STORY 01 - Creating Users

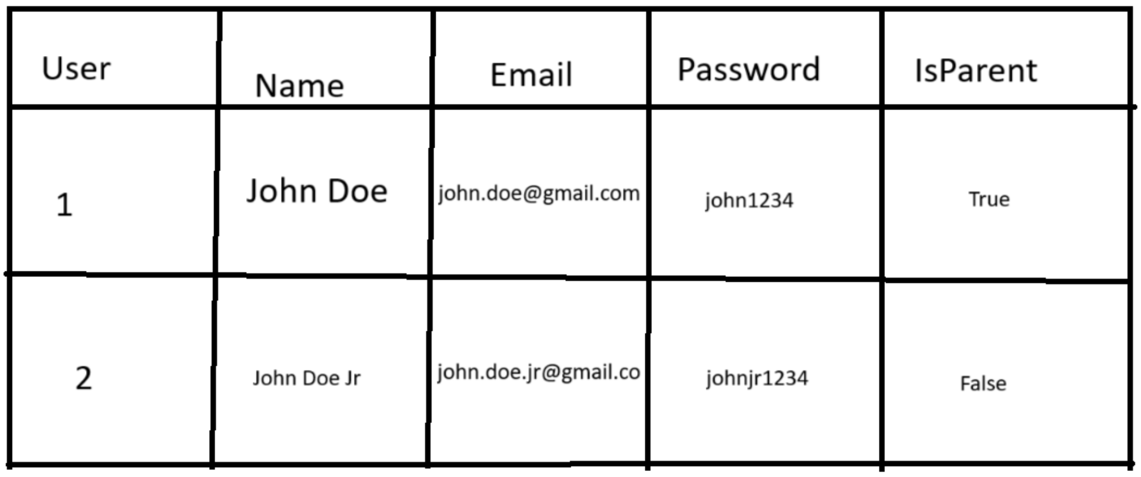
Description

As a person,

I want to register as a user in the application,

So that I can use the app.

Wireframe



User	Name	Email	Password	IsParent
1	John Doe	john.doe@gmail.com	john1234	True
2	John Doe Jr	john.doe.jr@gmail.co	johnjr1234	False

Acceptance Criteria

- A user must have a name, password, email and a role ‘admin’, ‘parent’ or ‘child’.
- “Name” must not be empty
- “Password” must be 8 characters
- “Email” must be a valid format
- “Role” must be ‘admin’, ‘parent’ or ‘child’
- Appropriate error messages must be displayed when these values are incorrect.

STORY 02 - Creating A Family

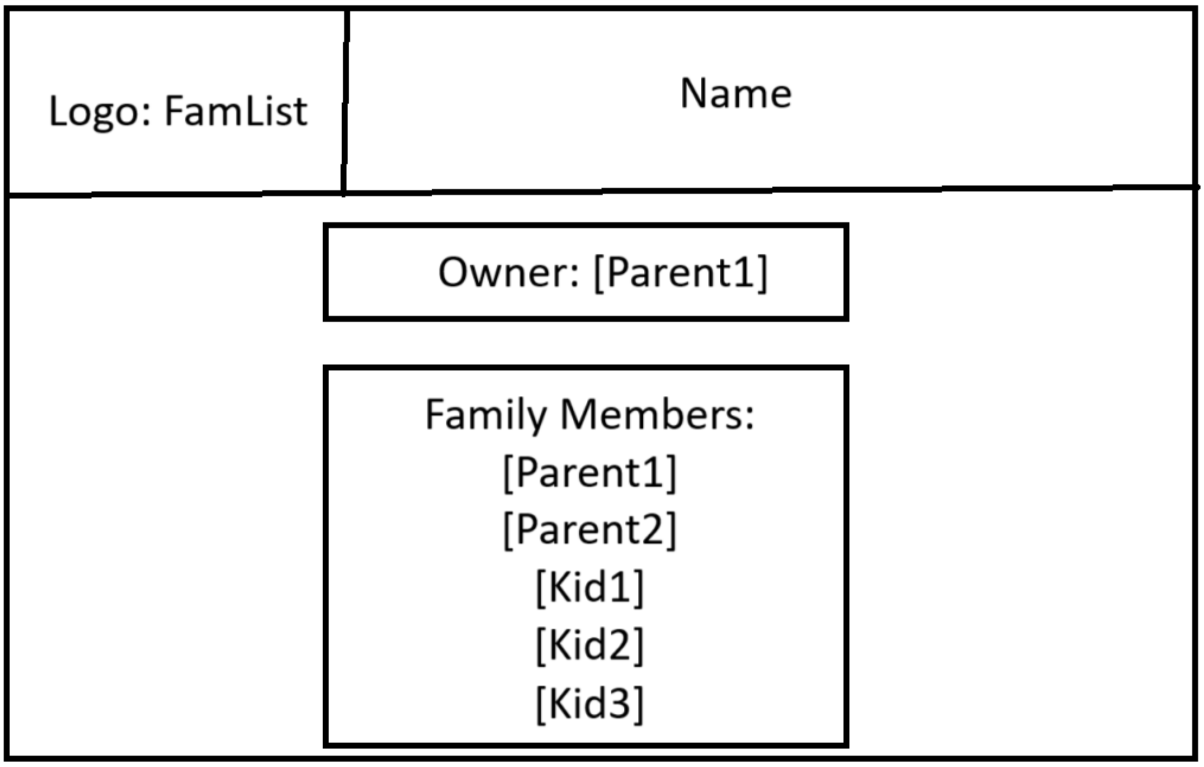
Description

As a parent,

I want to create a family in the application,

So that I have a family.

Wireframe



Acceptance Criteria

- A user can only create a family if they're a parent.
- A family must have a name, a list of family members and an owner.
- Name must not be empty
- Owner must be the person that created the family
- The list of family members must be empty but the owner will be added to the list.
- Only parents can add members to the list.
- Appropriate error messages must be displayed when these values are incorrect.

