

# Documentation iNeedPower

## CONTENTS

---

About the project.....	2
About the code .....	2
File structure .....	2
How to read this.....	3
Login.....	3
Page components .....	5
Home.....	5
Profile and Userpage.....	5
Projectpage .....	5
ProjectDisplay .....	6
EditProject.....	6
NewProject.....	7
SearchPage.....	7
ProblemPage .....	8
Leaderbord.....	8
Header.....	8
Wrapping up .....	8

## ABOUT THE PROJECT

---

This web application provides a platform where students can show projects they are working on or are planning to start soon. The creators of these projects can search for other people on the application that could help them. They also can ask questions to other people, comment on projects, ask for help, like and join other projects. The whole application is centered on projects and users and gives the possibility to everyone to find interesting projects based on topics they like.

*To install the project please look to the [readme.md](#) file.*

*If you have any question about our application or about this documentation, please feel free to send me a mail to this address [nicolas.pecher@student.ehb.be](mailto:nicolas.pecher@student.ehb.be) I'd be very happy to answer.*

## ABOUT THE CODE

---

For this project we mainly used React, Node.js and Express. We also used some npm packages such as:

- React-router-dom
- React-router
- semantic-ui-react

For our server side we used:

- bcryptjs
- body-parser
- cors
- dotenv
- express
- mysql

All other external resources we used are referred to on the place they were used with a link to the web page where they were found.

## FILE STRUCTURE

---

/: this file, our server file with our connection to the database and the queries to it.

/ineedpower: React application with all react generated files.

/ineedpower/src: index.js file, index.css file, components folder, CSS folder, fonts folder and pictures folder.

/ineedpower/src/components: all the components of our application placed in folders in function of what they are used for. Look to it, most of the folder names describe fairly good what is happening in there.

## HOW TO READ THIS

---

The rest of this document will describe as detailed and precise as possible the working of the main components of the application. The goal of this is to make it easy for other programmers to add functionalities or adapt our code. Hope this will help you!

## LOGIN

---

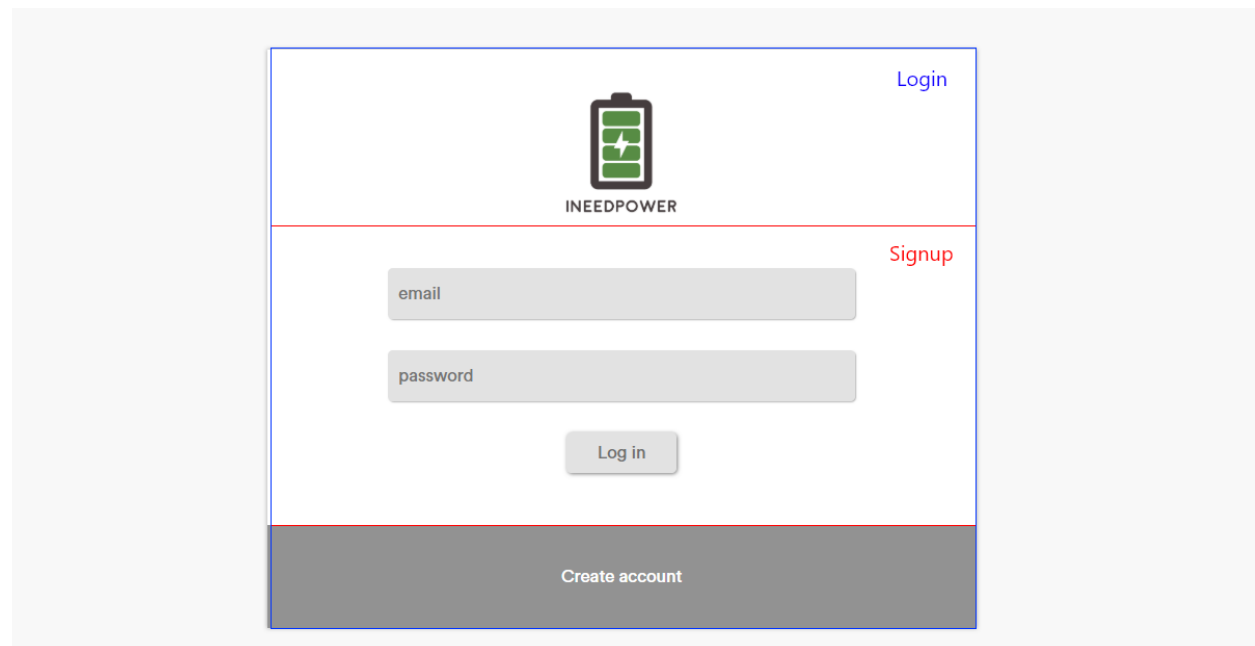
Our login is displayed by the login class situated in Componants/Login/Login.js

