INF-2900: Software Engineering

Anna Eline De Vito,
Emilie Steen,
Marie Alette Stenhaug,
Sera Madeleine Elstad,
Skjalg Alexander Slubowski

May 15, 2024

Contents

1 Introduction

1.1 The Idea and vision

KnitHub was conceived by Sera, a knitter and developer, who often lost track of her knitting projects and yarn supplies. This situation highlighted a common issue in the knitting community - organizing and managing knitting projects and materials effectively. The concept behind KnitHub was to develop a web-based platform that would not only streamline the knitting process but also provide a comprehensive system for managing projects, patterns, and supplies. The platform's key features - a resource page, yarn and needle inventory systems, a pattern page, and a project tracking page - are all a part of this vision. By simplifying the practical aspects of knitting, KnitHub makes it easier for users to access and discuss their patterns, manage their inventories, and oversee their projects, all in one place. With KnitHub, the vision of creating a unified and simplified knitting experience has become a reality.

1.2 Description of the project and goals

The primary goal of KnitHub is to improve the management of knitting activities for avid knitters. We saw an opportunity to create a place for knitters, such as ourselves to keep all necessary information about all our projects and inventory.

KnitHub integrates multiple features designed to optimize the knitting experience:

1. **Pattern Organization**: KnitHub allows users to upload, organize, and annotate your digital knitting patterns in one central location. This feature simplifies tracking of modifications and personal touches to each pattern.
2. **Inventory Management**: Say goodbye to manual searches through your yarn and needles. KnitHub's inventory system keeps you informed about what you have, what you need for upcoming projects, and what you are using in you ongoing projects.
3. **Project Tracking**: KnitHub's project dashboard allows users to simply monitor their knitting progress. The platform displays finished, ongoing, and upcoming projects, allowing for improved planning and management. Each project includes information about the project, calculators, the recipe, and counters.
4. **Resource Page**: KnitHub also has a resource page where you can search up a variety of abbreviations, use the calculators, and see some Instagram photographs to find ideas for your next project.

Additional features for admin users include:

1. **User Management**: Admins can view all user profiles, manage admin rights, ban or unban users, and sort user lists by name or search by email for easy navigation
2. **Communication Handling**: Admins have access to manage and respond to user inquiries through the 'Contact Us' form, including active, pending, and completed interactions.
3. **Newsletter Subscription Overview**:Admins can view and manage the list of newsletter subscribers, enhancing engagement and community interactions.
4. **Other**: Admins can also see how many yarn cards that are uploaded to the page, and how many needles of the 4 predefined types that are uploaded.

1.3 The Group

The team consists of Eline, Emilie, Marie, Sera, and Skjalg, all with different knitting experience and development skills. Throughout the project development, all team members have actively engaged in both front-end and back-end development.

Sera came up with the idea for KnitHub, she defined the key components and functionalities that form the basis of KnitHub's services. Marie had previous experience with HTML, CSS, and web development, therefore she took on the major responsibility for writing the CSS code.

Skjalg wanted to work on the backend, so he took the responsibility for it. He focused on developing functionalities that would support and enhance KnitHub's features. Eline was in charge of connecting the front and back-end components, providing seamless integration and improving the user experience. Emilie was important in ensuring that all little aspects of the project worked seamlessly together. She also oversaw the file structure, ensuring that everything remained clear and organized throughout the development process.

2 Technical Background

2.1 API

An Application Programming Interface (API) allows for communication between different software components. By defining a set of methods and data structures, an API allows developers to interact with a specific component or service without needing to understand its internal workings. In the context of web applications, a Web API employs HTTP protocols to enable interaction between the internet-connected services [1].

KnitHub supports both the Instagram Graph API and the MailerSend API [2]. The Instagram Graph API is integrated with OAuth 2.0 authentication and uses the me/media endpoint to fetch and show a grid of Instagram posts, increasing user engagement with dynamic content. It handles data in JSON format, which allows for quick parsing and management of media content. It also includes a MailerSend API that allows for sending emails. Mailersend is a cloud-based service, allowing users to manage email operations through an API.

2.2 Stack

2.2.1 Frontend

React is a JavaScript library for building user interfaces. It use a virtual DOM [3] to optimize rendering and improve performance, making it efficient for dynamic data updates. The architecture is component-based which allows developers to build reusable UI components that manage their state, leading to organized and manageable code [4].

Material-UI (MUI) is a React component package that includes ready-to-use components to assist speed up development. It provides a diverse set of components that can be altered including buttons, cards, dialog boxes, and more [5].

2.2.2 Backend

.NET is a software development framework developed by Microsoft, offering a comprehensive platform for building various types of applications, including web, desktop, and mobile. It provides a rich set of packages and tools that facilitate efficient development, deployment, and management of applications across different platforms and devices. Among the different tools provided are **Entity framework**. Entity Framework simplifies database interactions by allowing .NET to treat the database as a .NET object. This eliminates the need for writing of direct queries to the database, which simplifies the development process. In addition **dependency injection** (DI) is a core element in .NET. It is a design pattern used to enhance modularity and maintainability. DI archives this by injecting the dependencies required by a class from an external class, instead of the class creating them individually. This allows for flexible configuration of components and can help make an application more scalable.[6]

3 The Development Process

3.1 Methodologies - Agile development

3.1.1 Scrum

Scrum is an agile project management style that focuses on collaboration, accountability, and incremental progress toward a specific goal. We adopted Scrum to maintain agility in developing our product, KnitHub. This approach involves dividing the work into short periods called sprints, during which the development team focuses on completing specific tasks to contribute effectively to the project's goals. The process of these sprints and other Scrum activities is illustrated in the Scrum Cycle, see figure 1.

- **Product:** KnitHub which is the software product we developed.
- **Product Owner:** A team member responsible for defining product features and overseeing the completion of work.
- **Product Backlog:** A list of tasks such as bugs, features, and improvements. Our detailed backlog can be found in Appendix F.
- **Development Team:** A little group that organizes themselves. Our group size is five, which falls within the standard range for Scrum teams.
- **Sprint:** A three-week period during which we developed new features for KnitHub.
- **Scrum Meetings:** Regular meetings (held two to three times a week) where we discuss daily goals and review progress.
- **Scrum Master:** A coach who teaches the team how to use Scrum efficiently. The Scrum Master helps to facilitate meetings and address any difficulties that arise.
- **Velocity:** A measure of how much work we can complete throughout each sprint. This assisted us in planning and assigning tasks based on our capabilities.

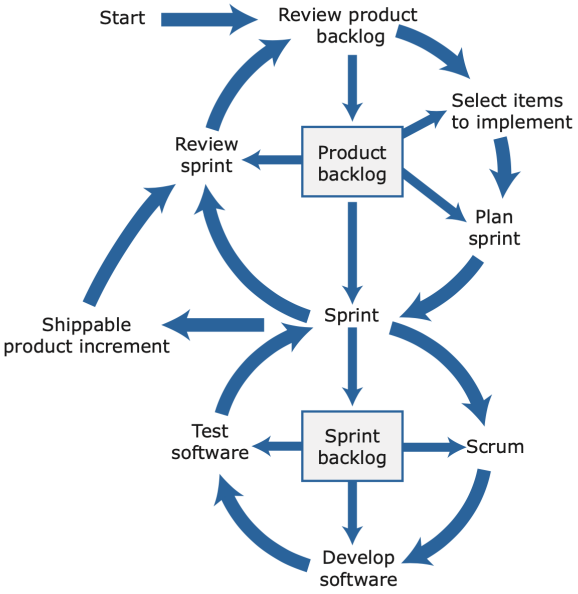

Figure 1: Scrum Cycle [7, p. 40]

We nominated Sera as the Product Owner at the start of the project because she came up with the idea for KnitHub. As a knitter, she understood what features would be most useful, which aligned with the Product Owner's responsibility of defining product features and overseeing their development. Even though we did not assign a Scrum Master, each sprint different team members took on the leadership role, hosting meetings and assisting with problem solving.

Initially, we had no idea what the ideal sprint length would be. After consulting with our TA Alexander, we settled on three-week sprints, giving us a total of six sprints for the project. For each sprint, we choose to focus on one big feature, which was made up of multiple smaller elements. The backlog F provides more information about each sprint. Account management is an example of a sprint; it includes log in and sign-up capabilities, as well as additional features required for users to establish and log in to their accounts.

We modified the usual stand-up Scrum meetings to match the limits of our course, naming them "Team Weekly". Daily sessions were impractical given the course's value of 10 credits. Instead, we choose to meet 1-2 times each week, in addition to a weekly session with the Teaching Assistant. At the end of each sprint, the last meeting included a round of constructive feedback from all team members, focusing on improvements for the next sprint.

3.1.2 Pair programming

Pair programming is a collaborative practice where two developers work together on each code unit. It was initiated with the belief that working in pairs enables mutual learning and helps catch each other's mistakes more efficiently [7, p. 37]. We chose to practise this methodology, particularly in the initial stages of the project, as it was a lot to dive into. Our belief was that pairing up might enhance productivity and lead to more effective time management. Which is why one can see in our backlog that more then one team-member is assigned to different tasks.

3.2 Test-Driven Development

According to the book "test-driven development (TDD) is an approach to program development that is based on the general idea that you should write an executable test or tests for code that you are writing before you write the code" [7, p. 291]. For our project the test-driven development was mostly used by the back-end team. This is because creating tests for the frontend before it is actually implemented is challenging, since the frontend tests often involves rendering buttons, checking if specific text appears on the screen, etc.

During the backend development process, the team focused on establishing unit tests that tested the desired functionality of individual components before implementing them. This method gave developers a clear understanding of what the group intended each component to do. Before expanding the program, the team needed to write down and talk about what they intended the backend to do and how it should function. Once this was established, we could proceed

to implement tests and the actual code. Defining a precise behavior was difficult at first because we had never done it before and were unsure of how everything should operate. As we changed our strategy and expected outcomes during the development process, numerous tests had to be redone.

The benefits of adopting TDD to create our application are that the code is more reliable because almost everything is tested, and it provided the backend team with better structured development phases because they understood what they were working with. However, because this was all new to us, some tests had to be redone, and the team had to think abstractly about problems before implementing code.

3.3 DevOps

When developing KnitHub we used both Continuous Integration (CI) and Continuous Deployment (CD) approach, essential in maintaining a steady and efficient workflow. We used GitHub for source code management, which it makes collaboration and code sharing among team members easy, see section 3.6 for more details. We also used Scrum and pair-programming, which worked well alongside our DevOps processes. This improved the team productivity and project management, since it allowed us to divide the development work into separate sprints, making it easier to implement changes and improvements quickly based on feedback from earlier sprints. More details of how we did this can be found throughout this report.

3.4 Notion

Notion is a platform where users can create project boards, databases, and share notes and documents with other invited users. It offers tools for task management, project tracking, to-do lists, and much more. Throughout our project, Notion has played a crucial role in writing meeting notes from various meetings, setting up the backlog, brainstorming ideas, and tracking the status of tasks. At the beginning of the project, a mood-board was created to visualize the different features envisioned for the website, allowing all team members to contribute images, color palettes, and other ideas.

At the sprint planning meeting, features that will be worked on for that sprint will be added and given points. To do this we use a backlog that is divided into 5 stages (rows) and it is also divided into sprints (columns). How the backlog is structured, and more details about it can be read in section 3.5. Notion has also been used to track meetings. Figure 2 shows a part of the meeting-notes list on notion; the full log and meeting notes can be found in G. Each of these contains information about what type of meeting it was (TA, status/new weekly, or sprint), who was present, and notes for that specific meeting.

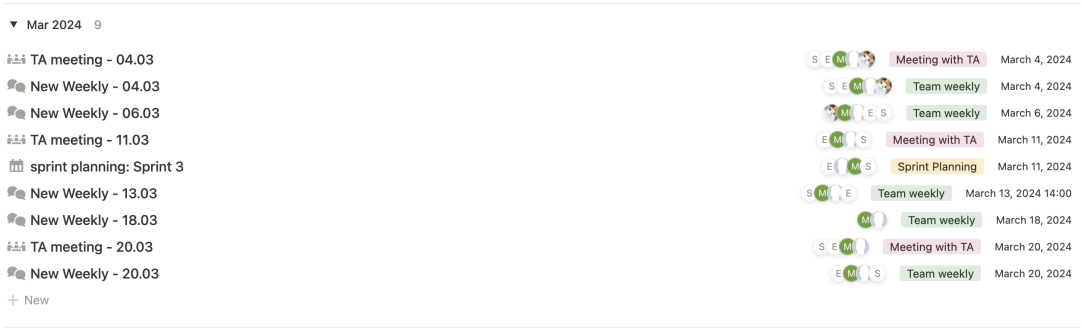

Figure 2: A partial list of meeting notes

3.5 User-stories and Backlog planning

To be able to create user stories and a backlog we had to write some personas (appendix B) and scenarios (appendix D). They laid the foundation for forming the user-stories (appendix C).

Before being categorized and given a priority, each user story was entered into the user story map in Miro. The MoSCoW approach, which divides our user stories into must-, should-, could-, and will-not-have categories, was used to prioritize the tasks. By doing this, we were able to determine what features were necessary for the software to function. These features was the first to be added to the backlog. To manage the backlog of the project, Notion was

used. "Not started," "In progress," "Ready for review," "Ready for merge," and "Done" are the five categories used to group the work in the backlog.

Each sprint planning meeting we used planning poker to add tasks covering up to approximately 30 points. All the tasks was set to "Not started", and every team-member was assigned to tasks, so all tasks would be covered. During the sprints, when a team-member starts on a specific feature, the status of the task was be changed in the backlog. When they believed they were done with the task, the status is set to ready for review, and then someone else from the team looked through the code, and if everything looked good, it was moved to ready for merge to main. When the code and functionality is tested by another team-member with success, the status is changes to done. The complete backlog for the project can be found in the appendix F.

3.6 GitHub

To enable team members to independently develop features without interfering with each other code, GitHub has been used. In the beginning a branch was created for each team member. However, this proved not to be a great solution. Sometimes a team member would work on multiple features at the same time, or multiple members would work on the same feature. Therefore a different solution was chosen. Instead of creating a branch for each team member, a branch was created when someone began implementing a new feature. The branch would be given an appropriate name, corresponding to the feature to be implemented.

In addition, Git has been used for source control. Once a feature was working, the feature would be committed and pushed to the branch. If someone had made an error, breaking the the functionality of the application, it would be easy to revert to a previous state where the application worked. Once the code is tested and everything looked good, the code from the branch would be merge into the **main** branch. If the feature was completely finished, the branch was deleted. This was done to have a clean repository, with few branches at the same time.

3.7 Time and effort of team members

Each team member has worked more or less the same amount throughout the course. As there are five different schedules, each team member's workload varies from week to week and sprint to sprint. However, when looking at the entire project, we have shared the workload equally.

4 Design

4.1 Languages

Before the coding of the project could start, the team had to decide which technologies we wanted to use. Many different languages were discussed, but the team landed on React as the frontend and .NET as the backend framework, with a SQLite database.

