Project Report

Group 6

Introduction

Ringu is a full stack web application for managing call lists. The main purpose of the application is to provide a user friendly tool to collect and organize contact information in a structured way. The user can create an account to be able to create, edit and delete call lists and save the changes.

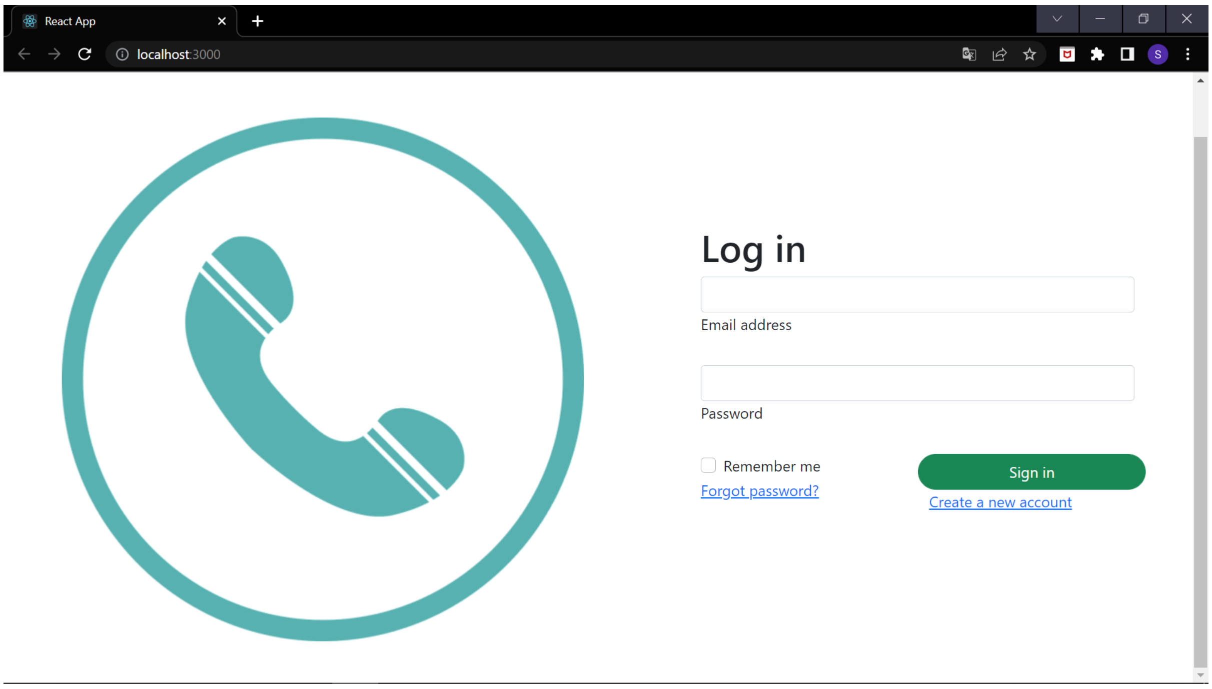
A link to git repository is found here: https://github.com/Maddzenx/web_app.git

A list of use cases

1. A user can register an account
2. A user can sign in/sign out from the application
3. A user can create a call list
4. A user can edit a call list
5. A user can delete a call list
6. A user can create a contact
7. A user can edit a contact
8. A user can delete a contact
9. A user can display its call lists
10. A user can display its call lists' contacts