Class login:

class Signup

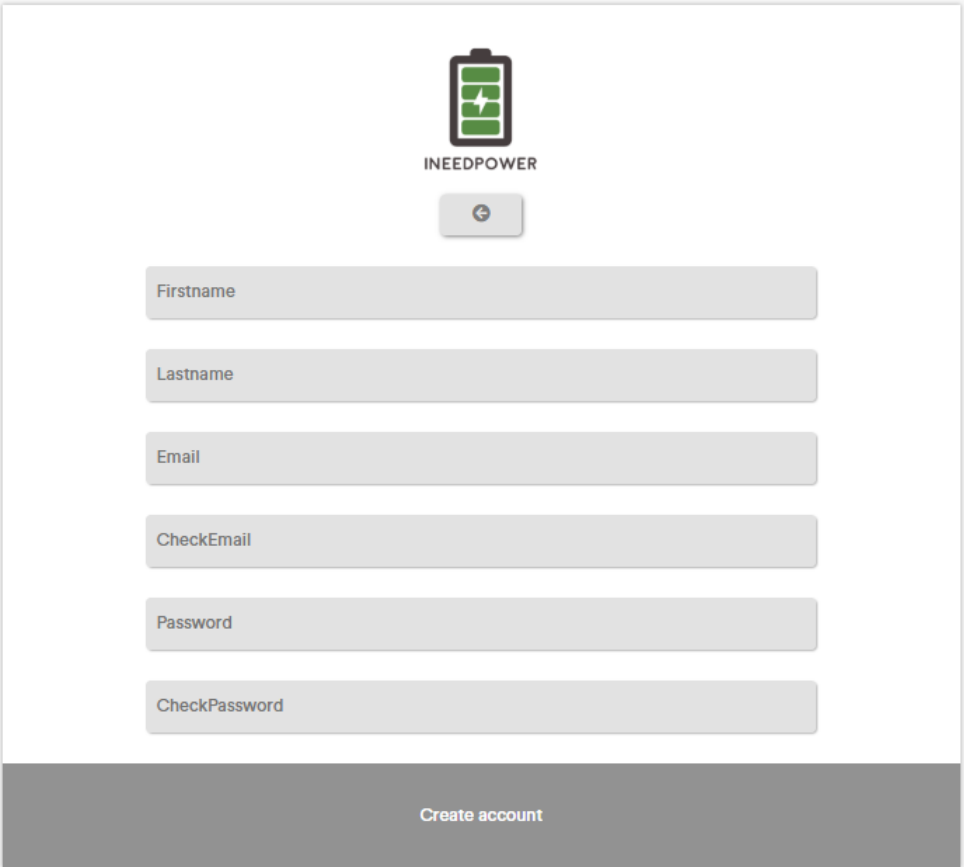
class NewAccount

class GoogleLogin (made for eventual extension of our login system, not implemented)



Login has a version state that determines which component is shown when the user clicks on create account the version changes to “newAccount” which displays the NewAccount component.

This is how the CreateAccount component looks. The user fills in the fields, on submit the data is checked and the user is created. The user is shown a popup (newAccountPopup) with a confirmation message of the creation of the account and the version of the login is changed to show back the “default” version.