4.1.1 .NET

We chose to use the .NET framework because we saw it as an opportunity to learn a framework widely used in different workplaces. None of the group members had any prior experience with .NET, but several members had received summer job offers where they would be working with .NET among other technologies. A big reason for choosing .NET, was the **Entity Framework** provided by .NET. Entity Framework simplifies database interactions by allowing .NET to treat the database as a .NET object. This eliminates the need for writing of direct queries to the database, which simplifies the development process.

4.1.2 React

Everyone in the group had some experience with coding in React, and therefore the choice of this language was natural. With React, we could easily reuse our components across different files, making it easier to organize our code and files. When looking at our code, one can view that we reuse many components, such as input fields, buttons and containers. Additionally, React offers several libraries that we have benefited from during our development, which will be explained further in the implementation section of this report.

4.2 Architecture

Knithub follows a Model-View-Controller design pattern as shown in figure 3. To display data to a user, React is used to handle the view. React also has a part in the controller section, by sending HTTP requests to the .NET backend. The .NET backend has the main responsibility as controller. It handles incoming requests and responds to them appropriately. This is either in the form of an error code and message or with output data the view can display to the user. To handle the request, the controller has to call upon the model. In our application, this is in the form of calling services. These services process the requests and handle database operations.

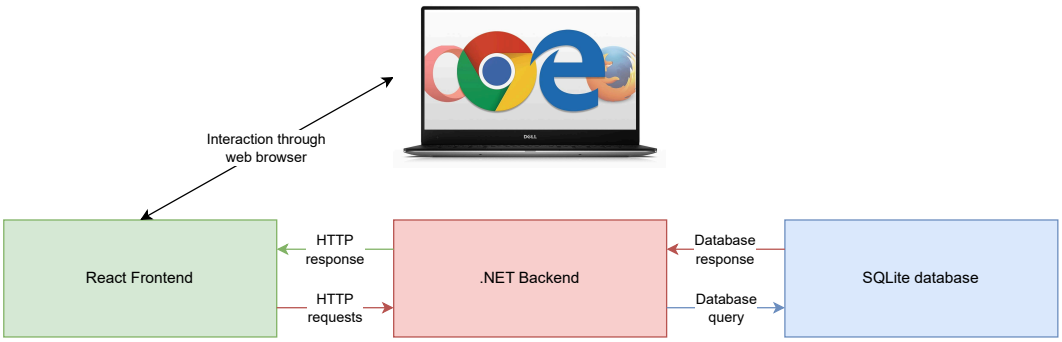

Figure 3: Flowchart of how our application handles user interaction

4.2.1 Front

KnitHub's front-end is intended to create a user-friendly and visually appealing user interface (UI). This section of the application handles all user interactions, including graphics, buttons, and text forms, using React's component-based architecture. This strategy divides the UI into components, or building blocks, with each performing a specialized purpose, simplifying updates and maintenance. Components are arranged hierarchically, with some elements nested within others, to keep the program organized and secure. For example, the "Navbar" contains links viewable only to logged-in users or administrators, such as the "Profile" or "Admin Page," which protect sensitive areas of the program.

The central element of this architecture is the App component, which manages the routing system for the entire application, as illustrated in figure 4. It utilizes the Routes and Route components from react-router-dom to establish the navigational paths users take to access different pages. Essentially, the App component functions as the central coordinator that directs user navigation to various parts of the application based on the URL path. The NavBar and footer components primarily function as the navigational interfaces, offering links and buttons that facilitate user movement throughout the application. The NavBar adjusts its display based on user status, showing different navigation options for logged-in users and distinguishing between admin and regular user roles.

States and hooks in React are used to make the app more responsive. States allow the program to dynamically maintain and adapt the user interface based on user inputs. Hooks allow functional components to manage state and side effects without having to convert into class components.

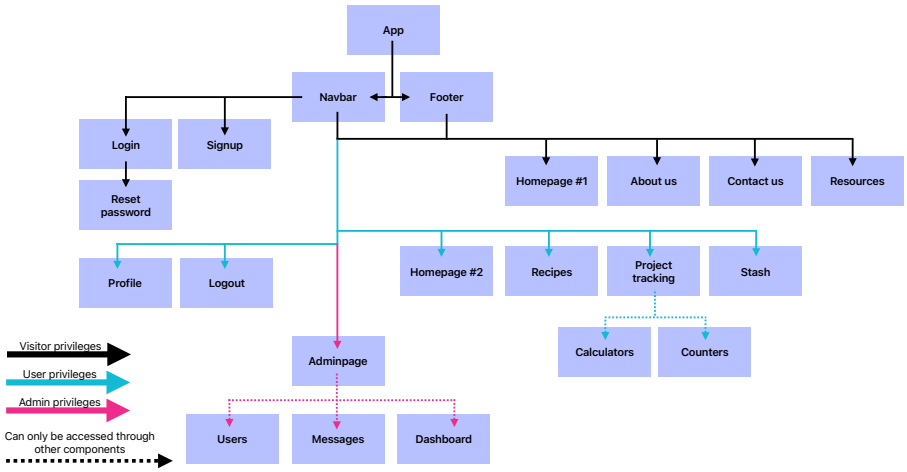

Figure 4: Component architecture, front-end

4.2.2 Back

Backend structure
As mentioned earlier, our backend is divided into three main components: the SQLite database, the controllers, and the services. Additionally, Data Transfer Objects (DTOs) are used to send information between these components.

Data Transfer Objects are used to send wanted and necessary data between the layers in the backend. They can be used to filter information sent between the different layers.

The controllers define what API endpoints exist, and can be called upon. They validate incoming requests, ensuring they contain the necessary data for processing. Once a request is validated, the controller invokes the necessary services to manage data manipulation and retrieval. In addition, the controllers are responsible for responding to the client or frontend calling to the endpoint. The response should be in the form of valid data, a message of successful operation, or an error message.

The services manage the logic and processing of data received from the controller. They have the primary responsibility of authentication and security within the backend. In addition, they are responsible for handling database interaction, including data retrieval, manipulation, and error management.

Lastly, the database is responsible for storing data in a logical manner.

Database structure
The database has been set up to use the **UserId** as the main key throughout all tables. The UserId is defined and used as a primary key in the *UserLogIn* table. All other tables except *Newsletter* reference this key as shown in figure 5. This allows the linking of different elements to a specific user, helping the retrieval of elements for a specific user. It also allows the deletion behavior of a table to be set to cascade, ensuring all data belonging to a user, if the entry for the user is removed from *UserLogIn*. The *Newsletter* table is standalone, as it does not need to reference a user, due to its independent logic.

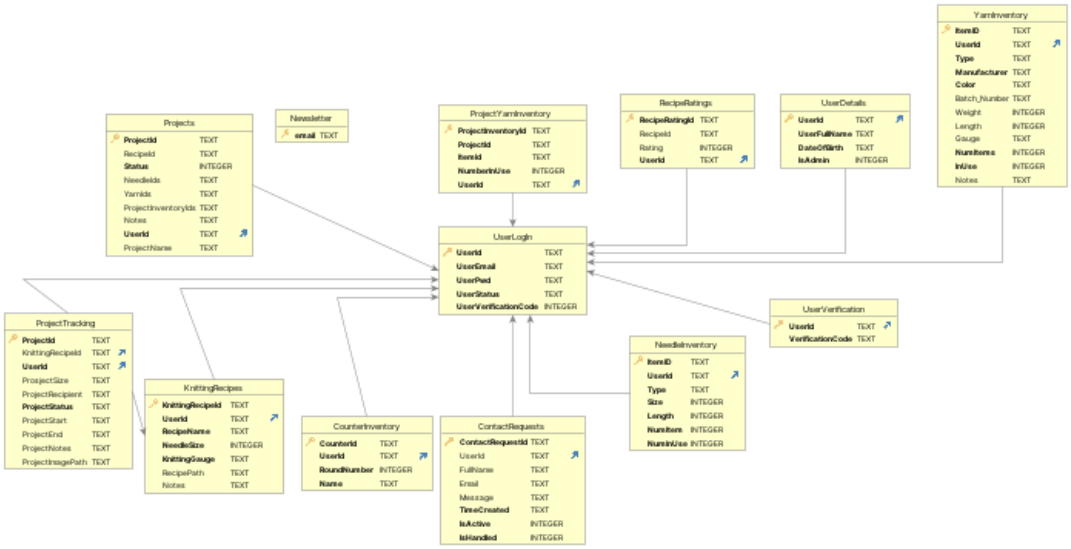

Figure 5: Database schema visualized.
The keys represent the primary key of the table, and blue arrows represent a foreign key

4.3 Non-functional attributes

4.3.1 Security and Privacy

KnitHub incorporates security considerations into both the frontend and backend components, providing a robust and secure user experience. This strategy enhances the entire security architecture.

JWT-token
For session management, KnitHub uses JSON Web Tokens (JWT). When a user logs in or registers, a token with encoded session data is generated from the backend and is then saved in session-storage in the frontend. This token is critical for authentication and is used to communicate with the backend, once a user is logged in. It is also used to grant a user admin access and allow the user to access and view the admin page. This token is cleared from the session storage, when a user logs out. The integrity of tokens is confirmed using the 'jwt-decode' library. This process occurs during the first load and whenever authentication verification is required. If token decoding fails, the application logs the issue and resets the admin state.

Since JWT-tokens cannot be redacted once generated, the tokens have an expiration time of 8 hours. This should be more time than a user would use in a single session. This also means that if a token is compromised, an attacker does not have access to the token for an extended period of time.

When the tokens are generated in the backend, they are signed using a secret key. The token can be accessed and read by anyone who can intercept it, but it cannot be altered. Altering the token without the secret key would render the token invalid, as the information contained would be corrupted and unreadable. The key used for signing the tokens is a 512-bit randomly generated key.

Given the potential limitations of client-side controls, our backend separately verifies user privileges for all critical actions and access requests. This means that even if an attacker gets past the frontend security measures, the backend verification acts as a crucial second line of defense, detecting and preventing unauthorized access and modification.

React Router
React Router renders components based on the user's authentication state. This configuration ensures that sensitive routes, such as administrative actions, only are accessible if the decoded token confirms administrator privileges. This conditional rendering increases security by preventing unauthorized access.

Hashing
In order to securely store user passwords in the database, and mitigating password leaks even if the database is compromised, a hashing technique has been used. To do this, the Microsoft Identity package has been utilized. This package offers functionalities to hash passwords for users and to verify passwords against the hashed values stored in the

database. In addition this package handles the salting of passwords. This ensures that users with the same passwords have different hash values, thereby enhancing the applications security and mitigating rainbow table attacks.

4.3.2 Responsiveness

KnitHub makes use of Bootstrap and React Hook Form to provide a better user experience throughout the application. The combination of these two technologies improves the website's usability. AJAX, through Axios, improves interactivity by retrieving backend data asynchronously, minimizing reloads and increasing speed, ensuring that our platform remains updated and functional.

4.3.3 Reliability

KnitHub maintains reliability by handling errors effectively, validating inputs, and logging them. The implementation had been testing with both unit and integration tests to discover and resolve bugs prior to release.

4.3.4 Usability

KnitHub has been created to be user-friendly. We used Material-UI (MUI) libraries to create a practical and visually appealing design with reusable components that maintain a consistent look throughout our application. This not only enhances the visual appeal, but it also helps in user navigation, making the interface more intuitive and straightforward to use. KnitHub also gives users clear and immediate feedback. This includes error notifications during form validation and confirmation alerts for successful actions such as subscriptions.

4.3.5 Maintainability

All of the code, both front and back end, has clear names that clarify what it does, as well as good comments that make it easy to read and understand. The majority of the front-end code is also built utilizing the same buttons, input-fields, boxes, and other components, which simplifies code maintenance by eliminating the need to alter the same code many times.

KnitHub is built with a modern framework, React, with libraries such as MUI (Material-UI), which improves maintainability by standardizing the codebase. As previously said, React is a component-based architecture that allows for code structure and reuse, whereas MUI delivers pre-designed UI components that follow the Material Design standards. This technique maintains consistency in the user interface and speeds up development, making the software more modular, understandable, and easier to maintain over time.

The backend is structured into 4 main parts. First is the data section, that contains all information about the database. This section handles the structure of the database, as well as the entities that define the tables. Using Entity Framework from .NET makes updating the database quite easy, as it handles the migrations when updating the database. Secondly are the data transfer objects (DTOs). They are used to send information between the models, views and controllers. They enable to send only specific information between the different sections, as well as defining what information needs to be present between the different sections. The third part are the services. They are responsible for modifying and retrieving information from the model (database). To do this it utilizes the DTOs. Lastly are the controllers. They are responsible for the interaction between the view and model. They handle the incoming requests and call the different services. The responses and errors from the services are handled in the controller, before passing the information to the client using HTTP responses. In addition to these parts, an own test solution has been created for the backend. All unit and integration tests are put here.

Using this structure along with entity framework to handle that migration of the database makes it easy to know where code should be added when creating new functionality for the application. The test application also provides a platform to test and verify code before putting the new functionality into production.

5 Implementation

5.1 Authentication

Users must have an account and be signed in to take advantage of the majority of KnitHub's features. Most of the authentication was implemented during the first sprint (see F.1).

5.1.1 Create an account (sign up)

The sign-up process is designed to be secure and easy to use. Users begin by filling out a form that requests their name, email, birthday, and a password. The form employs strict validation rules to ensure the accuracy and security of the information provided. For instance, the password must meet specific security requirements, including at least one number, one lower-case letter, one upper-case letter, and one special character. The email validation checks for a valid email format that includes an "@" symbol and a dot, ensuring that email addresses are correctly formatted before submission. These validations are enforced through regular expressions within the form validation logic to minimize security risks.

Additionally, the form requires that all input fields must be filled in before submission. This prevents the backend server from receiving any invalid or empty submissions, ensuring that only complete and validated data is processed. Upon successful validation and submission of the form, the data from the user is packaged and sent to a backend server via an HTTP POST request. This communication is facilitated by Axios, a promise-based HTTP client.

When the crate user request it received, the request content is verified to have content, and the birth date is valid. If the request is bad, an bad request error is sent to the frontend. After this, the backend can begin processing the request. A check to see if a user already exists with the given email is performed. If this is the case, a conflict error is returned to the frontend. If a user with the given email does not exist, the user information will be added to the database. The password the user provided will be hashed before adding it to the database. A JWT-token is then generated for the user to be returned to the frontend. Before the token is sent with an "Ok" status to the frontend, a 6-digit verification code is generated using the token. This verification is sent to the user by email, and is used to verify that the email belongs to the user.

5.1.2 Log in

Logging into the application is simple and secure. The login component creates a user interface where existing users can enter their email address and password. This input is validated through a validation method to ensure that the email format is correct and the password entry is not empty.

