Bachelor of Computer and Information Sciences

Contemporary Issues in Software Engineering
Semester 2, 2024

ASSIGNMENT 1A: Worksheet 4 (30% of Ass1A)

Next.js Project

Deliverables and Due dates:

You are required to complete the Worksheet and keep evidence as you do it by taking screenshots of your work, as well as explanations.

This worksheet should be Checked off and uploaded to Canvas by end of Tutorial Week 5.

Introduction

This worksheet introduces you to Next.js, routing URLS & different page components.
You will also get practice in making a drop-down input, an input form, and a display table.

Everything will be done from the frontend React, with some dummy data files in the frontend.

For the SPEED product development, you should be able to create a server backend with nest.js/node.js, connect this backend to MongoDB Atlas, and connect the React front end to the backend.

Overview

This is just an overview – the actual instructions are later

1. Setup create-next-app frontend (Using Next.js)
2. Setup Git and GitHub
3. Create some simple pages
4. Create some dummy data
5. Change the SE-Practices display page component to use a dropdown component and table component
6. Change the Submit-Article page component to use a form component
7. Deploy with Vercel

1. Open VSCode and Open the CISE_REPOS folder

2. Create folder “worksheet4” under the CISE_REPOS folder

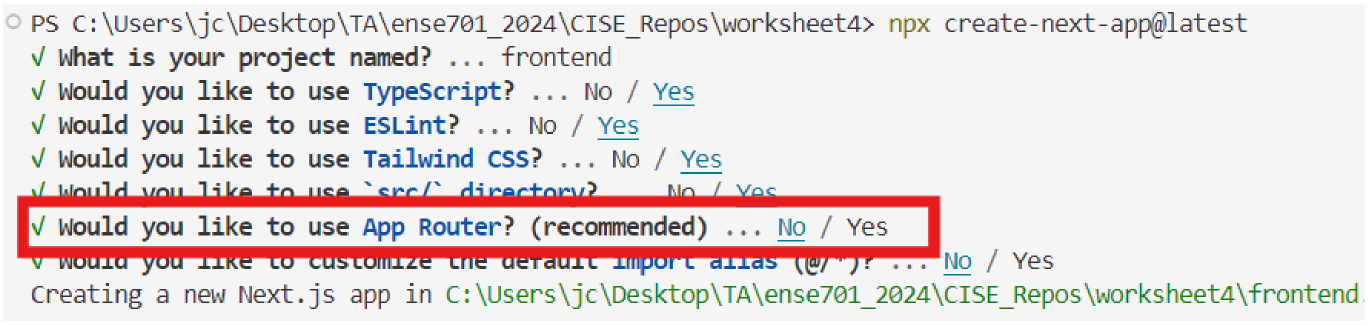
3. In VS Code create folder “backend” under “worksheet4” folder

4. Open a new terminal (CLI) in folder in “worksheet4” folder

5. In this folder:

   > npx create-next-app@latest frontend
   (This assumes you have node.js installed on your machine from previous worksheets)

   NOTE: We used App Router in worksheet 3 (the one recommended by NextJS), and in worksheet 4 we want to show you the other one: the Pages Router. You can still use the app router, but you will need to adapt the file/folder/code accordingly. You are free to use any one in assignment B.

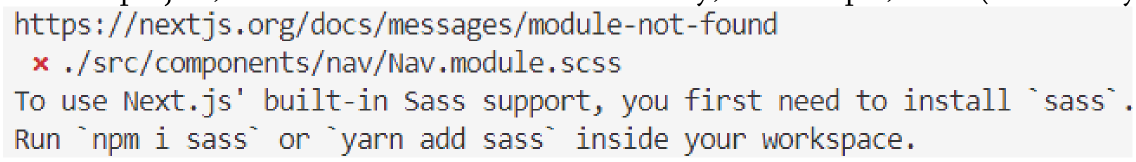


```
○ PS C:\Users\jc\Desktop\TA\ense701_2024\CISE_Repos\worksheet4> npx create-next-app@latest
  √ What is your project named? ... frontend
  √ Would you like to use TypeScript? ... No / Yes
  √ Would you like to use ESLint? ... No / Yes
  √ Would you like to use Tailwind CSS? ... No / Yes
  √ Would you like to use `src/` directory? ... No / Yes
  √ Would you like to use App Router? (recommended) ... No / Yes
  √ Would you like to customize the default import alias (@/*)? ... No / Yes
  Creating a new Next.js app in C:\Users\jc\Desktop\TA\ense701_2024\CISE_Repos\worksheet4\frontend.
```