STORY 03 - Creating Shopping list

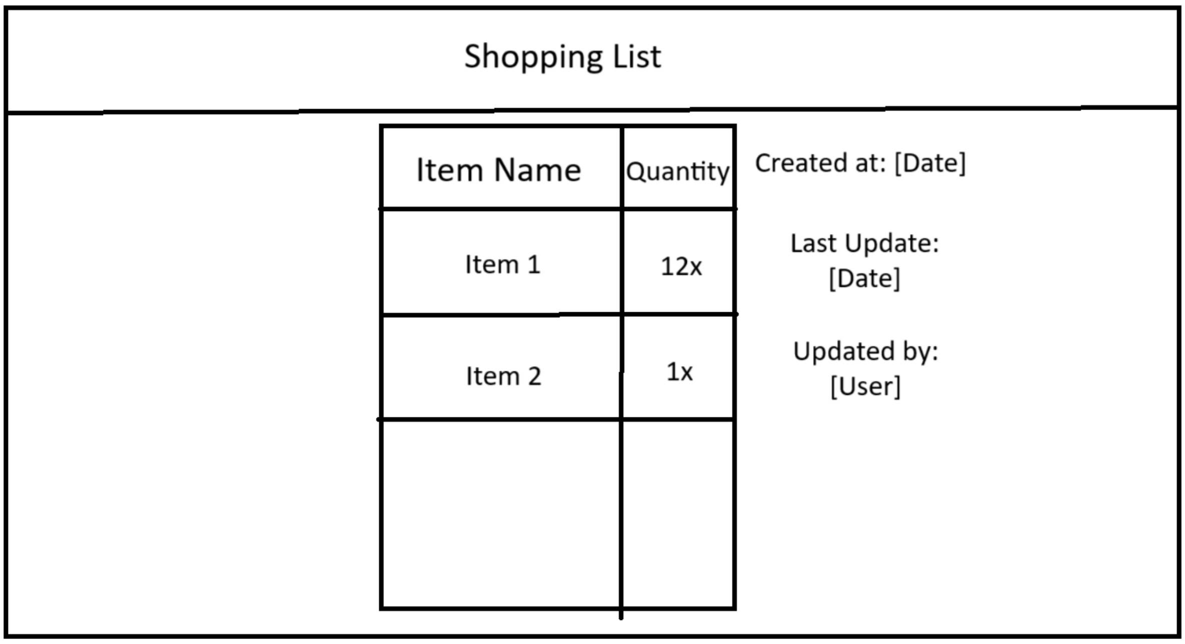
Description

As a family owner,

I want to create a shopping list,

So that everyone in the family can add items to the shopping list.

Wireframe



Acceptance Criteria

- A shopping list has a list of items, a creation date, a date which shows the last update and who updated it.
- The list of items is empty initially
- The creation date is the date of when the shopping list was created.
- When an item is added to the shopping list it shows a date of when the list was last updated and by whom.
- Appropriate error messages must be displayed when these values are incorrect.

STORY 04 - Creating Item

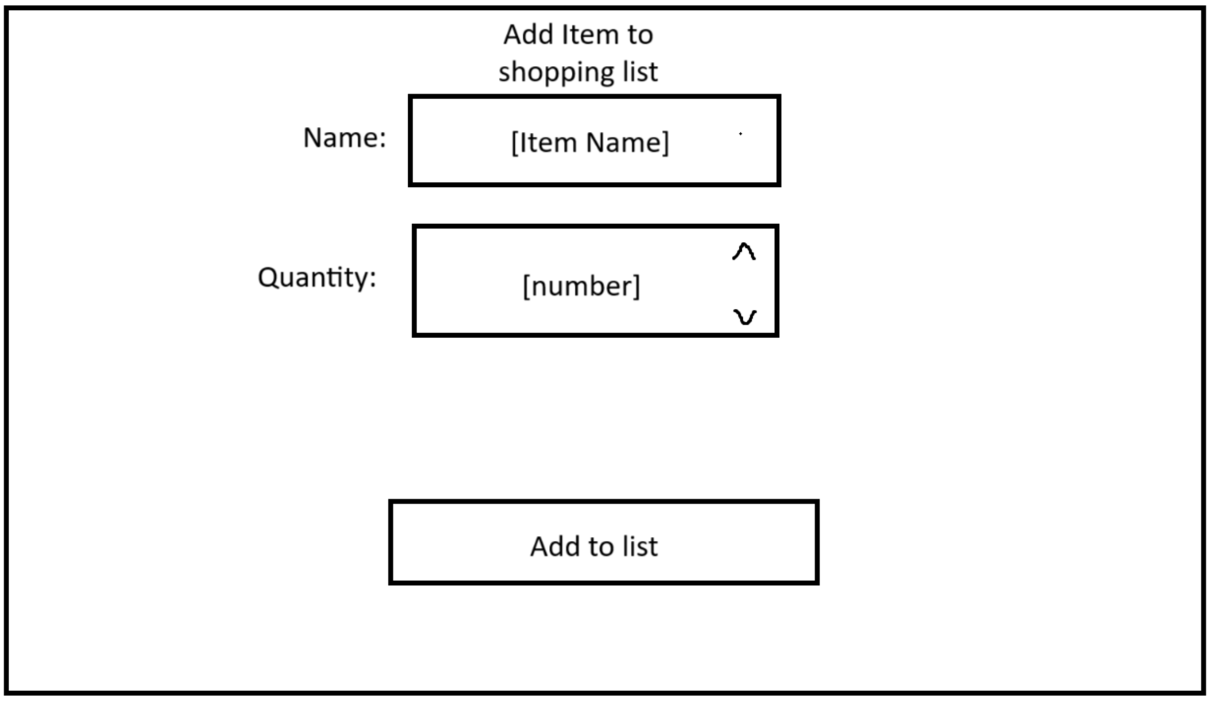
Description

As a family member,

I want to create an item,

So that I can add this item to the shopping list.

Wireframe



Acceptance Criteria

- An item has a name and a quantity
- Name cannot be empty
- Quantity cannot be 0
- Appropriate error messages must be displayed when these values are incorrect.