If the credentials are authenticated locally, they are sent to the server for authentication. This is, like the registration procedure, done via Axios. Once the backend receives the request, it starts handling the request. It attempts to retrieve the login information from the database. Checks are performed to check if the user exists with the given email, and if the user is banned. If one of these cases are true, the controller will respond with an error. However, if the user exists, a check to see if the provided password matches the hashed password in the database is done. This is done using the passwordhasher. If the password does not match, the controller will return an error. If the passwords match, a JWT-token is generated and sent to the frontend. For security reasons, the backend will not provide information if the login failed because of a wrong password or non existing user with the given email. The server responds with a token that contains the verification and admin status of the user. If the credentials are correct and the account is verified, the user is forwarded to the home page. If the user is not confirmed, a window will appear where the user can enter their verification code. If logging in is not possible, the front-end displays an appropriate message to the user. For example, if either the password or email address is incorrect or the user does not exist, the user will receive a notification stating that 'Login failed. Check your username and password and try again.'. If the user is banned, a custom message is displayed, encouraging them to contact the support team if they believe the ban was incorrect.

5.1.3 Verify the account

When a user creates an account a user token is generated after inserting the user information into the database. This token is further used to create an entry in the userverification table. When inserting an entry to the verification table, a random 6-digit **verification code is generated** and stored in the table along the user's id. This code is then sent to the user's email using Mailersend.

Verification is required upon signup for new users and login for non verifies users. It is implemented through a modal interface, where the user can insert the verification code, or close the modal. It is designed so users can opt to skip verification when signing up and be rerouted to the homepage. At the users profile, there is a button where the user can verify its account at any time, by inserting the verification code received on mail. When the user proceeds with clicking the "Verify" button after inserting the code, a PATCH request is sent to the backend API endpoint, and passes the usertoken and verification code along with it. The backend will verify that the usertoken and verification code is present. The absence of one or both of these will make the backend return a bad request response to the

frontend. If the request is ok, the backend will verify the token, and extract the user id. The user's entry is then retrieved from the backend, using the user id and verification code. If no match is found for this combination, a not found response is sent to the frontend. If the code is found, the user status is updated to "verified", and the verification entry for the user is removed from the database. If an "Ok" response is received in front end, the front will update the token userstatus in session-storage.

Unverified users get the verification option every time they proceeds with a log in. There is that risk with our implementation that some users never verifies their account, but it will also provide a more flexible user experience by not strictly blocking users who choose to skip verification initially.

5.1.4 Log out

In the navigation bar on the website, the "Log Out" option is implemented to provide the users with the choice to either log out or cancel the procedure. When the user clicks the button, a confirmation modal is displayed over the page the user is on. If the user proceeds and confirms the logout, the authentication token is removed from the Session Storage, and the user is logged out from their account. Following this, the user is redirected back to the login page. However, if the user chooses to close the confirmation modal, the action is canceled and they remain on the current page.

5.2 User profile

When the profile page is loaded, Axios retrieves user details such as full name and email address from a backend service and displays them in view mode. This section provides a clean, read-only view of the user's profile data, improving the user experience by providing clear visibility into their personal information. If the user has not yet been verified, a button to that allows the user to verify their account is displayed.

5.2.1 Change information and password

The editing functionality is triggered by a user action, toggling the component to an edit mode. This mode presents users with editable form fields pre-populated with existing data. Input validation is performed to ensure that all fields, especially security-sensitive ones like passwords, are correctly filled before submission. Changes are submitted to the backend via an Axios PATCH request. When the request is received by the backend, the backend will first validate the token, to extract the userId. Checks are then performed, to verify what fields should be updated. An extra check is performed when the password is provided. This check verifies that the provided password hash value matches the one in the database. If no fields are provided to update, or the password does not match, an error is sent to the frontend. If the update is successful, the backend will send an "Ok"-response to frontend, along with what fields have been updated. The frontend will upon successful updates refresh the displayed profile information and provide positive feedback through a success alert. However, if any errors are encountered during the process generate descriptive error messages to guide the user accordingly.

5.2.2 Delete account

Deletion of the profile is managed through a dedicated modal that confirms the user's intent to delete their account. This critical action requires explicit user confirmation to prevent accidental deletions. Upon confirmation, an Axios DELETE request is made to the backend, and upon successful deletion, the user is logged out and all session information is cleared to maintain security. The application then redirects the user to the home page, ensuring a clean exit from the account.

The database has already been set up where the *UserLogIn* table is the main table, where the *userId* has been defined. All other tables that has data belonging to the specific user have the *userId* as a foreign key. In the context, the delete behavior has been set to *cascade*. This means that the entry is removed from *UserLogIn*, all other tables should delete entries where the *userId* matches the one of the user to be deleted.

5.3 Admin

Users with admin status will gain access to an admin page that unlocks some extra features. This page has various functions for application management and is an important component because it provides a dashboard that is designed for managing various aspects of the application, including users, messages, and newsletters.

5.3.1 Dashboard overview

Only administrators can access the dashboard, which displays different statistics for the users, allowing the administrators to effectively monitor and manage the application. Data for the dashboard is obtained through API calls with Axios, which are set up within the useEffect hook to ensure that data is collected upon component mounting or when the relevant state change occurs.

5.3.2 User Management

Administrators can manage registered users via the "View Users" page, which contains search, sorting, and pagination options. Axios is used to retrieve user data via a secure backend API. Administrators can also adjust users' statuses, including adding or removing admin permissions and changing user bans. These tasks utilize particular API requests that updates the database properties, which includs *'isAdmin'* and *'UserStatus'*. Changes are mirrored in the frontend through React state updates, which keep the UI consistent with the backend state.

Banning users
Skkkriv

5.3.3 View Messages

Administrators can easily manage questions from the contact us form via the message page. Messages are classified and sorted based on their active and handled statuses using GET request parameters, which improves data retrieval from the backend. The administrator can find messages in the specified state using the request's name or email address. To update the status of messages, PATCH requests are used to change the *'IsActive'* and *'IsHandled'* properties. When administrators respond to messages, the responses are structured and attached to existing message string (Response). This results in a systematic and consistent record of interactions, which is necessary for administrative tracking and user communication.

5.3.4 Newsletter

Administrators may monitor and control the subscriber list, which simplifies newsletter management. Subscriptions can be cancelled using the delete subscriber request. The front-end manages these interactions via reactive updates, in which the state changes based on search inputs or actions like deletions.

5.4 Recipes

One of the main features on KnitHub, is the recipe management, where users can upload and view their recipes anytime on the website. The Recipe management is designed in a way so the user can upload recipes and view them in two different ways. The PDF viewer can fetch and open a specific PDF from the database for the viewer, with the possibility to open the PDF in a new tab or rate the recipe. The rating is designed so the user can enumerate their recipes by giving them 1 to 5 hearts, and can be updated whenever a user opens the PDF. It is also implemented a search mechanism, that gives the user the opportunity to search through their recipes based on name, or filter them by needle-size and/or knitting gauge.

The main upload component handles the overall recipe functionalities implemented, including state management to control and display the actual upload modal, and the list of uploaded recipes. The actual file upload process uses state hooks to track the chosen files and the details uploaded by the user for specific recipes. The upload component handles these file selections, input changes for the recipe information, and the upload process through an HTTP POST request using Axios. Error handling and a feedback mechanism were implemented to inform users of the upload status of their recipes. The component responsible for displaying the list of uploaded recipes uploaded, retrieves the recipes in a list from the backend and allows users to sort and view the details of each recipe. The user can also sort or delete their recipes if desired. We use the same cards in the recipe management that is used on the admin page.

5.5 Project tracking

This project features the capability to create and manage various knitting projects. Each project includes all relevant notes, necessary needles or yarn, and the recipe. The user interface allows for easy management of knitting projects, with the ability to create, read, update, and delete projects.

Project data is displayed in an organized way for simple project tracking. Backend API calls maintain data consistency and security. Axios is employed to implement all CRUD operations for the projects. The user interface is immediately refreshed when a project is updated or deleted.

Users can add new projects by clicking on a "+" button. They can select what they need from their inventory of needles, yarn, and recipes when creating a new project. This selection helps users keep track of their inventory usage and manage their resources effectively. The inventory reflects what is currently in use and what is available while a project is planned or ongoing. Completed projects remove used yarn from the inventory. Projects can be sorted by type and filtered based on their status using a switch menu. Users can mark projects as completed, change their status, and update them with new notes, needles, or yarn as they progress.

5.5.1 Calculator

When working on a project, it might be necessary to make some calculations. KnitHubs calculators are located on the project tracking page, so the knitters easily can access the calculators while the project is on the screen. The calculators require all input fields to have positive numbers as input, otherwise an error message appears. Both the increase and decrease calculator informs the knitter of how many stitches should be on the needle after doing the knitting the calculator recommends.

The **yarn calculator** was implemented for the user to efficiently calculate the required number of skeins needed to achieve the desired new length based on the length of the yarn the user will use on the project. The purpose of the yarn calculator is that knitters do not purchase more or less yarn then needed. The calculator requires that the user insert the yarn length (L_{recipe}) and number of skeins in the recipe (N_{recipe}), along with the yarn length of the yarn the user wants to use (L_{new}). The calculation involves first determining the total length of the original yarn listed in the recipe (T_{original}), achieved by multiplying the length by the number of skeins ($T_{\text{original}} = L_{\text{recipe}} \times N_{\text{recipe}}$). The required number of skeins (N_{required}) is then calculated by dividing the total length by the new length, and rounding the result up to the nearest whole number:

$$N_{\text{required}} = \lceil \frac{T_{\text{original}}}{L_{\text{new}}} \rceil$$

The **increase calculator** is designed to help knitters in determining the new stitch count and stitch pattern after increasing the stitches on the needle. The calculator helps the knitter get a secure and even increase in the stitches. Users need to input the initial number of stitches on the needle (S_{initial}) and the number of increases to be made (I). The calculation computes the new stitch count (S_{new}) by adding the initial number of stitches and the number of increases ($S_{\text{new}} = S_{\text{initial}} + I$). Subsequently, the new stitch pattern is determined by dividing the new stitch count by the number of increases, illustrating the stitches between each increase. The result is rounded down to ensure it is an integer because you do not have half stitches.

$$S_{\text{new}} = S_{\text{initial}} + I$$

The **decrease calculator** is designed to help knitters calculate the new stitch count and stitch pattern after decreasing the stitches on the needle. The knitter has to insert the number of stitches on the needle (S_{initial}) and the number of decreases to be made (D). The new stitch count S_{new}) is calculated by subtracting the number of decreases from the number of stitches on the needle ($S_{\text{new}} = S_{\text{initial}} - D$). The new stitch pattern for the decreases is calculated by determining the number of stitches between each decrease. By dividing the new stitch count by the number of decreases, the stitch pattern is calculated, and any remainder is stored for adjustment.

$$S_{\text{new}} = S_{\text{initial}} - D$$

The resulting stitch pattern gives instructions to knit a given number of stitches followed by knitting two stitches together to decrease the stitches. If there is any remainders, the result specifies how they should be handled. The decrease calculator helps knitters achieve even stitch decreases.

5.5.2 Counter

For the Counter, the React components is integrated with the backend services, allowing the users to easily get access to the counter functionality. The counter is designed for users to create, modify and manage the counters at any time. The condition of use the events are handled, and the state is maintained using reach hooks and a custom counter hook. It is implemented so users can add, edit and delete the counters. When the user proceeds with any of these actions, a sequence of backend calls is triggered, and the session token is used for authentication. The data is encoded, headers are added, and it is transmitted over a fetch API to create or refresh a counter. The user receives visual confirmation and the counter is updated instantly if the procedure is successful. However, in the event of an error, the user is notified via an alert that shows an error message on the screen.

5.6 Stash

The stash page allows the user to upload both needles and yarn, giving them control over what they have laying about the house. The yarn page allows you to move between yarn and needle stash using a switch-container.

On the yarn page, the user will find a simple interface where they can view the yarn they have uploaded, these will look like cards, and add it into your stash by pressing the "+" button. The user can also change or delete existing yarn in their stash by clicking on the connected card. When the yarn is updated or deleted, an API request is sent to the backend, ensuring data consistency and security. If the process is successful, the user interface will be updated instantly. If an error occurs, the user receives an alert.

On the Needles page all knitting needles are displayed in a table, where users can add new entries by clicking the "+" button and adding the necessary details. The needles come in four pre-defined types, including an "other" option which allows users to specify a custom type. The table can be sorted by needle type, and the filter options in the drop-down menu are dynamically adjusted to display just the types in stock. Each row includes full information and the opportunity to delete it. Interactions such as creating or deleting a needle trigger API calls to the backend. The frontend manages the user interface using React hooks and MUI components. This setup allows for validation of input fields, immediate error feedback, and updates throughout the UI upon successful operations. The API calls ensure data integrity and handle errors smoothly, enhancing user experience with alert components for notifications.

5.7 Home

5.7.1 Navbar

The navigation bar is a key part of the website. Like a roadmap, it gives users a list of the main sections, pages, and features. It changes based on whether the user is logged in or not, adjusting what it shows accordingly.

Because the navigation bar changes to suit the user, it makes using the site easier. Users can click on the items in the navigation bar to move around the site. This makes the site easy to use and helps users find what they want or discover new things.

The navigation bar is at the top of the website, so it's one of the first things a user sees and it's always easy to reach. It also changes size to fit the screen, making the site easy to use on different devices, from computers to phones.

Even when the screen size changes, the navigation bar stays at the top of the page. It's always there, no matter where the user scrolls, making it easy to move between pages. This makes the site easier to use and improves the user's experience.

5.7.2 Homepage

The homepages on KnitHub is separated so visitors and logged-in users view them differently. This is done to provide a more personalized homepage designed specifically for the logged in user. The visitors homepage features what you can get access to as a logged-in user.

Upon your initial visit to the KnitHub website, you are greeted by a landing page that serves as an introduction to the services and features offered by our application. This landing page is curated to provide the user with a warm welcome from our dedicated team. It also provides an overview of who we are, our mission, and the potential benefits and conveniences you can enjoy when you choose to become a registered user with us.

We are not just about knitting, but we also aim to inspire the user with creative ideas and the latest trends in the knitting world. To achieve this, we have prominently displayed our Instagram feed on our landing page. It showcases

an array of impressive and inspiring works from our community members, which we hope will spark your creativity and encourage you to join our thriving community.

When a user successfully logs in to KnitHub, they are immediately taken to a personalized homepage that provides statistics about their projects and stash. This page contains information like their name, the amount of yarn skeins and needles in their stash, and the number projects and recipes uploaded. Each of these boxes are linked to a corresponding page, so if the user clicks on the box displaying yarn skeins, they will be directed to their yarn stash page. The user token is required to access this data from the backend. It is used to obtain the profile information by submitting a GET request using Axios. This ensures that when retrieving project and inventory data, the frontend remains synchronized with the backend, allowing the user to get real-time data from their profile and inventory statistics. For design purposes and to enhance user engagement, an Instagram feed has also been integrated at the bottom of the home page (the implementation of this can be found in section 5.9.2).

