Hack4Impact UIUC - MERN Template

Background

The majority of Hack4Impact projects are web based, and from those, are built using the MERN stack. MERN is one of the most widely used stacks for modern web development. Composed of MongoDB, Express, React, and Node, MERN represents the full stack from the frontend to the backend.

Provided Set Ups

- Sexy README
- Frontend and backend skeletons
- Frontend testing with Cypress
- Backend testing with Jest
- Frontend and backend linting with ESLint
- Continuous integration tests with CircleCI

Project Structure

The frontend code is contained in `client/`, and `api/` contains the backend code. Each corresponding directory contains the respective configurations for testing and linting, as well as a `package.json` file with the necessary production and development packages.

README

A sexy readme is provided. A couple things to change include

- Nonprofit logo (replaces Hack4Impact logo)
- Team member images & links

Client

Navigate to `client/`

To begin to set up the client, navigate to the `/public` directory. This will contain project metadata, such as the favicon and Apple touch icon. In addition, you can change the title of your project in `index.html`.

In `package.json`, change the fields as needed (including project name, GitHub repository, and license).

Running Client Locally

To run, `yarn` to install dependencies, and then `yarn start` to start the development server. It will be located on `http://localhost:3000`.

In `/src`, the React components will be developed. Given is a single `Home` component located in `/src/pages` which connects to the backend. It simply just renders an `h1` header and two `p` tags. If the API is not connected, the second tag will render the default state that indicates that the API is not connected. If it is connected, the state gets updated upon API fetch.

Linting, Typechecking, and Testing

The linting and testing frameworks have been provided.

Linting uses ESLint linter, with Prettier and Airbnb style guides. Customization for specific rules can be entered in `.eslintrc.json`, and style changes in `.prettierrc`. To run linting, `yarn lint` within the `client` directory.

Type checking uses Flow, which can be run with `yarn flow`. For a file to be detected by flow, add `// @flow` at the top. By default, type checking is not enabled in continuous integration as it hinders fast project development.

Frontend testing uses Cypress. There is a default configuration for Cypress set up. For your project, please navigate to the Hack4Impact UIUC Cypress dashboard (which can be reached by signing in with GitHub to Cypress), and enable your project. You will recieve a project id, which you should enter in `cypress.json`. Tests can be found in `client/cypress/integration` To run Cypress tests, `yarn test` in the `client` directory.

API

Navigate to `api/`

In `package.json`, change the fields as needed (including project name, GitHub repository, and license).

Here in `/src`, there are many folders. `/src/api` contains the code for fetching from MongoDB (currently commented out), `/src/middleware` contains the code for error handling, `/src/models` contains the MongoDB models, and `/src/routes` contains the routes for the endpoints of the API.

There is also configuration for three seperate envirments: production (`production`), development (`dev`), and testing (`test`). Each of these have their own database, which can be configured within the `/config` directory. Inside, create three `.env` files: `production.env`, `dev.env`, and `test.env`. Each should contain the following field, with unique values (a seperate database for each environment).

```
MONGO_URL=mongodb://<username>:<password>@<id>.mlab.com:<id/project>
```

`production` is set on deployment, `dev` is set whenever running locally, and `test` is set when tests are running. These environments are automatically set based on the task within `app.js`.

Running API Locally

To run, `yarn` to install dependencies, and then `yarn start` to start the development server. It will be located on `http://localhost:9000`.

Currently there is a single endpoint at `/api/home` which returns the text for home.

Linting and Testing

The linting and testing frameworks have been provided.

Linting uses ESLint linter, with Prettier style guides. Customization for specific style changes in `.prettierrc`. To run linting, `yarn lint` within the `api` directory.

Backend testing uses Jest. There is a default configuration for Jest set up. Tests can be found in `api/test`. Simply run `yarn test` to run the tests.

Continuous Integration

Continuous integration is just the practice of automatically running tests for every single code push, which can be automated by services such as

CircleCI. To enable these, simply go to the CircleCI dashboard (found at https://app.circleci.com/pipelines/github/hack4impact-uiuc) and sign in with your GitHub. You should then be able to select Hack4Impact organization in the top left corner, where you'll see all of the current pipelines for the projects. Find and set up your project, and select that there is already a configuration.

In `.circleci/config`, there is an outline of the CI jobs that are used. These include:

- `frontend_lint`: Ensuring `yarn lint` within the `client/` directory
- `frontend_test`: Ensuring `yarn test` within the `client/` directory
- `backend_lint`: Ensuring `yarn lint` within the `api/` directory
- `backend_test`: Ensuring `yarn test` within the `api/` directory

`frontend_test` is a modification of Cypress CircleCI orb for `cypress/run`, which depends on `frontend_test_install`, a modification for `cypress/install`.

If you have set up Cypress in the dashboard and gotten your project id, add it to the CircleCI environment variables and add the `record: true` flag.

By default for `frontend_test`, parallelism is turned off. To enable parallelism, add the flags

- `parallel`: `true`
- `parallelism`: `4`