STORY 01 - Creating Users

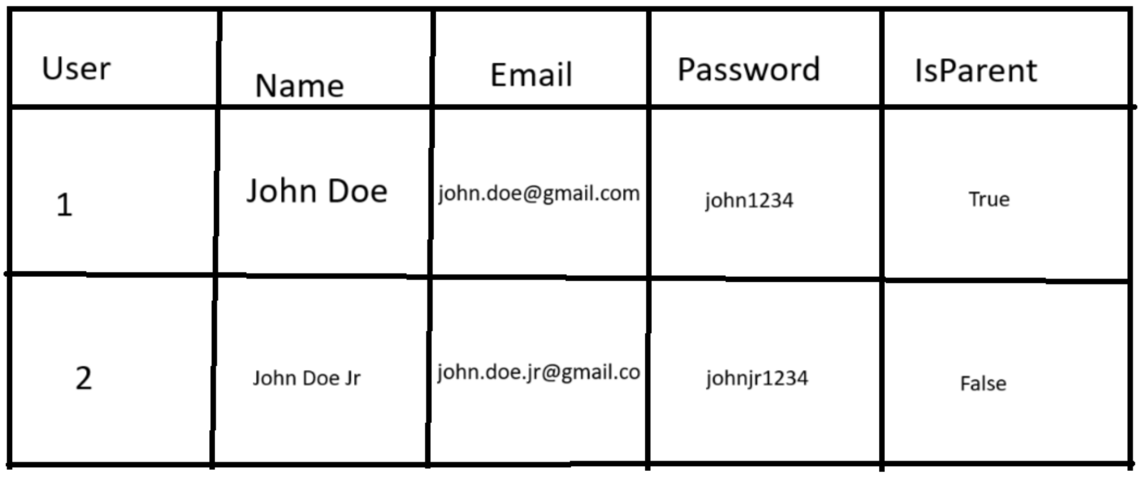
Description

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 indicate if they are a parent or not.
- "Name" must not be empty
- "Password" must be 8 characters
- "Email" must be a valid format
- "IsParent" must be true or false
- Appropriate error messages must be displayed when these values are incorrect.