STORY 05 - Creating A Homepage

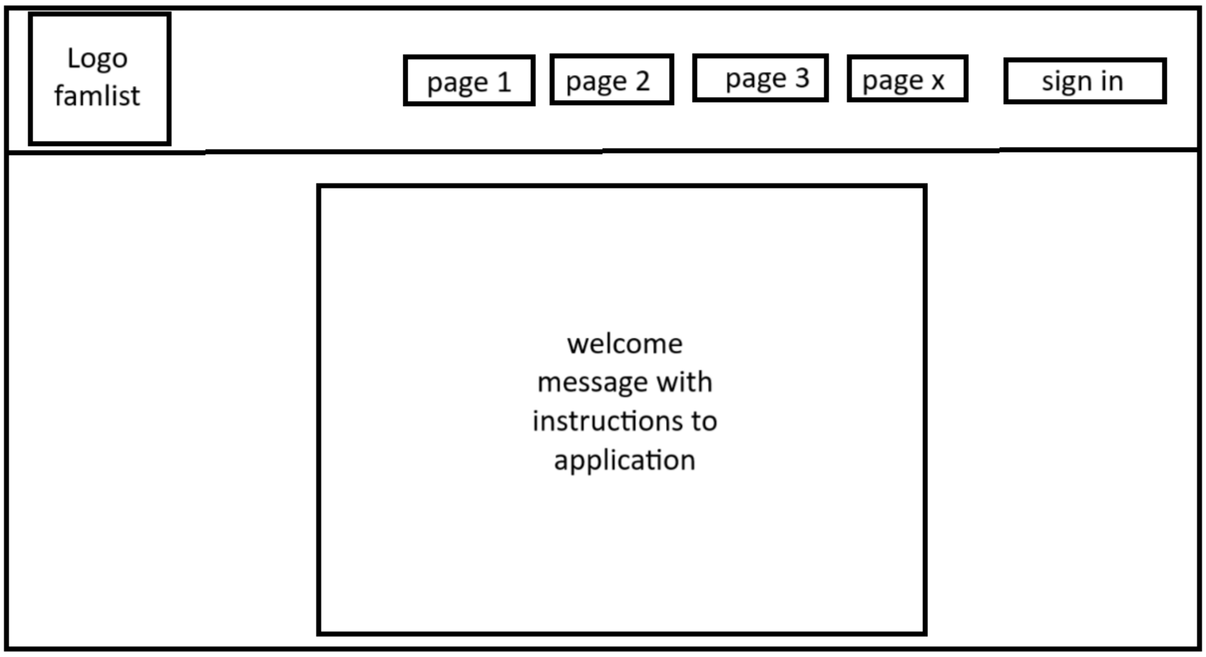
Description

As a user,

I want to access the homepage,

So that I can get an overview of the website’s content and features.

Wireframe



Acceptance Criteria

Frontend:

- The homepage has a navigation bar which displays the other sections of the application including a sign in button and logo.
- The homepage should display a welcome message or brief introduction to the application.

STORY 06 - Getting All Users

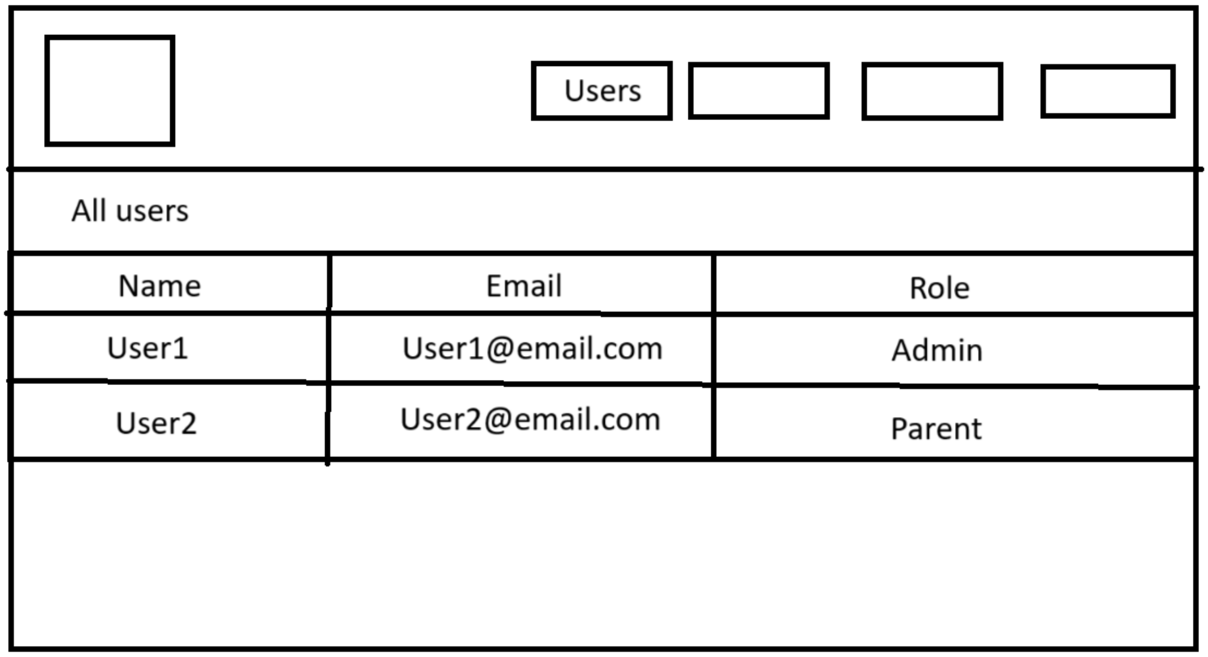
Description

As an administrator,

I want to be able to retrieve all users,

So that I am able to see all the users.

Wireframe



Name	Email	Role
User1	User1@email.com	Admin
User2	User2@email.com	Parent

Acceptance Criteria

Frontend:

- There is a page called ‘Users’ where you’re able to see all the users on the application
- All the users are stored inside of a table

Backend:

- A user.db.ts in the repository folder.
- The repository has a list of users.
- A user.service.ts in the service folder.
- A controller user.routes.ts in the controller folder.
- Add a function which retrieves all the users from the repository.
- All business validation logic happens in the service.
- Document all the requests with swagger.
- Create service tests which test the different functions.
- Appropriate error messages must be displayed when errors occur.

