

Liebherr Service App Documentation

Project summary:

- The navigation through the whole application is performed using b-nav elements (Vue bootstrap); consequently additional navigation elements can be freely added accordingly.
- Language switcher (vue-i18n) is applied through the application. The user can choose English or German language based on his preference.
- The application is device friendly (responsive). It was tested and verified on different types of devices using online tools.

The application has two main areas, which you can navigate to, on the home page:

- Unregistered Area:

The area includes two components:

Products: It encapsulates two vue-carousel-3d elements. Routing is implemented for each slide in the two carousels, including one example information page:-

- Liebherr components carousel
(An example information page is implemented for the Diesel motor component using b-table)
- Commercial spare parts carousel.

Contact: A static image of contact information.

- Registered Area:

A secure authentication process was implemented. Bcryptjs (hash and salt methodology) was utilized to encrypt the user given password.

Mongodb was used to store the relevant user data (user name, password, email, and id).
The area comprises two components; signup and login.

Sign up: The user should fill up the required information (read through b-form-input elements) to register. After successfully signing up, the user is automatically navigated to the login component.

Login: Registered users can login by typing in their user name and password (b-form-input). After successfully logging in, the user is automatically navigated to the account component, which is presented below.

- **Account:** The page contains two routes; failure plan and chat function.
- ❖ **Failure Plan:** An example plan was implemented (Fehlersuchplan P1019). The working flow of which is as follows:
 1. Input the id "P1019-00" in the Id field.
 2. The first step, that the technician should do, will be shown automatically in the Input and Output fields.
 3. The technician adds the result of the step, he performed (" i.o.", " n.i.o." or "<empty string>") in the input field.
 4. The next step will be shown on the output field automatically.
 5. Continue likewise until you get the message "submit results".
 6. After submitting, a report page will be displayed (routing only done; not implemented)
- ❖ **Chat:** An open area for chatting was implemented, in which logged in technicians can call for support on necessary matters. (Socket i.o was utilized to implement the chatting function.)

Implemented requirements:

- 1) The app is usable with a smartphone ("Responsive Design").
- 2) The app has an open and a closed area for registered users.
- 3) The app is multi-lingual (english/german).
- 4) The app implements the Corporate Identity (CI) of Liebherr.
- 5) The app shows an implementation example of the "Fehlersuchplan".
- 6) The app includes a support area (chat function for registered users) in addition to an inviting display for products using 3d Carousels.

Installation Manual:

Visual Studio code, NodeJS, npm, vuejs

VSCode and Vuejs must be installed then the project should be imported.

See VSCode Vue Tutorial :

<https://code.visualstudio.com/docs/nodejs/vuejs-tutorial>

Prerequisite: NodeJS and npm must be installed on the computer.

- `npm install -g @vue/cli`
- `vue create <project name>`
- `cd <project name>`
- `npm run serve`
- `npm install vue-router`

Client-Framework Vue.js:

- Vue.js was used as a client framework for frontend.
- Vue-bootstrap, css, javascript were used during the course of development.
- Routing: (Router.js, Guard.js)
- Internationalization: (I18n.js, Translation.js, constants (trans.js), locales (de.json, en.json))
- CSS: custom.scss, mycss.css
- Vue Components: Components folder

Multi-Language Support:

For Multilanguage support we have used Vue-I18n plugin and vue-i18n – starter

Github <https://github.com/dobromir-hristov/vue-i18n-starter>

Usages:

(In project root dir) npm install vue-i18n

“Default language, supported languages and fallback language can be setup inside Constants/trans.js.”

```
export const DEFAULT_LANGUAGE = 'de'
```

```
export const FALLBACK_LANGUAGE = 'en'
```

```
export const SUPPORTED_LANGUAGES = ['de', 'en']
```

Server-Side Programming / NodeJs:

NodeJs, API Functions, Express Routing, MongoDB and Express were used.

Express app skeleton:

Install (globally for all users):

```
$ npm install express-generator -g
```

Websocket:

Websocket was used for real time interaction in chat.

Socket.io allows us to emit and receive custom events.

Install Socket.io module

```
npm install --save socket.io
```

<https://socket.io/docs/#Using-with-Express>

To install all of the dependencies

All of the dependencies must be installed to run the project successfully.