STORY 02 - Creating A Family

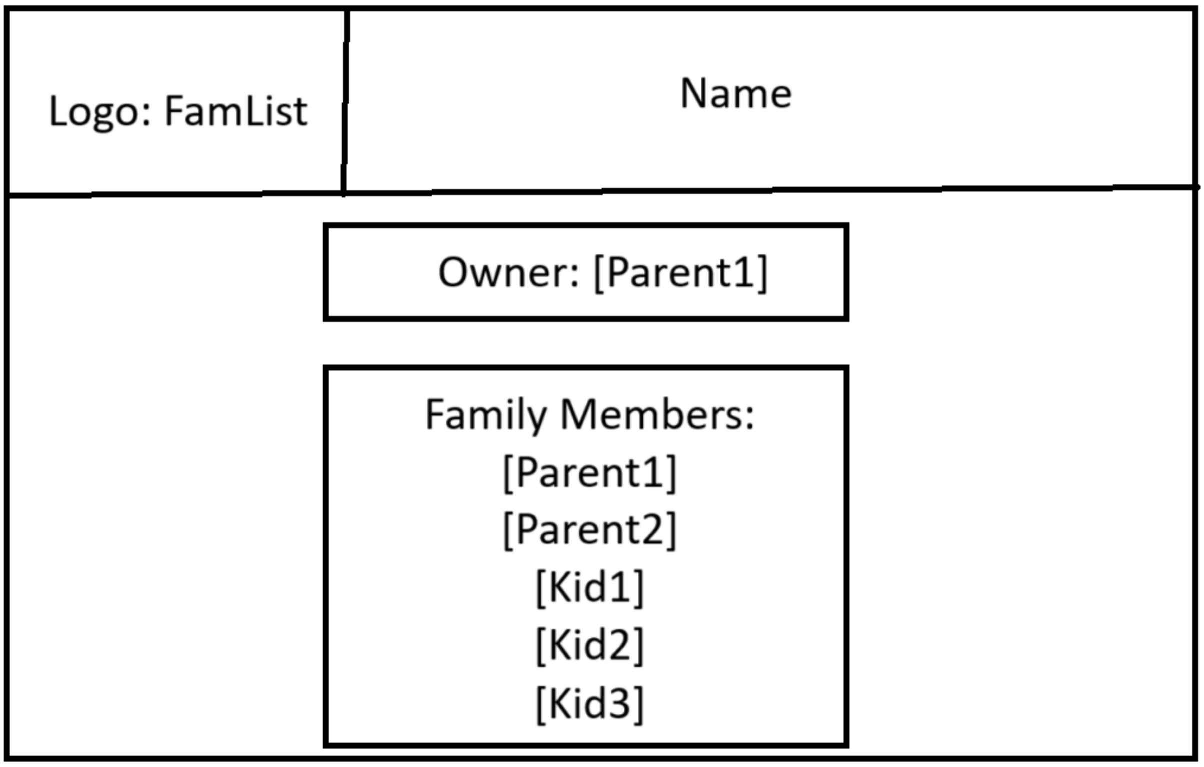
Description

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

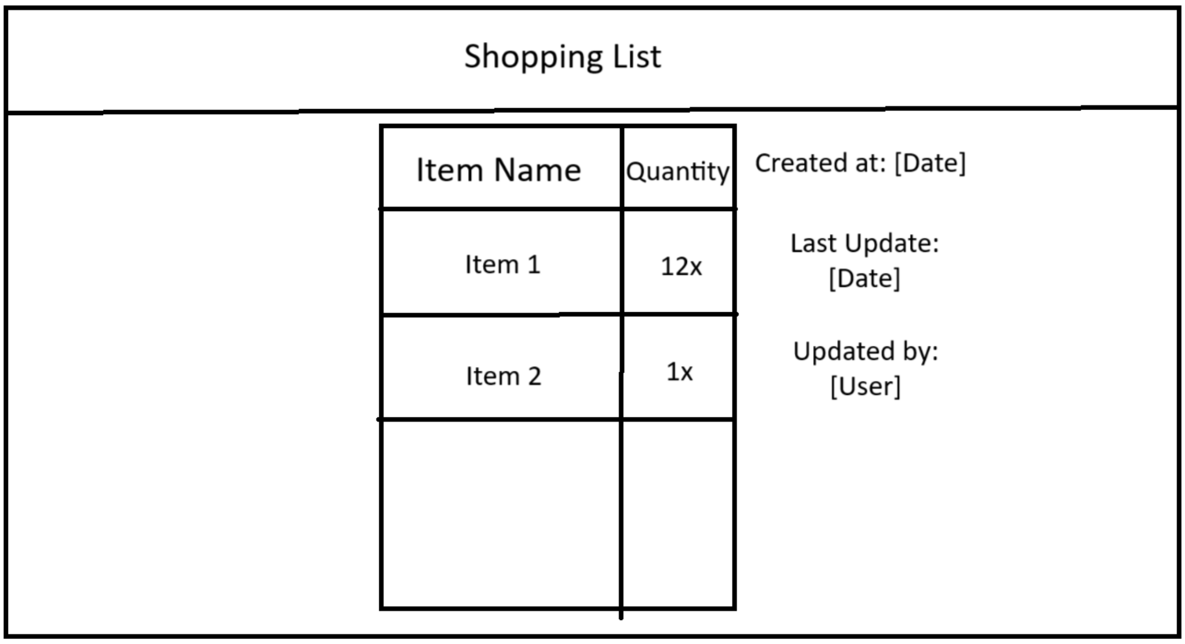
Description

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
- When adding the same item twice the quantity of the item is increased instead of adding the same item twice.
- 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