STORY 07 - Creating a user

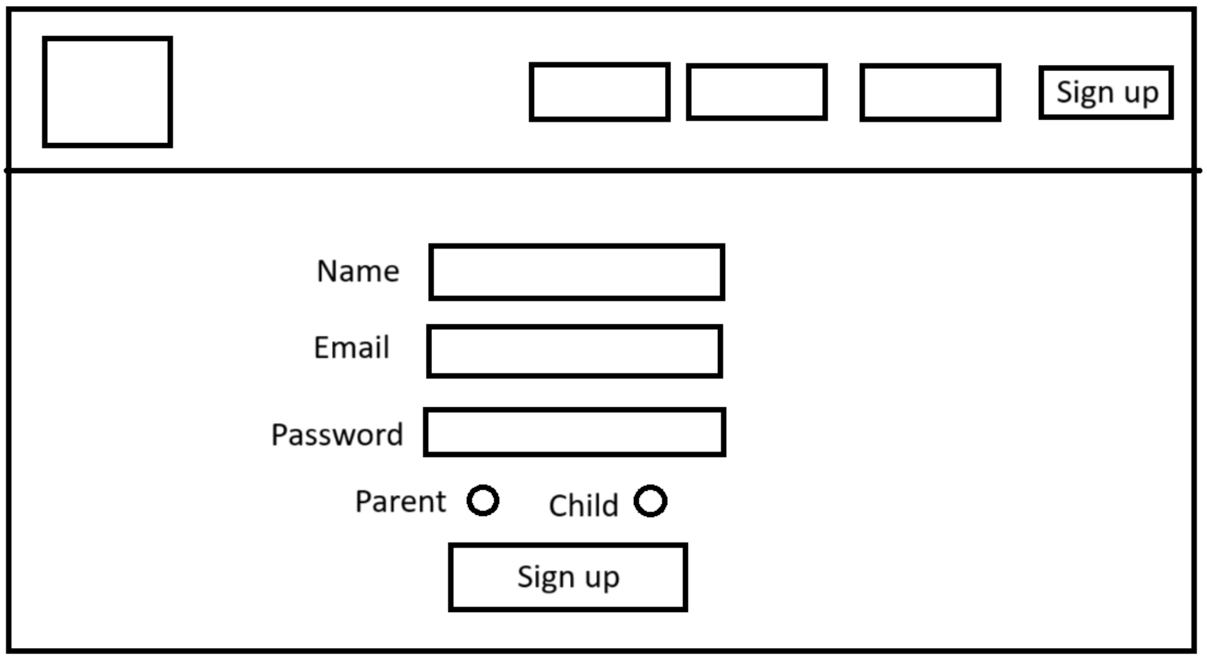
Description

As a user,

I want to be able to become a user,

So that I am able to access the application.

Wireframe



Acceptance Criteria

Frontend:

- There is a sign up page.
- The page has a form for making your user account.
- The fields are:
  - Name
  - Email
  - Password
  - Parent or Child radio button
- All fields are required.
- There is frontend validation for every field.
- We send the information to the backend via a json.

Backend:

- In the controller create a POST request which receives a json of a user as a requestbody.
- The user-object gets transferred from the controller to the service.

- The service checks if the user already exists.
    - If the user exists, an error message is thrown, otherwise it gets pushed to the repository.
- In the repository add a function which adds the user to the list of users.
- All business validation logic happens in the service.
- Document all the requests with swagger.
- Create service tests which test the different functions.
- Appropriate error messages must be displayed when errors occur.

STORY 08 - Getting All Families

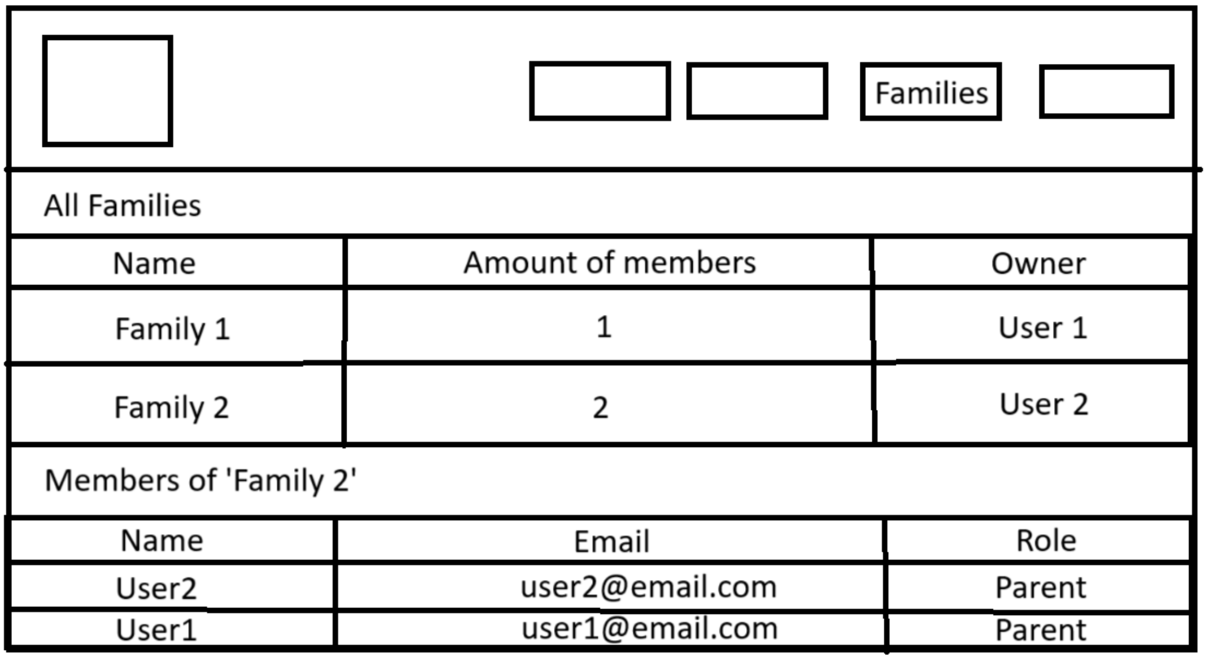
Description

As an administrator,

I want to be able to retrieve all families,

So that I am able to see all the families.