5.8 Contact Us

The "Contact Us" page is intended to simplify user communication with an interactive interface that combines front-end and back-end components. This interface allows a smooth transition from user input to data processing and response management, improving both user experience and administrative efficiency. When users visit the "Contact Us" page, they are presented with a form where they may submit their question. To send in the question, they must enter some information, such as their name, email address, and message. This form makes use of the react-hook-form to guarantee that all inputs are entered correctly before submission. The form data is securely transferred to the back-end using an Axios POST request.

At the backend, the Contact Controller validates these requests, before saving the valid data in the database. If the form submission succeeds, the Set Alert component shows a success message, indicating that the message was successfully sent. If the submission fails, an error alert is generated, prompting the user to try again. The data received by the back-end is contained within a Contact Request Dto. This encapsulation guarantees that only critical information is processed, which improves security by reducing the exposure of extraneous data pieces. Once approved and stored, contact requests are saved in the database for future retrieval and administration by admins.

Additionally, the "Contact Us" page features an FAQ section, which employs an accordion-style format [1]. This interactive component gives users fast access to look at answers of frequently asked questions. The page also includes static contact information such as an email, phone number, and physical address, which is accompanied by an image.

5.9 Resources

On KnitHub both visitors and user can find a resources page. This is a page useful for both beginner and experienced knitters since it provides information and inspiration for their knitting projects. This page has three primary components: a searchable table of knitting words, a calculator, and an Instagram feed with knitting-related posts.

5.9.1 Knitting Terms Table

The knitting terms table allows users to quickly look up knitting abbreviations and their full definitions. The feature takes advantage of React's use state to keep the search text and filtered results updated. When a user types something into the search box, an on Change event updates the search text state, which triggers a use effect hook. This hook filters a predefined list of knitting terms, matching them to the user input, and displays relevant phrases that match their input.

5.9.2 Instagram

The Instagram feed uses the Instagram Graph API to fetch and show a grid of knitting-related posts. When entering the page, or refreshing it, a fetch request with an Instagram access token is used to retrieve posts and media URLs. A defined number of posts are randomized to display a variety. If anything happens during data retrieval, these errors are handled by assigning an error state that notifies users of the problem. The page also has a loading indicator that appears when posts are fetched, improving the user experience by clearly expressing the loading progress. Note that the Instagram token expires June, 14th.

[1] Vertically stacked headers that reveal more details when clicked on.

5.10 About us

The vision for the about-us page was to present the purpose, different features KnitHub has to offer and the team behind the platform. It includes some figures between the information to make the page get a more visually appealing design. The page begins with a welcome message and short explanation of why KnitHub was created, before the key features; *Resource page*, *Yarn stash*, *Needle stash*, *Pattern organization*, *Project tracking* and other smaller features is presented. Furthermore, the users who is not logged in, is encouraged to sign up by a link to the sign-up page placed in the middle of the page. Finally, the team is presented by their role and background in the same way the features was presented at the beginning of this page.

5.11 Footer

The Footer component on Knithub is designed to give users a comprehensive and interactive experience at the bottom of each page. This section is crucial for enhancing user engagement through social media links, a subscription form, providing essential contact information, and easy access to pages like about, contact us and resources.

5.11.1 Newsletter

The user can use the subscription form located on the left side of the Footer to sign up for the newsletters. This form provides feedback on issues such as wrong emails or empty fields by validating email inputs against standard formats using the 'useForm' hook. Upon successful submission, a POST request is sent to the back-end to register the subscriber's email. The backend performs a check to whether the email is already in the database or not. If the email is not in the database, it is added. If the email is already there, a conflict response is sent to the frontend. If successful, users receive an alert confirming their subscription.

6 Testing

6.1 Frontend testing

When testing the front-end code, we have used the Jest and React testing libraries. We have unit tested some of the individual components like the Counter, Calculator, Needle and Yarn, to ensure that all components functioned correctly by verifying incrementing, decrementing and proper data display operations.

When we tested the interactions between different components, such as the Custom button and Input field, we did ensure correct data flow and user interaction sequences. The navigation bar component was tested specifically across the various user authentication states, to verify that the link displays and accessibility was correct. Additionally, we tested the application manually to provoke any errors within the UI to identify and handle any potential failure points efficiently. By taking this proactive approach it helped us ensure that the application could gracefully handle unexpected user inputs and system errors.

To monitor any unintended changes in the UI, we incorporated Snapshot tests which helped us maintaining consistency throughout the whole development process. By combining these test approaches, we made sure the frontend's performance and user experience was optimal, giving a stable and reliable application interface.

6.2 Backend testing

To be able to test the code from our Strikkeapp, a new test project was created using .NET. This test project includes references to the Strikkeapp project, allowing it to utilize its functionality. The tests contained in this project are split into unit- and integration tests. The vast majority of tests are integration tests. The reason for this is that it was hard to test the functionality of the services without an actual database.

To test the individual components, an in-memory database has been used. This provided an in-persistent database that would only be stored in the computer´s memory, and no I/O to disk would be needed. The database would be seeded to provide data that can be utilized during testing. For each test, a newly seeded database is created to provide consistent data for the tests. In addition to the in-memory database, mocks have been used extensively. Mocks enable the creation of "mock" components, where the input and output of the component can be controlled. This ensures predictable responses from the components used in the component to be tested. When testing the application's services, every component was mocked. However, during testing controllers, the service would not be mocked. This approach was chosen to ensure that the services and controllers worked together as expected.

One test that was viewed as extra important was the deletion of a user account. When a user deletes their account, all entries belonging to the user in the database should be deleted. Therefore an own test has been written. Here the database seeds the entire database with entries belonging to a single user. The user was then deleted, and checks were performed to ensure that each table in the database was empty.

Throughout the development process, tests were written before implementing the actual component. This was done to follow test-driven development as mentioned in section 3.2.

7 Future Work

7.1 Unimplemented functionality from the backlog

Functionalities like improving the user interface for mobile devices, ensuring a responsive and adaptive design that scales seamlessly across different screen sizes. This will improve accessibility, especially for users who primarily use the application from their mobile phones.

Implementing a feature that allows users to reset their passwords without being logged in. This functionality will involve verification steps such as sending a link to the user's registered email to verify their identity, thereby enhancing the security framework while maintaining user accessibility.

Developing a function on the admin page to send out emails to all subscribers on the newsletter list will streamline communication and keep the community engaged. This functionality will support mass emailing while ensuring compliance with data protection regulations, promoting a reliable and efficient channel for updates and promotions.

Creating a platform where users can see responses to their 'Contact us' form submissions will increase transparency and trust. The idea involves emailing users a link that mirrors the admin page view, showing the reply form 'KnitHub' and allowing users to reply directly. This two-way communication model will significantly improve customer service and user satisfaction.

7.2 Ideas for further development

Introducing social features like adding friends, sharing finished projects, and integrating a leader board to showcase users who have knitted the most, can greatly enhance the community aspect of the app. These features will encourage interaction, competition and cooperation between users, potentially increasing daily active users and fostering a vibrant community.

Allowing users to purchase knitting patterns directly from the pattern page could give users easy access to a diverse selection of knitting designs. This feature will not only increase user engagement by simplifying the process of acquiring new patterns, but also establish the application as a one-stop shop for knitting enthusiasts.

7.3 Security

If we had more time we should have enhanced the security of the application. Under are some examples of improvements we could have made to avoid certain attacks.

Today, we use open APIs that offer unrestricted access, which can be exposed to unauthorized data retrieval or manipulation. Implementing closed APIs with strong authentication and authorization checks ensures that only trusted identities have access to the data. This prevents unauthorized access and ensures secure data exchange between the frontend and backend.

SQL injection attacks happen when an attacker inserts or modifies SQL queries using the clients input data. There are several techniques to protect against such attacks, one of them is to utilize pre defined statements and parameterized queries rather than directly creating SQL instructions from user input. One can also employ Object-Relational Mapping (ORM), which is a programming technique that maps the date between an object-oriented language and a relational database [8]. This helps the developers manipulate the database with objects rather than SQL queries.

With future iterations, it is critical to address session hijacking vulnerabilities where an attacker can steal or manipulate session tokens to gain unauthorized access to a user's account. Implementing robust session management practises such as secure token storage and regular session timeouts will mitigate such risks [9].

Protection against cross site scripting attacks should also be a priority. This involves sanitizing and validating all user input to prevent malicious scripts from being injected into the website, which could compromise user data and the

integrity of the application [10]. Using content security policy headers can further help mitigate these risks by limiting the resources the client is allows to load[11].

8 Discussion

8.1 Performance

Write about the results from our tests? Vise hvor mye testene er på front og back.- sette resultat. Skrive at de funksjonene som ikke er testet i front, da at de ikke har testet, er testet av bruk å prøve å framprovosere feil. Vi har latt ulike studenter få prøve applikasjonen, underveis i utviklingen få prøve p gi tilbakemeldning.

Backend tests
The application's backend has a total of 273 tests, with line coverage of 96,16% and branch coverage of 89,34% as shown in figure 6 in section A. The coverage report shows numbers that are much smaller than our calculations. The reason for this is that it covers all lines of code in the application. Some parts of the code, such as the migrations files and the program file are not able to be tested. This results in these having very low test percentages. Therefore, calculations have been performed to find the actual coverage percentage. These calculations can be seen in figure 6.

We are satisfied with our coverage percentages. All components have been tested to ensure they work with valid data. In addition, most edge cases have been tested for every component. Unfortunately, tests for database exceptions have not been created. This has not been done, as we have not been able to find a way to write tests where database exceptions were produced successfully. If these tests were created, the coverage percentages would likely be 100&.

Frontend tests
We have tested the frontend of the application, and our test output indicates a great coverage with 27 test suites and a total of 131 individual tests, that all passes within 15 seconds. Most files has gone through automated tests, but the remaining ones has been manually tested.

8.2 Code structure

We have structured our frontend code by storing most of the CSS files in a separate folder, with individual files for text, boxes, buttons, etc. We have stored all tests in one directory, which contains sub-folders. The components is grouped in their own folder, further divided into logically categorized subfolders such as UI, Navigation, etc. Our 'App.js' and 'index.js' files are also places in the 'src' directory, but outside of other folders as they hold the entire frontend. Furthermore, we have a 'pages' folder where each page has its own directory, containing relevant CSS files that conflict with the CSS in the separate folder, and specific page-related files. If we were to do this any differently, we would have all CSS gathered at one place, but as we attempted to do this towards the end, it conflicted with other CSS files when placed in the same CSS directory.

8.3 Problems and choice of solution

8.3.1 GitHub

Merge Requests
Throughout our development process, as mentioned, GitHub has been used for coding collaboration and code management. The team members entered the project with different levels of experience with GitHub. Due to this, we had some internal problems throughout our development process. We intended to use GitHub's pull request functionality. However, setting up branch protection rules could not be done in the provided repository. We therefore set up a system in Notion to keep track of merges. Here, we would manually set the statuses of the branches and could set them up as "ready for review" and "ready for merge", and thereby have other team members review the code before merging it into the main branch. In the beginning, this worked well for our group, however as we got more into the project, the threshold for merging smaller features without it being reviewed by another team member was lowered.

Sending files manually
In the beginning, some team member had problems pushing their code to the main branch. Therefore we sometimes sent files to each other, where one team member would add another team member's file to their branch before pushing it to the main branch. Many of these problems were a result of in-expertise and knowledge of how to use GitHub correctly. These problems became less of an issue, once all team members started getting comfortable using Git.

Different OS's
Since all team members did not use the same OS, we had problems with how folders would be named, pushed and retrieved from GitHub. During the development, computers using Windows and MacOS have been used. These computers have handled naming differently when pushing to GitHub. This resulted in our repository having two folders within the backend. One with capitalized (S) and one without capitalization (s). In turn, this resulted in problems when running the backend on MacOS, since it would have two folders with different parts of the code in it. On Windows, it worked fine, and all code would be located in the same folder. To fix this, the GitHub online editor was used. All files were moved into a single folder with a capitalized S, and the other folder was removed. In addition, we had problems if a team member ended a folder or file name with a white space. MacOS handled this fine, while Windows was not able to retrieve the file from GitHub. To fix this, we needed to go through the files with problems and remove white spaces.

8.4 Reflections

8.4.1 Reflection on team

The team has together created the web application for knitters. Throughout the project, we have maintained good collaboration and helped each other whenever someone was stuck. This has taught us a lot about the importance of teamwork, and the great value of having different perspectives in problem-solving. For us, good communication and the weekly check-ins has been crucial in our development process. We are grateful for this learning experience and now we feel ready to work in teams in the future.

8.4.2 Reflection on complexity

Utilizing Microsoft Azure
The complexity of the backend could be improved by moving the database and storage of recipes from local machine storage, to a remote service such as Azure. Instead of a local SQLite database, an Azure SQL database could be used. Additionally, recipes could be stored using Azure BLOB storage. This change is particularly advantageous for a production environment ensuring consistent data across multiple machines. Any changes or insertions performed by one team member would immediately be accessible to all team members.

In the future work section 7, we have listed some features we would like to add to KnitHub, if the application would be developed further. These would increase the complexity of the application.

8.5 Lessons learned

We learned a lot as a group this semester while working on this task. We've learned about effective team collaboration and how useful technologies like GitHub and Notion are for maintaining clear workflows and regular communication. Regular meetings have been essential for ensuring consistency and quick problem solving. The combination of Agile and DevOps was critical, as continuous integration and deployment pipelines improved efficiency in operations. It allowed for quick, dependable feature upgrades while still developing the product.

The project gave us hands-on experience with technologies like.NET and React, teaching us how to deal with and overcome technical challenges, which was crucial for our development. We learned about automated testing, including Test-Driven Development (TDD), which was critical in maintaining high code quality and efficiency.

9 Conclusion

Our team has over the period of 4 months used agile methods to developed the web application KnitHub. KnitHub is a user-friendly platform, with various features that help knitters keep track of their knitting projects and inventory. During this time, we focused on creating the most important features that both new and experienced knitters might find useful. The team has designed the application's interface to be user friendly, and easy for users to navigate through various features such as project tracking, pattern management, inventory control, among other features KnitHub offers. Additionally, we integrated a resource page that includes some abbreviation and their meanings, along with different calculators that assist knitters in making the calculations needed for their projects. This features are valuable for both beginners and experienced knitters.

10 Disclosure

Throughout the assignment, the group has used AI tools differently to help with our task. All team-members have used ChatGPT as a tool to find and identify problems with our code. ChatGPT has been a great tool to solve errors when we have discovered issuses that we was unable to identify ourselves. We have also used the AI Quillbot, to help us with language and sentence structure for our report.

Most of the images displayed on the web-page itself is retrieved from Ksuview, a creator that publishes her art on Canva [12].

References

[1] GeeksforGeeks. What is an API (Application Programming Interface). *GeeksforGeeks*, February 2024. [Online; accessed 15. May 2024]
URL: `https://www.geeksforgeeks.org/what-is-an-api`.

[2] Instagram Platform - Documentation - Meta for Developers, May 2024. [Online; accessed 15. May 2024]
URL: `https:/developers.facebook.com/docs/instagram`.

[3] Virtual DOM and Internals – React, May 2024. [Online; accessed 10. May 2024]
URL: `https://legacy.reactjs.org/docs/faq-internals.html`.

