pull request (PR), they would also ask for a review from one or several of the other members. The reviewers would pull the code to their local machine and go through established code review guidelines (Appendix: F) while reviewing the changes. If everything looked good and passed the guidelines, it would be approved. The original author would then be notified, and could finally integrate the code to the stable master branch.

For Bro's back-end module there was a bit of a different process involved. Strapi does not necessarily require writing any code apart from the initial database and deployment configuration, as it is primarily used as a graphical interface for content management. As Strapi, the CMS, was deployed to Heroku and was thereby put into a production environment, changes to the *content-types* could not be made dynamically. Instead changes had to be done in a local development environment and be pushed as PRs to the project's Github repository. After being reviewed through the same process as with the front-end module, it could finally be integrated into the stable master branch. Upon merging, Heroku would automatically deploy the new version to the production environment through the CD configuration.

2.8.4 Product Rollout

Bro's front-end module was designed for both Android and iOS under the name "Bro - Flyktninghjelp". To be able to deploy to these operating systems, the application had to comply with the standards set by the Google Play and Apple App store. Both of these repositories have strict on design quality, userdata and monetization. During development most of these criteria were fulfilled or did not apply to the product.

Both the Google Play and Apple App store have some sort of internal testing program available. This is so that the application can be beta-tested before it is actually deployed to the consumers. In this case it was quite useful, as most of the actual content (at time of product hand off) was not been inserted into the application. The creation of such content is up to the refugee service in Lier municipality. This means that the staff there can add and remove content as much as they want, until the product reaches a satisfactory state. Upon reaching such a state, Selamewith Eng can deploy the application, full well knowing how it looks.

At the time of product hand off, the application was in Google Play's "Internal Testing"-program. Selamewit Eng has been given instructions on how to add users to the internal testing environment. In addition, Eng has also received instructions on how to publicize the application to the stores. This way she can do so when the refugee service reach a satisfactory amount of content for application. All accounts and emails related to the application were created by the refugee service, to ensure that they had absolute control of Bro, when

the time came for the handover came.

Unfortunately, due to an oversight, the team had not anticipated the length of the Apple developer account review process [4]. Which according to some users on their forums usually takes about two weeks, but could take even longer. This meant that we would not be able to submit Bro for Apple's application review in time for the project deadline. Considering the good relationship the team has built with Selamwit Eng over the course of the project, they have informed her that they will assist in launching Bro's front-end module to the Apple App Store upon receiving Apple developer license approval.

2.9 Code Testing

Code quality and testing is an important aspect of software development. Ensuring an error free, bug free and crash-resistant application is at the core of user-centered design.

The usability of Bro is of the upmost importance due to the lack of a permanent development staff at the refugee service. If unexpected behaviour occurs after deployment, the staff at the refugee service will not have the means to solve this by themselves. Good test coverage is therefore needed for the product to be reliable. This is something the team will accomplish through rigorous quality assurance routines.

2.9.1 Unit Testing

Unit tests, test smaller pieces of code and will expose any flaws in the functionality it covers. These remain useful throughout the development process as they will ensure that any changes to the code do not change the desired outcome.

Unit tests were an important part of Bro's development, due to the aforementioned lack of a maintenance team. By ensuring consistent logic via unit tests, the team could confidently assume that the logic would not distinctly change when implementing new features.

2.9.2 Widget Testing

Widget testing is an important part of ensuring a smooth user experience. Where the unit tests ensure consistent logic, the widget test ensure consistent interaction. The separation of buisness logic and presentation is made explicit in the BLoC architecture. Thus widget testing takes care of ensuring the quality of the presentation package.

Having widget tests ensures that any changes to the codebase will not affect how the user experiences the application. As user experience and interaction is key to this application, the widget tests are a way of keeping unwanted UX-altering changes from affecting the project.

2.9.3 Null Safety

The use of nullable values in Bro's front-end module was a concern during programming. Having an application crash for an unknown reason has a huge effect on the user experience. Combine this with Lier municipality's reduced ability to fix the problem, due to a lack of developmental staff, it poses a danger to the robustness of the application.

On March 3rd 2021 Flutter 2 was announced, which implemented the sound null safety feature from the Dart programming language [9]. This was a huge break, as it meant that the team could spend less time on dealing with null-instances. It also meant that it made it easier during development, as one could keep the dirty, mutable states inside the business logic. This meant that the views would explicitly tell which values needed null-handling and which values could not be null. Taking the time to implement this feature, was

seen as a huge win, as it let the team spend less time on null-handling during further development. It also assisted the team in creating a more robust application.

2.10 Finished Product

As the Bro project came to an end the team had managed to develop and deploy both the front-end to Google Play and the back-end module to production environments. The staff in Lier municipality had been granted access to Strapi (Bro's back-end module) with their own accounts and was creating content as the front-end module was pending approval from both Google Play, and Apple App store. The team had written a comprehensive user manual (Appendix E) for how content should be made, and partook in digital meetings to train the staff in the new system. Instructions for how Selamewit Eng could finally release and publicize the application when the content was ready, were also conveyed.

2.10.1 Bro's Front-End Module

The centerpiece of the project, the front-end module, ended up consisting of four primary views: the home, resource, course and settings views. It is fully functional on both Android and iOS devices, and very close to the original vision of the application.

The home view (figure 2.31(a)) displays an introductory message from the staff in the refugee service at Lier municipality, as well as two accordions containing recommended courses and resources. The recommended courses and resources are specifically selected by the staff to promote certain content.

The resource views are a bit more comprehensive and are comprised of three parts, a category carousel, a resource list view, and the resource detail view. The categories (figure 2.31(b)) have big pictures symbolizing their contents, and can be navigated between with either swipes or taps. Within each category the resource-list view (figure 2.31(c)) can be found, with tiles including the cover photo and a short description for each resource. The resource-detail view (figure 2.31(d)), includes the same description and cover photo, as well as references with explanatory texts, and appended documents. Upon clicking a link in a reference an integrated device specific browser is opened with the correct URL. This view is placed "on top" of the application, meaning the user does not have to exit the app, and can easily navigate back. The same goes for the documents, however, they do open the appropriate default app for the file type, exiting Bro.

The course-list view (figure 2.31(e)) is comprised of all available courses in the selected language. Each tile includes a title, a description, a number representing the sum of slides in the course, a time estimate, and the associated category. Upon clicking a tile, the user is redirected to the course-detail view (figure 2.31(f)), containing pictures and useful information. The cards can be navigated between by use of the arrow buttons or swipe gesture. Additionally, the user can reveal information about the subject by tapping on the pictures (figure 2.31(g)). After having gone through the slides and studied the material the user is redirected to the quiz portion (figure 2.31(h)) of the course. The quiz view presents several options to the user, along with the question text. When the user has their choice, a screen displaying whether or not the alternative was correct along with a clarifying text is displayed (figure 2.31(i)).

Lastly, there is the settings view (figure 2.31(j)). Originally, a profile page was planned (as shown in figure 2.13), but this was not prioritized and a simpler settings page was created. The finished product included a single setting, language. Tapping the language input box, shows a drop-down menu where the user can change the application language. Upon doing so the content from Bro' back-end module would be re-fetched and updated accordingly in the front-end module.

2.10.2 Bro's Back-End Module

To fulfill Selamwit Eng's wishes about being able to create content for the application after launch it was convenient to make use of a already finished product that had the capabilities the project required. Strapi, the CMS technology used in Bro's back-end module, has many such capabilities and provided the team with a solid backbone for Bro's front-end module.

The user manual, written by the team describing instructions for how Strapi is and will be used can be found as Appendix E. Figure 2.32 shows the interface where Bro's *content-types* were created. These include *category*, *course*, *course group*, *language*, *publisher*, *resource*, and *resource group*. The relationship between these *content-types* can be seen in figure 2.33, and describes the structure the team implemented through the use of the CMS.

2.10.3 Under the Hood

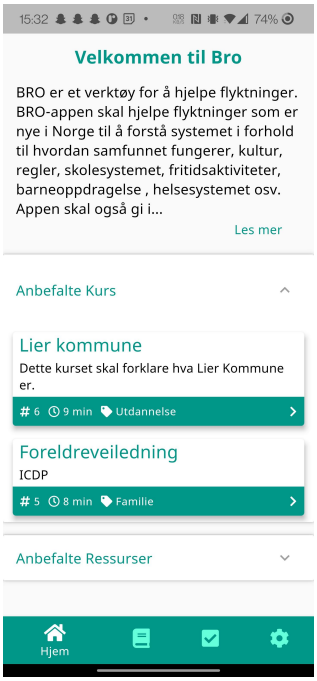
Despite having to retrofit BLoC in sprint 3, the implementation worked well and created a solid foundation for further development. By using BLoC the team achieved immutable states, and separation of logic and presentation. This allowed the team to work more efficiently on code which would otherwise be overlapping. It also made the project more readable and facilitated for further development in the future.

The sound null safety feature released with Flutter 2 created a large challenge as the team decided to implemented the necessary changes to make use of the new functionality. Although the implementation required a lot of work, it resulted in a far more robust application.

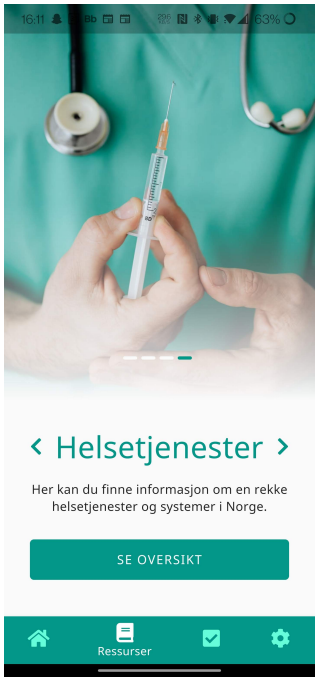
By implementing these features into the product structure, the application has laid a solid foundation for further development. The separation of logic and presentation in BLoC, ensures that it is easy to implement features, while null-safety assists in making a robust product. Combining these with the cross-platform capability of Flutter and the ease of use of Strapi ensures that the application's potential is not limited by the technologies used.

2.10.4 Success Metrics

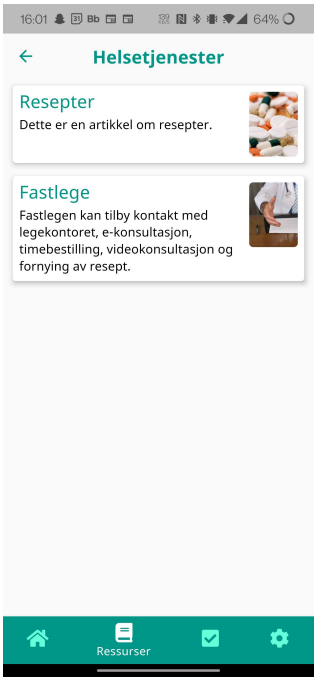
As the success of most of the metrics defined in table 1.1 were not possible to determine before getting Bro into the end user's hand, it was difficult to conclude something concrete. However, the Bro system does have the capabilities to provide refugees with useful information, as well as reducing potential physical documentation. Additionally, it was outfitted with the tools required to achieve the other goals. The applications ability to satisfy the remaining goals will largely depend on the quality of the content produced by Lier municipality. Judging from the feedback from both the user test, and customer meetings the product has reached a satisfactory result.



(a) Home view



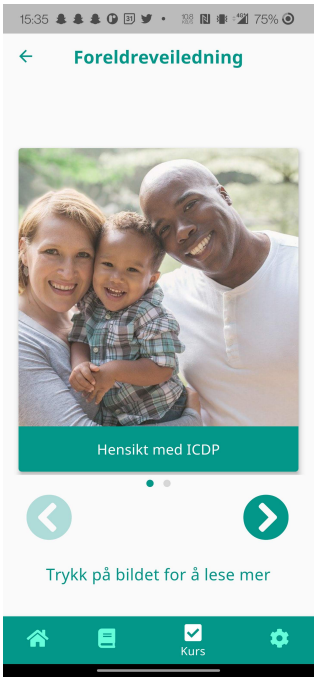
(b) Category carousel view



(c) Resource-list view

(d) Resource-detail view

(e) Course-list view



(f) Course-detail view

(g) Course-detail view tapped

(h) Course-quiz view

(i) Course-quiz view correct answer

(j) Settings view

Figure 2.31: Final screenshots from the Bro front-end module.