Setting up **axios**: `npm init -y npm i axios`

Setting up **babel**: `npm install babel-cli babel-core --save-dev`

Setting up **bcryptjs**: `npm install bcryptjs`

Setting up **body-parser**: `npm install body-parser`

Setting up **bootstrap**: `npm install bootstrap@3`

Setting up **bootstrap-vue**: `npm i bootstrap-vue`

Setting up **cache-loader**: `npm i cache-loader`

Setting up **concurrently**: `npm i concurrently`

Setting up **connect-flash**: `npm i flash`

Setting up **copy-webpack-plugin**: `npm i copy-webpack-plugin`

Setting up **core-js**: `npm i core-js`

Setting up **cors**: `npm i cors`

Setting up **css-loader**: `npm i css-loader`

Setting up **dotenv**: `npm i dotenv`

Setting up **ejs**: `npm i ejs`

Setting up **express**: `npm install express`

Setting up **express-ejs-layouts**: `npm i express-ejs-layouts`

Setting up **express-session**: `npm i express-session`

Setting up **i18n**: `npm i i18next`

Setting up **jquery**: `npm i jquery`

Setting up **jsonwebtoken**: `npm i jsonwebtoken`

Setting up **mongoose**: `npm i mongoose`

Setting up **node-sass**: `npm i node-sass`

Setting up **npm**: `npm i npm`

Setting up **passport**: `npm i passport`

Setting up **passport-jwt**: `npm i passport-jwt`

Setting up **passport-local**: `npm i passport-local`

Setting up **sass-loader**: `npm i sass-loader`

Setting up **socket.io**: `npm i passport-jwt`

Setting up **uws**: `npm i uws`

Setting up **vue**: `npm install vue`

Setting up **vue-carousel-3d**: `npm i vue-carousel-3d`

Setting up **vue-router**: `npm i vue-router`

Setting up **vue-socket.io**: `npm i socket io`

Setting up **vuex**: `npm i vuex`

Setting up **@vue/cli-plugin-eslint**: `npm i @vue/cli-plugin-eslint`

Setting up **@vue/cli-service**: `npm i @vue/cli-service`

Setting up **babel-eslint**: `npm i @babel-eslint`

Setting up **eslint**: `npm i eslint`

Setting up **eslint-plugin-vue**: `npm i eslint-plugin-vue`

Setting up **nodemon**: `npm install nodemon`

Setting up **vue-template-compiler**: `npm install vue-template-compiler`

User manual

To successfully run the project, mongoDB must be installed on your pc. Then create a new database or you can use mongoDB test database. Afterwards you have to fix the database path and name on the keys.js file in the project directory. After that you have to open the project folder (liebherr1-master) using visual studio code. Here liebherr1-master is the project name. Finally, to run the project you have to write the following commands in the project terminal

- npm run serve
- node app.js
- chat.js

After running those commands, a localhost link will appear, on which you can click and the project will be opened in a new browser tab. If any dependencies are missing, the terminal will provide a message stating which dependencies should be installed. All of our dependencies list including the associated installation commands are shown above.

Technical remarks:

Connection between server side and client side:

- In the main.js a file a vue Instance is created that uses the store from store.js.
- Store.js is using vuex (getters, actions, mutations) to import functions which are implemented in warehouse/Auth.js.
- Auth.js is using axios to manage http requests.
- Http requests are then dealt with using express router.
- Users.js is using the express router.