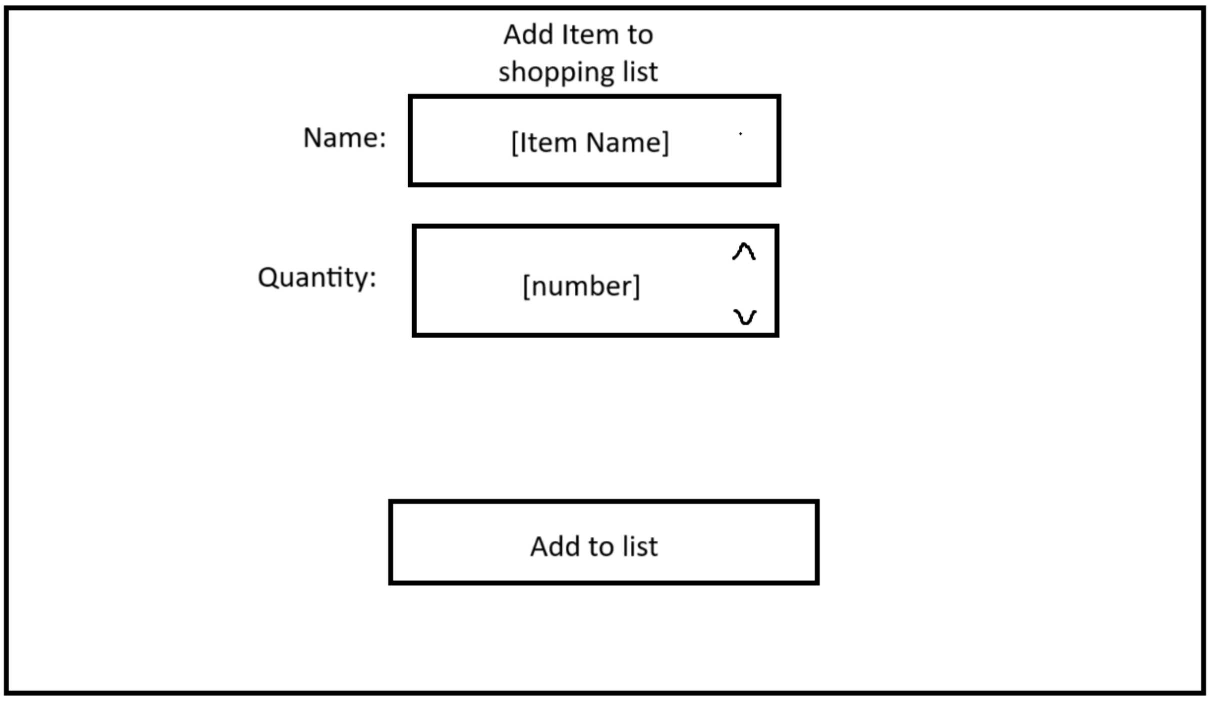
Description

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
- When an item with the same name is added to the same shopping list, the quantity goes up by the quantity of the new item.
- Appropriate error messages must be displayed when these values are incorrect.