[4] React, May 2024. [Online; accessed 10. May 2024]
URL: `https://react.dev`.

[5] MUI: The React component library you always wanted, May 2024. [Online; accessed 10. May 2024]
URL: `https://mui.com`.

[6] Rick-Anderson. Dependency injection in ASP.NET Core, May 2024. [Online; accessed 13. May 2024]
URL: `https://learn.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-8.0`.

[7] Ian Sommerville. *Engineering Software Products*. Pearson Education Limited, 1st edition, 2021.

[8] GeeksforGeeks. What is Object-Relational Mapping (ORM) in DBMS? *GeeksforGeeks*, February 2024. [Online; accessed 12. May 2024]
URL: `https://www.geeksforgeeks.org/what-is-object-relational-mapping-orm-in-dbms`.

[9] What is Session Hijacking | Types, Detection & Prevention | Imperva, December 2023. [Online; accessed 15. May 2024]
URL: `https://www.imperva.com/learn/application-security/session-hijacking`.

[10] What Is Cross Site Scripting (XSS) and How Does It Work? | Synopsys, May 2024. [Online; accessed 14. May 2024]
URL: `https://www.synopsys.com/glossary/what-is-cross-site-scripting.html#C`.

[11] Content Security Policy (CSP) - HTTP | MDN, March 2024. [Online; accessed 14. May 2024]
URL: `https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP`.

[12] Ksuview – Canva, May 2024. [Online; accessed 12. May 2024]
URL: `https://www.canva.com/p/ksuview/?utm_medium=referral&utm_source=creator_share&utm_campaign=creator_share`.

A Coverage Report

What	Line Coverage	Branch Coverage
Controllers	94,7	86,3
Models	100	100
Recipes	96,6	75
Services	89,5	85,4
User (models)	100	100
SUM	**96,16**	**89,34**

Figure 6: Calculations for test coverage

Summary

Sponsor Star

Information

Parser:	Cobertura
Assemblies:	1
Classes:	86
Files:	55
Coverage date:	15.05.2024 - 18:43:43

Line coverage

63%

Covered lines:	2726
Uncovered lines:	1580
Coverable lines:	4306
Total lines:	6917
Line coverage:	63.3%

Branch coverage

85%

Covered branches:	574
Total branches:	672
Branch coverage:	85.4%

Method coverage

Feature is only available for sponsors

Upgrade to PRO version

Risk Hotspots

Assembly

Filter:

Assembly	Class	Method	Crap Score	Cyclomatic complexity
Strikkeapp	Strikkeapp.Services.ProjectService	PatchProject(...)	31.75	26
Strikkeapp	Strikkeapp.Services.UserInfoService	UpdateProfileInfo(...)	24.26	24
Strikkeapp	Strikkeapp.Services.InventoryService	UpdateYarn(...)	22.14	22
Strikkeapp	Program	<Main>$(...)	20	4
Strikkeapp	Strikkeapp.Services.ProjectService	CreateProject(...)	18.58	18

Coverage

Collapse all | Expand all

By namespace, Level: 2

Grouping:

Select coverage types & metrics

Filter:

	Line coverage					Branch coverage		
Name	**Covered**	**Uncovered**	**Coverable**	**Total**	**Percentage**	**Covered**	**Total**	**Percentage**
Strikkeapp	**2726**	**1580**	**4306**	**9565**	**63.3%**	**574**	**672**	**85.4%**
-	**0**	**57**	**57**	**90**	**0%**	**0**	**4**	**0%**
Program	0	57	57	90	0%	0	4	0%
Strikkeapp.Controllers	**576**	**32**	**608**	**1207**	**94.7%**	**202**	**234**	**86.3%**
Strikkeapp.Controllers.ContactController	33	0	33	75	100%	9	10	90%
Strikkeapp.Controllers.CounterController	79	6	85	162	92.9%	32	38	84.2%
Strikkeapp.Controllers.InventoryController	166	10	176	319	94.3%	62	72	86.1%
Strikkeapp.Controllers.NewsletterController	31	3	34	75	91.1%	9	12	75%
Strikkeapp.Controllers.ProjectController	28	0	28	65	100%	0	0	
Strikkeapp.Controllers.RecipeController	60	2	62	125	96.7%	24	26	92.3%
Strikkeapp.Controllers.RecipeRatingController	19	0	19	49	100%	4	4	100%
Strikkeapp.Controllers.UserInfoController	41	2	43	79	95.3%	16	18	88.8%
Strikkeapp.Controllers.UsersController	119	9	128	258	92.9%	46	54	85.1%
Strikkeapp.Data	**211**	**21**	**232**	**538**	**90.9%**	**0**	**4**	**0%**
Strikkeapp.Data.Context.StrikkeappDbContext	129	0	129	178	100%	0	0	
Strikkeapp.Data.Dto.ContactRequestDto	6	0	6	12	100%	0	0	
Strikkeapp.Data.Entities.ContactRequest	8	0	8	26	100%	0	0	
Strikkeapp.Data.Entities.Counter	4	0	4	19	100%	0	0	
Strikkeapp.Data.Entities.KnittingRecipes	7	0	7	26	100%	0	0	
Strikkeapp.Data.Entities.NeedleInventory	7	0	7	29	100%	0	0	
Strikkeapp.Data.Entities.Newsletter	1	0	1	9	100%	0	0	
Strikkeapp.Data.Entities.ProjectEntity	9	0	9	32	100%	0	0	
Strikkeapp.Data.Entities.ProjectTracking	3	14	17	43	17.6%	0	2	0%
Strikkeapp.Data.Entities.ProjectYarnInventoryEntity	5	0	5	32	100%	0	0	
Strikkeapp.Data.Entities.RecipeRatingEntity	4	0	4	16	100%	0	0	
Strikkeapp.Data.Entities.UserDetails	4	0	4	22	100%	0	0	
Strikkeapp.Data.Entities.UserLogIn	5	7	12	32	41.6%	0	2	0%
Strikkeapp.Data.Entities.UserVerification	7	0	7	23	100%	0	0	
Strikkeapp.Data.Entities.YarnInventory	12	0	12	39	100%	0	0	
Strikkeapp.Migrations	**0**	**1298**	**1298**	**1441**	**0%**	**0**	**0**	
Strikkeapp.Migrations.FinishedDb	0	824	824	924	0%	0	0	
Strikkeapp.Migrations.StrikkeappDbContextModelSnapshot	0	474	474	517	0%	0	0	
Strikkeapp.Models	**350**	**0**	**350**	**2484**	**100%**	**34**	**34**	**100%**

	Line coverage					Branch coverage		
Name	Covered	Uncovered	Coverable	Total	Percentage	Covered	Total	Percentage
Strikkeapp.Models.AddNeedleRequest	10	0	10	91	100%	6	6	100%
Strikkeapp.Models.AddYarnRequest	16	0	16	91	100%	6	6	100%
Strikkeapp.Models.BanUserRequest	7	0	7	75	100%	2	2	100%
Strikkeapp.Models.CounterResult	11	0	11	95	100%	0	0	
Strikkeapp.Models.CreateCounterRequest	6	0	6	95	100%	2	2	100%
Strikkeapp.Models.CreateCounterResult	13	0	13	95	100%	0	0	
Strikkeapp.Models.CreateUserRequest	16	0	16	75	100%	4	4	100%
Strikkeapp.Models.DeleteCounterRequest	5	0	5	95	100%	0	0	
Strikkeapp.Models.DeleteItemRequest	6	0	6	91	100%	2	2	100%
Strikkeapp.Models.DeleteItemResult	13	0	13	114	100%	0	0	
Strikkeapp.Models.GetCountersResult	13	0	13	95	100%	0	0	
Strikkeapp.Models.InventoryGetResult	15	0	15	114	100%	0	0	
Strikkeapp.Models.InventoryResult	13	0	13	114	100%	0	0	
Strikkeapp.Models.LogInUserRequest	6	0	6	75	100%	2	2	100%
Strikkeapp.Models.MailResult	11	0	11	18	100%	0	0	
Strikkeapp.Models.NeedleInventoryDto	6	0	6	114	100%	0	0	
Strikkeapp.Models.NewsletterResult	9	0	9	35	100%	0	0	
Strikkeapp.Models.NewsletterSubscribersResult	13	0	13	35	100%	0	0	
Strikkeapp.Models.ProjectCreateModel	6	0	6	25	100%	0	0	
Strikkeapp.Models.ProjectModel	7	0	7	25	100%	0	0	
Strikkeapp.Models.RecipeRatingModel	2	0	2	9	100%	0	0	
Strikkeapp.Models.TokenResult	13	0	13	19	100%	0	0	
Strikkeapp.Models.UpdateAdminRequest	7	0	7	75	100%	2	2	100%
Strikkeapp.Models.UpdateCounterRequest	6	0	6	95	100%	0	0	
Strikkeapp.Models.UpdateInventoryResult	22	0	22	114	100%	0	0	
Strikkeapp.Models.UpdateItemRequest	8	0	8	91	100%	4	4	100%
Strikkeapp.Models.UpdateProfileInfoResult	13	0	13	39	100%	0	0	
Strikkeapp.Models.UpdateYarnRequest	16	0	16	91	100%	2	2	100%
Strikkeapp.Models.UserInfoServiceResult	15	0	15	39	100%	0	0	
Strikkeapp.Models.UserResultDto	3	0	3	75	100%	0	0	
Strikkeapp.Models.VerificationRequest	6	0	6	52	100%	2	2	100%
Strikkeapp.Models.VerificationResult	13	0	13	52	100%	0	0	
Strikkeapp.Models.VerificationResultCreate	13	0	13	52	100%	0	0	
Strikkeapp.Models.YarnInventoryDto	11	0	11	114	100%	0	0	
Strikkeapp.Recipes	**57**	**2**	**59**	**250**	**96.6%**	**3**	**4**	**75%**
Strikkeapp.Recipes.Models.RecipeInfo	15	1	16	38	93.7%	3	4	75%
Strikkeapp.Recipes.Models.RecipePDFResult	13	0	13	58	100%	0	0	
Strikkeapp.Recipes.Models.RecipePatch	3	1	4	38	75%	0	0	
Strikkeapp.Recipes.Models.RecipeServiceResult	13	0	13	58	100%	0	0	
Strikkeapp.Recipes.Models.RecipeServiceResultGet	13	0	13	58	100%	0	0	
Strikkeapp.Services	**1458**	**170**	**1628**	**2895**	**89.5%**	**335**	**392**	**85.4%**
Strikkeapp.Services.AutoMapperProfile	22	0	22	43	100%	0	0	
Strikkeapp.Services.ContactService	103	0	103	194	100%	26	26	100%
Strikkeapp.Services.CounterService	113	20	133	248	84.9%	20	20	100%
Strikkeapp.Services.InventoryService	301	44	345	586	87.2%	66	74	89.1%
Strikkeapp.Services.MailService	41	3	44	80	93.1%	5	6	83.3%
Strikkeapp.Services.NewsletterService	52	11	63	127	82.5%	9	10	90%
Strikkeapp.Services.ProjectService	146	19	165	281	88.4%	55	72	76.3%
Strikkeapp.Services.ProjectYarnInventoryService	183	17	200	332	91.5%	46	64	71.8%
Strikkeapp.Services.RecipeRatingService	31	2	33	69	93.9%	6	8	75%
Strikkeapp.Services.RecipeService	109	19	128	237	85.1%	25	28	89.2%
Strikkeapp.Services.TokenService	55	5	60	104	91.6%	4	6	66.6%
Strikkeapp.Services.UserInfoService	71	7	78	136	91%	30	30	100%
Strikkeapp.Services.UserService	177	13	190	340	93.1%	34	36	94.4%
Strikkeapp.Services.VerificationService	54	10	64	118	84.3%	9	12	75%
Strikkeapp.User	**74**	**0**	**74**	**660**	**100%**	**0**	**0**	
Strikkeapp.User.Models.BanUserResult	13	0	13	110	100%	0	0	
Strikkeapp.User.Models.DeleteUserResult	11	0	11	110	100%	0	0	
Strikkeapp.User.Models.GetUsersDto	5	0	5	110	100%	0	0	
Strikkeapp.User.Models.GetUsersResult	13	0	13	110	100%	0	0	
Strikkeapp.User.Models.UpdateAdminResult	15	0	15	110	100%	0	0	
Strikkeapp.User.Models.UserServiceResult	17	0	17	110	100%	0	0	

Generated by: ReportGenerator 5.2.5.0
15.05.2024 - 18:43:46
GitHub | reportgenerator.io

B Personas

Sissel, the unorganized Knitter
Sissel, 45 years old, lives in Trondheim with her family. Her home has a storage room dedicated to her hobbies, which is stocked with yarn and fabric. She works as a school secretary and does a number of administrative tasks on a regular basis. Sissel has a bachelor's degree in Business and Leadership, which gives her a strong basis for successful resource management and project leadership.

Her house is well-organized, with the exception of her knitting supplies, which she wishes to regain. Sissel considers knitting a relaxing activity, but due to a lack of organization and tracking, she always has at least three unfinished projects on her needles. She also has a tendency to buy yarn on sale, which increases her yarn stockpile and gives her little control over what she has and does not have. She wants to take control of her yarn stash, organize her knitting needles more systematically, and have a clear picture of her current projects so she can track and finish them faster.

Sofie, the Seasonal Knitter
Sofie is an 18-year-old hairdresser living alone in a small studio apartment in Oslo. She is a very artistic person, as seen by her employment as a hairdresser and her home, which she has decorated herself. Sofie's winter interest is knitting big sweaters with colorful designs. She prefers knitting in the winter and frequently puts it away when summer arrives. Every winter, she struggles to resume her projects since she has forgotten crucial elements such as project specifications and where she keeps her patterns.

Sofie wants to improve her knitting experience by finding a digital solution that would allow her to manage her knitting more efficiently. She is looking for an online platform that will allow her to keep track of her knitting projects as the seasons change. Essential features would be the ability to securely store and immediately access her knitting recipes and project specifics, such as yarn type, needle size, and project status. This platform would ideally allow her to continue where she left off, eliminating the need to restart projects.

Sam, the creative-soul and beginner knitter
Sam is a 24-year-old graphic designer living in a shared flat in London. He is a creative soul who enjoys exploring different forms of art. Recently, he has discovered a new interest – knitting. As a beginner, he finds knitting to be a relaxing hobby that he can enjoy in the comfort of his home, especially during the colder months.

However, Sam finds it challenging to keep track of his projects and often struggles to remember how much he has done. He often forgets important details such as the type of stitch he was using, where he left off, or where he stored his needles and yarn. As a beginner, he also struggles with understanding different knitting patterns and often needs help with abbreviations and techniques.

Sam want to have a place to learn and keep up with this new life as a knitter. He is looking for an online platform where he can store information about his knitting projects, including the type of yarn and needles he is using, the stitch patterns, and his progress, as well as having one place for all types useful info. This platform would ideally make his knitting experience more enjoyable and less frustrating, and help him grow as a knitter.