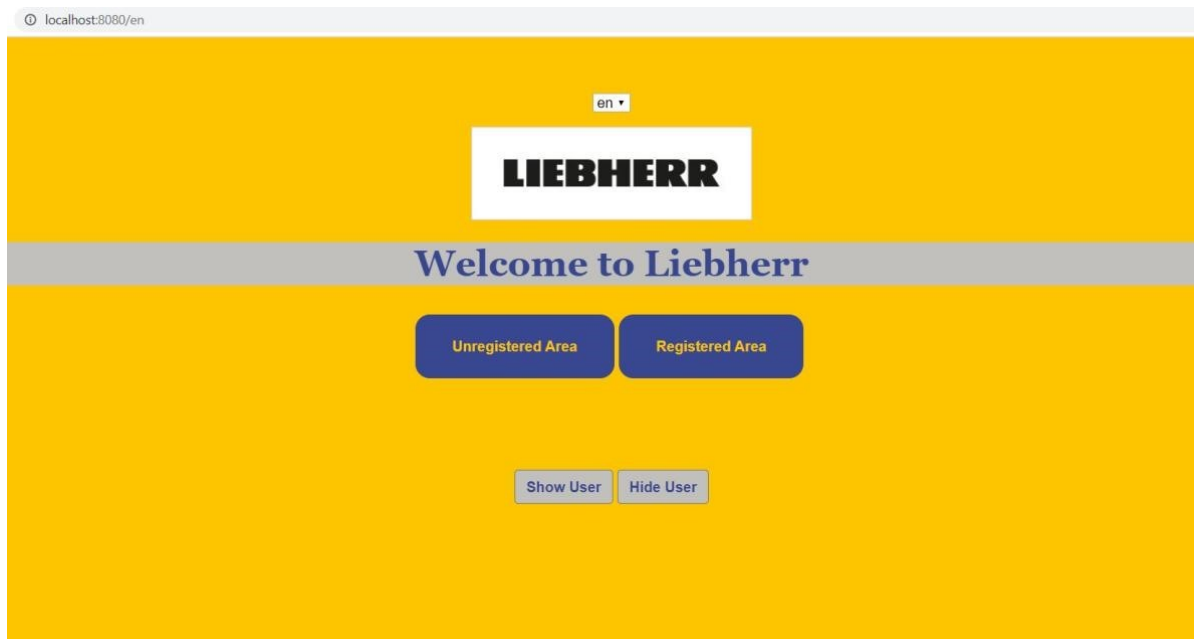
- In app.js a new express instance is instantiated which uses routes from Users.js.

Authentication process:

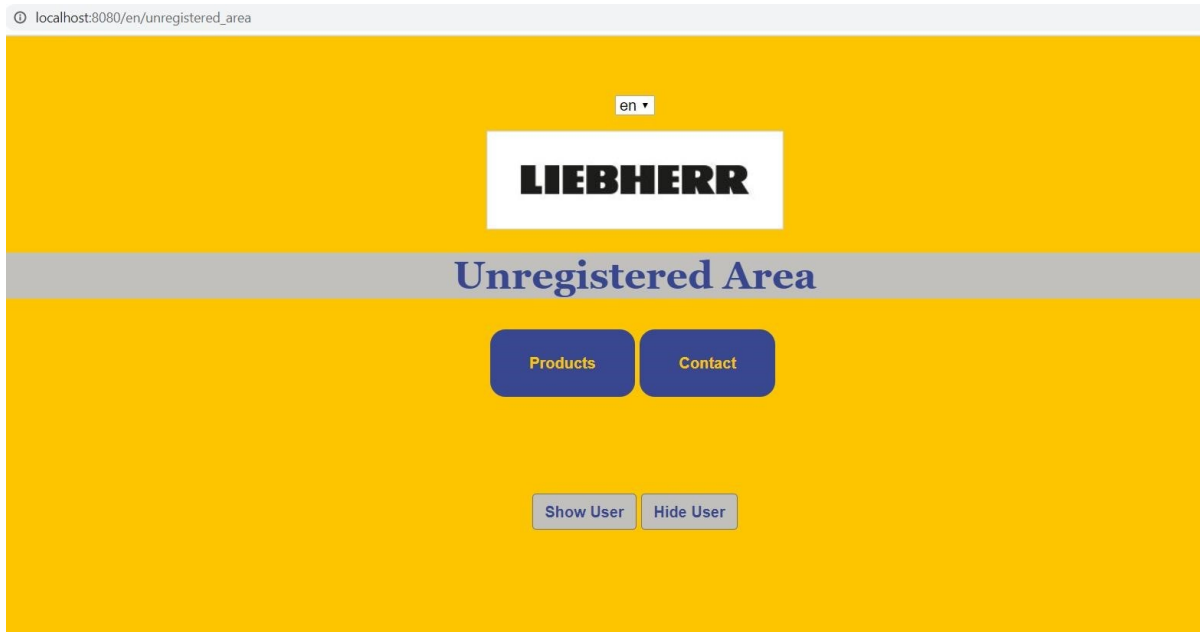
- JWT assigns a token for each user login.
- `getProfile()` in Auth.js is using the `jwt-strategy` developed in Passport.js as follows:

JWT -> (Assigns) Token -> (Checked by) Passport -> (Access) Profile

Screen shots of our project:



Home



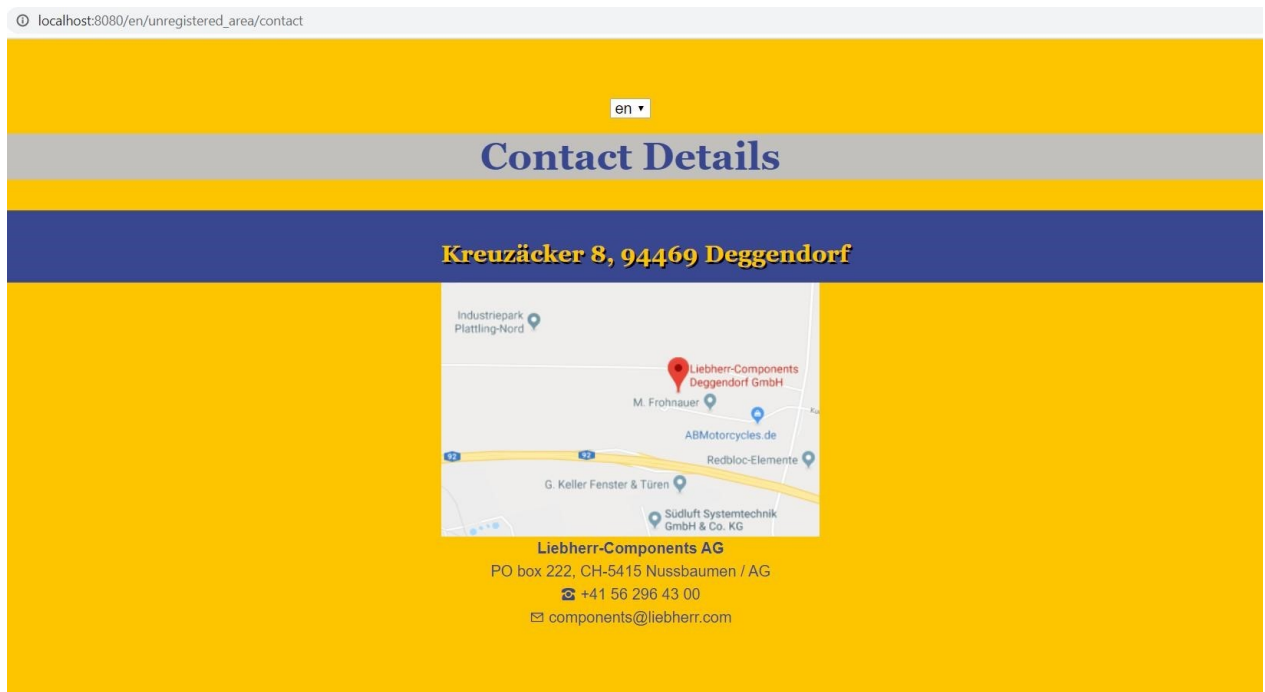
Unregistered Area



Products



Products



Contact

localhost:8080/en/registered_area

en ▼

LIEBHERR

Registered Area

Login Register

Show User Hide User

Registered area

localhost:8080/en/registered_area/register

en ▼

LIEBHERR

Register

Name	Enter your name
User name	Enter your preferred user name
E-Mail	Enter your E-mail
Password	Enter your password
Confirm Password	Retype your password

Submit

Register

en ▼

LIEBHERR

Login

Name :

abdo

Password :

Submit

Login

en ▼

LIEBHERR

Personal Account

Failure plan

chat

logout

Account

localhost:8080/en/registered_area/account/chat

en ▼

Welcome! please select a username

Username...

Chat

localhost:8080/en/registered_area/account/failureplan

en ▼

LIEBHERR

Failure search plan

Failure search plan id	<input type="text" value="id"/>
Output step	<input type="text" value="Output the next flow step"/>
Input	<input type="text" value="Enter result of your current step"/>

Failure plan

A short video about the application also uploaded in the [git repository](#)