25th January 2024
Version: 9695eaa

Software Architecture Report

Contents

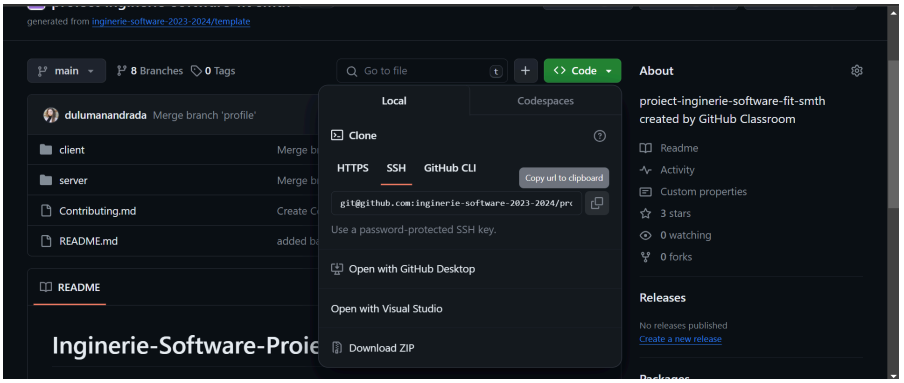
1. Purpose of the software project

PowerUp is a software project with a clear mission, to eliminate financial barriers to a healthy lifestyle. It focuses on personalized weight management, informative content, and community engagement. By making health accessible to all, the project combines cutting-edge technology with evidence-based information to guide users on their unique well-being journey.

At this point, the project presents user experience with registration process, a user-friendly profile management system with edit option, calculate Body Mass Index (BMI), Basal Metabolic Rate (BMR), and Total Daily Energy Expenditure (TDEE) and comment on articles.

2. Running the project

To run the project locally, clone this repository.



In the terminal, run 'npm install'. After this, split the terminal: in one type *cd client* and in the other *cd server*. In each one type 'npm start' for deploying the project locally. This way is different from the regular process for backend because we add a script that uses *nodemon.*

```
Compiled successfully!

You can now view client in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.174.1:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

```
PS C:\Users\bianc\is\proiect-inginerie-software-fit-smth> cd server
PS C:\Users\bianc\is\proiect-inginerie-software-fit-smth\server> npm start

> server@1.0.0 start
> nodemon server

[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Server started on port 3001
DB Connected successfully
```

```
PS C:\Users\bianc\is\proiect-inginerie-software-fit-smth> cd client
PS C:\Users\bianc\is\proiect-inginerie-software-fit-smth\client> npm start
```

In *client*, "build": "react-scripts build" performs a series of tasks to prepare application for deployment: transpiling, bundling, minification and asset management. In *server*, "build": "npm run lint" is using ESLint which is a tool that analyzes the code for potential errors.

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "start": "nodemon server",
  "dev": "nodemon server"
},
```

server (backend)

```
"scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject"
},
```

client (frontend)

It's important to mention that *package.json* should be deleted and then type in the command prompt *npm install* to install all the dependencies and libraries.

The **backend server** is running on port 3001 and the **frontend server** is running on port 3000.

If you want to contribute to our software project, please check our Contribution guide.

3. Application entry points

Data sources

- MySQL database

Table: users

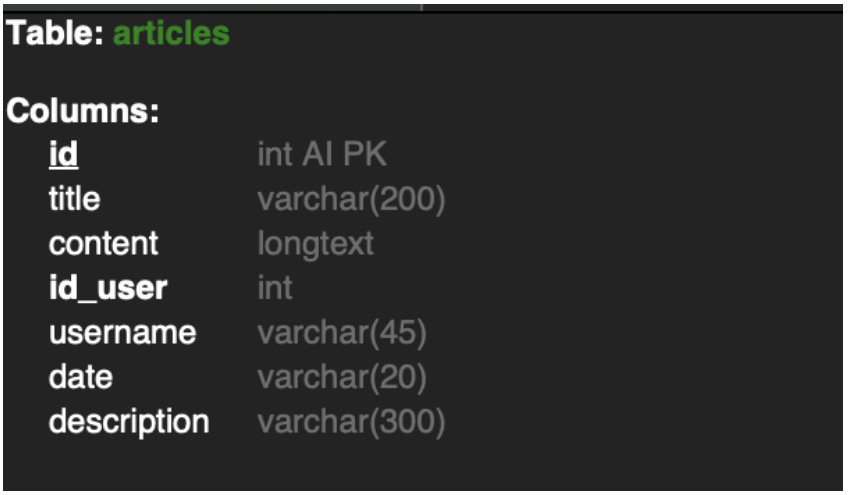
Columns:

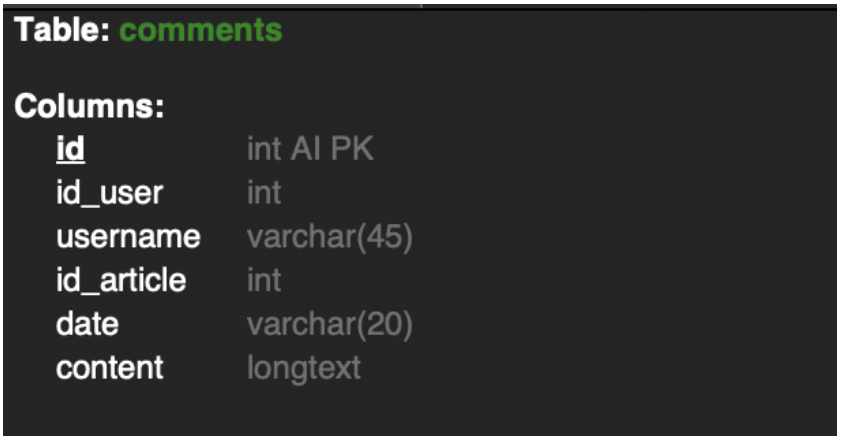
iduser	int AI PK
mail	varchar(100)
username	varchar(100)
password	varchar(100)
profileimage	varchar(200)
tokenEmail	varchar(200)
emailVerified	int
tokenPassword	varchar(200)