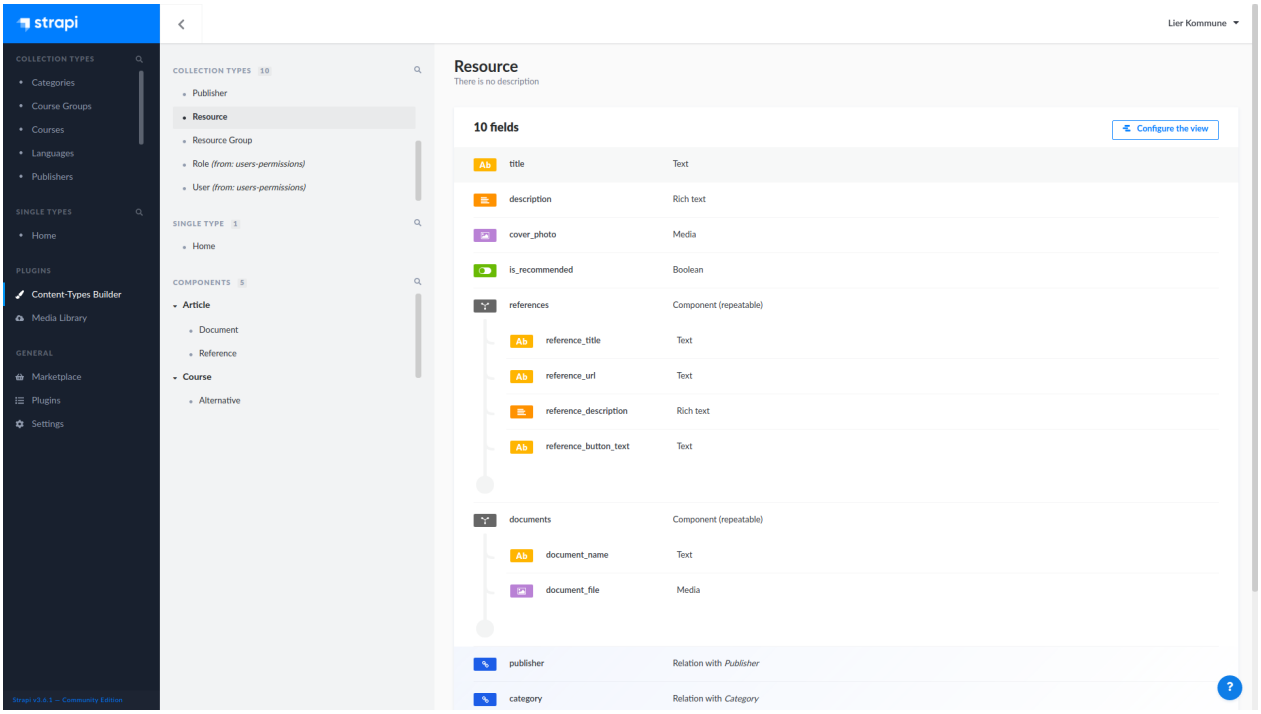


Figure 2.32: Strapi content-type creation interface

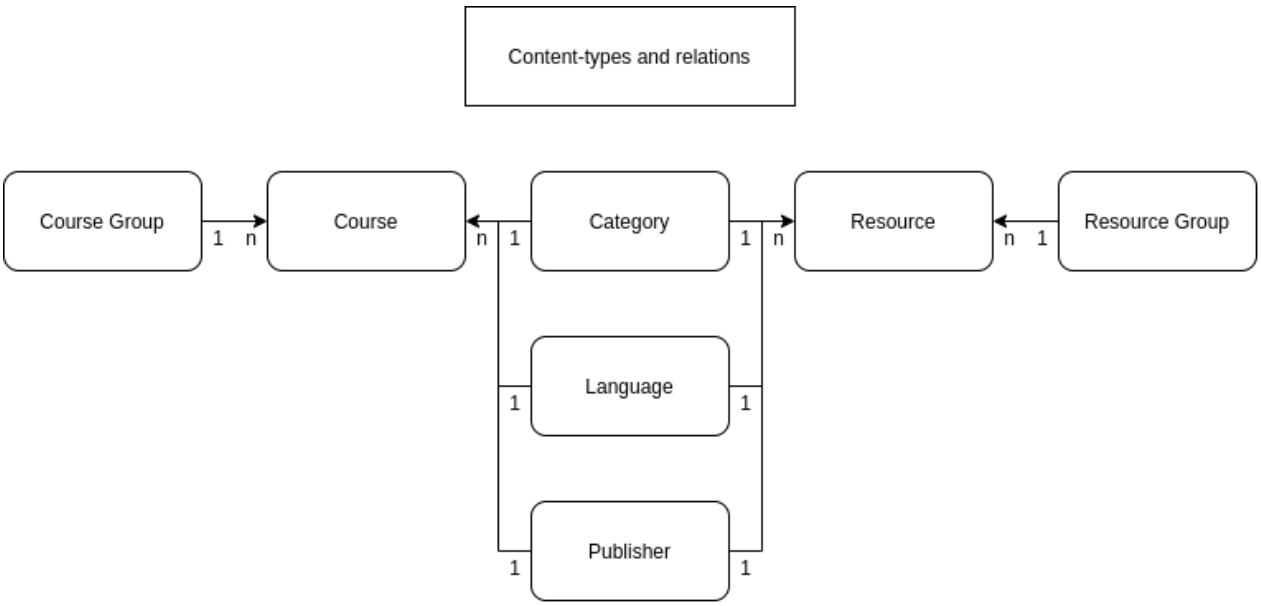


Figure 2.33: Content-type relationships

2.11 Product Summary

During the pre-study the team found many applications with similar functionality but none offered exactly what Selamewit Eng was looking for. As there was no alternative solution it proved necessary to develop a new solution with inspirations from applications with some overlapping functionality.

The technology chosen by the team made provided a new set of challenges, but also fit the teams application perfectly. Most team members had no previous experience with the Flutter framework and the Dart language. Normally this would cost the team a significant amount of time, but the framework had many features that sped up development without adding complexity. Additionally, the setup process was very developer friendly and thereby headache free. Strapi served its purpose by providing a simple to user interface for the content creators.

Since the team's time with the refugee service was limited it was necessary to ensure that the staff would be able to add, edit and remove content. This goal was reached by utilizing the Strapi interface, which leaves the lifespan of Bro up to the content creators.

The main goal of developing Bro was to help the refugee service share useful information to the refugees. This was achieved by developing the article system. The articles contain relevant information and useful references, that remain accessible to the refugees even outside of the classroom. The course system also provides refugees with an interactive learning platform which doubles as another point of exposure to the Norwegian language.

When the team first started on the development process, measurements of success were laid out (1.1). The team has provided Lier municipality with all the necessary tools, in Bro's front-end module and back-end module, to achieve these goals. From this point on it is up to the content creators to produce content and hopefully provide refugees with a great platform.

Chapter 3

Process

3.1 Development Process

For a software development team to work efficiently, certain rules and guidelines should be followed. During the first meetings, the team discussed and decided upon a methodology, collaborative working hours and product management tools.

3.1.1 Methodology (Scrum)

The team chose Scrum as the core methodology. This was a conscious decision based on the technical aptitude of the product owner, as well as the need for an agile work style. This enabled the team to make decisions regarding the task priorities as well as including the product owner as an integral part of the the development cycle [5].

The sprints were set at 2 weeks long. This gave the team the maximal amount of flexibility without an unnecessary amount of overhead due to meetings at the start and end of each sprint. The need for flexibility was based on the vagueness of the product idea and the project's initial lack of functional requirements.

Sprints

The beginning of each sprint on Mondays, started out with a sprint retrospective followed swiftly by a sprint planning meeting. This was done to create a quick feedback-loop in which the team was able to identify problems from the previous sprint and solve them in the same session. This also helped in creating more accurate sprint predictions in size, scope and workload.

With the sprints being 2 weeks long, the sprint was concluded at the Sunday two weeks from the initial sprint meeting with its own sprint review. These meetings were primarily centered around reviewing the work done and whether or not the team had achieved the sprint goal. Sprint reviews were also used to demonstrate the work done.

Daily Stand-Ups

In an attempt to secure a common understanding of project status, there were also daily stand-ups communicating task progress. As most days had meetings or collaborative working hours, these stand-up were folded into those activities. For the days that did not have such activities, a stand-up meeting was held.

The intention of the daily-stand ups was to create a space where one could request assistance if one were to get stuck, and to give the team an understanding of the progress of the other team members. These

stand-ups also established an arena to create a common direction. This indirectly also ensured consistent project progress.

Adjustments

The team had a lot of experience working together and planned the process thoroughly at the start of the project. Therefore few significant changes were made during development, however pair programming was introduced during sprint 2 and had a significant positive impact on the teams performance during physical work sessions. Originally, pair programming was introduced to deal with large tasks that were hard to complete alone and could not properly be split into sub-tasks because of dependencies and conflicts. After good results with pair programming in sprint 2 the team decided to keep using the approach in later sprints.

3.1.2 Collaborative Working Hours

The team had an already established group dynamic, due to collaboration on previous projects. This let the team focus more on working rather than spending time on organization. It also helped the team organize more effectively, as they already knew how every individual works best. Thus, the optimal balance between group sessions and individual work was quickly found.

The project work was centered around team sessions (Scrum, working sessions etc.) and individual working hours. The use of individual working hours let the members work on tasks in their own time while also solving the issue of non-overlapping schedules. Daily stand-ups were also added to ensure progress and make rapid decisions.

Workshops

In addition to meetings, it was quite important to be able to collaborate, especially when working on related tasks. To achieve this, collaborative working hours were set up to make sure that there was a common time slot where everyone was available to have discussions and work together. This ensured that the teamwork would be efficient and the project progress would not be halted due to uncertainties.

Timetable

After the preferred working scheme was decided, the team identified the universally available hours. The balance between individual work and collaborative work sessions was struck and the team settled on the arrangement described in the timetable below. Note that the timetable is used to show the important meetings during **both weeks**. Thus, there were some times which exist but are not listed in the timetable. One example of this would be the "sprint planning"-meeting during the first week of the sprint. As it was not logical to have sprint planning in the middle of the sprint, this meeting was replaced with a daily stand-up.

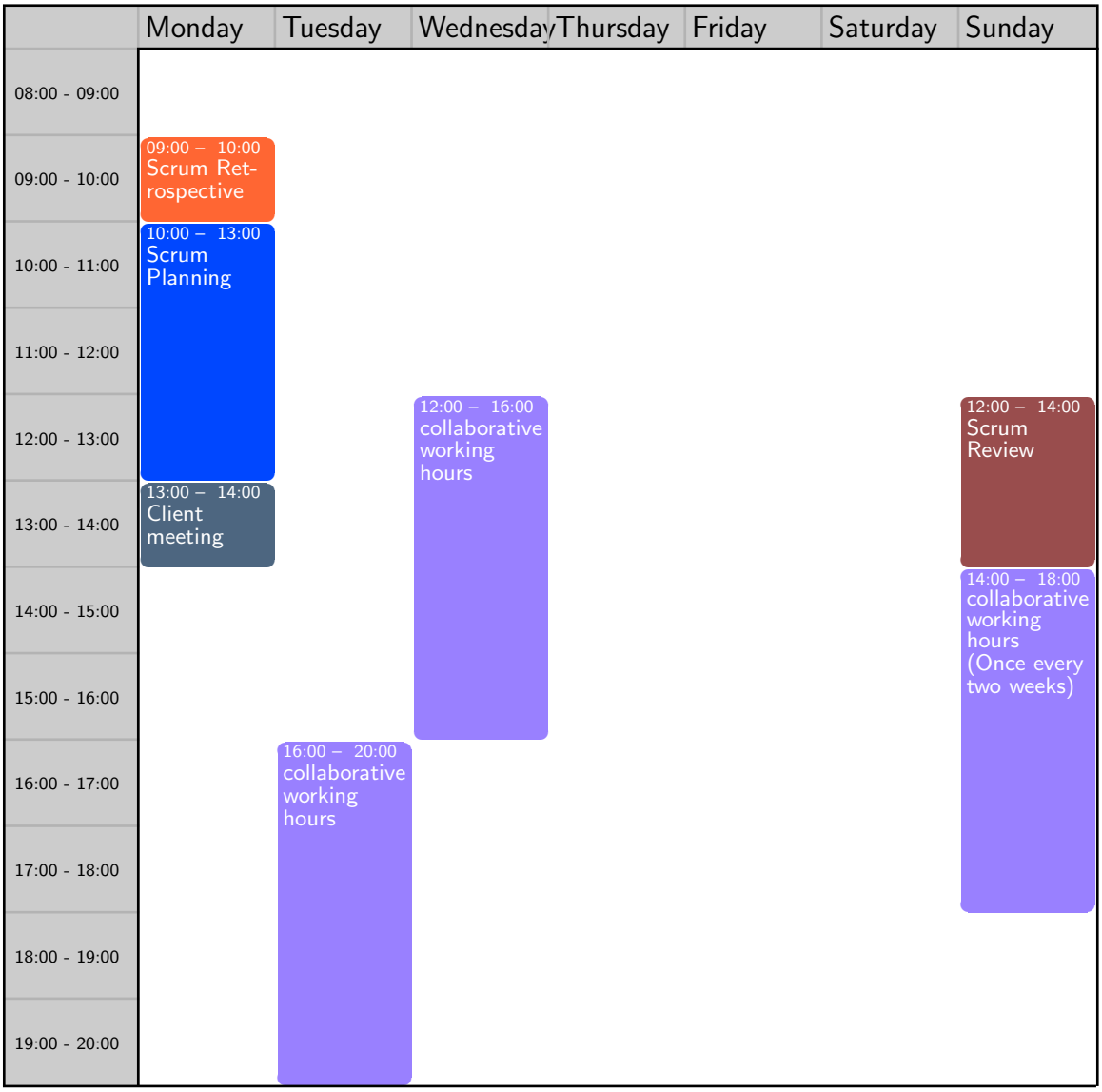
One sprint contains the following (as shown in table 3.1):

- One hour of retrospective
- Three hours of Scrum planning
- One hour of client communication meetings. This would only exist for the one week without Scrum planning
- Twenty hours of collaborative working hours. Note that the Sunday session is only once every two weeks.
- Two hours of Scrum review

The team agreed in advance to spend 20 hours per week on this project. This totals to 40 hours per student, each sprint. The team has planned 27 hours of organized work time per sprint. The team agreed that each individual member would find the final 13 hours of work time within their own schedule.

f

Table 3.1: Scrum timetable overview



3.1.3 Product Owner Communication

Since Selamewit Eng could relate to both the perspective of a refugee and the perspective of a refugee service worker, it was of especially high importance for the team to utilize this understanding. To take maximal advantage of this Selamewit Eng was added to the development teams Slack and Jira work-spaces. Selamewit Eng was also kept in the loop via mail and Microsoft Teams meetings.

