Questionnaire

- Ein privates Git Repository mit dem Sourcecode (z.B. Github) oder Code per Zip Archiv (inkl. .git Verzeichnis)
- Ein kurzes How-To, wie man die Webapp bauen und lokal ausführen kann.
- Etwaige Dokumentation in beliebiger Form.
- Ein Vorschlag, welche CRUD Funktionen ein mögliches REST-Backend zur Verfügung stellen müsste, wenn die Datenhaltung (Fragen, Antworten, Iterationen) auf einem Server erfolgen würde.

How-To

1. Clone Repository

```
git clone
```

2. Install Dependencies

```
npm install
```

3. Start Project

```
npm run dev
```

4. Run Tests

```
npm run test
```

Used Frameworks and Libraries

- Vite + React
- Javascript
- Material UI
- Bootstrap
- Jest + React-Testing-Library

Documentation

Structure:

```
src/
components/
iterationShowcase.jsx
...
views/
CreateIterationView.jsx
DashboardView.jsx
...
App.jsx
main.jsx
index.html
```

The app is divided into two main sections, each with its own set of responsibilities. `App.jsx` is the top-level component that renders the app sections. It shows the current view of the application which holds a component that we change depending on the context (`DashboardView.jsx` or `CreateIterationView.jsx`). We pass a prop to each view to change the current component displayed in `App.jsx`.

The `DashboardView.jsx` fetches and shows a list of all iterations from the local storage. If we click on an iteration from the list it gets loaded into the `IterationShowCase.jsx` where we can see more details and delete the selected iteration. The `DashboardView.jsx` features a button that uses the `startView` prop to open the `CreateIterationView.jsx` in `App.jsx`.

The `CreateIterationView.jsx` allows users to answer the questions from the questionnaire. It loads questions and answers from an JSON-file (`data/questions.json`). If we click the `Create` button after all questions are finished, the `CreateIterationView.jsx` saves the answers in the local storage. If we click on the `Cancel` button, it saves an unfinished version of the iteration in the local storage.

The JSON-file `data/questions.json` includes all questions and answers of the questionnaire. We can change existing and add new questions in this file.

All actions related to the local storage are handled by `src/helpers/localStorageHelper.js`.

Tests are written for the functionality of the `DashboardView.jsx` component in `DashboardView.test.js`. It tests if

- the list of iterations is rendered
- the create new iteration button is working
- the iteration showcase is rendered when we click an iteration from the list and
- the clear all iterations button is working.

CRUD Funtionen für REST-Backend

Tables: Questionnaire, Iterations

Prerequisites:

1. Questionnaire can not be updated with app
2. Mutliple iterations of questionnaire per user are possible
3. Each iteration has one-to-many relationship with user

Create:

- **Create Iteration:**

  POST: {url}/iterations

Read:

- **Get Questionnaire:**

  GET: {url}/questionnaire

- **Get all my Iterations:**

  GET: {url}/iterations

- **Get my Iteration:**

  GET: {url}/iterations/:id

Update:

- **Update Iteration:**

  PUT: {url}/iterations/:id

Delete:

- **Delete Iteration (depending if "isDeleted" flag used):**

  PUT: {url}/iterations/:id

  DELETE: {url}/iterations/:id