Wireframe



All Families

Name	Amount of members	Owner
Family 1	1	User 1
Family 2	2	User 2

Members of 'Family 2'

Name	Email	Role
User2	user2@email.com	Parent
User1	user1@email.com	Parent

Acceptance Criteria

Frontend:

- There is a page called ‘Families’
- When a clicking on it, all families will be shown in a table.
- When clicking on a family in the table, you’ll be able to see which family members are in there.

Backend:

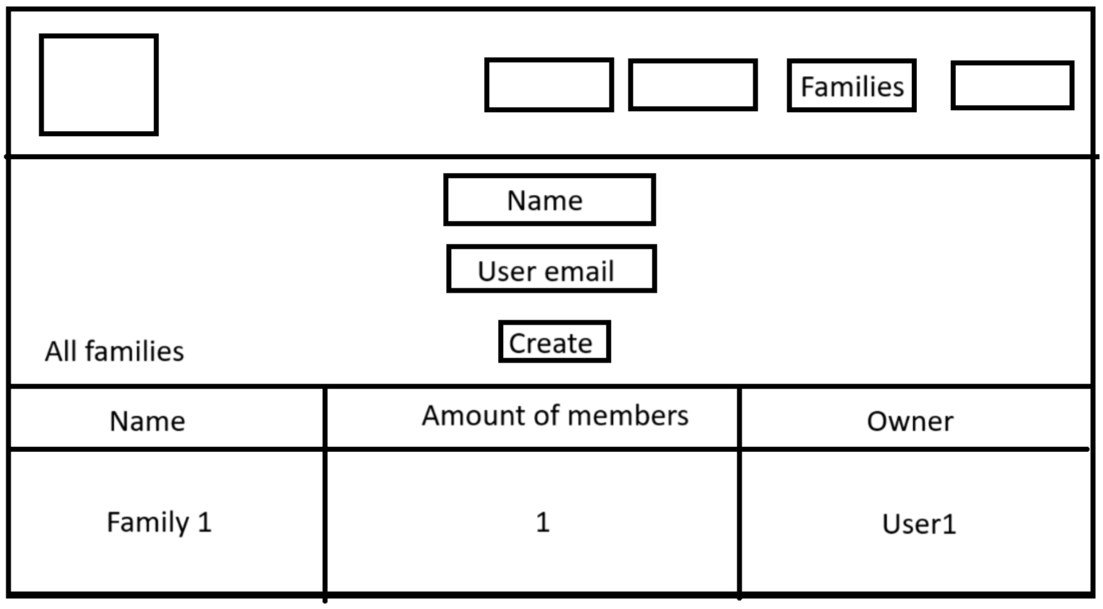
- A family.db.ts in the repository folder.
- The repository has a list of families.
- A family.service.ts in the service folder.
- A controller family.routes.ts in the controller folder.
- Add a function which retrieves all the families from the repository.
- All business validation logic happens in the service.
- Document all the requests with swagger.
- Create service tests which test the different functions.
- Appropriate error messages must be displayed when errors occur.

STORY 09 – Creating a family

Description

As a user,
I want to be able to create a family on the platform,
So that I can have a family on the platform.

Wireframe



Acceptance Criteria

Frontend:

- On the family page, there is a 'Create family' button.
- When clicking on it, it will ask you for a name and a user email.
- The name and the user email will be sent to the backend. A user with this email must exist in the database, it can't be a random email address.
- All fields are required to create a family.

Backend:

- In the controller there is a post function which receives the name and the user email and sends it to the service.
- In the service, the user gets fetched by the email and a new family object is created. The family object is validated and then it's sent to the repository.
- In the repository, the family is added to the list of different families.
- A user can create multiple families and a family can have multiple users.
- All business validation logic happens in the service.
- Document all the requests with swagger.
- Create service tests which test the different functions.
- Appropriate error messages must be displayed when errors occur.

STORY 10 - Database Implementation

Description

As an admin,
I want to have a database for my different class models,
So that I can access my data through a database.

Wireframe

/

Acceptance Criteria

Backend:

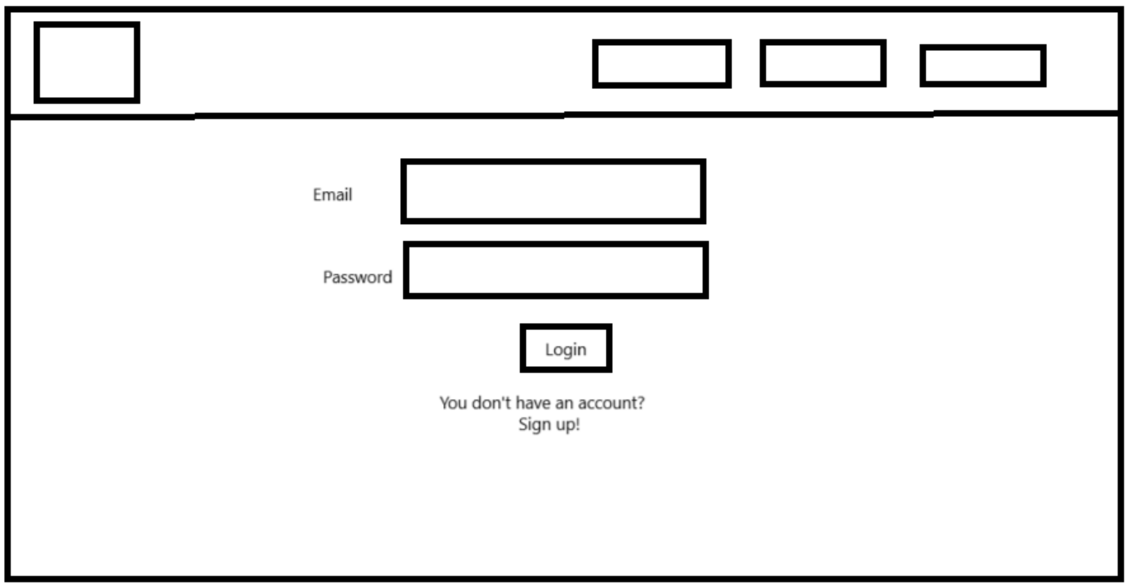
- The repository arrays are replaced by a database
- Create a schema.prisma in the repository folder
- Create table models for user, family, item and shoppingList and make sure the relations are correct.
- Create a seed.ts file with example data.
- Make sure every service function works.