The product owner was invited to every Scrum planning to provide feedback and confirm that the direction and changes being made in the sprint were in accordance with the wishes of the product owner.

On Mondays in the middle of a sprint, there were no sprint planning meetings. To prevent long feedback-loops, we set up customer meetings during these same hours to substitute for a lack of Scrum planning meeting.

3.1.4 Product Management Tools

Proper tools to manage a project like Bro is important when working in teams. This is especially important when external factors, such as COVID-19 safety protocols, force the team to work remotely.

Version Control

Version control is critical, particularly for a mostly distributed team such as in this project. Version control is useful to keep the code conflict-free and ensure painless merging, which would otherwise be very difficult when working remotely. Git was essential to separate a production ready version from development, and to prevent code deletion. Github was selected because the project can be hosted there for free, and it features a plethora of integrations with other tools used in this project, such as Jira. The Github project organization can be found under the name LIER01.

Wiki and Kanban Boards

Jira allows to keep a wiki and sprint and report kanban boards in one place. After trying some other services (such as Trello) the team also found that Jira was easiest to use for sprints. Kanban boards and had some important features, such as integration with Github, which made it easier to get an overview of progress. Additionally the product owner could be invited to the service, and thereby have the ability to check in outside of meetings.

The wiki was hosted through Atlassian's Confluence, and features templates, referencing for user stories, and much more.

Communication

Slack was chosen as the formal communication channel. This is because the channel and message reply system in slack allows for good structure and it is easy to reference conversations. It also proved more efficient for rapid back and forth communication with supervisor and customer than what can be achieved via email.

3.2 Team Organization

Team organization is an important aspect of starting a new project. To create an outline for the teams collaboration, the team wrote a team contract and assigned roles for each member. The team contract creates guidelines for what team members can expect and demand from each other, in terms of effort and general attitude.

3.2.1 Team Dynamic

Previous to the project, the team had worked together in other subjects. Having this previously established group dynamic meant that the team knew how they cooperated best and made fixing conflicts easy. This group dynamic also meant that the team could bring up issues without being concerned about creating a negative work environment. In turn, this meant that the team could dedicate most of their energy on working together on the project. If the team did not already have a good team dynamic, time would had to be spent on building up a team dynamic and creating common understanding.

3.2.2 Team Contract

The team members discussed different topics, which were: Communication, Work Ethics, Process, Attitudes, Expectations, Decisions and Encouragement. From the discussions, a team contract was written and signed by all the team members (Appendix A).

3.2.3 Roles and Responsibilities

Table 3.2 shows the roles the team defined, along with the responsible member and the responsibilities of that role.

Table 3.2: Role definitions

Role	Role holder	Role responsibilities
Team leader	Amund	Customer and supervisor contact. Attending team leader meetings. Being the main spokesperson in customer and supervisor meetings.
SCRUM-master	Amund	Leading the SCRUM meetings (sprint planning, daily stand-up, retrospective)
Meeting reporter	Patrick	Setting up the meeting documents before each meeting and assigning meeting secretaries.
Report manager	Lotfi	Keeping an overview of what was worked on and who was writing which part.
Quality assurance	Vegard	Delegating code review responsibility between members and keeping the work load even.
Room booking	Patrick	Making sure the team has a room for every meeting.
Welfare contact	Hjalti	Solving internal conflicts. In charge of team welfare.
UX manager	Hjalti	In charge of Figma design and consistent design implementation.

3.3 Project Management

In any project it is important to have an overall plan as to the execution. It strengthens efficiency by eliminating uncertanty within the team. It also lets teams make long-term decisions knowing that it will pay off.

The sections below describe the project management decisions made in the project. They are mainly centered around the Project plan and the risk analysis (discussed in section 3.3.2). Both of these provide the team with a holistic view, and helps in planning ahead for any known and unknown incidents.

3.3.1 Project Plan

The project plan (table 3.3) is an important aspect of any team oriented project. It identifies deadlines, milestones and creates perspective for the team. Having a project plan creates a sense of assurance, as the team could make developmental decisions which had a long term payoff. Having such a plan also creates incentives for good structural solutions, as it helps the team see beyond the current sprint and see the entire development process.

Having such a plan also creates a single source of truth for the team. It let the team plan around expected events, knowing that it may impact development. It also helps in focusing work, as it creates a solid structure for the project. Knowing what is the next milestone or deadline is important for the team when making prioritizing work. Every task is important, although some task may be more important than others.

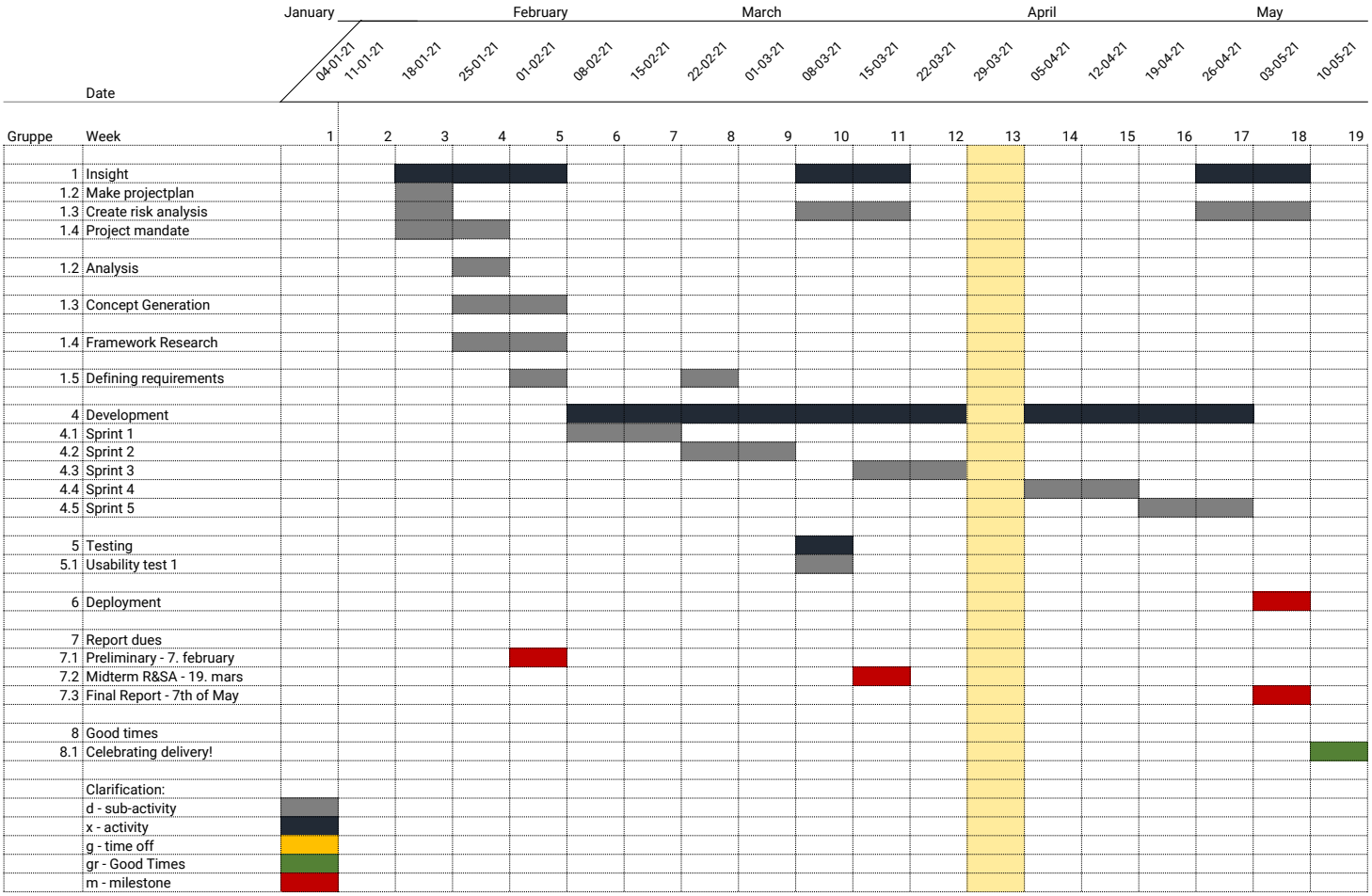
It is worth noting that this project plan is just that; a plan. While it outlines how the team expected

to work throughout the semester, it was fully expected for the plan to change. Being able to adapt is an important part of an agile development process. Thus, changes were made to the plan to adapt to the situation in that point in time.

Changes to the Project Plan

One such change was delaying the initiation of sprint 3 by one week. The reason for this was the looming midterm report in addition to performing user tests. The team knew from sprint 1 that a user test would be possible, but the time and date was not set. During the beginning of sprint 2 the team was given the opportunity of having a user test, but the team only had four days to finish the preparations. Dedicating time to proper user tests is essential in creating a user-centered product. This is particularly important in an application such as Bro, where user experience is at the core of the application. Thus, the team decided it was an easily justifiable decision to delay sprint 3 in order to ensure resourceful user tests. Thus, sprint 3 was set back to start at week 11, not week 10 as originally suggested.

Table 3.3: Project plan



3.3.2 Risk-Analysis

Table 3.4 describes the various risks involved in the development process of Bro.

Risk-Analysis Reflections

During sprint 5 the team had a very reduced capacity due to the availability of the team members. All team members had major deliveries in other courses and spent on average about half as much time on IT2901 compared to earlier sprints according to the time table (Appendix **??**). Fortunately, the team had planned ahead and put in a lot of work prior to sprint 5 and the finished product was therefore not significantly impacted by the reduced capacity.

A risk the team did not plan for was the risk of using a relatively new technology (Flutter) which did not have a lot of documentation. As none of the team members had used the technology before, the team encountered a few problems they had not prepared for. Learning how to use Flutter properly took a bit longer than the team expected and the first sprint achieved less than planned. The team also had to put in considerable work in sprint 3 to apply the BLoC architecture which they did not initially realize they should have implemented. If the team had planned ahead and implemented risk management measures they might have avoided having to retrofit BLoC and could perhaps have had a better estimation of the time required to learn Flutter. For example, a planning session for finding best practices and architectures for Flutter at the start of the project could have saved a lot of time in the long run.