Table: userinfo

Columns:

iduser	int PK
firstname	varchar(255)
lastname	varchar(255)
current_weight	decimal(5,2)
goal_weight	decimal(5,2)
dateofbirth	date
height	decimal(5,2)





- Local Storage

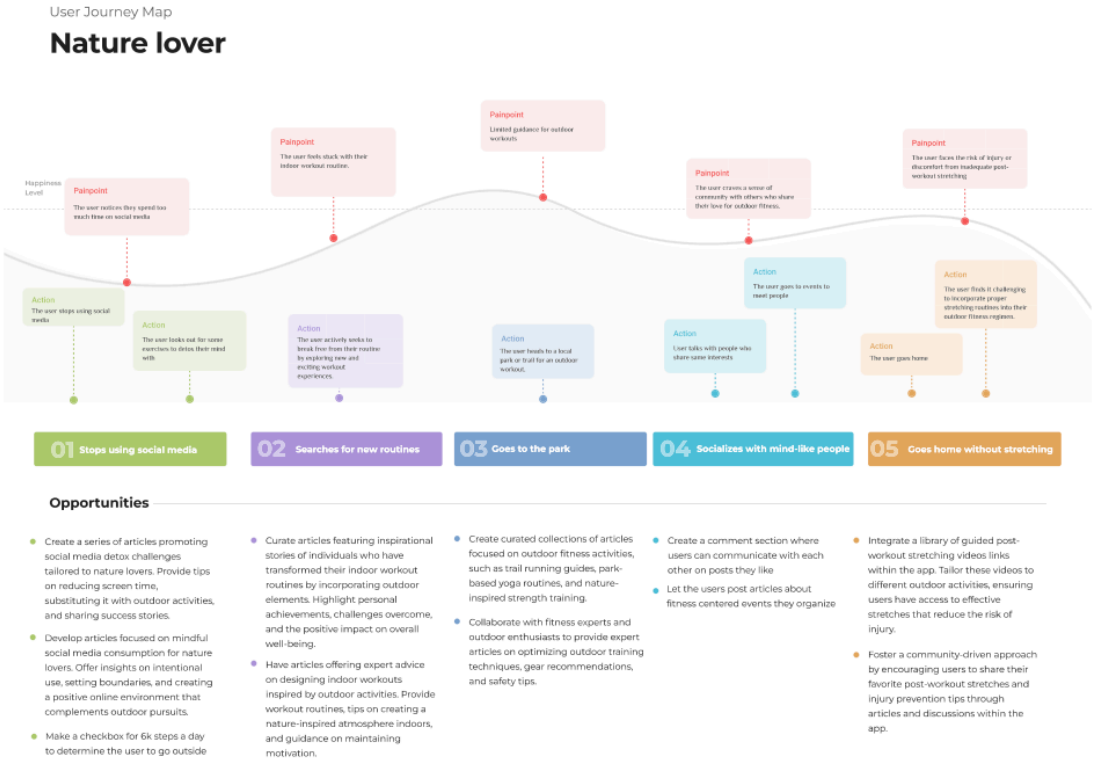
Data inputs

- login and register forms
- calculators (BMI, BMR, TDEE)
- edit profile

Configuration files

- package-lock.json
  - automatically generated
  - records the exact version of each installed package and its dependencies
  - ensures that the same versions are installed across different environments
- package.json
  - contains metadata about the project, including its name, version, description, author, and more.
  - lists project dependencies along with their allowed version ranges.
  - includes scripts, which can be executed with *npm run*
  - specifies configuration settings and other project-related information.

4. High level diagrams of the architecture



5. Deployment plan

Application Components

- Backend Server
    - The backend server runs on Node.js and is accessible at `localhost:3001`.
    - API endpoints are available for frontend communication.

- Frontend Application
    - The React frontend runs on `localhost:3000`.
    - The application interacts with the backend via API calls.

- Database
    - MySQL Workbench is used for database management.
    - The application relies on a local MySQL database.

Local Development Setup

- Backend
    - Start the backend server with `npm start` in the backend directory.

- Frontend
    - Run the frontend development server with `npm start` in the frontend directory.
    - Connect to the backend API at `localhost:3001`.

- Database
    - Use MySQL Workbench to connect to the local database.

Configuration

- Environment Variables
    - set environment variables for database connection and API endpoints.

- Local Storage:
    - the application utilizes local storage for client-side data storage.

Database Management

- use MySQL Workbench for database schema management
- execute queries and scripts to manage the database

For **deployment steps** and **build process** see **chapter 2.**

a. How the CI/CD pipeline works.

6. External dependencies

- testing libraries:
    - jest-dom
    - react
    - user-event
- axios
- bootstrap
    - react-bootstrap
- react
    - dom
    - icons
    - router-dom
    - scripts
    - react-quill
- web vitals
- bcrypt
- cookie parser

- cors
- express
- mysql
- nodemon