Story 11 – Login Page 2

Description:

As a user,
I want to be able to sign in or log in,
So that I am able to access the platform.

Wireframe:



Acceptance Criteria:

Frontend:

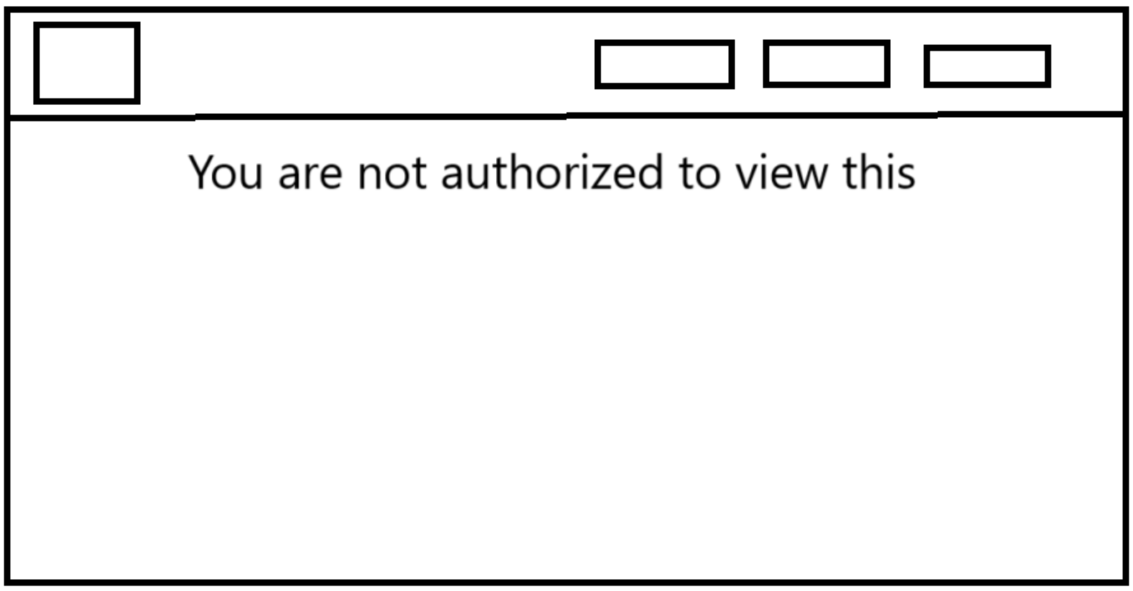
- When clicking on the log in button you're provided with the login form: email, password, a 'Log in' submit button and a small button under it saying ''You don't have an account? Sign up!" which leads you to the sign up form.
- When signing/logging in, there's a status message saying 'Successfully logged in! Redirecting you to the home page!' and after 2 seconds it redirects you to the homepage.
- If the user is trying to log in with a non-existent email or a wrong password, there's an error message that pops up.
- The user is stored inside of the session storage as a json.
- When the user is logged in, in the navigation bar there's a welcome message with the name of the user.

Story 12 – Backend and Frontend Security

Description:

As an administrator,
I want to have different roles that have different authorizations on the website,
So that not everyone has access to everything.

Wireframe:



Acceptance Criteria:

Frontend:

- When opening the site and you're not logged in, you're only able to choose to login or register.
- Different roles are able to view the site in a different matter -> Backend section for more explanation.
- JWT token and user are saved in the session storage.

Backend:

- Passwords of users are encrypted with Bcrypt.
- JWT token authentication for different routes with Swagger.
- No authentication needed for Swagger documentation, status and login/registration.
- Roles:
  - Child:
    - Able to view their own families.
    - Able to only add items to shopping list.
    - Able to leave the family.

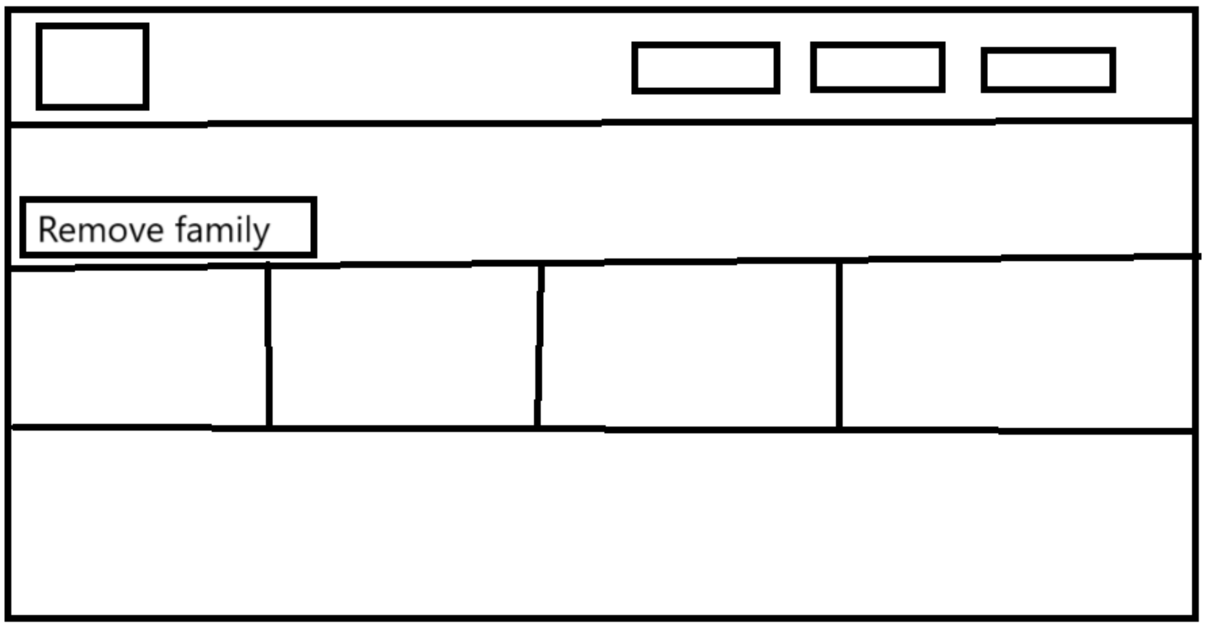
- Parent:
    - Able to view their own families.
    - Able to create shopping lists.
    - Able to add items to shopping list.
    - Create new families.
    - Able to add or remove family members.
    - Able to leave the family. But since the parent is the owner, the family gets removed as well.
- Admin:
    - Can view all members on the site.
    - Can create families and assign a leader to them, including themselves.
    - Can add or remove family members from all different families.
    - Can add or remove shopping lists from all different families.