Table 3.4: Risk factors

ID	Description of risk	Reason for consequence estimate	Consequence	Likelihood	Reason for estimate of likelihood	Risklevel	Measures for risk management	Consequence after measures	Likelihood after measures	Remaining risk
R1	Team member drops out	If a team member drops out, the team risks losing valuable knowledge and the workload for the remaining members will be increased.	Very high	Low	The team members already know each other and work well together. They assume they can catch potential problems before they arise.	Moderate	The team will use the group contract as well as good cooperation and communication to make sure they support each other if problems arise. As they might not be able to deal with unexpected personal problems they will try to share information and save progress often to reduce potential consequences.	high	low	moderate
R2	Illness	If a team member gets sick the team risks temporarily losing valuable knowledge and the workload for the remaining members will be temporarily increased.	Very high	Moderate	Short-term illness is very likely and there is a low chance of more serious long lasting illness. The team therefore decided to put plausibility in the middle at moderate.	High	By sharing knowledge continually the team will avoid relying to heavily on individual team members with a lot of specific knowledge on a part of the project. That way, they will have an easier time continuing the work even if the member with main responsibility is unavailable.	Moderate	Moderate	Moderate
R3	Availability of team members	The team members have a lot of time consuming courses this semester, this in addition to other responsibilities may reduce availability. If they do not plan their work well enough the project could suffer from it.	High	High	Throughout the semester personal and academic situations could arise, which would collide with the project.	High	To avoid this the team will strive to keep the work on the project well planned and structured. This requires good communication, regular status updates, and getting control over any situations that arise quickly.	High	Moderate	Moderate
R4	Too high expectations from product owner	Unrealistic expectations of the end result could lead to increasing the scope of the project past the capacity of the team which could cause low quality code and unfinished functionality if the team feels pushed to achieve this scope.	High	Low	This is not a professional setting where the team is paid for the work, and the product owner expects the team to be inexperienced. The product owner also knows that the team has other courses they need to work on.	Moderate	The team will communicate well with the product owner and explain why it is not achievable for the team, and try to find a manageable solution if conflict arises.	Moderate	Low	Low
R5	The team underestimates task requirements	If this were to happen the team's backlog would probably grow out of control, and it would quickly become a struggle to finish the project.	High	Moderate	This is likely as this is the first project of its size the team has attempted.	Moderate	The team will be using scrum to mitigate this risk by carefully planning out task estimates using tools such as planning poker and sprint reviews.	High	Low	Moderate
R6	Meetings get off track or are not taken seriously	If meetings are not used properly it could cause the team to fall behind on deliveries.	Moderate	High	The team is a close group of friends, which could create a too casual setting where team members lose focus and start talking about unrelated topics.	Moderate	Well-structured meetings and discussing expectations for the teamwork early should set a good standard for how meetings would be conducted.	Moderate	Moderate	Moderate
R7	Application does not pass code review	The application does not satisfy the requirements set by the iOS store and/or google play. Thus, the result is that we will be delayed in our release in addition to having fix the issues brought up by the review process	Very high	Moderate	The requirements seem simple enough, and require coding based on "best-practice", which is something which was within our scope anyway. Though, some of the requirements seem somewhat abstract, which means that we may down-priotizise something that is quite imporant for the reviewer	High	Throughout the coding process we will attempt to code in relation with Apple's coding requirements in mind. This is due to the fact that Apple's requirements are more stringent than Google's. Thus, we will most likely succeed in getting the applications approved on both iOS and Google play. If we still do fail, it will also reduce the size of needed changes.	High	Low	Moderate

3.4 Product Owner Cooperation

Any developmental process requires effective product owner communication. The product owner has to describe their needs and what they want out of a product or the developers will not have anything to develop.

3.4.1 Product Scope

During the initial planning phase after the team established communication with Selamewit Eng the team found out that there was high expectations for Bro. During the first few meetings with Selamewit Eng, the team experienced additional staff at Lier municipality wanted to get involved with the meetings. Since Selamewit Eng and the staff at Lier municipality had no experience with software development, the team noticed that the scope could quickly get out of hand.

After having a meeting with a legal expert in the refugee service, Selamewit Eng and one of the leaders in the refugee service, the team managed to somewhat reduce this problem. During this meeting it was agreed that features requiring GDPR compatibility would be scrapped in their entirety. This helped with removing a big part of the non-essential features from the project, like a messaging functionality and calendar integration. This decision also removed the need to invest precious development time in legal research.

The team also realized that other staff at the refugee service in Lier municipality had very different fields of expertise. They would therefore come with a lot of ideas for Bro. These ideas would often seem useful for the refugees and the staff, but also deviated greatly from the initial mission statement. After a few meetings the team and Selamewit Eng agreed that it was for the best to keep the meetings between fewer people. There was also an agreement to scrap many of the ideas other staff members came forward with and limit the project to the core functionalities. Namely the informational articles and the course functionality.

3.4.2 Sprint Planning

Communication with product owner is an essential part of sprint planning. As explained by Kniberg, sprint planning requires a careful balance between three variables; scope, importance and estimate. Scope and importance are set by the customer and decide what the developers do each sprint, while estimate is set by the team of developers [5].

During the first sprint the team struggled with getting the most out of Selamewit Eng when it came to organizing the sprint. This problem was present because the team had little experience with developing a product alongside a product owner, especially one that had no experience with software development. Team members found it difficult to ask the right questions and explain the scrum methodology, making the sprint planning difficult.

This problem was less prevalent during the second sprint, but was still a minor hindrance. For the sprint 2 planning session the team decided to take more time to explain the process and take more responsibility for moving the conversation in the meetings. This allowed the team and Selamewit Eng to get a clearer picture of how the next two weeks would unfold.

After a meeting with the project supervisor Serena Lee-Cultura, the team was told that the right approach was not too different from what the team attempted for sprint 2. Namely, that a team with a less experienced product owner should take more of a lead during the early stages of development and take time to clarify the process for the product owner. Then, during the later stages of development, when the product owner gains more of a understanding, Selamewit Eng will be able to take a bigger role in the meetings.

3.5 Project Diary

In this section the team provided short summaries of the various stages of development.

3.5.1 Planning Phase (25.01.2021 - 08.02.2021)

The project planning phase was the time period where the team first set themselves into the project and its requirements. During this phase the team got their initial impressions of Selamewit Eng and the course in general. With these impressions, the team was able to make their initial choices. Including how often to meet Selamewit Eng, which platforms to use for communication between team members, Selamewit Eng and Serena Lee-Cultura. During the second week the pre-study got more practical, at this time the team researched viable frameworks and technologies for creating Bro.

3.5.2 Sprint 1 (08.02.2021 - 22.02.2021)

Name: The Running Baboon
Goal: Complete initial setup and design

Completed Features:

- Figma design
- Set up development environment
- Physical architecture design

Challenges:

The sprint suffered from the team committing too much time into implementing features. When team members started development on features that were not already designed, there was a lot of uncertainty on how they should have been implemented. The team also underestimated the challenges that arose while using new technologies. Implementing new functionalities takes more time when a team member is still not used to the framework and the language it uses.

Successes:

The biggest success of sprint 1 was getting the design completed and approved by Selamewit Eng. This greatly increased the team members confidence in the direction the project moved in. Finishing the design also gave the team members more certainty when it came to the implementation of the functionalities they worked on.

Conclusion:

During the first week of sprint 1 the team was setting up the development environment and creating the first drafts for the design. At the same time the team also started started implementing features. In retrospect this was disastrous since team members were attempting to implement features where the design had not matured before even beginning to learn the technologies. The team understood this misstep in week 2 of the sprint and spent more time refining the design.

The main takeaway from this sprint was to ensure more direction in each sprint. The team agreed to trim down the user stories included in each sprint to a core of immediately needed issues, before potentially pulling more from the product backlog. One way the team thought this principle could have improved sprint 1 would be to dedicate the first week to design, in the form of a short design-sprint. In addition to this the team accepted that there was a need to spend more time learning about and using the Flutter framework.

3.5.3 Sprint 2 (22.02.2021 - 08.03.2021)

Name: The Bongo Stampede
Goal: Implement course system

Completed Features:

- Strapi content-types created
- Home page for app created
- Course cards implemented
- Swiping between course cards implemented

Challenges:

During sprint 2 the team experienced discomfort with pulling user stories from backlog into the sprint, fearing repeating the mistakes from sprint 1. This problem occurred when team members were waiting for dependant user stories to be resolved. This hesitancy cost the team development time that would be better spent working on new user stories. By being more proactive assigning user stories, the team could bypass this issue in its entirety.

Another problem the team encountered was the low frequency in commits. When other team members do not commit changes to the branches they are working on regularly it makes overseeing related work difficult. This also extended to the team being late with upstream pull requests. A lot of completed subtasks and user-stories were not shared before the end of the sprint.

Write about the user test

Successes:

Sprint 2 had several successes. The team managed to implement more user stories than last sprint and felt more comfortable using the new technologies. When it came to work methodology the team also felt that an increased use of pair programming lead to more comfortable and effective work sessions.

Conclusion:

Sprint 2 was far more successful than Sprint 1. Reducing the amount of user stories to implement and centering them around one main functionality lead to a sprint with more direction. This lead to new user stories being picked up from the backlog to fill the weekly 20 hour goal. The team members felt that there were positive changes in work methodology, compared to the previous sprint.

3.5.4 User Test Week (08.03.2021 - 15.03.2021

The team had planned user tests at the 11th of March and had the midterm delivery deadline at the 12th of march. Up until this week the team had produced a lot of documentation relevant to the report (meeting notes, sprint reflections, etc.) which was not yet formalized and included in the report. The team realized that completing the report and preparing the user tests would require a lot of time, and they would not have much time to work on the product. Since sprint 3 was planned to be one week longer than the rest of the sprints they decided to delay the sprint by one week. This way the team could put all their resources into the user tests and report.

3.5.5 Sprint 3 (15.03.2021- 05.04.2021)

Name: Jetting Yeti
Goal: Implement Navigation and finish course system

Completed Features:

- Navigation system with navigation bar
- Category system for articles
- Recommendations on front page
- Course list
- Question cards in courses

Challenges:

Sprint 3 underwent without any unexpected challenges. One factor that slowed down development was the challenges of implementing the BLoC pattern, this however was be accounted for when planning the sprint.

Successes:

The biggest success of sprint 3 was the implementation of the BLoC pattern. While architectural changes can be demanding to implement the team managed to pull through with both these changes and the sprint goal. This was achieved by reducing sprint scope during planning. This ensured that the team could implement the BLoC pattern properly and still finish all the user stories in the sprint.

Conclusion:

The biggest takeaway from sprint 3 was that it was a good move to reduce sprint scope when planning to make changes to the architecture. The desired changes in architecture as well as the desired functionality were successfully implemented.

3.5.6 Sprint 4 (05.04.2021- 19.04.2021)

Name: The Dodging Quagga
Goal: Implement Localization and Articles

Completed Features:

- Localized versions of articles and courses in CMS
- User language preference on articles and courses
- article view with styling
- Improved navigation system
- System for handling static files in articles

Challenges:

The team attempted to use a development branch during this sprint. This proved challenging as certain functionalities were dependant on each-other and the development branch allowed team members to get untested, lower quality code into the project. This created a mess with branches with great conflicts between themselves and the master, leading to significant delays.

Successes:

The fourth sprint was very successful, due to being placed at the end of the semester this sprint needed to contain all major functionality that was still missing from Bro. The team planned to have a content dense sprint and managed to successfully implement all planned functionalities.

Conclusion:

The team decided to stop using the development branch and agreed that it would have been wiser to stick with the previously successful workflow at this late point in the development process.

3.5.7 Sprint 5 (19.04.2021- 02.05.2021)

Name: The Storming Starfish
Goal: Bugfixing and deployment

Completed Features:

- localized text direction for right-to-left scripts
- Improved front page design
- Bugfixing and refactoring
- User manual for administrators

Challenges:

The deployment of Bro was delayed beyond this sprint due to factors in the refugee service. The staff members at Lier municipality that could acquire licences for uploading Bro to Google play and App store were not available during the sprint. The actual deployment of the application happened during the week after the fifth and final sprint.

Successes:

The bugfixing part of the fifth sprint went according to plan and the team ended the sprint with a minimum viable product ready for deployment. During the last sprint the team also started teaching the staff at Lier municipality how to use the Strapi back-end and wrote a user manual to go with it. Doing this early in the last sprint proved to be a good move, so the user manual could be improved upon before product handover.

Conclusion:

The fifth sprint resulted in a finished product that was ready for deployment. The actual deployment had to be delayed due to internal factors at the refugee service in Lier municipality. The deployment took place the week after the fifth sprint.

3.5.8 Deployment Week

The deployment week was characterized by setting up systems for product handoff. As the project complied in large part with the App-Store and Google Play guidelines, most of the coding done were tweaks adjusting for deployment. The deployment to Google Play was quite painless as most of the development was done on android. Deployment to App-store was more challenging, as deployment requires Apple products.

Both products were tested and verified to comply with platform-specific permissions. The Android version was uploaded for internal testing and instructions were given to Selamewit om deployment. As the Apple developer verification process was too sluggish, Bro was not uploaded to the App-Store. That being said, the application has been tested and compiled, and is awaiting deployment upon Apple developer confirmation.

3.6 Process Summary

The process has been sculpted by the time spent on planning. By basing the workflow on the Scrum methodology, the team was able to quickly adjust to changes while still retaining a structure. Updates regarding individual and common progress was shared on a near daily basis due to scrum-meetings and collaborative working hours. These factors made the implementation of changes and adjustments easy by ensuring the team members always had a common understanding of the status in the development process.

Each sprint presented the team with different challenges and learning experiences. During the first sprint the team had some growing pains getting used to the workflow with Selamwit Eng. The team and Eng were both inexperienced in our respective roles, which made it initially difficult to communicate. With time came experience and between sprint 2 and 3, the team had established efficient routines for working with Eng. Sprint 2 went slow and steady. The internal working routines took root and the team members became comfortable in their roles. The user test week was an action packed period were the team got an unique experience in having a user test through an interpreter. In Sprint 3, everything process related went smoothly, but implementing BLoC pattern proved to be a technical challenge, although a learning experience. In sprint 4 the team made a mistake in changing the successful workflow. A development branch was created, which turned out to have several more cons than pros, and was therefore removed in the following sprint. Sprint 5 was characterized by a refactor of the code base to implement language selection across all areas of Bro, tying up loose ends.