STORY 05 - Creating A Homepage

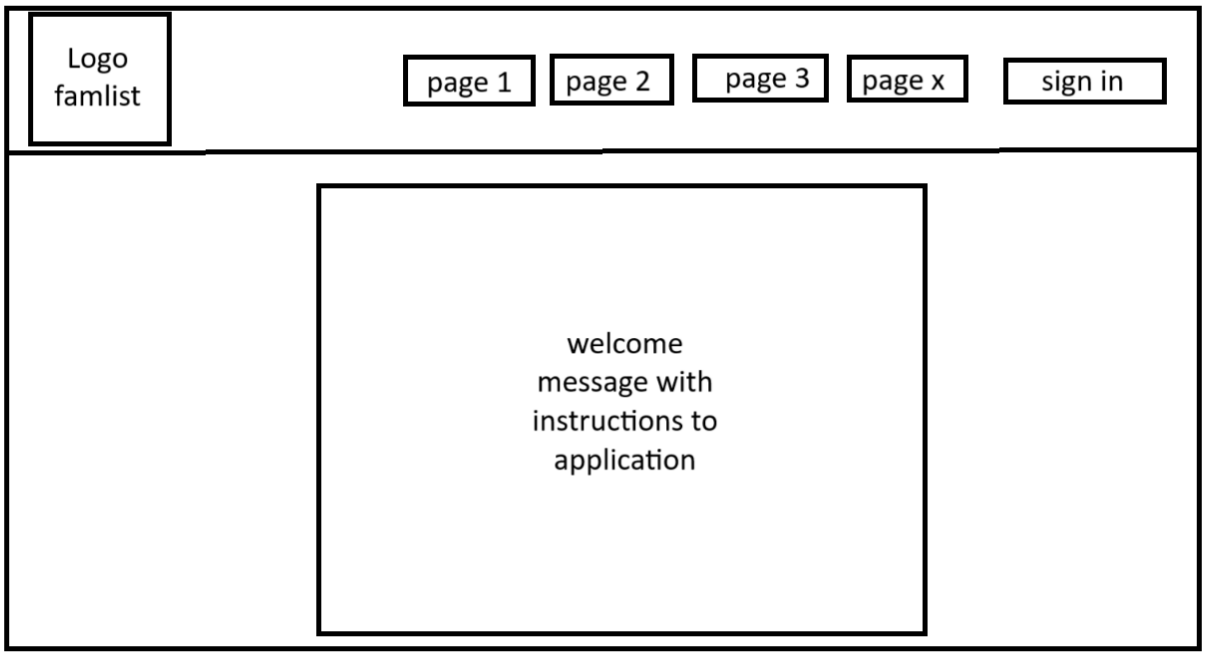
Description

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

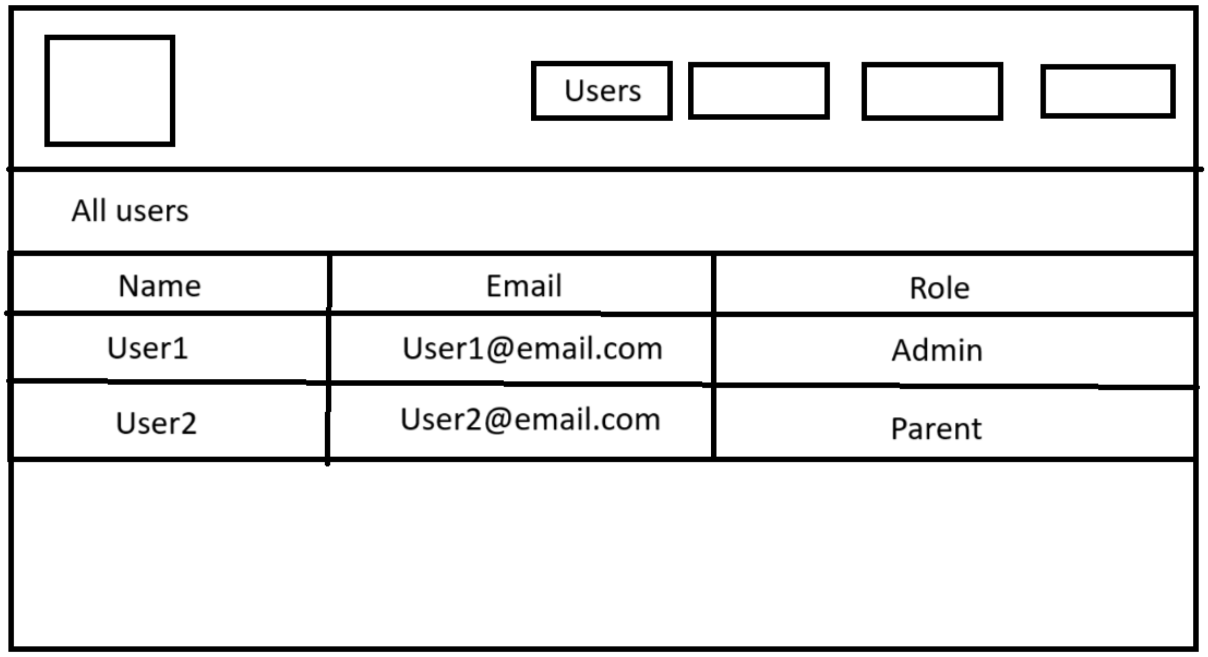
Description

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

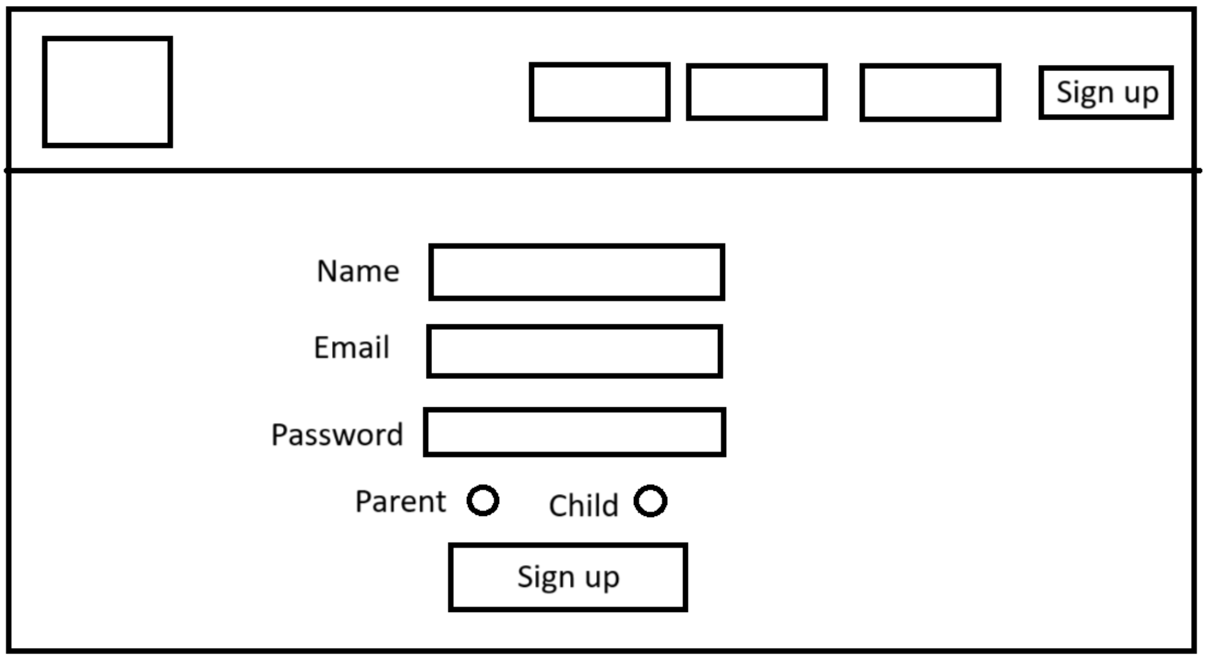
Description