The image shows a web form for creating an account. At the top center is a logo consisting of a green battery icon with a white lightning bolt inside, and the text "INEEDPOWER" below it. Below the logo is a small grey button with a circular arrow icon. The form contains six input fields, each with a label on the left: "Firstname", "Lastname", "Email", "CheckEmail", "Password", and "CheckPassword". At the bottom of the form is a wide grey button labeled "Create account".

After logging in the user data is lifted up to the application class situated in `/ineedpower/index.js`. Index is the parent class of all the components, the active user is saved in the state of this component and is sent by props to most of the child components. These child components are placed in the `BrowserRouter` component which gives us the possibility to switch from one child component to another without having to pass by the parent.

## PAGE COMPONENTS

---

With page component we are referring to direct children of the application component. These children are the components that display a page for example the home page displayed by the Home component.

Page components contain other components and have at least one component in common with the other components, this component is the Header component which we will talk about a bit further.

These components also have a redirect component that redirects the user to the login if the current user is not known.

## HOME

---

The home component is situated in `/components/home.js`. The home has child components that are named on base of what they display (these components are only used for clarity purposes, these components only render a `ProjectDisplay` component with 3 props: title, fetch and user).

This component was made to give users an overview of the created projects based on their topics. Other topics can very easily be added by adding another `ProjectDisplay` component.

## PROFILE AND USERPAGE

---

Both components can be found in `/components/user/nameOfComponent.js`. The purpose of these components is to display all the data of a user. The only difference between those two is that they sent a different prop (owner) to their child components. The owner prop is true in the profile component, this gives us the possibility to make the profile editable, adding links, competences, creating projects, edit bio and field of study.

Projects are displayed with the `ProjectDisplay` component and the fetch prop we send returns all the projects in which the user is owner or participant.

## PROJECTPAGE

---

The Projectpage component can be found in `components/projects/Projectpage.js`. This component displays the page to which users are sent when they click on a project.

Not that much to explain about this component, but if you have questions don't hesitate.

## PROJECTDISPLAY

---

This component can be found in `components/projects/ProjectDisplay.js`. This is what is displayed by this component the fetch happens here and the `title` prop that is received is used to describe what is fetched

The boxes in which the `projectData` appears is displayed by the `Project` component in `Projects.js`

### Liked projects



## EDITPROJECT

---

The `EditProject` component can be found in `componants/projects/CreateProject.js`. This file is a bit chaotic and the splitting of this file in multiple files would be one of my priorities if I had more time to work on this project, but for the moment everything works and is divided in several components in function of what they display and how they work.