The foundation built during the pre-study allowed the team to work closely together when required and provide creative individuality when desired. The daily stand-ups and collaborative working hours were critical in creating a common goal and keeping everyone updated. The individual working hours gave the flexibility to produce results, even when the team had different schedules. By using the established team dynamic to create a solid working environment, the team could quickly get to work. While this eagerness may have gone overboard during early development, the willingness to grab a hold of these issues made sure that the group would grow from these missteps. The communication with Selamewit Eng emboldened the team to be creative and implement unique solutions. Taking all of this into account, one would not be wrong to say that the product is a direct result of the cooperation within the team and the collaboration with Selamewit Eng.

Acknowledgments

The team would like to thank Serena Lee Cultura for invaluable insight and expertise. They would also like to thank Selamewit Eng for her passion regarding Bro and her perseverance during an unfamiliar process. Without their contributions this would have been far more challenging, and the project would not have been such a success.

Appendix A

Team contract

Team contract for LIER01

Communication

- Say what you mean, it is allowed to disagree.
- Don’t mock others’ contributions before a discussion has been held.
- There should be an agenda before every meeting, as well as a record of the meeting afterwards.
- Long discussions should be held outside of meetings (follow the meeting agenda).
- Make sure to notify your absence, or other tardiness in good time.

Work Ethics

- All changes to the master branch shall only be done through pull requests.
- Follow the guidelines for good code on the github wiki before you merge.

Process

- Keep Jira updated.
- Anyone can change code, but major changes are decided as a team.
- Use our tools well.
- Focus on completing the goal of the sprint.

Attitudes

- Be prepared and take meetings/work seriously
    - Don’t do other work during meetings
- Inform the team if you encounter problems
    - Be willing to aid team members who get stuck.
- Prioritize work that is most critical to the most important project goals and those tasks that display more knowledge in the subject.

Expectations

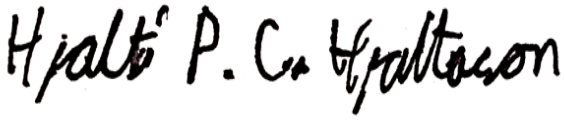
- The team expects that each member uses at least 20 hours a week on this course divided between various activities including meetings, common work hours and independent work sessions.
- The team will strive for a high grade in the subject, preferably an A.

Decisions

- All major decisions should be discussed thoroughly and be formalized by a unanimous vote. Major decisions include, major stack changes, adding or removing features, and changes to this contract.
- All minor decisions should also be discussed, but will be decided by a vote where the greater party wins if necessary.

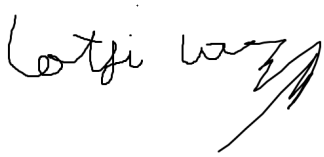
Encouragement

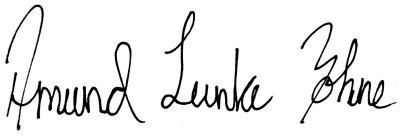
- If a team member fails to do their tasks or aggregate a total of 1 hour of tardiness during a week. They will be penalized with making “encouragement cake” for the team. The cake will be served during working hours on sundays.



Patrick Moen Allport

Hjalti Percy Casimis Hjaltason





Lotfi Amin Lazreg

Amund Lunke Røhne

Vegard Rognstad Smines

Appendix B

Consent Form

Brukertest for appen Bro

Dette er en henvendelse om å delta i en brukertest hvor formålet er å få tilbakemelding på et design av appen Bro. Videre vil du få informasjon om hva formålet med produktet er og hva din deltakelse vil innebære.

Formål med produktet
Formål med Bro vil være å være et hjelpemiddel som flyktningtjenesten i Lier vil ta i bruk i introduksjonsprogrammet for flyktninger.

Appen vil bestå av to hoveddeler, en del med artikler om forskjellige emner som for eksempel helsesystemer, økonomi, og barn og miljø. Her vil du kunne lese nyttig informasjon og se dokumenter som flyktningetjenesten legger ut som vedlegg. Du vil også finne linker til nyttige sider som kan gi en dypere forståelse om temaet.

Den andre delen vil bestå av en interaktiv læringsplattform, hvor du vil kunne lære mer om emner på et quiz-format. Hvor du først får informasjon, og deretter noen oppfølgingsspørsmål for å hjelpe deg å repetere kunnskapen.

Hvem er vi?
Vi er en gruppe på fem studenter som går bachelor i informatikk ved NTNU. I samarbeid med flyktningtjenesten i Lier kommune har vi i oppgave å utvikle appen Bro.

Hvorfor blir du bedt om å delta?
Du er en av noen få som har blitt bedt om å delta. Dette fordi du følger introduksjonsprogrammet i Lier kommune og kan gi verdifulle tilbakemeldinger.

Hva vil deltakelsen innebære for deg?

- Taleopptak
- Kort intervju
- Tilbakemeldingsskjema

Hvis du velger å delta i denne brukertesten, vil dette involvere at du forklarer hvordan du hadde løst en rekke oppgaver via en design prototype vi har utviklet. I tillegg vil vi ha et kort intervju hvor vi du får stilt spørsmål. Til slutt har vi et tilbakemeldingsskjema som omhandler hvordan du syns designet var å bruke.

Hvordan vil din personlige informasjonen bli brukt?
Vi vil bruke det korte intervju for å få litt bakgrunn om din erfaring med lignende produkter. Dette vil bli brukt sammen med taleopptak for å finne svakheter ved vår applikasjon og dermed utvikle den videre. Utførelsen av testen og informasjonen vi får henter ut av den vil bli dokumentert i en rapport som blir skrevet i forbindelse med gruppens bacheloroppgave. All informasjon som kan knytte deg som deltager til testresultatene vil bli fjernet og totalt anonymisert.

Deltakelse er frivillig

Appendix C

Observer Form

Person som testes: Person1
Person som observerer: Vegard, Hjalti

Tid	Problem	Årsak	Forslag til løsning
14:14	Forsøkte å trykke på hjemmeknappen i navbaren når vi er på hjemme		
14:20	Gikk til kurs under testcase der vi ønsket å nå artikkel om utdanning		
14:25	Deltager spør hva knappene under bildet er på kurs	Oppløsning eller ukjent med konvensjoner? Vi som må vurdere konvensjonene vi bruker?	

Forvirrende:
Likte:
Ikke likte: Vil ha tall på svar alternativene.

Frie observasjoner: Møtte problem med at skjermstørrelse/oppløsning gjorde at deltagerne ikke kunne se tekst og knapper.
Liten viktighet om grid eller listeformat

Person som testes: Person2
Person som observerer: Hjalti, Vegard

Tid	Problem	Årsak	Forslag til løsning
14:40	Trykket på kurs istedenfor artikkel når testcase var å finne artikkel	Uklar funksjonalitet? Samme betydning på arabisk?	
14:43	Ville først trykke se oversikt (mens han var i helsetjenester),,		Bytte tekst til artikkeloversikt?

	men fant ut at pilen skulle trykkes på		
14:48	Testsubjekt ville trykke på det som er synlig på siden. Han tror ikke det går an å scrolle.		Scrollbar på siden? Ha en pil som viser man kan scrolle ned?Tredje testsubjekt foreslo dette.

Forvirrende:
Likte: : Person sa appen ser bra ut, veldig profesjonell.
Ikke likte:

Person som testes: Person3
Person som observerer: Hjalti, Vegard

Tid	Problem	Årsak	Forslag til løsning
14:45	Ser ikke forskjell mellom kurs og artikler på hjemmesiden		

Forvirrende: Linjene rundt boksene på quiz cards var utydelige.
Likte: :
Ikke likte: Person3 lurer på hvordan bytte språk. Veldig viktig mener hun, inkludert for navigasjonsikoner.(skal komme først) Person3 synes budsjett og økonomi er veldig likt. Krever litt tålmodighet, informasjon nærmere bildet hadde vært ideelt. Altså, bør ikke trenge å trykke på bildet for å se. Trykk for å lese mer bør være på begynnelsen av siden, ikke nederst. Ikke sikkert hun gidder å lese hele veien ned.

Frie observasjoner: FIkk gjennomført test med mye mindre leding enn første testsubjekt. Tredje testsubjekt følte det var vanskeligere å se hva som var naturlig uten å ha en mobil i hånda.

Har dere brukt lignende før? Person3 har ikke brukt lignende før. Hun sier det er veldig bra og at det ikke finne noe som dette fra før.

Person som testes: Person4
Person som observerer: Hjalti, Vegard

Person4 hadde ingen breakdowns. Testsubjektet var veldig erfaren med teknologi, som er trolig årsaken.

Likte: Person4 sa appen ser bra ut, veldig profesjonell.
Forvirrende:
Ikke likte:

Konlusjon fra utviklere:
Generell reduksjon av innhold. Flytkningene må bevisst lese innhold, så vær sparsom med tekst
Spesielt hovedsiden

Appendix D

System Usability Score

Table D.1: Usability test results

Participant	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	SUS Score
p1	3	4	4	4	4	3	3	2	3	3	52.5
p2	3	2	3	3	4	4	3	3	4	2	57.5
p3	2	3	3	3	3	2	2	2	3	4	47.5
p4	2	3	4	1	4	2	2	4	4	2	60.0
Average	2.5	3.5	4	2.5	4	2.5	2.5	3	3.5	2.5	56.3

Questions:

1. Jeg tror jeg vil bruke denne appen ofte
2. Jeg syns appen var unødvendig komplisert
3. Jeg syns appen var lett å bruke
4. Jeg tror en teknisk person må hjelpe meg å bruke appen
5. Jeg syns at de forskjellige funksjonene i appen fungert bra sammen
6. Jeg syns det var for mye som var inkonsekvent i appen
7. Jeg tror at de fleste vil lære seg å bruke appen fort
8. Jeg syns appen var tungvint å bruke
9. Jeg var veldig selvsikker med å bruke appen
10. Jeg måtte lære mye for å bruke appen

Appendix E

CMS User Manual

Brukermanual for administrasjonspanel

Innlogging

For å legge til innhold i Bro bruker man et nettbasert administrasjonspanel som kan finnes på denne adressen. Bruk kontoen du har fått tilgang til for å logge inn her.

Generelle tips

- **Media filer:**

  Når bilder skal brukes i applikasjonen er det viktig at de er av god kvalitet og i høy oppløsning. Det finnes mange gode og gratis kilder for slike bilder. Under utviklingen av Bro ble Pexels brukt mye, og kan anbefales.

- **Opplasting av bilder/filer:**

  Bilder og filer kan lastes opp på to måter. Enten ved skapelsen av nytt innhold hvor det trengs, eller gjennom **Media Library** menyen, som kan finnes under **Plugins** på venstre side av skjermen. Eneste du trenger å gjøre er å trykke på **Upload assets** knappen, og velge filene fra din lokale datamaskin.

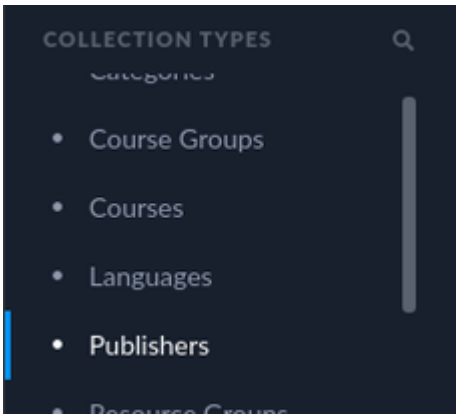
- **Strapi**

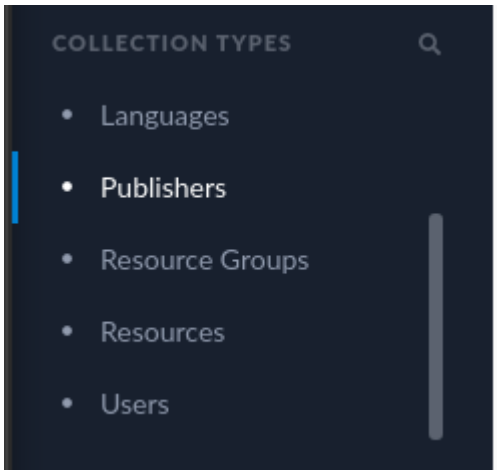
  Administrasjonspanelet er laget med en teknologi som heter Strapi. Brukermanualen nedenfor beskriver hvordan vår implementasjon av Strapi fungerer. Om det er noe utover dette som er uklart, kan man referere til Strapi Documentation. NB, denne er veldig teknisk og kan virke forvirrende.

- **Hjelp! Jeg finner ikke kurset/ressursen min i Bro appen!**

  Dette er mest sannsynlig på grunn av manglende relasjonelle felter i kurset/ressursen. Gå inn på det gjeldende innholdet og kontroller at feltene er korrekte (de som finnes på høyre side). Om dette ikke løser problemet, er det som regel lurest å gå igjennom forklaringen nedenfor steg for steg, og finne feilen.