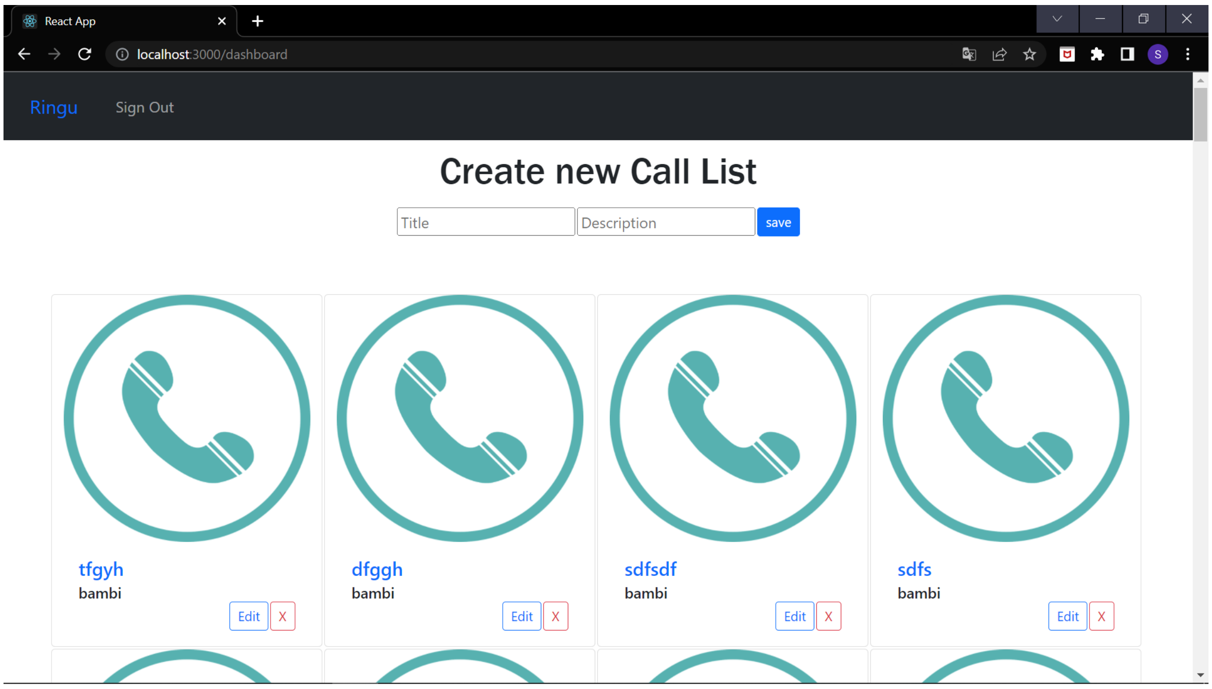
User manual

The start page of the application is a sign in page. The user can either sign in or create a new account.



[Screenshot of the application's main page]

If the user chooses to sign in, a new page of a dashboard will show. However, if the user chooses to create a new account the user will be directed to the registration page to fill in account details. After the registration, the user will be directed to the dashboard.