C User stories

We used Miro to create our user-board, and we used MoSCoW method to categories all features into must have (pink), should have (purple), could have (blue), will not have (grey). This is displayed in figure 7, but because it is really small, we have also added them into a list that can be displayed under.

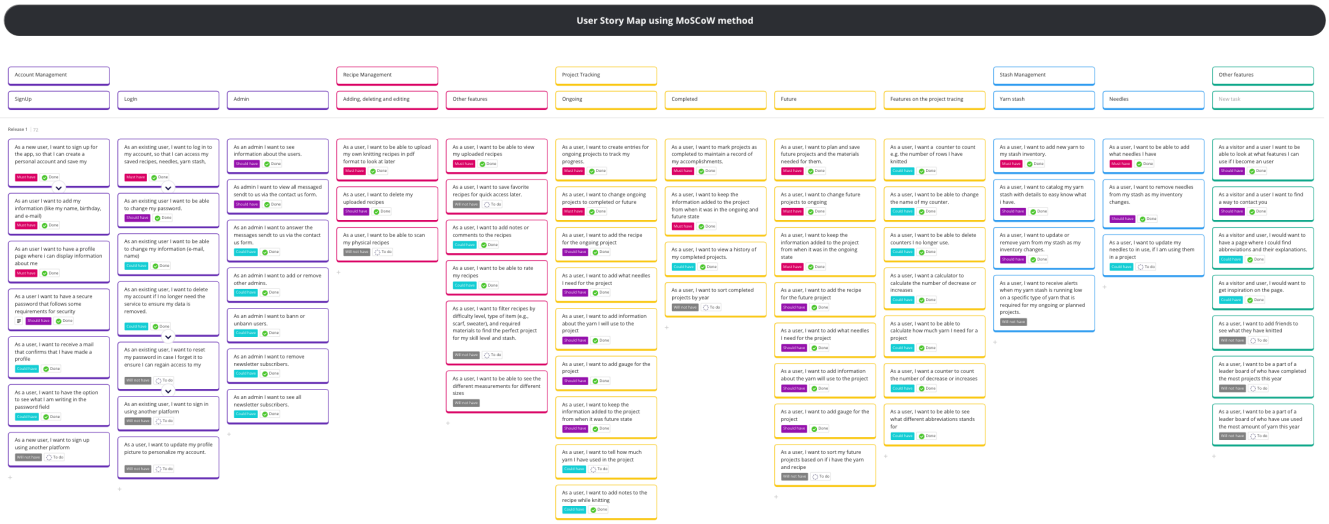

Figure 7: User Story Board

Must Have

- As a new user, I want to sign up for the app, so that I can create a personal account and save my knitting projects and preferences.
- As an user I want to add my information (like my name, birthday, and e-mail)
- As an user I want to have a profile page where i can display information about me
- As an existing user, I want to log in to my account, so that I can access my saved recipes, needles, yarn stash, and project progress.
- As a user, I want to be able to upload my own knitting recipes in PDF format to look at later.
- As a user, I want to be able to view my uploaded recipes.
- As a user, I want to create entries for ongoing projects to track my progress.
- As a user, I want to change ongoing projects to completed or future.
- As a user, I want to mark projects as completed to maintain a record of my accomplishments.
- As a user, I want to keep the information added to the project from when it was in the ongoing and future state.
- As a user, I want to plan and save future projects and the materials needed for them.
- As a user, I want to change ongoing projects to ongoing.
- As a user, I want to keep the information added to the project from when it was in the ongoing state.
- As a user, I want to add new yarn to my stash inventory.
- As a user, I want to be able to add what needles I have.

Should Have

- As a user I want to have a secure password that follows some requirements for security
- As an existing user I want to be able to change my password.
- As an admin, I want to see information about the users.
- As a user, I want to delete my uploaded recipes.
- As a user, I want to add the recipe for the ongoing project

- As a user, I want to add what needles I need for the project
- As a user, I want to add information about the yarn I will juste to the project
- As a user, I want to add gauge for the project.
- As a user, I want to keep the information added to the project from when it was future state.
- As a user, I want to add the recipe for the future project
- As a user, I want to add information about the yarnI will juste to the project
- As a user, I want to catalog my yarn stash with details to easy know what i have.
- As a user, I want to update or remove yarn from my stash as my inventory changes.
- As a user, I want to remove needles from my stash as my inventory changes.
- As a visitor and a user, I want to be able to look at what features I can use if I become a user.
- As a visitor and a user, I want to find a way to contact you.

Could Have

- As a user, I want to receive a mail that confirms that I have made a profile
- As a user, I want to have the option to see what I am writing in the password field.
- As an existing user I want to be able to change my information (e-mail, name)
- As an existing user, I want to delete my account if I no longer need the service to ensure my data is removed.
- As an existing user, I want to delete my account if I no longer need the service to ensure my data is removed.
- As an admin I want to add or remove other admins.
- As an admin I want to bann or unbann users.
- As an admin I want to remove newsletter subscribers.
- As an admin I want to see all newsletter subscribers.
- As a user, I want to save favorite recipes for quick access later.
- As a user, I want to add comments to the recipes
- As a user, I want to be able to rate my recipes
- As a user, I want to view a history of my completed projects.
- As a user, I want to sort completed projects by year
- As a user, I want a counter to count e.g. the number of rows I have knitted
- As a user, I want to be able to change the name of my counter.
- As a user, I want to be able to delete counters I no longer use.
- As a user, I want a calculator to calculate the number of decrease or increases
- As a user, I want to be able to calculate how much yarn I need for a project
- As a user, I want a counter to count the number of decrease or increases
- As a user, I want to be able to see what different abbreviations stands for
- As a user, I want to update my needles to in use, if I am using them in a project
- As a visitor and user, I would want to have a page where I could find abbreviations and their explanations.
- As a visitor and user, I would want to get inspiration on the page.

Will Not Have

- As a new user, I want to sign up using another platform.
- As an existing user, I want to reset my password in case I forget it to ensure I can regain access to my account.
- As an existing user, I want to sign in using another platform
- As a user, I want to update my profile picture to personalize my account.
- As a user, I want to be able to scan my physical recipes.
- As a user, I want to filter recipes by difficulty level, type of item (e.g., scarf, sweater), and required materials to find the perfect project for my skill level and stash.
- As a user, I want to be able to see the different measurements for different sizes.
- As a user, I want to receive alerts when my yarn stash is running low on a specific type of yarn that is required for my ongoing or planned projects.
- As a user, I want to be a part of a leader board of who have completed the most projects this year.
- As a user, I want to be able to calculate how much yarn I need for a project.
- As a user, I want to be a part of a leader board of who have used the most amount of yarn this year.
- As a user, I want to sort my future projects based on if I have the yarn and recipe.
- As a user, I want to add friends to see what they have knitted.
- As a user, I want to be a part of a leader board of who have use used the most amount of yarn this year

D Scenarios

Scenario 1: Sissel's Knitting Organization Overhaul
Sissel, the 45-year-old school secretary from Trondheim, struggles with keeping her knitting hobby organized. Her storage room is filled with yarn and incomplete projects. Sissel has a lot of yarn and unfinished knitting projects, but keeping track of everything is difficult. She buys more yarn than she needs because she forgets what she already has.

To reclaim control, Sissel decided to reorganize her inventory. She begins by physically organizing her yarn by brand, type, and color and storing it in labeled bins. Sissel also uses KnitHub to arrange materials and projects. She records each type of yarn and project details in KnitHub, establishing a profile for each ongoing project that includes current progress comments and next steps.

This improves Sissel's knitting experience. With her physical area and her KitHub profile organized, she can easily start up where she left off on any project, and she can easy find yarn for a new project in her inventory. KnitHub's user-friendly interface enables her to conveniently track her progress and plan new projects.

Scenario 2: Sofie's Seasonal Knitting Solution
Sofie, the Seasonal Knitter, has difficulty remembering all of the the details of her knitting projects when winter arrives. She frequently gets stuck trying to figure out where she left off or where she placed her patterns, which makes knitting less fun.

To make things easier, Sofie decides to use KitHub. Sofie can use this app to add details about each project, such as the yarn she's using, the size she is knitting her sweater in, and how far along she is. When winter arrives, she can open the app and easily see what she needs to do next with each project.

Sofie can now resume her knitting with ease thanks to the app. It helps her stay organized and makes knitting more pleasurable throughout the year.

E Contributions

E.1 Collective

E.2 Individual

F Sprint Backlogs

Under is some screenshots from the backlog, divided into sprints. All the

F.1 Sprint 1: Account Management

Name	Tags	Sprint	Person	Front or back	Stoorypoints	Priority
Create a logout button that askes if you are sure you want to sign out, and remove token	Done	1: Account Management	Emilie	Frontend	2	Medium
Create a table for signup information	Done	1: Account Management	Skjalg A Slubowski, Emilie	Backend	5	HIGH!!
Create a table for login information	Done	1: Account Management	Skjalg A Slubowski, Eline De Vito	Backend	5	HIGH!!
Rewrite the CSS so we use the same colors, fonts, etc. everywhere	Done	1: Account Management	Sera Madeleine Elstad, Marie Alette Stenhaug, Emilie	Frontend	3	Low
Create a page where the user can log in	Done	1: Account Management	Eline De Vito, Sera Madeleine Elstad	Frontend	3	HIGH!!
Create a page where the user can sign up, and toggle between login and sinup	Done	1: Account Management	Sera Madeleine Elstad	Frontend	5	HIGH!!
Make a navbar so the user can navigate to different pages	Done	1: Account Management	Marie Alette Stenhaug	Frontend	3	HIGH!!
distinguish between logged in and not logged in for what shows in the navbar	Done	1: Account Management	Sera Madeleine Elstad, Eline De Vito	Frontend	3	Medium
Make the navbar fold pretty for smal screens	Done	1: Account Management	Marie Alette Stenhaug	Frontend	3	Low

Figure 8: Backlog from sprint 1: Account Management

F.2 Sprint 2: Recipe Management

Name	Tags	Sprint	Person	Front or back	Stoorypoints	Priority
Write tests for uploading recipes	Done	2: Recipe Management	Marie Alette Stenhaug	Backend	3	Low
Create a place where recipes and the given information is stored	Done	2: Recipe Management	Skjalg A Slubowski, Emilie	Backend	5	HIGH!!
Make it possible to view the recipes stored in that persons database	Done	2: Recipe Management	Marie Alette Stenhaug	Frontend	5	Medium
Create a place where the user can upload recipes	Done	2: Recipe Management	Sera Madeleine Elstad	Frontend	5	HIGH!!
Create the basic layout for the about page and information of the website	Done	2: Recipe Management	Sera Madeleine Elstad, Emilie	Frontend	5	Medium
Create the basic layout for the contact-us page	Done	2: Recipe Management	Sera Madeleine Elstad, Emilie	Frontend	3	Low
Setup a profile page where you can display your name and email	Done	2: Recipe Management	Eline De Vito	both	5	Low

Figure 9: Backlog from sprint 2: Recipe Management

F.3 Sprint 3: Project Tracking

Name	Tags	Sprint	Person	Front or back	Stoorypoints	Priority
Create a database that holds information on a project and its state	Done	3: Project Tracking	Skjalg A Slubowski	Backend	5	HIGH!!
Create a yarncalculator	Done	3: Project Tracking	Emilie	Frontend	3	Low
Connect recipe page to ongoing projects	Done	3: Project Tracking	Marie Alette Stenhaug	Backend	2	Medium
Create a page to fill in information on yarn (only frontend)	Done	3: Project Tracking	Eline De Vito	Frontend	3	Low
Create an increase- and decrease calculator	Done	3: Project Tracking	Emilie	Frontend	3	Low
Create a counter for increasing and decreasing etc. (frontend)	Done	3: Project Tracking	Eline De Vito	Frontend	3	Low
Create a page that displays all information on the project along with its state	Done	3: Project Tracking	Marie Alette Stenhaug	Frontend	5	HIGH!!
Create a switch bar where the user can change state of a project	Done	3: Project Tracking	Sera Madeleine Elstad	Frontend	3	Medium
Create a handler for messages that comes in the contact us page	Done	3: Project Tracking	Sera Madeleine Elstad	Backend	3	Low

Figure 10: Backlog from sprint 3: Project Tracking

F.4 Sprint 4: Stash Management

Name	Tags	Sprint	Person	Front or back	Stoorypoints	Priority
Create a verification mail for new users when signing up	Done	4: Stash Management	Emilie, Skjalg A Slubowski	Backend	5	Medium
Create a way for users to delete their account	Done	4: Stash Management	Eline De Vito	both	2	Medium
Create a database for the needles and yarn in stash	Done	4: Stash Management	Skjalg A Slubowski	Backend	5	HIGH!!
Create a verification mail handler in frontend	Done	4: Stash Management	Emilie, Sera Madeleine Elstad	Frontend	2	Medium
Create a personalized home page for logged in users, and one for visitors	Done	4: Stash Management	Emilie, Marie Alette Stenhaug	Frontend	5	Low
Create a way to rate the recipes uploaded	Done	4: Stash Management	Marie Alette Stenhaug	both	2	Low
Create a way to delete needles and yarn in stash	Done	4: Stash Management	Eline De Vito, Sera Madeleine Elstad	both	2	Low
Create a page where users can fill in information on your needles	Done	4: Stash Management	Eline De Vito	Frontend	3	Medium
Divide between yarn and needle stash in frontend	Done	4: Stash Management	Sera Madeleine Elstad, Eline De Vito	Frontend	2	HIGH!!
Create an admin page where admins can view all users, their mail, name, state, messages and subsctibers	Done	4: Stash Management	Sera Madeleine Elstad	both	5	Low

Figure 11: Backlog from sprint 4: Stash Management

F.5 Sprint 5: New Features

Name	Tags	Sprint	Person	Front or back	Stoorypoints	Priority
View PDF in 2 differnt ways	In progress	5: New features	Marie Alette Stenhaug	Frontend	3	Medium
Setup a place to look at messages that is sent to us via the contact form on the webpage	Done	5: New features	Sera Madeleine Elstad, Skjalg A Slubowski	both	3	Low
Change email, password or name for your profile	Done	5: New features	Eline De Vito, Emilie	both	3	Low
Create the layout for the report	Done	5: New features	Sera Madeleine Elstad, Emilie	Report	2	Low
Save the counter for increasing and decreasing etc. in backend	Done	5: New features	Eline De Vito	Backend	3	Low
Set up an overview of subscribers in frontend for admins	Done	5: New features	Sera Madeleine Elstad	Frontend	2	Medium
Set up a display for KnitHubs instagram photos at the website	Done	5: New features	Sera Madeleine Elstad	Frontend	2	Low
Make it possible for users to view the password they are writing in	Done	5: New features	Sera Madeleine Elstad	Frontend	2	Low
Create a database for subscribers on the newsletter	Done	5: New features	Skjalg A Slubowski	Backend	2	HIGH!!
Add a footer so users can navigate easily to contact-us, about-us, and a subscription box for the newsletter	Done	5: New features	Sera Madeleine Elstad, Emilie	Frontend	3	Medium
Be able to answer messages sendt in from the Contact-us form	Done	5: New features	Sera Madeleine Elstad, Skjalg A Slubowski	both	3	Low

Figure 12: Backlog from sprint 5: New Features

F.6 Sprint 6: Report and cleanup

Name	Tags	Sprint	Person	Front or back	Stoorypoints	Priority
Cleanup code	Not started	6: Report and cleanup	Emilie, Marie Alette Stenhaug, Skjalg A Slubowski, Eline De Vito, Sera Madeleine Elstad	both	5	HIGH!!
Finish report	In progress	6: Report and cleanup	Emilie, Marie Alette Stenhaug, Skjalg A Slubowski, Eline De Vito, Sera Madeleine Elstad	Report	25	HIGH!!