- **Hjelp! Jeg finner ikke riktig fane under “Collection-types”!**
  Området under “Collection-Types” er en lengre liste, som en kan bla seg opp og ned på. Dersom en ikke finner den fanen en leter etter, så må en kanskje bla opp eller ned i denne lista.





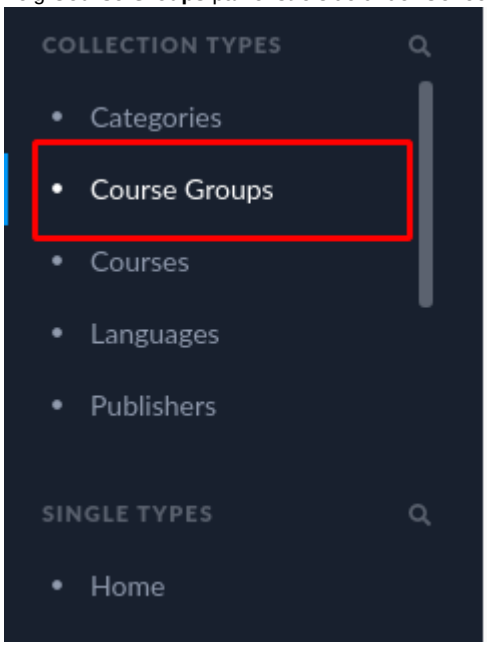
Kurs

Denne seksjonen forklarer hvordan man legger til nye kurs samt oversatte versjoner av eksisterende kurs.

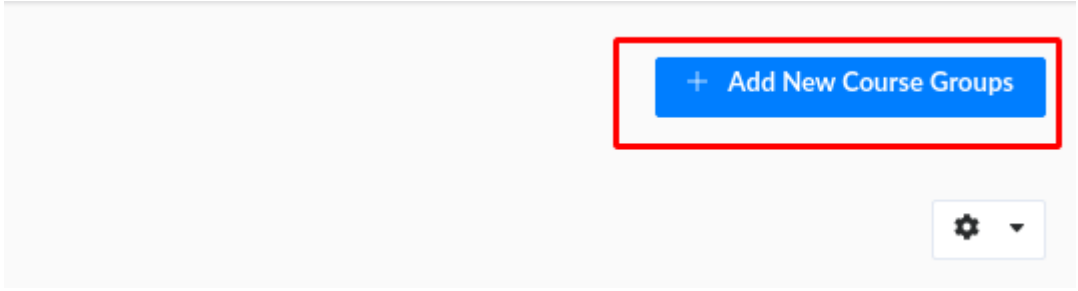
Lage et nytt kurs

Her kan du lese om hvordan man legger til et helt nytt kurs. Ønsker du å legge til en oversettelse, se seksjon for “Legge til et nytt oversatt kurs“.

1. Velg **Course Groups** på venstre side under **Collection-Types.**

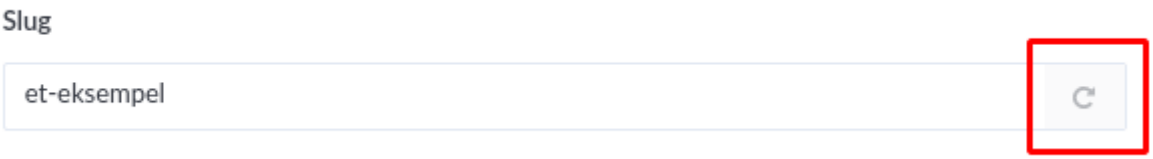


2. Trykk på **Add New Course Groups** øverst til høyre på siden.



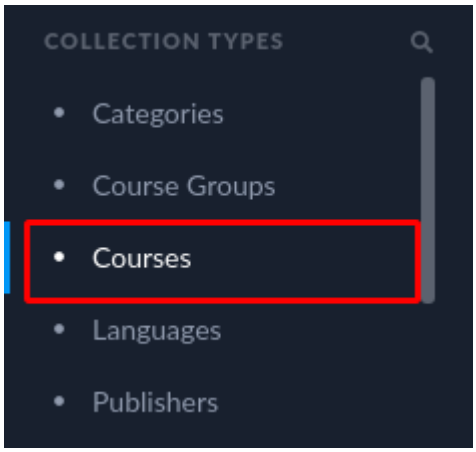
3. Skriv inn et navn på den nye kurs gruppen under **Name** feltet, feks den norske tittelen på kurset. **Slug** feltet skal fylle seg ut automatisk basert på **Name** feltet. Husk verdien av **Name** feltet, da dette vil bli brukt igjen senere.

4. Dersom **Slug**-feltet ikke fyller seg ut automatisk, så kan en trykke på **Regenerate**-knappen til høyre for **Slug**-feltet. Dette vil da fylle ut **Slug**-feltet

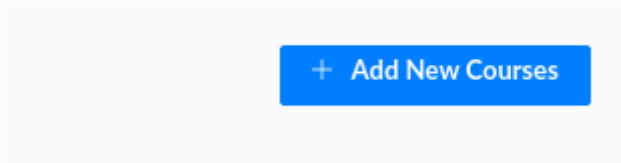


5. Trykk på **Save** knappen, øverst til høyre. Deretter trykk på **Publish**.

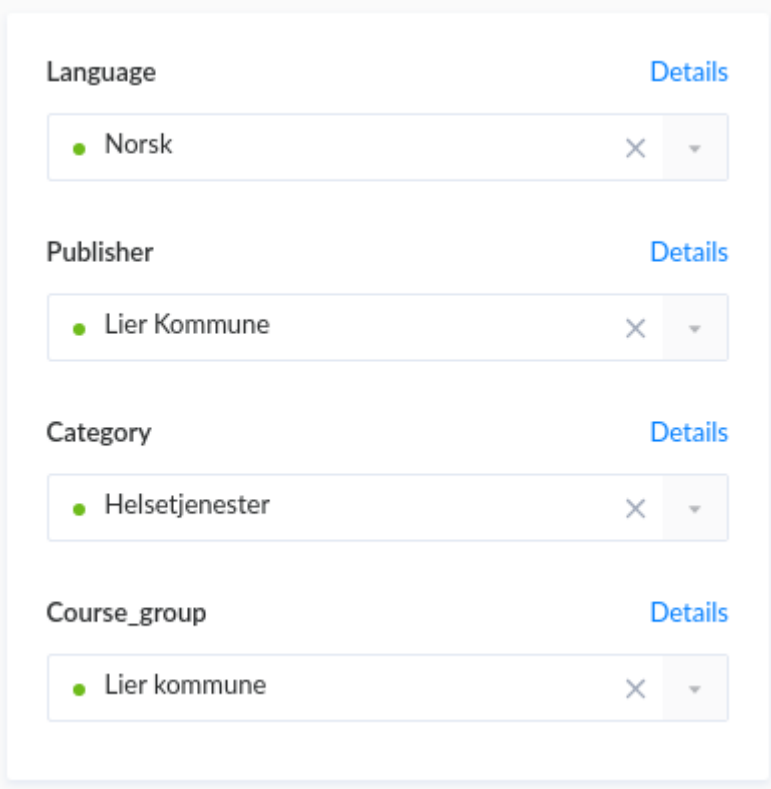
6. Velg **Courses** på venstre side under **Collection-Types.**



7. Trykk på **Add new Courses** øverst til høyre på siden.



8. Fyll inn korrekt informasjon i feltene, forklaring finnes i tabellen under. Tabellen heter “ Forklaring av felter i Kurs”
9. Velg riktige relasjoner i relasjonsfeltene på høyre side. Dersom en eller flere av disse ikke er publisert eller lagt inn, så vil kurset ikke vises i applikasjonen:
    a. **Language** bestemmer språket på dette kurset (skal stå som Norsk hvis dette er første kurset i gruppen).
    b. **Publisher** bestemmer utgiver på dette kurset.
    c. **Category** bestemmer kategori på dette kurset.
    d. **Course_group** bestemmer hvilken gruppe dette kurset tilhører, her skal gruppen som ble laget tidligere bli valgt.



10. Trykk **Save** øverst til høyre på siden.
11. VIKTIG! Når du er klar for å gi ut kurset i Bro, trykk på **Publish**, øverst til høyre på siden. Før dette vil kurset ikke vises i appen.

Forklaring av felter i Kurs (Course):

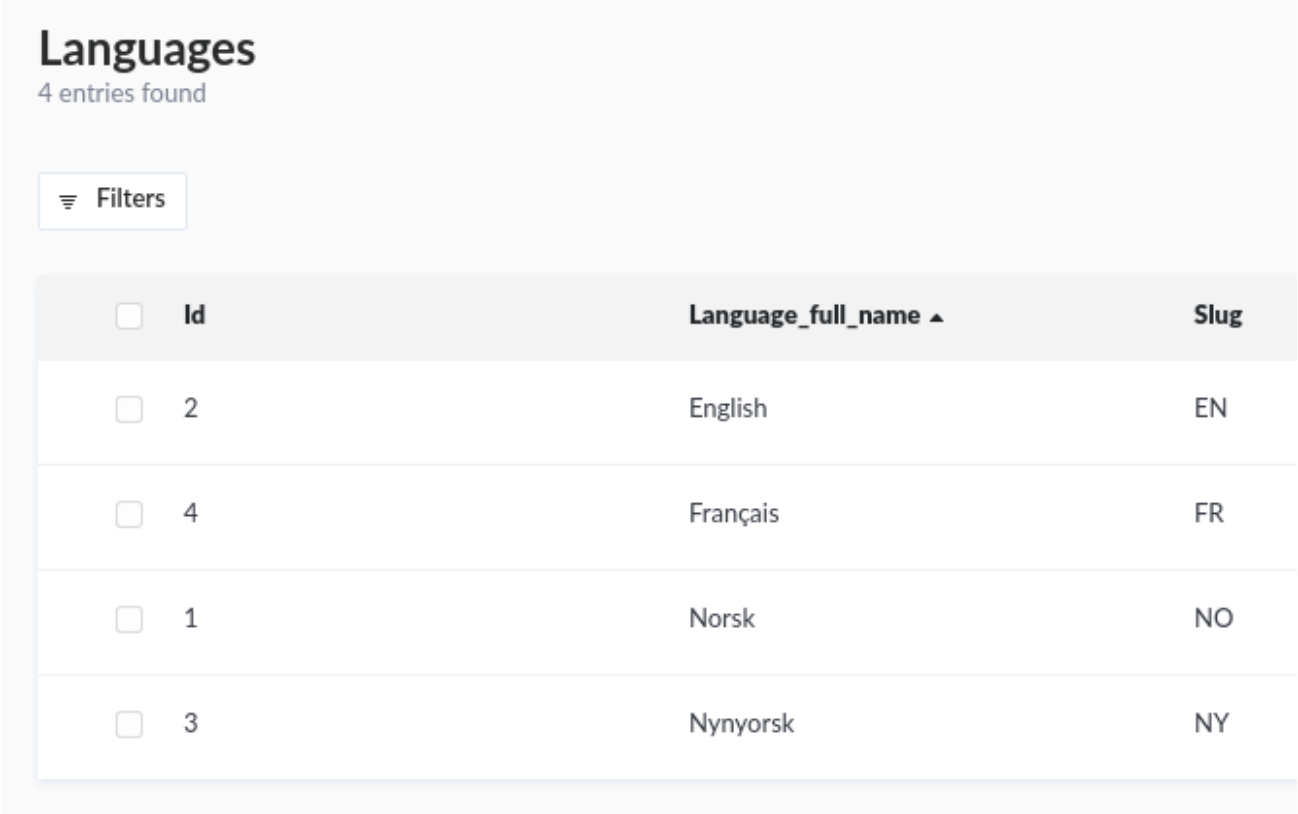
Feltnavn	Beskrivelse
Title	Kursets tittel
Description	Kort, forklarende tekst om kurset.
Slides	Gruppering av informasjonskort.
Slides -> Title	Tittel på gjeldende informasjonskort.
Slides -> Description	Lengre tekst med det gjeldende informasjonskortet sitt innhold.
Slides -> Media	Bilde for gjeldende informasjonskortet.
Questions	Gruppering av kurs spørsmål.
Questions -> Question	Spørsmålteksten for gjeldende spørsmål.
Questions -> Alternatives	Gruppering av alternativer for gjeldende spørsmål.
Questions -> Alternatives -> Alternative_text	Svartekst for gjeldende alternativ
Questions -> Alternatives -> Image	Bilde for gjeldende alternativ. **Dette feltet er ikke påkrevd.**
Questions -> Alternatives -> Is_correct	Skal være “ON” hvis alternativet er korrekt, “OFF” hvis alternativet er feil.
Questions -> Clarification	Utdypende og forklarende tekst for gjeldende spørsmål.
Is_recommended	Skal være “ON” hvis kurset bør vises på “hjem”-skjermen, “OFF” hvis ikke. **Merk**: Det kan maksimalt være 4 kurs som er satt som anbefalt samtidig.