STORY 13 - Deleting a family

Description

As a user,
I want to be able to delete my family,
so that I can remove the family from the application.

Wireframe



Acceptance Criteria

Frontend:

- When selecting a certain family additionally to all the family members it also shows you a red button that says ‘Remove family’.
- When interacting with the button, a prompt pops up asking the user if they’re sure they want to delete the family and that this option cannot be undone.
- The family id gets sent to the backend.

Backend:

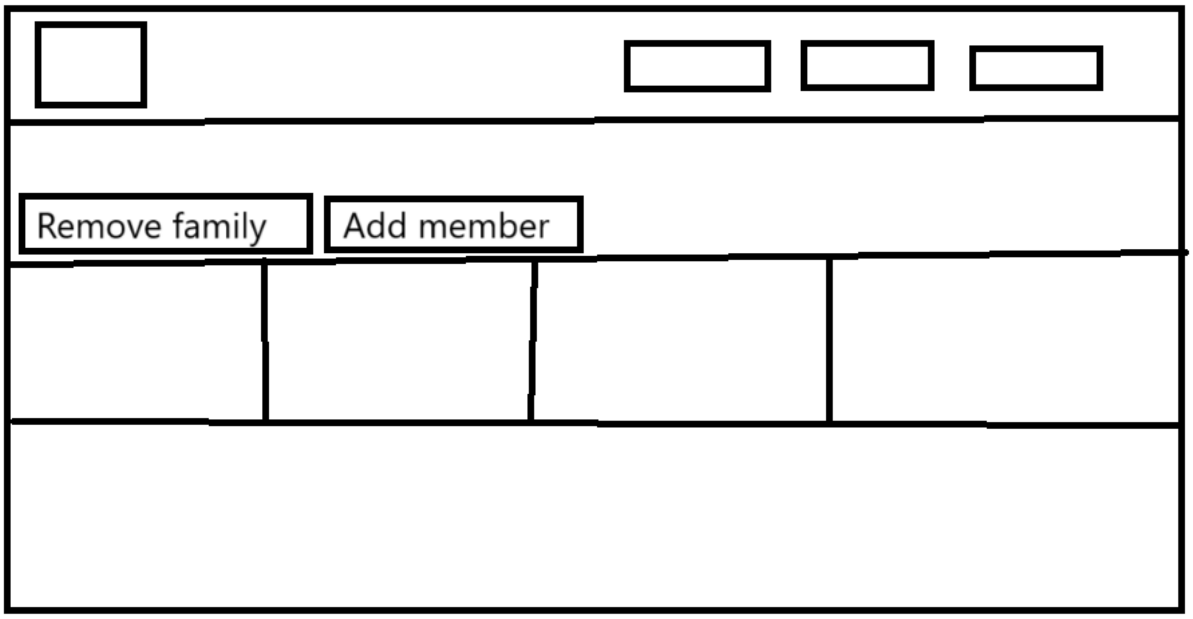
- The family object gets an id which will be used to distinguish each family from each other.
- In the family controller a delete request is called and given the Family ID as path variable, and then the service is called.
- The service checks if there’s a family with that certain ID, if it doesn’t exist, an error gets throws, otherwise the repository looks for the family and deletes it.
- All business validation logic happens in the service.
- Document all the requests with swagger.
- Create service tests which test the different functions.
- Appropriate error messages must be displayed when errors occur.

STORY 14 - Adding a user to a family

Description

As a family owner,
I want to be able to add family members to my family,
So that I can have more family members inside of my family.

Wireframe



Acceptance Criteria

Frontend:

- When selecting a certain family, there's a button that says 'Add a family member'.
- When clicking on the button, you're asked to provide an email address of the member you want to add to the family.
- The email must be a valid email address and an address that's also in the database.
- The email and the family id are then sent to the backend.

Backend:

- A new put request is created in the family controller which takes the family id and the user email as path variables and then it sends them to the service.
- In the family service, the id is checked if there's a family that exists with that id and the email is checked to see if there's an user that exists with that email.
- In the family repository the user object is then added to the list of family members of the family object.
- All business validation logic happens in the service.
- Document all the requests with swagger.
- Create service tests which test the different functions.