Figure 13: Backlog from sprint 6: Report and cleanup

F.7 Unimplemented features from the backlog

Name	Tags	Sprint	Person	Front or back	Stoorypoints	Priority
Make all pages look good also for phone view	Not started				5	
Reset the password without being logged in	Not started			Other	3	Medium

Figure 14: The unimplemented features from the backlog

G Meetings

We have decided to include only the TA and sprint planning meetings. This is because these meetings cover the majority of the topics discussed at our weekly meetings. The figures below show how many meetings were held, what type of meeting they were, when they occurred, and which team members attended each meeting. Eline is the cat image, Skjalg is the person, Sera is the S, Emilie is the E, and Marie is the M.

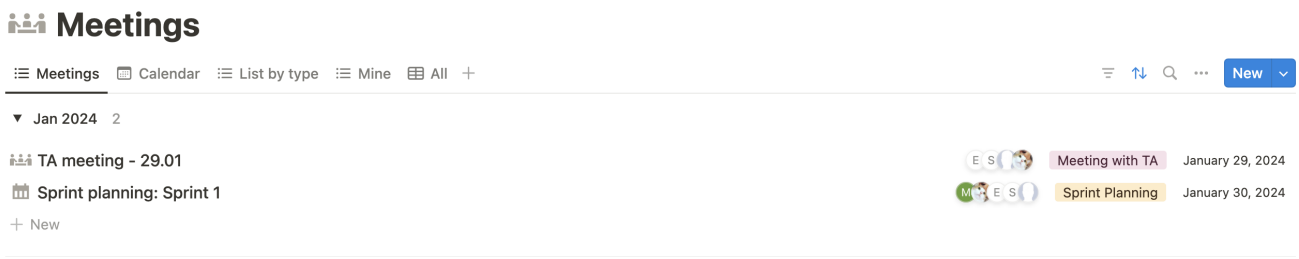

Figure 15: Overview of the meetings in January

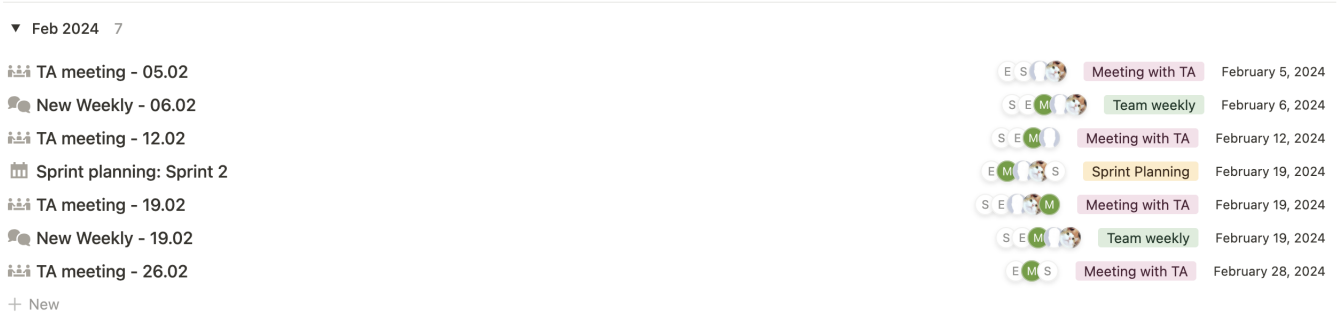

Figure 16: Overview of the meetings in February

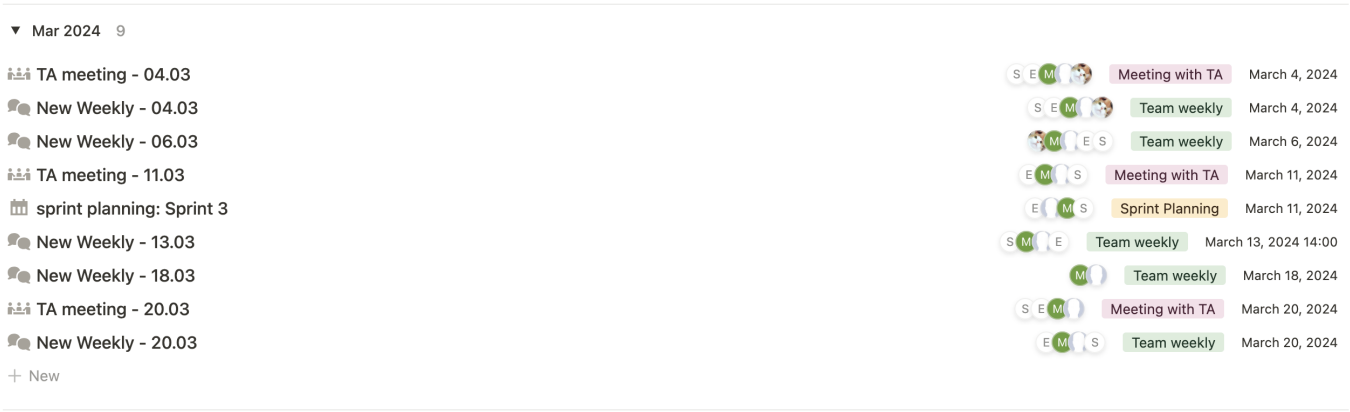

Figure 17: Overview of the meetings in March

Figure 18: Overview of the meetings in Aril

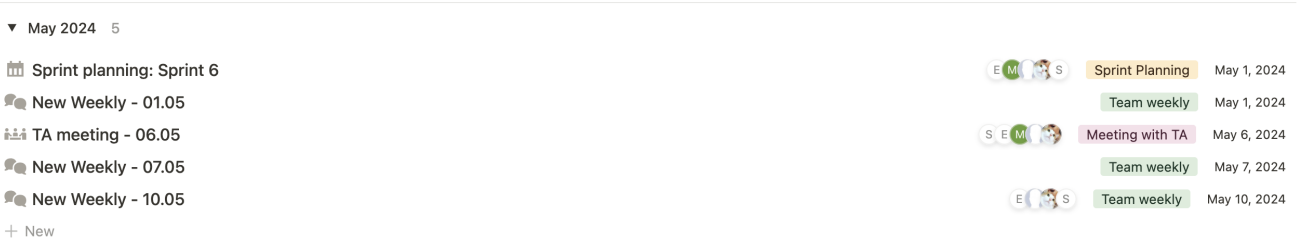

Figure 19: Overview of the meetings in May

TA meeting - 29.01

Attendees	Ⓔ Emilie Ⓢ Sera Madeleine Elstad Skjalg A Slubowski Eline De Vito
Event time	@January 29, 2024
Type	Meeting with TA

Meeting Report

Meet our TA for the project.

We presented our idea for the website and the features we envision - Knitting webpage.

The website should include:

- Homepage
- Login/Signup page
- Verificate profile handling
- A personal profile page
- A recipe page
- Project tracker (done, in progress, upcoming)
- Needle and yarn stash
- + other features within these pages

Discussed Technologies

- **Frontend**: React, HTML.
- **Backend**: .NET with SQLite, or Django.

Sprint planning: Sprint 1

Attendees	Marie Alette Stenhaug Eline De Vito Emilie Sera Madeleine Elstad Skjalg A Slubowski
Event time	@January 30, 2024
Type	Sprint Planning

Sprint Planning 🚀

ACCOUNT MANAGEMENT

Setting up the new sprint: 📆 Tuesday, January 30th - Monday, February 19th.

Goals 🎯

- Ensure that everyone has access to the GitHub, and create one individual branch for each team member.
- Draft a progress plan.
- Go through the User board and agree on the MOSCOW prioritization.
- Set up a work session for next week:
  - Work session next week February 6th.
- Set a date for next session planning meeting.
  - The next sprint planning meeting is February 19th, 10.00-12.00

ToDos for this sprit:

1. Create a table for signup iformation.
2. Create a table for login information.
3. Create a page where the user can log in.
4. Create a page where the user can sign up, and toggle between login and signup.
5. Rewrite the CSS so we use the same colors, fonts, etc. everywhere.
6. Create a navigation bar so the user can navigate between different pages.
7. Create a logout button that askes if you are sure you want to sign out, and remove token
8. Distinguish between logged in user and visitor for what shows on the navigation bar.
9. Make the navigation bar fold pretty for small screens.

📋 User board

We have now created a User Board that is divided into five main features, each with sub-features. The user stories is categorized by the **Must-have**, **Should-have**, **Could-have**, and **Will-not-have** (MOSCOW) criteria.

During this meeting, we collectively reviewed all user stories we had created before the meeting, and determined their priority. Each main feature will represent future sprints, and we will focus on completing the **Must-haves** before moving on to the next sprint.

Additionally, we will consolidate longer user stories as we progress, including persona identification and grouping similar features. We have usea miro.com for the User Board:

https://miro.com/app/board/uXjVN0az3Xc=/?share_link_id=785449674662

📆 Progress Plan

We have initiated a progress plan for our ongoing work. We aim to allocate a minimum of 8 hours per week for coding and project-related tasks.

Agreed Upon Dates/Times for Team Updates 🗓️

- Mondays at 9:00 ⏰
- Wednesdays at 13:30 ⏰

Tasks Assigned to Team Members 👩‍💻👨‍💻

Backlog KnitHub

Aa Name	Sprint	Person	# Stoorypoints
Create a table for login information	1: Account Management	Skjalg A Slubowski Eline De Vito	5
Make the navbar fold pretty for smal screens	1: Account Management	M Marie Alette Stenhaug	3
Create a logout button that askes if you are sure you want to sign out, and remove token	1: Account Management	E Emilie	2

Aa Name	Sprint	Person	# Stoorypoints
Rewrite the CSS so we use the same colors, fonts, etc. everywhere	1: Account Management	Ⓢ Sera Madeleine Elstad M Marie Alette Stenhaug Ⓔ Emilie	3
distinguish between logged in and not logged in for what shows in the navbar	1: Account Management	Ⓢ Sera Madeleine Elstad Eline De Vito	3
Create a page where the user can sign up, and toggle between login and sinup	1: Account Management	Ⓢ Sera Madeleine Elstad	5
Make a navbar so the user can navigate to different pages	1: Account Management	M Marie Alette Stenhaug	3
Create a page where the user can log in	1: Account Management	Eline De Vito Ⓢ Sera Madeleine Elstad	3
Create a table for signup information	1: Account Management	Skjalg A Slubowski Ⓔ Emilie	5

TA meeting - 05.02

Attendees	Ⓔ Emilie Ⓢ Sera Madeleine Elstad Skjalg A Slubowski  Eline De Vito
Event time	@February 5, 2024
Type	Meeting with TA

Meeting Report

🏆 Accomplishments Since Last Meeting

- Agreed upon the project's technologies we want to use:
  - Frontend: React and HTML.
  - Backend: .NET with SQLite.
- Created user stories for the features we want on the website.
- Initiated the implementation of Login and Signup functionality on both frontend and backend.
- First sprint has started.

🔄 Agile Approach

We've embraced Agile principles:

- Each sprint equals 30 points.
- Story points (1-13) will be assigned to user stories via an estimation matrix.
- Aim to start each sprint with approximately 30 story points.

🔜 Upcoming Changes

We'll implement meaningful branching for organized development, e.g., "LoginSignup" for Login and Signup features, instead of one branch for each person on the team.

TA meeting - 12.02

Attendees	Ⓢ Sera Madeleine Elstad Ⓔ Emilie M Marie Alette Stenhaug Skjalg A Slubowski
Event time	@February 12, 2024
Type	Meeting with TA

Meeting Report

🏆 Accomplishments Since Last Meeting

- Integrate developed components and modules to establish connections.
- Integrated navigation bar with sign-up and sign-in functionalities.
- Has created a table for both login and signup information.

🛠️ What We Did

- Discussed our user stories.
- Discussed the backend implementation.

➡️ SOON Upcoming Changes

- We will introduce additional user roles, such as admin, signed-in user, not signed-in user.

🎯 Goals for the Next TA Meeting

- Complete the first code sprint.

TA meeting - 19.02

Attendees	Ⓢ Sera Madeleine Elstad Ⓔ Emilie Skjalg A Slubowski Eline De Vito M Marie Alette Stenhaug
Event time	@February 19, 2024
Type	Meeting with TA

Meeting Report

🏆 Accomplishments Since Last Meeting

- We completed the first code sprint, Account management.
- We have planned our next code sprint, Recipe management.

➡️ SOON Upcoming Changes

- We have to write tests for the frontend and backend code.

🎯 Goals for the Next TA Meeting

- Review unit tests.
- Begin coding the user stories for sprint 2.

Sprint planning: Sprint 2

Attendees	Ⓔ Emilie Ⓜ Marie Alette Stenhaug Skjalg A Slubowski Eline De Vito Ⓢ Sera Madeleine Elstad
Event time	@February 19, 2024
Type	Sprint Planning

Sprint Planning 🚀

RECIPE MANAGEMENT

Setting up a new sprint: 📅 Monday, February 19th - Thursday, March 8th.

Goals 🎯

- Establish a recipe bank 📚
- Each user should have a unique recipe bank 🏦
- Users should be able to categorize PDFs into different folders, stored under their user account 📂

ToDos for this sprint:

1. Create a place in the database where recipes and added information can be stored.
2. Create a page where the user can upload recipes.
3. Make it possible to view the recipes stored in the users database.
4. Write tests for uploading recipes.
5. Create a basic layout for the contact-us page.
6. Create a basic layout for the about page and information of the website.
7. Create a profile page where you can display your name and email.

Also Need to Achieve ✅

- Find a way to stay logged in 🔐

Agreed Upon Dates/Times for Team Updates 🗓️

- Mondays at 9:00 ⏰
- Wednesdays at 13:30 ⏰

Tasks Assigned to Team Members 👩‍💻👨‍💻

Backlog KnitHub

Aa Name	Sprint	Person	# Stoorypoints
Setup a profile page where you can display your name and email	2: Recipe Management	Eline De Vito	5
Write tests for uploading recipes	2: Recipe Management	Marie Alette Stenhaug	3
Create the basic layout for the contact-us page	2: Recipe Management	Sera Madeleine Elstad Emilie	3
Create the basic layout for the about page and information of the website	2: Recipe Management	Sera Madeleine Elstad Emilie	5
Create a place where recipes and the given information is stored	2: Recipe Management	Skjalg A Slubowski Emilie	5
Create a place where the user can upload recipes	2: Recipe Management	Sera Madeleine Elstad	5
Make it possible to view the recipes stored in that persons database	2: Recipe Management	Marie Alette Stenhaug	5

TA meeting - 26.02

Attendees	Ⓔ Emilie M Marie Alette Stenhaug Ⓢ Sera Madeleine Elstad
Event time	@February 28, 2024
Type	Meeting with TA

Meeting Report

🏆 Accomplishments Since Last Meeting

- We have implemented file upload functionality in frontend.
- We have reviewed unit tests.
- We have started the coding on sprint 2.
- Tokenized login for persistent login sessions.