Legge til et nytt oversatt kurs

Her kan du lese om hvordan man legger til en oversettelse av et eksisterende kurs. Før du fortsetter, godkjenn at det følgende stemmer.

- Språket du ønsker å bruke er lagt til i **Languages** under **Collection-Types.**

  Her kan vi eksempelvis se at vi har lagt til: “Engelsk”, “Norsk” “Nynorsk” og “Fransk”

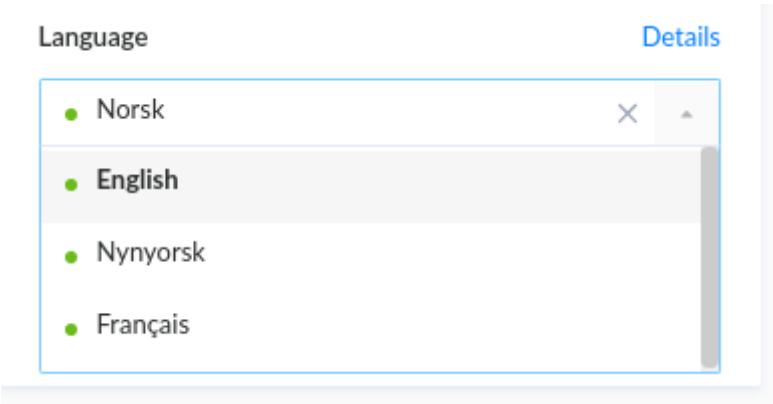


- En Norsk versjon av kurset eksisterer allerede i **Courses** under **Collection-Types.**

Slik går du deretter frem for å lage en oversatt versjon av kurset ditt.

1. Trykk på **Courses** under **Collection-Types.**
2. Finn kurset du ønsker å oversette i listen.
3. Trykk på “papir” ikonet helt til høyre på samme linje som kurset, for å duplisere kurset og begynne å redigere.

4. Endre det relasjonelle **Language** feltet på høyre side til det ønskede språket.



5. Oversett feltene i kurset. Referer til tabellen i seksjonen om å “Lage et nytt kurs”, om du trenger hjelp med dette.
6. Når du er ferdig med oversettelsen, trykk på **Save** øverst til høyre på siden.

7. VIKTIG! Når du er klar for å gi ut oversettelsen av kurset i Bro, trykk på **Publish**, øverst til høyre på siden. Før dette vil ikke oversettelsen vises i appen.

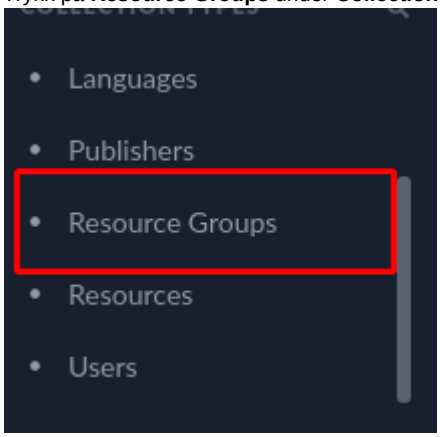
Ressurser

Denne seksjonen forklarer hvordan man legger til nye ressurser samt oversatte versjoner av eksisterende ressurser.

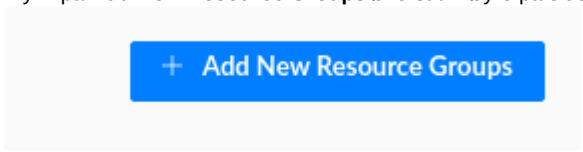
Lage en ny ressurs

Her kan du lese om hvordan man legger til en ny ressurs i Bro.

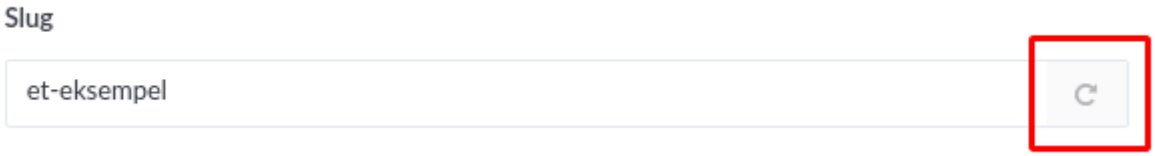
1. Trykk på **Resource Groups** under **Collection-Types** på venstre side av skjermen.



2. Trykk på **Add New Resource Groups** øverst til høyre på siden.

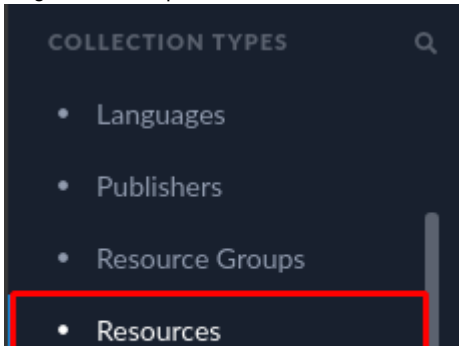


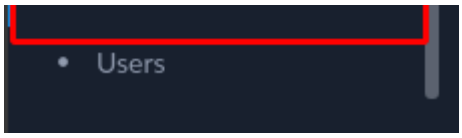
3. Skriv inn et navn på den nye ressurs gruppen under **Name** feltet, feks den norske tittelen på ressursen. **Slug** feltet skal fylle seg ut automatisk basert på **Name** feltet. Husk verdien av **Name** feltet, da dette vil bli brukt igjen senere.
4. Dersom **Slug**-feltet ikke fyller seg ut automatisk, så kan en trykke på **Regenerate**-knappen til høyre for **Slug**-feltet. Dette vil da fylle ut **Slug**-feltet



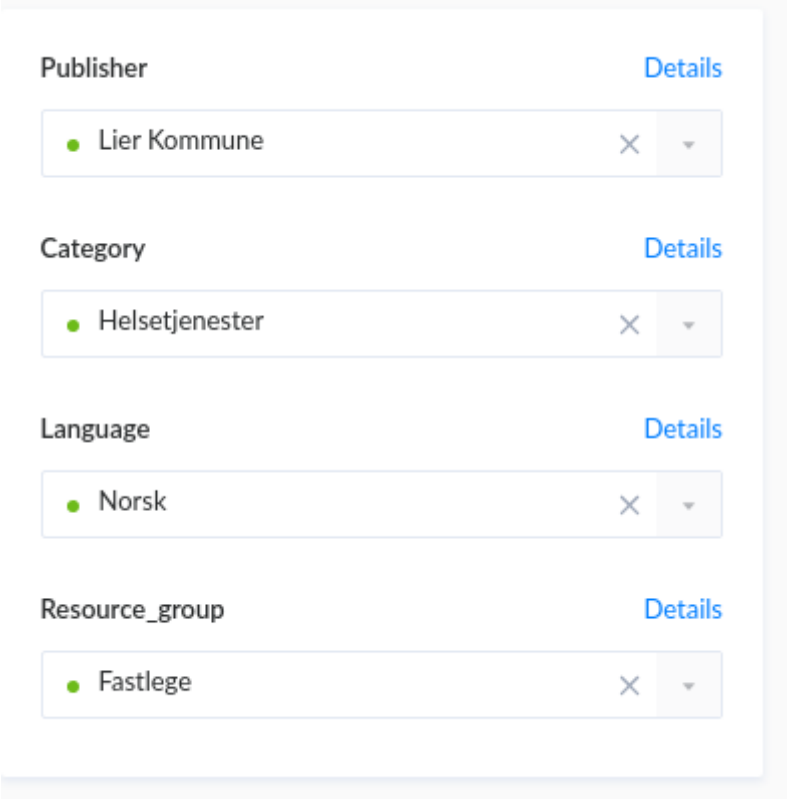
5. Trykk på **Save** knappen, øverst til høyre. Deretter trykk på **Publish**.

6. Velg **Resources** på venstre side under **Collection-Types.**





7. Trykk på **Add new Resources** øverst til høyre på siden.
8. Fyll inn korrekt informasjon i feltene, forklaring finnes i tabellen under. Tabellen heter " Forklaring av felter i Ressurser"
9. Velg riktige relasjoner i relasjonsfeltene på høyre side. Dersom en eller flere av disse ikke er publisert eller lagt inn, så vil kurset ikke vises i applikasjonen:
    a. **Language** bestemmer språket på dette ressursen (skal stå som Norsk hvis dette er første ressursen i gruppen).
    b. **Publisher** bestemmer utgiver på dette ressursen.
    c. **Category** bestemmer kategori på dette ressursen.
    d. **Resource_group** bestemmer hvilken gruppe denne ressursen tilhører, her skal gruppen som ble laget tidligere bli valgt.



10. Trykk **Save** øverst til høyre på siden.
11. VIKTIG! Når du er klar for å gi ut ressursen i Bro, trykk på **Publish**, øverst til høyre på siden. Før dette vil ressursen ikke vises i appen.

Forklaring av felter i Ressurser (Resources):

Feltnavn	Beskrivelse
Title	Tittel på ressursen
Is_recommended	Skal være "ON" hvis ressursen bør vises på "hjem"-skjermen, "OFF" hvis ikke.
Description	Kort beskrivelse av ressursen.
Cover_photo	Bilde som vises på toppen av ressursen.
References	Gruppering av referanser til eksterne kilder.
References -> Reference_title	Tittel på gjeldende referanse.
References -> Reference_description	Utdypende forklaring om gjeldende referanse.
References -> Reference_button_text	Teksten som vil vises på knappen for gjeldende referanse.

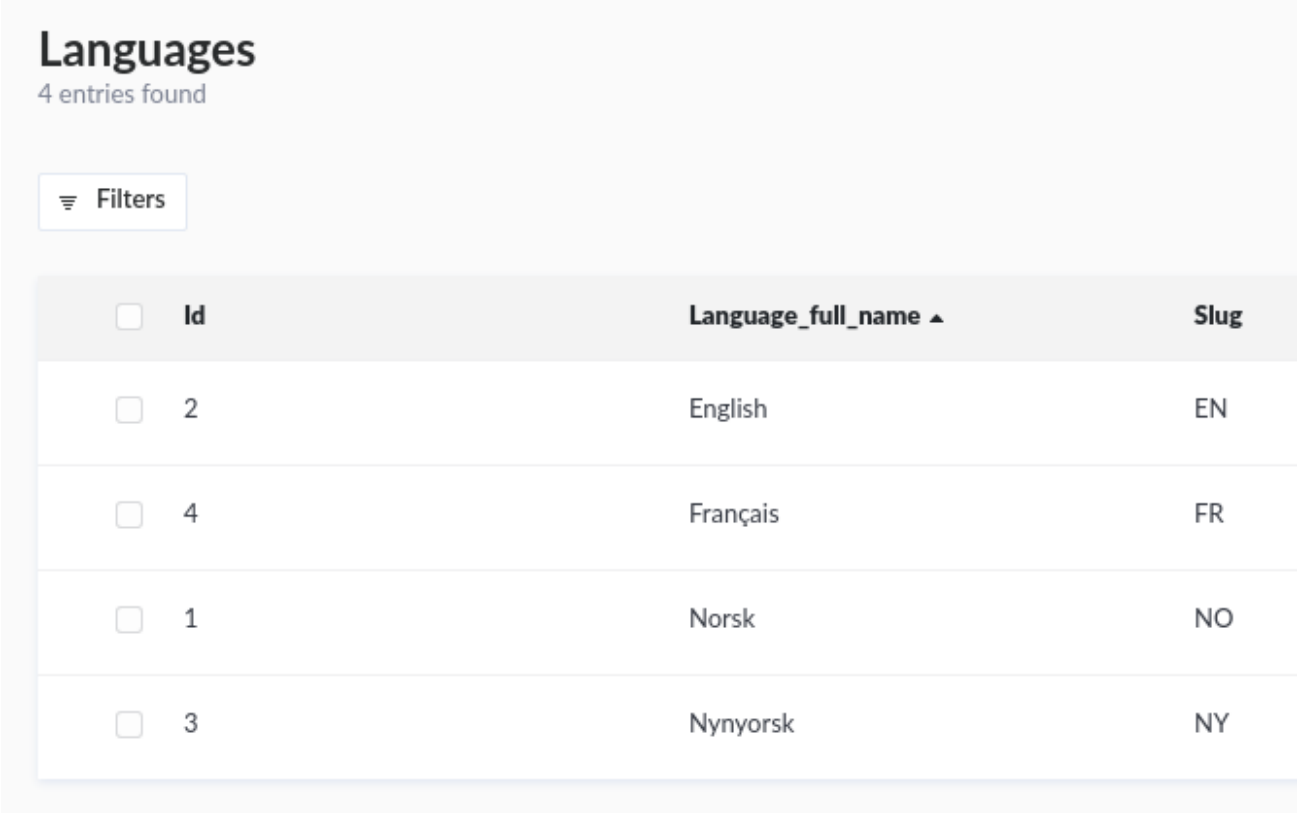
References -> Reference_url	Linken til referansen på formatet. “https://eksempel.no”
Documents	Gruppering av vedlagte .pdf dokumenter. Dette feltet er ikke påkrevd.
Documents -> Document_name	Navnet på gjeldende dokument som vil vises i Bro.
Documents -> Document_file	Dokumentfilen som skal legges ved for det gjeldende dokumentet.