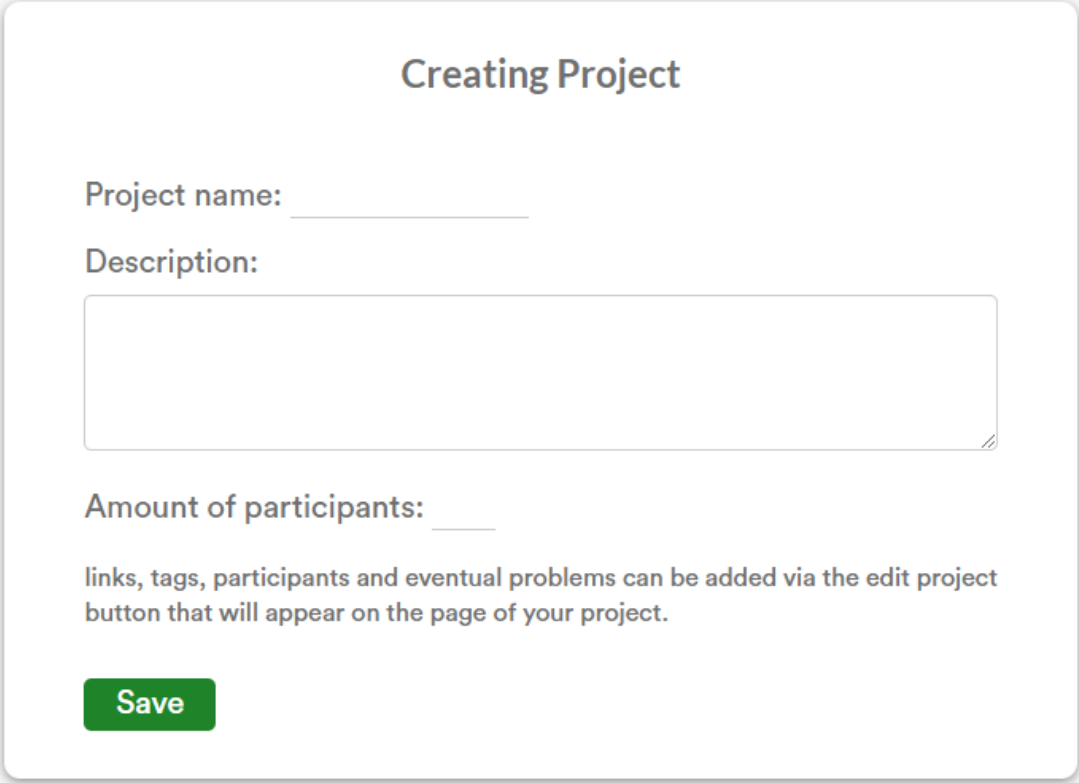
This component gives the owner of a project the possibility to add links, tags and problems. It also gives him the possibility to edit the description of the project and to accept user requests.

This component also gives the owner the possibility to delete comments and participants.

## NEWPROJECT

---

This component can be found in `componants/projects/MakeNewProject.js`.



The image shows a 'Creating Project' form. It has a title 'Creating Project' at the top. Below it are three input fields: 'Project name:' followed by a text input, 'Description:' followed by a larger text area, and 'Amount of participants:' followed by a text input. Below these fields is a paragraph of text: 'links, tags, participants and eventual problems can be added via the edit project button that will appear on the page of your project.' At the bottom left of the form is a green 'Save' button.

This is displayed by the child component `NewProjectData` when the state of `saved` is `false`. The user fills in the fields, the data is checked and if everything is ok the `saved` is changed to `true` and the `ProjectPopup` component is rendered. `ProjectPopup` shows a confirmation text.

## SEARCHPAGE

---

The search page is displayed by the `SearchPage` component in `componants/search/SearchPage.js`. The user is automatically sent to this page when he searches something in the header on the homepage. The search term is sent in the header as a hash to the searchpage, then the results (projects, users or problems) are displayed.

On that page you can search something else and select project, user or problem to only have results of that type. This is handled and displayed by the `SearchInput` component.

## PROBLEMPAGE

---

The problem page is displayed by the ProblemPage component in `components/problems/Problempage.js`. This page is accessed by clicking on a problem on a problem page or on the search page. The page displays a box with a problem (Problem component) and all the answers that were given to the problem (comment component in `components/comments/comments.js`).

## LEADERBORD

---

This component is situated in `components/leaderboard/Leaderbord.js` and is displayed when the user clicks on leaderboard in the header. The ranking of the users (based on rating) is displayed by `DisplayLeaderbordUser` and the ranking of projects based on the likes they have is displayed by `DisplayLeaderbordProject`.

## HEADER

---

The header component was one of the first components we made this explains why it is a bit chaotic, but the working of it is very simple. When we create a new page, we want the header to be displayed in a different way, for example on the home page we want a search bar and on the profile page we want a back button and a title that display “profile”. The solution we found was to send a prop (version) to the component and in function of that we display a different version of the header.

## WRAPPING UP

---

I hope all of this gives you a better understanding of the flow of our application and that it clarified the working of some specific components. All the components listed are the most important ones and are not particularly focused on functionality. Our functional components (like, comment, rate, ...) are generally smaller to make them as reusable as possible.

Thank you for reading,

Nicolas Pecher