🎯 Goals for the Next Meeting

- Find a way to store recipes in the backend along with its other information.
- Find a way to display the uploaded recipes in the frontend.

TA meeting - 04.03

Attendees	Ⓢ Sera Madeleine Elstad Ⓔ Emilie M Marie Alette Stenhaug Skjalg A Slubowski Eline De Vito
Event time	@March 4, 2024
Type	Meeting with TA

Meeting Report

🏆 Accomplishments Since Last Meeting

- We have fixed our CSS code to ensure consistency across all pages.
- We have started to look at different ways we can view the recipes in PDF format.
- We store the recipes locally.

➡️ SOON Upcoming Changes

- Continue to write tests for the code.

🎯 Goals for the Next Meeting

- Complete the second code sprint.

TA meeting - 11.03

Attendees	Ⓔ Emilie Marie Alette Stenhaug Skjalg A Slubowski Ⓢ Sera Madeleine Elstad
Event time	@March 11, 2024
Type	Meeting with TA

Meeting Report

🏆 Accomplishments Since Last Meetin

- We completed the second code sprint, Recipe management.
- We have some small adjustments on our features we have to make still.

🛠️ What We Did

- We showed how one can upload recipes on the website.
- Showed that users can go to their profile page and view their mail and name.

🎯 Goals for the Next Meeting

- Plan the next code sprint.

sprint planning: Sprint 3

Attendees	Ⓔ Emilie Skjalg A Slubowski Marie Alette Stenhaug Ⓢ Sera Madeleine Elstad
Event time	@March 11, 2024
Type	Sprint Planning

Sprint Planning 🚀

PROJECT TRACKING

Setting up a new sprint: 📅 Monday, March 11th to Friday, March 22nd.

Goals 🎯

- Enable users to track their projects 📝
- Facilitate storing project information 📂
- Implement a yarn calculator to determine yarn requirements for different yarn types 🏀
- Introduce a universal row counter for project progress tracking 🔄

ToDos for this sprint:

1. Create a database that holds information on a project and its state.
2. Create a yarncalculator.
3. Connect recipe page to ongoing projects.
4. Create a page to fill in information on yarn.
5. Create an increase- and decrease calculator.
6. Create a counter for increasing and decreasing etc.

7. Create a page that displays all information on the project along with its state.
8. Create a switch bar where the user can change state of a project.
9. Create a handler for messages that comes in the contact us page.

Also Need to Achieve ✅

- Set up backend functionality to receive messages from the contact us page 📨

Agreed Upon Dates/Times for Team Updates 🗓️

- Mondays at 9:00 ⏰
- Wednesdays at 13:30 ⏰

Tasks Assigned to Team Members 👩‍💻👨‍💻

Backlog KnitHub

Aa Name	Sprint	Person	# Stoorypoints
Create a handler for messages that comes in the contact us page	3: Project Tracking	Ⓢ Sera Madeleine Elstad	3
Connect recipe page to ongoing projects	3: Project Tracking	M Marie Alette Stenhaug	2
Create a yarncalculator	3: Project Tracking	Ⓔ Emilie	3
Create an increase- and decrease calculator	3: Project Tracking	Ⓔ Emilie	3
Create a page to fill in information on yarn (only frontend)	3: Project Tracking	Eline De Vito	3
Create a counter for increasing and decreasing etc. (frontend)	3: Project Tracking	Eline De Vito	3

Aa Name	Sprint	Person	# Stoorypoints
Create a page that displays all information on the project along with its state	3: Project Tracking	M Marie Alette Stenhaug	5
Create a switch bar where the user can change state of a project	3: Project Tracking	Ⓢ Sera Madeleine Elstad	3
Create a database that holds information on a project and its state	3: Project Tracking	Skjalg A Slubowski	5

TA meeting - 20.03

Attendees	Ⓢ Sera Madeleine Elstad Ⓔ Emilie M Marie Alette Stenhaug Skjalg A Slubowski
Event time	@March 20, 2024
Type	Meeting with TA

Meeting Report

🏆 Accomplishments Since Last Meeting

- We have planned and started coding on the thrird sprint, Projact tracking.
- We have fixed some features that remained from the last code sprint.
- Contact us now works.
- The user can now change their password on their profile page (must be logged in).
- The Yarn calculator is complete, just some CSS faults.
- The recipes-cards is complete.
- We have set up the admin page.

SOON Upcoming Changes

- Find a way to answer messages from the user.
- Set up a connection between users/admins and contact-us
    - If you have a user, you get a response, if you are not a user; you get a link to a page where you can answer.

🎯 Goals for the Next Meeting

Sprint planning: Sprint 4

Attendees	Ⓢ Sera Madeleine Elstad Ⓔ Emilie M Marie Alette Stenhaug Skjalg A Slubowski
Event time	@April 3, 2024
Type	Sprint Planning

Sprint Planning 🚀

STASH MANAGEMENT

Setting up a new sprint: 📅 Thursday, April 4th - Wednesday, April 17th.

Goals 🎯

- Create the stash page.
- Make it possible for users to add and remove yarn and needles.
- Set state on needles, if they are in use or not.
- Users can view which needles and yarn they have in their stash.

ToDos for this sprint:

1. Create a verification mail for new users when signing up.
2. Create a verification mail handler in frontend.
3. Create a way for users to delete their account.
4. Create a database for the needles and yarn in stash
5. Create a personalized home page for logged in users, and one for visitors
6. Create a way to rate the recipes uploaded.
7. Create a way to delete needles and yarn in stash.
8. Create a page where users can fill in information on your needles.
9. Divide between yarn and needle stash in frontend.
10. Create an admin page where admins can view all users, their mail, name, state, messages and subsctibers.

Agreed Upon Dates/Times for Team Updates 🗓️

- Mondays at 9:00 ⏰
- Wednesdays at 13:30 ⏰

Tasks Assigned to Team Members 👩‍💻👨‍💻

Backlog KnitHub

Aa Name	Sprint	Person	# Stoorypoints
Divide between yarn and needle stash in frontend	4: Stash Management	Ⓢ Sera Madeleine Elstad Eline De Vito	2
Create a database for the needles and yarn in stash	4: Stash Management	Skjalg A Slubowski	5
Create an admin page where admins can view all users, their mail, name, state, messages and subsctibers	4: Stash Management	Ⓢ Sera Madeleine Elstad	5
Create a page where users can fill in information on your needles	4: Stash Management	Eline De Vito	3
Create a way to delete needles and yarn in stash	4: Stash Management	Eline De Vito Ⓢ Sera Madeleine Elstad	2
Create a way to rate	4: Stash Management	M Marie Alette Stenhaug	2

Aa Name	Sprint	Person	# Stoorypoints
the recipes uploaded			
Create a way for users to delete their account	4: Stash Management	Eline De Vito	2
Create a personalized home page for logged in users, and one for visitors	4: Stash Management	Ⓔ Emilie Marie Alette Stenhaug	5
Create a verification mail handler in frontend	4: Stash Management	Ⓔ Emilie Ⓢ Sera Madeleine Elstad	2
Create a verification mail for new users when signing up	4: Stash Management	Ⓔ Emilie Skjalg A Slubowski	5

TA meeting - 08.04

Attendees	Ⓢ Sera Madeleine ElstadⒺ Emilie Skjalg A Slubowski
Event time	@April 8, 2024
Type	Meeting with TA

Meeting Report

🏆 Accomplishments Since Last Meeting

- We have mostly completet the previous sprint, just has some features we have to complete.
- We have fixed some of the bugs we have found.
- We have planned and started the forth code sprint, Stash management.

➡️ SOON Upcoming Changes

- Create a table for needles and yarn, and be able to delete and edit them in front.
- Fix the merge conflicts we have.
- Set the three claculators into a switch container in the recipes.

🎯 Goals for the Next Meeting

- Make all features work together.

TA meeting - 15.04

Attendees	Ⓔ Emilie Eline De Vito Ⓢ Sera Madeleine Elstad
Event time	@April 15, 2024
Type	Meeting with TA

Meeting Report

🏆 Accomplishments Since Last Meeting

- We have merged mostly all branches with main so all have access to most code.

🎯 Goals for the Next Meeting

- Complete the forth code sprint.
- Have our easter break ❤

Sprint planning: Sprint 5

Attendees	Ⓢ Sera Madeleine Elstad Ⓔ Emilie M Marie Alette Stenhaug Skjalg A Slubowski
Event time	@April 17, 2024
Type	Sprint Planning

Sprint Planning 🚀

NEW FEATURES

Setting up a new sprint: 📅 Wednesday, April 17th - Wednesday, May 1th.

Goals 🎯

- Finish everything so all components work simlessly.
- Get all code up on main.

ToDos for this sprint

1. Setup a place to look at messages that is sent to us via the contact form on the webpage.
2. Change email, password or name for your profile.
3. Create the layout for the report.
4. Save the counter for increasing and decreasing etc. in backend.
5. Set up an overview of subscribers in frontend for admins.
6. View PDF in 2 differnt ways.
7. Set up a display for KnitHubs instagram photos at the website.
8. Make it possible for users to view the password they are writing in.
9. Create a database for subscribers on the newsletter.
10. Add a footer so users can navigate easily to contact-us, about-us, and a subscription box for the newsletter.
11. Be able to answer messages sendt in from the Contact-us form.

Also Need to Achieve ✅

- Write more tests, in both frontend and backend.

Agreed Upon Dates/Times for Team Updates 🗓️

- Mondays at 9:00 ⏰
- Wednesdays at 13:30 ⏰

Tasks Assigned to Team Members 👩‍💻👨‍💻

Backlog KnitHub

Aa Name	Person	Sprint	# Stoorypoints
Setup a place to look at messages that is sent to us via the contact form on the webpage	Ⓢ Sera Madeleine Elstad Skjalg A Slubowski	5: New features	3
Create the layout for the report	Ⓢ Sera Madeleine ElstadⒺ Emilie	5: New features	2
Change email, password or name for your profile	Eline De VitoⒺ Emilie	5: New features	3
Save the counter for increasing and decreasing etc. in backend	Eline De Vito	5: New features	3
Be able to answer messages sendt in from the Contact-us form	Ⓢ Sera Madeleine Elstad Skjalg A Slubowski	5: New features	3

Aa Name	Person	Sprint	# Stoorypoints
Add a footer so users can navigate easily to contact-us, about-us, and a subscription box for the newsletter	Ⓢ Sera Madeleine Elstad Ⓔ Emilie	5: New features	3
Create a database for subscribers on the newsletter	Skjalg A Slubowski	5: New features	2
Make it possible for users to view the password they are writing in	Ⓢ Sera Madeleine Elstad	5: New features	2
Set up a display for KnitHubs instagram photos at the website	Ⓢ Sera Madeleine Elstad	5: New features	2
View PDF in 2 differnt ways	M Marie Alette Stenhaug	5: New features	3
Set up an overview of subscribers in frontend for admins	Ⓢ Sera Madeleine Elstad	5: New features	2

TA meeting - 22.04

Attendees	Ⓔ Emilie M Marie Alette Stenhaug Eline De Vito
Event time	@April 22, 2024
Type	Meeting with TA

Meeting Report

🏆 Accomplishments Since Last Meeting

- We can now differentiate between the different needles in stash.
- The user now get a Verification mail when signing up.
- mostly completed the forth coding sprint, Stash management.

🛠️ What We Did

- We showed our TA the code we have on main.

➡️ SOON Upcoming Changes

- Everyone have to complete any tasks that remains from the fifth code sprint.

🎯 Goals for the Next Meeting

- Write more tests in both frontend and backend.

TA meeting - 29.04

Event time	@April 29, 2024
Type	Meeting with TA

Meeting Report

🏆 Accomplishments Since Last Meeting

- Mostly completed coding sprint five, just have some features that needs to be completed.

🛠️ What We Did

- We went through our TAs report, to see how we would write our.

🎯 Goals for the Next Meeting

- Plan our last sprint for the project.

Sprint planning: Sprint 6

Attendees	Ⓔ Emilie Ⓜ Marie Alette Stenhaug Skjalg A Slubowski Eline De Vito Ⓢ Sera Madeleine Elstad
Event time	@May 1, 2024
Type	Sprint Planning

Sprint Planning 🚀

REPORT AND CLEANUP

Setting up a The last sprint: 📆 Wednesday, May 1th - Wednesday, May 15th.

Goals 🎯

- Create a backlog for the report and write it.
- Cleanup the code.
- Write a script to test the database.
- Hand in project ❤

Also Need to Achieve ✅

- Write the last tests.
- Fix the file structure of the code.
- Make sure all is deleted when a user delete the account.

Agreed Upon Dates/Times for Team Updates 🗓️

- Mondays at 9:00 ⏰
- Wednesdays at 13:30 ⏰

Tasks Assigned to Team Members 👩‍💻👨‍💻

Backlog KnitHub

Aa Name	Sprint	Person	# Stoorypoints
Finish report	6: Report and cleanup	Ⓔ Emilie Ⓜ Marie Alette Stenhaug Skjalg A Slubowski Eline De Vito Ⓢ Sera Madeleine Elstad	25
Cleanup code	6: Report and cleanup	Ⓔ Emilie Ⓜ Marie Alette Stenhaug Skjalg A Slubowski Eline De Vito Ⓢ Sera Madeleine Elstad	5

TA meeting - 06.05

Attendees	Ⓢ Sera Madeleine Elstad Ⓔ Emilie M Marie Alette Stenhaug Skjalg A Slubowski Eline De Vito
Event time	@May 6, 2024
Type	Meeting with TA

Meeting Report

🏆 Accomplishments Since Last Meeting

- We have written many tests for both backend and frontend.
- We have planned and started our last sprint.
- We have started writing our report.
- Most of the code is on main, some of it is missing due to merge conflicts that has not been resolved .

SOON Upcoming Changes

- Fix the merge conflicts.
- Write the report.
- Clenup the code.

🎯 Goals for the Next Meeting

- Finished the code so the complete project is on main.
- Have an almost completed Report.

TA meeting - 13.05

Attendees	Ⓔ Emilie Skjalg A Slubowski Eline De Vito Ⓢ Sera Madeleine Elstad
Event time	@May 13, 2024
Type	Meeting with TA

Meeting Report

🏆 Accomplishments Since Last Meeting

- We have written mostly everything on the report.
- We have commented our code.

➡️ Upcoming Changes

- Complete our report.
- Write the last tests.
- Write our discussion section together for the report.
- Hand in our project 💻