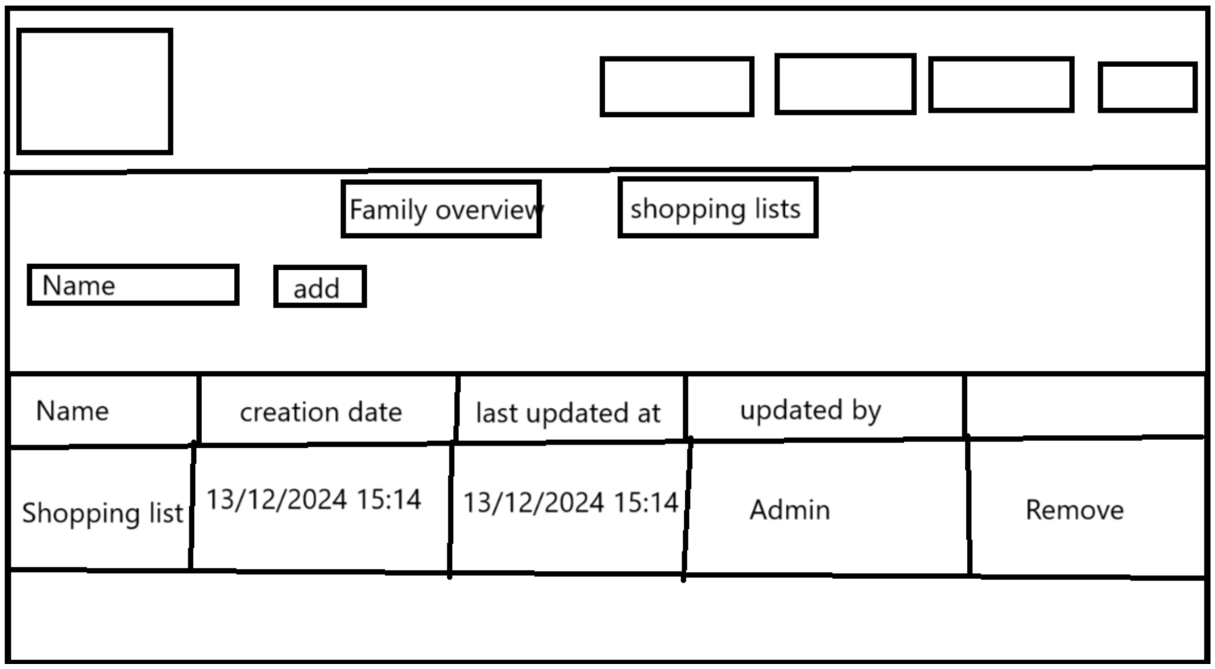
- Appropriate error messages must be displayed when these values are incorrect.

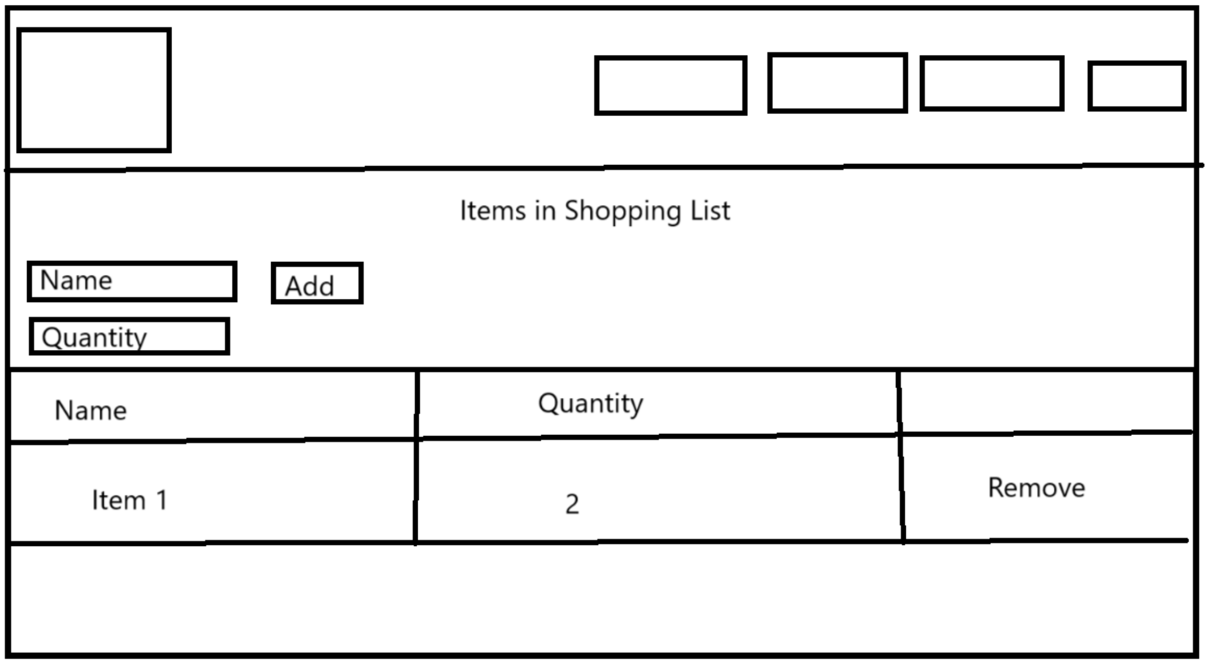
STORY 15 - Creating a shopping list in a family

Description

Wireframe

Acceptance Criteria





Frontend:

- Changes to family page: When clicking on a family, it opens up a new window.
- There are 2 tabs: Family Overview and Shopping Lists
- Family Overview:

    - Overview page with all family members.
    - The buttons to add/remove a member are there.
- Shopping Lists:
    - You’re able to see all the different shopping lists
    - On that page there is also a button to add a new shopping list.
    - When clicking on a shopping list, you’re able to view all the items and also add a new item or remove it.

Backend:

- A new HTTP / GET request to get all the shopping lists for a certain family.
- A new HTTP / GET request to get all the items of a certain shopping list.
- A new HTTP / POST request to create a new shopping list.
- A new HTTP / POST request to create a new item and add it to a shopping list.
- A new HTTP / DELETE request to delete a shopping list and all of its associated items.
- A new HTTP / DELETE request to delete a certain item from a certain shopping list.
- Appropriate error messages must be displayed when these values are incorrect.
- All new functions must be documented and authorized with Swagger.

Story 16 – Translation

Description:

As a user,
I want to access the website in different languages,
So that I can view the website in my native language.

Wireframe:

/

Acceptance criteria:

Frontend:

- I18n: Translate the entire website to:
    - EN
    - NL

Story 17 – HTTP protection Helmet

Description:

As an administrator,
I want to protect my HTTP traffic using Helmet,
So that potential hackers don't find loopholes in my application.

Wireframe:

/

Acceptance criteria:

Backend:

- Import helmet to the app.ts

Story 18 – React testing

Description:

As an admin,
I want to test my components,
So that I know they work correctly.

Wireframe:

/

Acceptance criteria:

Frontend:

- Test 2 components of your choice using the React Testing Library.

Story [X] (Optional) – Tailwind CSS

Description:

As a admin,
I want my website to be converted to Tailwind CSS,
So that my css work becomes easier.

Wireframe:

/

Acceptance Criteria:

Frontend:

- All of the CSS in the frontend is converted to Tailwind CSS.
- Improvements are also made for the CSS used.