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

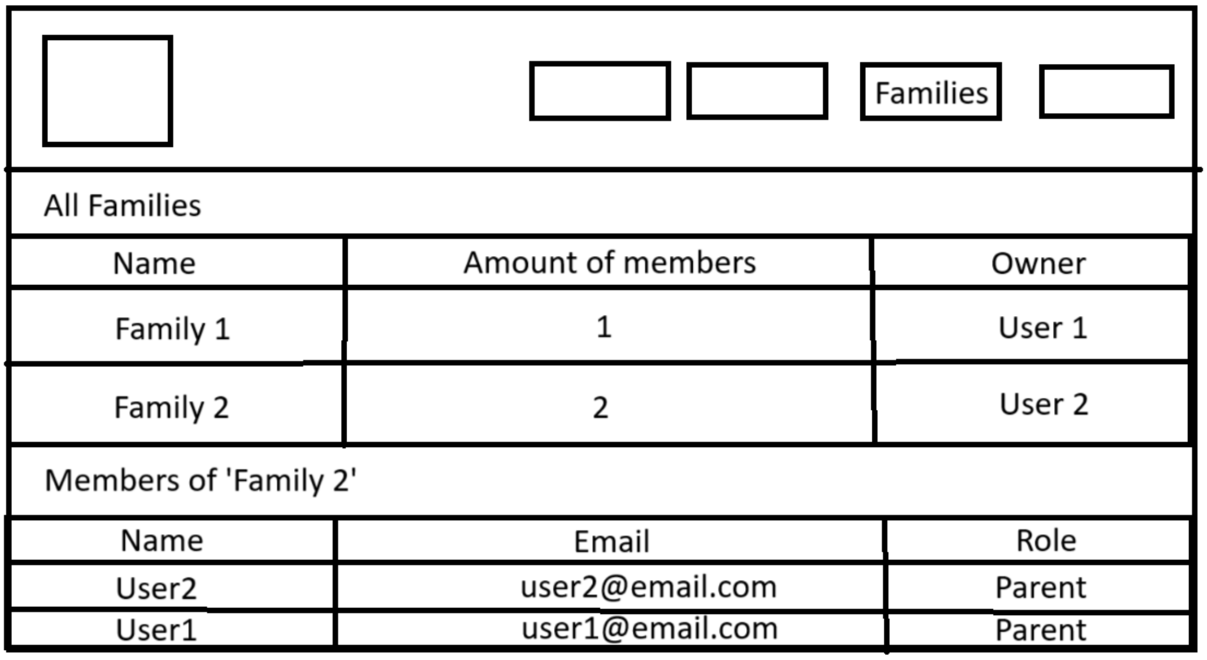
Description

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:

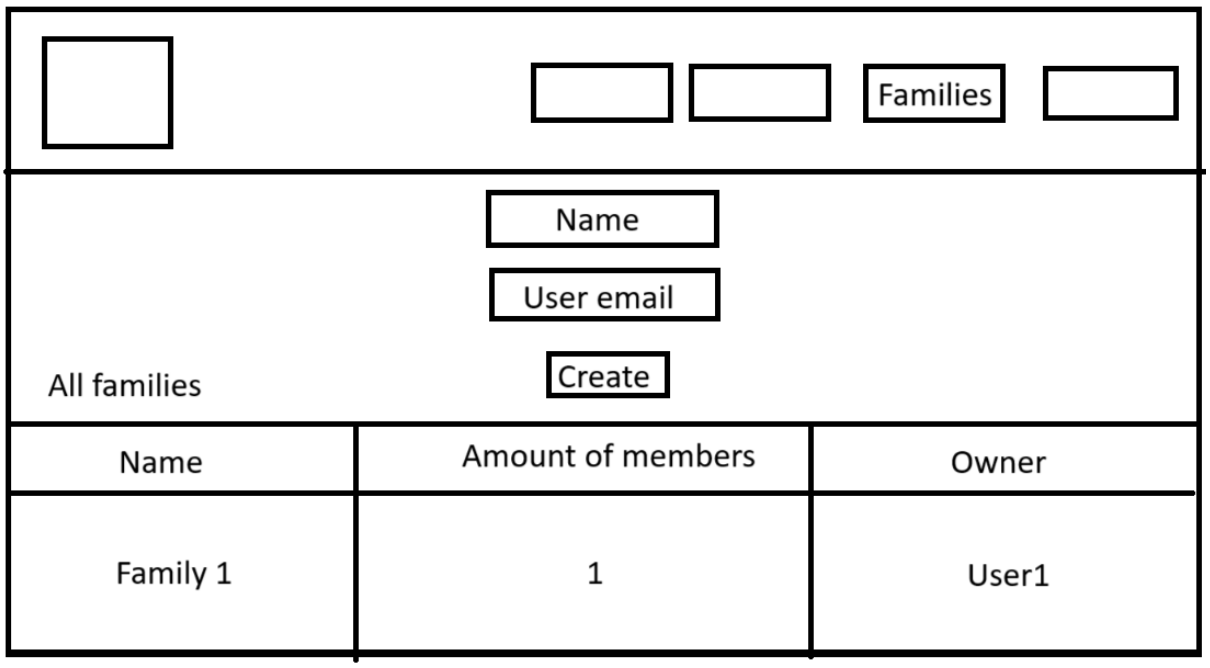
- A family.db.ts in the repository folder.
- The repository has a list of families.
- A family.service.ts in the service folder.
- A controller family.routes.ts in the controller folder.
- Add a function which retrieves all the families from the repository.
- All business validation logic happens in 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.
- 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.