You can test using the command – if this runs you are on the right track (cd into the right folder first!):

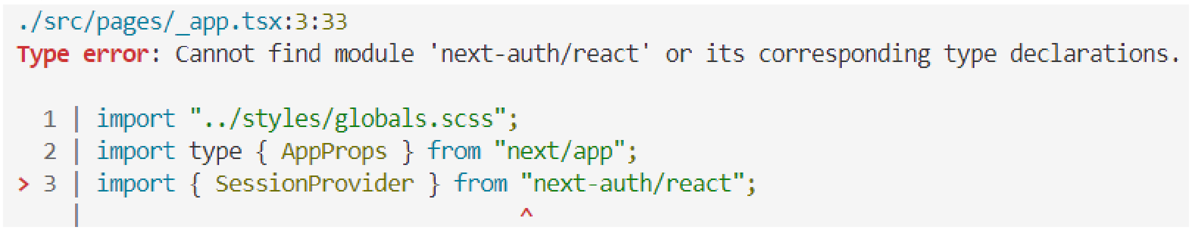
> npm run dev

6. Install extra dependencies. Some dependencies we used in this worksheet may be missing in the new project, so we need to install them manually, for example, SASS (for .scss style files).



```
https://nextjs.org/docs/messages/module-not-found
 × ./src/components/nav/Nav.module.scss
To use Next.js' built-in Sass support, you first need to install `sass`.
Run `npm i sass` or `yarn add sass` inside your workspace.
```

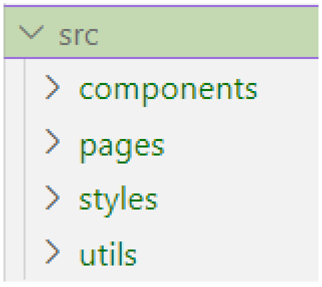
Or, next-auth:



```
./src/pages/_app.tsx:3:33
Type error: Cannot find module 'next-auth/react' or its corresponding type declarations.

  1 | import "../styles/globals.scss";
  2 | import type { AppProps } from "next/app";
> 3 | import { SessionProvider } from "next-auth/react";
    |                                 ^
```

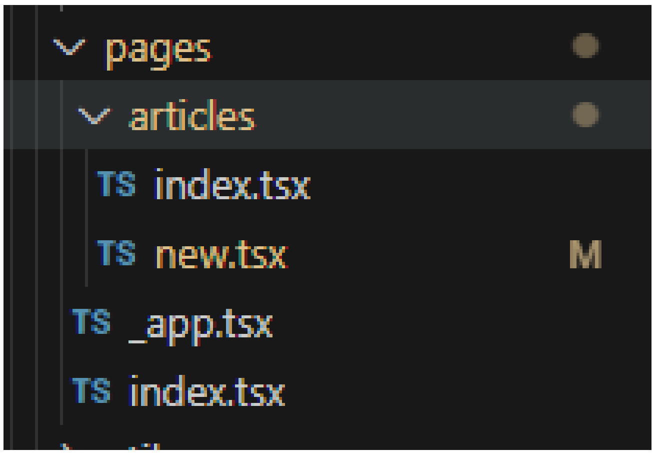
You can use the command “npm install xxx” to install the missing “xxx” package.
Here are the missing packages we identified: sass next-auth react-icons react-hook-form
Use the command to install them: npm install sass next-auth react-icons react-hook-form

7. Clean up our project to start from a clean slate. Simply remove everything in your "src" folder. Then create four folders as shown:



8. **Create some pages**

In the "pages" folder, create these files:



Copy the following code into their respective locations:

pages/index.tsx:

```
export default function Home() {
  return (
    <div className="container">
      <h1>Software Practice Empirical Evidence Database (SPEED)</h1>
    </div>
  );
}
```

pages/_app.tsx:

```
import "../styles/globals.scss";
import type { AppProps } from "next/app";
import { SessionProvider } from "next-auth/react";
import PopulatedNavBar from "../components/PopulatedNavBar";

function MyApp({ Component, pageProps: { session, ...pageProps } }: AppProps) {
  return (
    <SessionProvider session={session}>
      <PopulatedNavBar />
      <Component {...pageProps} />
    </SessionProvider>
  );
}

export default MyApp;
```

pages/articles/index.tsx

```
import { GetStaticProps, NextPage } from "next";
import SortableTable from "../../components/table/SortableTable";
import data from "../../utils/dummydata";

interface ArticlesInterface {
  id: string;
  title: string;
  authors: string;
  source: string;
  pubyear: string;
  doi: string;
  claim: string;
  evidence: string;
}

type ArticlesProps = {
  articles: ArticlesInterface[];
};

const Articles: NextPage<ArticlesProps> = ({ articles }) => {
  const headers: { key: keyof ArticlesInterface; label: string }[] = [
    { key: "title", label: "Title" },
    { key: "authors", label: "Authors" },
    { key: "source", label: "Source" },
    { key: "pubyear", label: "Publication Year" },
    { key: "doi", label: "DOI" },
    { key: "claim", label: "Claim" },
    { key: "evidence", label: "Evidence" },
  ];

  return (
    <div className="container">
      <h1>Articles Index Page</h1>
      <p>Page containing a table of articles:</p>
      <SortableTable headers={headers} data={articles} />
    </div>
  );
};

export const getStaticProps: GetStaticProps<ArticlesProps> = async (_) => {
  // Map the data to ensure all articles have consistent property names
  const articles = data.map((article) => ({
    id: article.id ?? article._id,
    title: article.title,
    authors: article.authors,
    source: article.source,
    pubyear: article.pubyear,
    doi: article.doi,
    claim: article.claim,
    evidence: article.evidence,
  }));


  return {
    props: {
      articles,
    },
  };
};

export default Articles;
```