Create new Call List

Title | Description | save

tfgyh
bambi
Edit X

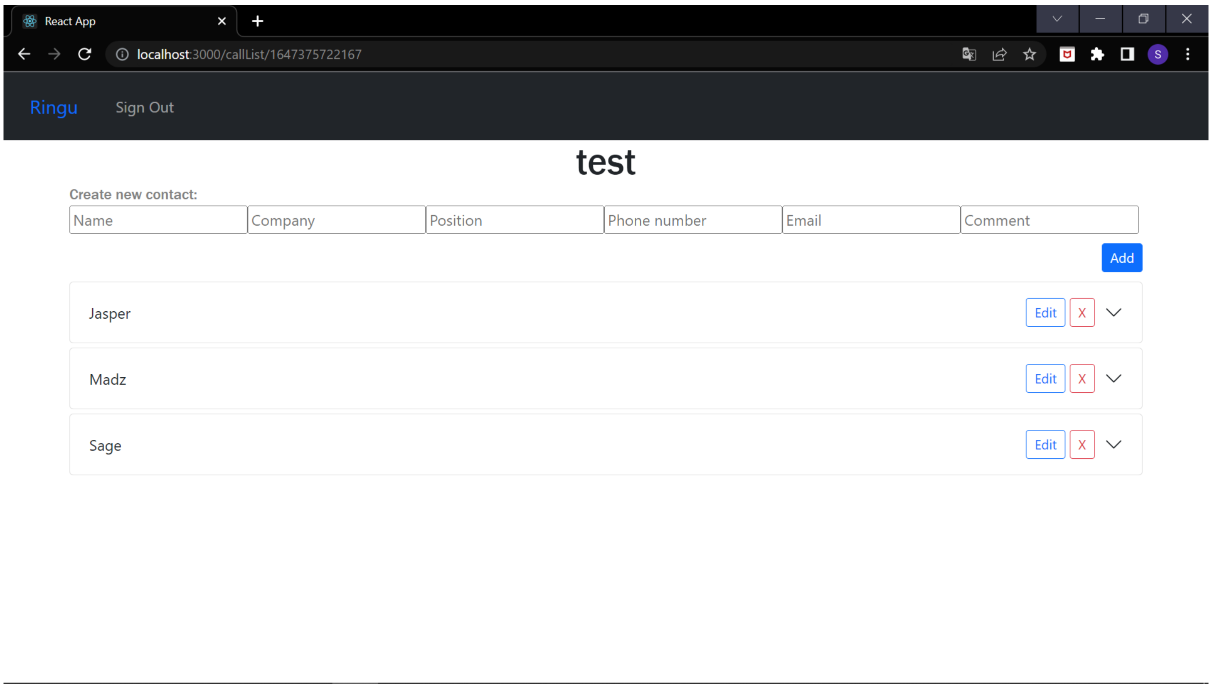
dfggh
bambi
Edit X

sdfsdf
bambi
Edit X

sdfs
bambi
Edit X

[Screenshot of *dashboard* page]

The dashboard has an input form where the user can fill in a title and a description to create a new call list. After filling in the form, the user can click on a button to confirm to create a new call list which will appear in the dashboard. The user can edit the title of a call list by clicking on the *edit* button and delete with the *delete* button. When the user clicks on the title of the *Call List* card it will be directed to a *Contact* page.



test

Create new contact:

Name | Company | Position | Phone number | Email | Comment

Add

Jasper — Edit X

Madz — Edit X

Sage — Edit X

[Screenshot of *Contact* page]

The *Contact* page has a list of contacts and also an input form to fill in new information to add new contacts. After creating a contact, it will be added to the call list and its contact information will be displayed. The user can edit the name and telephone number of a contact by clicking on the *edit* button. Moreover, the user can delete a contact by clicking on the *delete* button. The call list is an accordion which shows only the name when it is not expanded. Once the user expands an accordion field, information about a contact such as company, position, phone number, email and a comment will be displayed.

On top of the GUI is a navigation bar where the user can sign out. When the user clicks on the *sign out* button, the user will be directed to the start page.

Design

The MERN stack was used to build the web application. Regarding security of the application, the library bcrypt has been included to hash the passwords. The framework that has been used is React-Bootstrap.

Other technologies that have been included are React Developer Tool to inspect the application, also the web inspector was used to debug and see the console log updates. Moreover, the application Postman was used to test that the responses were fine. When creating the GUI for the application, Figma was used.

Specification of the API

	Function	**Accepted requests**	**Response for valid request**	**Response for invalid request**
Contact	getContact()	GET	200	500
	createContact()	POST	201	400 or 500
	editContact()	PUT	201	400 or 500
	deleteContact()	DELETE	201	400 or 500
	changeStatus()	PUT	200	400 or 500
CallList	getCallList()	GET	200	500
	getOneCallList()	GET	200	500
	createCallList()	POST	201	400 or 500
	editCallList()	PUT	201	400 or 500
	deleteCallList()	DELETE	201	400 or 500
User	createUser()	POST	201	400 or 500
	logInUser()	POST	201	400 or 500
	getUser()	GET	201	500

Responsibilities

Regarding the brainstorming, planning and formulating the use cases, we did it together as a group. After forming mutual goals, expectations and deciding on what functionality we plan on implementing, a mockup in Figma was created together.

When creating HTML and CSS files for the pages, we decided to divide the work. Jesper was responsible for the pages to sign in and create a new account and he also created CSS animations. Salvija was responsible for creating the page for displaying a call list with its contacts. Madeleine was responsible for creating the dashboard for the call lists.

The next part of the development process was to create the server layer including the model, router and service which we did together, which reminded us of the pair programming way of working. As we were pair programming, the observer and navigator roles were switched frequently in order to make sure that everyone contributed. Afterwards, unit tests and integration tests were created. As before, we did pair programming.

For the lab assignment 3 where we were supposed to create a react app, we worked together at first. However, we had come to a stage where we were sure that each of the group members understood the development process and was able to take on their own parts on the application so we decided to split the work to be able to work on different parts simultaneously.

Salvija was responsible for the use cases: sign in, sign out and create a new account. Also she integrated the database, redo the service layer and resolved errors in the router and tested the code. Madeleine was responsible for creating a call list, edit a call list and delete a call list and also tested the code. Jesper was responsible for implementing the use cases: create contact, edit contact, delete contact. Jesper finished his use cases so he helped with implementing creating a call list, edit a call list and delete a call list. Moreover, he also helped and implemented sign in/sign out and create new account.