Legge til en ny oversettelse av en eksisterende ressurs:

Her kan du lese om hvordan man legger til en oversettelse av en eksisterende ressurs. Før du fortsetter, bekfreft at det følgende stemmer:

- Språket du ønsker å bruke er lagt til i **Languages** under **Collection-Types.**

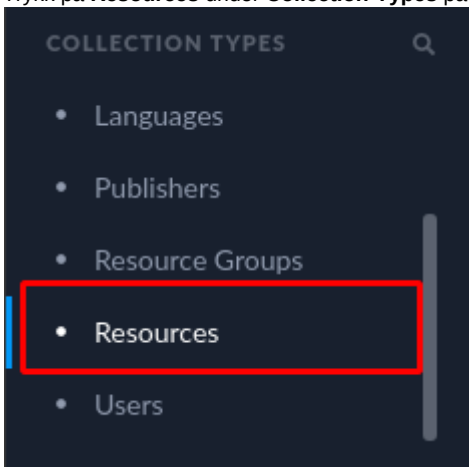
  Her kan vi eksempelvis se at vi har lagt til: “Engelsk”, “Norsk” “Nynorsk” og “Fransk”



- En Norsk versjon av ressursen eksisterer allerede i **Resources** under **Collection-Types**.

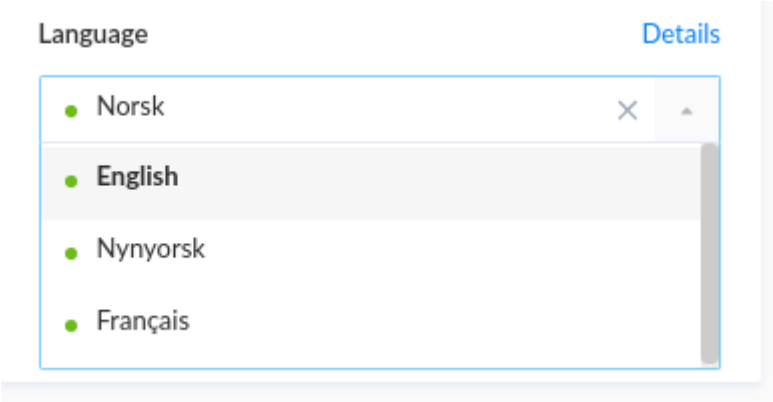
Slik går du deretter frem for å lage en lokalisert/oversatt versjon av ressursen din.

1. Trykk på **Resources** under **Collection-Types** på venstre side av skjermen.



2. Finn ressursen du ønsker å oversette i listen.
3. Trykk på “papir” ikonet helt til høyre på samme linje som ressursen, for å duplisere ressurser og begynne å redigere.

4. Endre det relasjonelle **Language** feltet på høyre side til det ønskede språket.

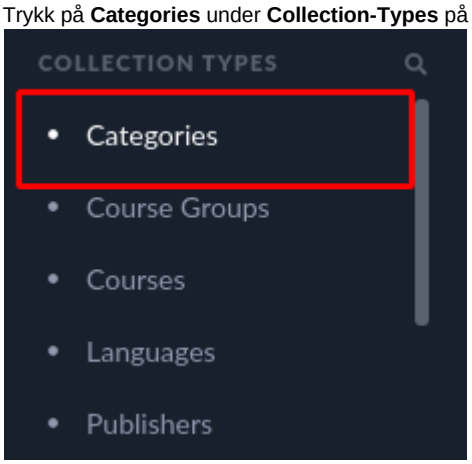


5. Oversett feltene. Referer til tabellen i seksjonen om å “Lage en ny ressurs”, om du trenger hjelp med dette.
6. Når du er ferdig med oversettelsen, trykk på **Save** øverst til høyre på siden.
7. VIKTIG! Når du er klar for å gi ut oversettelsen av ressursen i Bro, trykk på **Publish**, øverst til høyre på siden. Før dette vil ikke oversettelsen vises i appen.

Legge til en ny kategori

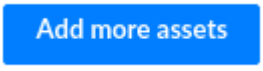
Her kan du lese om hvordan man legger til en ny kategori.

1. Trykk på **Categories** under **Collection-Types** på venstre side av skjermen.



2. Trykk på **Add New Categories** øverst til høyre på siden.

3. Fyll inn feltet **Category_name** med kategorinavnet.
4. Velg et bilde i **Cover_photo** feltet.
   a. En kan velge et bilde som allerede er lagt inn, eller legge til et nytt et ved å trykke på **Add more assets**-knappen.



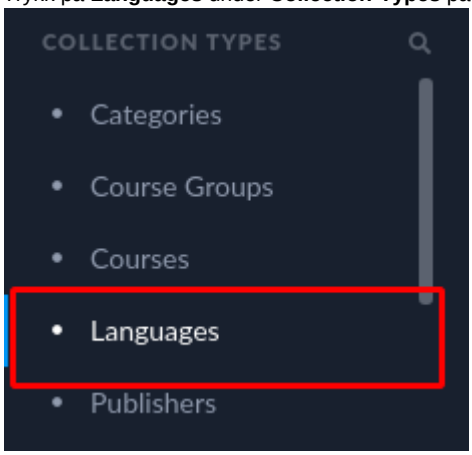
5. Fyll inn feltet **Description** med en kort beskrivelse om kategorien. Denne beskrivelsen er beskrivelsen brukeren ser på kategorioversikten.
6. Når du er ferdig med å fylle inn feltene, trykk på **Save** øverst til høyre på siden.

7. VIKTIG! Når du er klar for å gi ut kategorien i Bro, trykk på **Publish**, øverst til høyre på siden. Før dette vil ikke kategorien vises i appen.

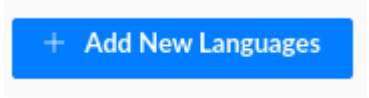
Legge til et språk

Her kan du lese om hvordan man legger til et nytt språk i Bro.

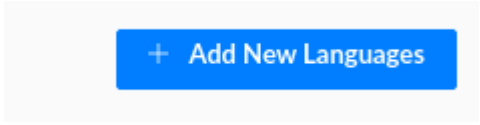
1. Trykk på **Languages** under **Collection-Types** på venstre side av skjermen.

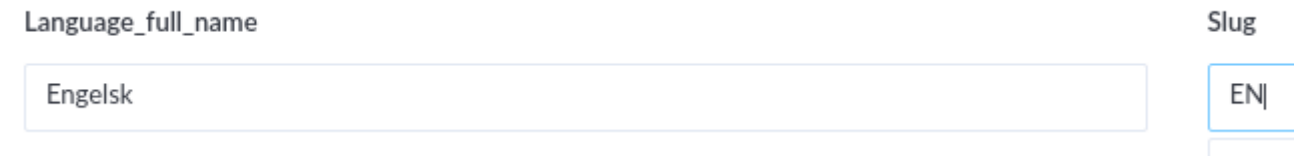


2. Trykk på **Add New Languages** øverst til høyre på siden.



3. Fyll inn feltet **Language_full_name**, med den korrekte oversettelsen av navnet (altså ikke på Norsk, Engelsk blir “English”). **Slug** feltet skal fylle seg ut automatisk basert på **Name** feltet. Men bør ikke overskride 2 bokstaver, fjern bokstaver om den gjør det.



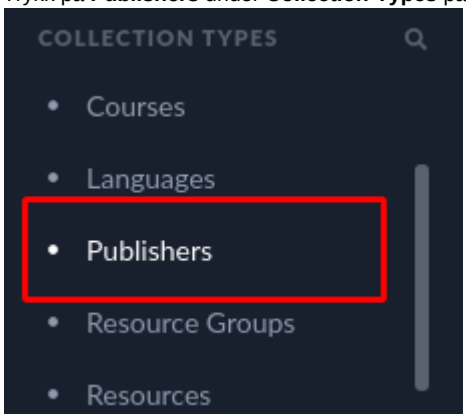


4. Trykk **Save** øverst til høyre på siden, og deretter på Publish for å kunne bruke språket i andre **Collection-Types**.

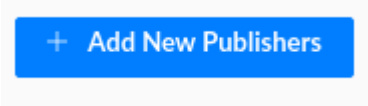
Legge til en utgiver

Her kan du lese om hvordan man legger til et nytt språk i Bro.

1. Trykk på **Publishers** under **Collection-Types** på venstre side av skjermen.



2. Trykk på **Add New Publishers** øverst til høyre på siden.

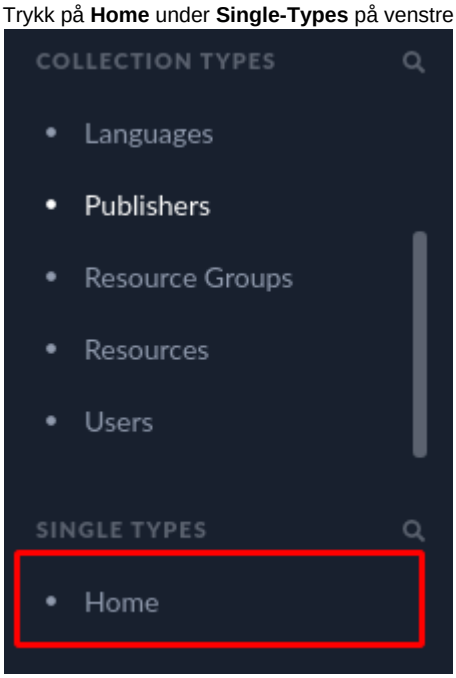


3. Fyll inn feltet **Name** med navnet på utgiveren (kommunen).
4. Trykk på **Avatar** feltet og legg inn bilde for utgiveren (kommune skjold/lignende).
    a. Dersom en ønsker å legge til nye bilder, så kan en trykke på **Add more assets**-knappen
5. Trykk **Save** øverst til høyre på siden, og deretter på Publish for å kunne bruke utgiveren i andre **Collection-Types**.

Endre innholdet på “hjem-skjermen”

Her kan du lese om hvordan man endrer innholdet på “hjem”-skjermen.

1. Trykk på **Home** under **Single-Types** på venstre side av skjermen.



2. **Header** feltet bestemmer hva som står i overskriften på “hjem”-skjermen i Bro.
3. **Introduction** feltet bestemmer hva som står i “beskrivelsen” på “hjem”-skjermen.
4. Etter du har gjort de ønskede endringene kan du trykke på **Save** øverst til høyre på siden.

Appendix F

Code review guidelines

- Perform manual functionality test.
- Does the code have unit tests?
- Is the code tested in end-to-end testing?
- Does the code complete all acceptance criteria?
- Is the code properly documented?
- Does the code comply with our non-functional requirements?
- Check that file names follow dart naming conventions.

Appendix G

Abbreviations

ASR	=	Architecturally significant requirements
BLoC	=	Business Logic Component
CD	=	Continuous deployment
CI	=	Continuous integration
CI/CD	=	Continuous integration and continuous deployment
CMS	=	Content management system
IT	=	Information technology
MVC	=	Model-view-controller
SUS	=	System Usability Scale
TTS	=	Text-to-speech
UI	=	User interface
UX	=	User experience

Table G.1: Abbreviations

Bibliography

[1] Agile Alliance. *User Story Template*. URL: https://www.agilealliance.org/glossary/user-story-template/. (accessed: 10.03.2021).

[2] Dart Community. *Effective Dart: Style*. URL: https://dart.dev/guides/language/effective-dart/style. (accessed: 03.05.2021).

[3] The Bloc Community. *Bloc Architecture*. URL: https://bloclibrary.dev/#/architecture. (accessed: 11.03.2021).

[4] eligherm. *Enrollment process time*. URL: https://developer.apple.com/forums/thread/31202. (accessed: 06.05.2021).

[5] Henrik Kniberg. *Scrum and xp from the trenches - 2nd edition*. lulu.com, 2015.

[6] Philippe Kruchten. "Architectural Blueprints — The "4+1" View Model of Software Architecture". In: *IEEE Software* 12 (1995).

[7] Donald A. Norman. *The Design of Everyday Things*. Basic Books, 2002.

[8] Dan Radigan. *Story points and estimation*. URL: https://www.atlassian.com/agile/project-management/estimation. (accessed: 10.03.2021).

[9] Chris Sells. *What's New in Flutter 2*. URL: https://medium.com/flutter/whats-new-in-flutter-2-0-fe8e95ecc65#0757. (accessed: 05.05.2021).

[10] Usability.gov. *System Usability Scale*. URL: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html. (accessed: 06.05.2021).