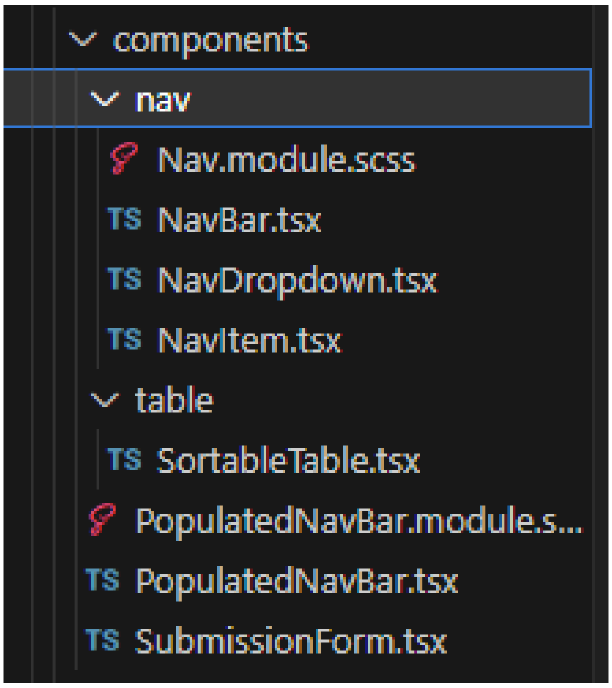
pages/articles/new.tsx

```
import { FormEvent, useState } from "react";
import formStyles from "../../styles/Form.module.scss";

const NewDiscussion = () => {
  const [title, setTitle] = useState("");
  const [authors, setAuthors] = useState<string[]>([]);
  const [source, setSource] = useState("");
  const [pubYear, setPubYear] = useState<number>(0);
  const [doi, setDoi] = useState("");
  const [summary, setSummary] = useState("");
  const [linkedDiscussion, setLinkedDiscussion] = useState("");

  const submitNewArticle = async (event: FormEvent<HTMLFormElement>) => {
    event.preventDefault();

    console.log(
      JSON.stringify({
        title,
        authors,
        source,
        publication_year: pubYear,
        doi,
        summary,
        linked_discussion: linkedDiscussion,
      })
    );
  };

  // Some helper methods for the authors array

  const addAuthor = () => {
    setAuthors(authors.concat([""]));
  };
```

```
const removeAuthor = (index: number) => {
  setAuthors(authors.filter((_, i) => i !== index));
};

const changeAuthor = (index: number, value: string) => {
  setAuthors(
    authors.map((oldValue, i) => {
      return index === i ? value : oldValue;
    })
  );
};

// Return the full form

return (
  <div className="container">
    <h1>New Article</h1>
    <form className={formStyles.form} onSubmit={submitNewArticle}>
      <label htmlFor="title">Title:</label>
      <input
        className={formStyles.formItem}
        type="text"
        name="title"
        id="title"
        value={title}
        onChange={(event) => {
          setTitle(event.target.value);
        }}
      />

      <label htmlFor="author">Authors:</label>
      {authors.map((author, index) => {
        return (
          <div key={`author ${index}`} className={formStyles.arrayItem}>
            <input
              type="text"
              name="author"
              value={author}
              onChange={(event) => changeAuthor(index, event.target.value)}
              className={formStyles.formItem}
            />
            <button
              onClick={() => removeAuthor(index)}
              className={formStyles.buttonItem}
              style={{ marginLeft: "3rem" }}
              type="button"
            >
              -
            </button>
          </div>
        );
      })}
      <button
```

```
          onClick={() => addAuthor()}
          className={formStyles.buttonItem}
          style={{ marginLeft: "auto" }}
          type="button"
        >
          +
        </button>

        <label htmlFor="source">Source:</label>
        <input
          className={formStyles.formItem}
          type="text"
          name="source"
          id="source"
          value={source}
          onChange={(event) => {
            setSource(event.target.value);
          }}
        />

        <label htmlFor="pubYear">Publication Year:</label>
        <input
          className={formStyles.formItem}
          type="number"
          name="pubYear"
          id="pubYear"
          value={pubYear}
          onChange={(event) => {
            const val = event.target.value;
            if (val === "") {
              setPubYear(0);
            } else {
              setPubYear(parseInt(val));
            }
          }}
        />

        <label htmlFor="doi">DOI:</label>
        <input
          className={formStyles.formItem}
          type="text"
          name="doi"
          id="doi"
          value={doi}
          onChange={(event) => {
            setDoi(event.target.value);
          }}
        />

        <label htmlFor="summary">Summary:</label>
        <textarea
          className={formStyles.formTextArea}
          name="summary"
          value={summary}
          onChange={(event) => setSummary(event.target.value)}
        />

        <button className={formStyles.formItem} type="submit">
          Submit
        </button>
      </form>
    </div>
  );
};

export default NewDiscussion;
```

8b) Now create the following files similar to how we created the pages: (Note if you haven't already please create the **components** folder first)



```
components
  nav
    Nav.module.scss
    NavBar.tsx
    NavDropdown.tsx
    NavItem.tsx
  table
    SortableTable.tsx
  PopulatedNavBar.module.s...
  PopulatedNavBar.tsx
  SubmissionForm.tsx
```

Create these files in the **nav** folder:

components/nav/NavBar.tsx

```tsx
import React from "react";
import styles from "./Nav.module.scss";

type Props = {
  children: React.ReactNode;
};

const NavBar = ({ children }: Props) => {
  return <nav className={styles.navbar}>{children}</nav>;
};

export default NavBar;
```

components/nav/NavDropdown.tsx

```tsx
import React from "react";
import styles from "./Nav.module.scss";

type Props = {
  children?: React.ReactNode;
};

const NavDropdown = ({ children }: Props) => {
  return <div className={styles.dropdown_container}>{children}</div>;
};

export default NavDropdown;
```

components/nav/NavItem.tsx

```tsx
import { useRouter } from "next/router";
import React from "react";
import styles from "./Nav.module.scss";

type Props = {
  route?: string;
  children: React.ReactNode;
  end?: boolean;
  dropdown?: boolean;
  onClick?: boolean | (() => void);
  style?: React.CSSProperties;
};

const NavItem = ({ children, route, end, dropdown, onClick, style }: Props) => {
  const router = useRouter();

  const navigate: React.MouseEventHandler<HTMLDivElement> = (event) => {
    if (typeof route === "string") {
      router.push(route);
    }

    event.stopPropagation();
  };
  return (
    <div
      style={style}
      className={`${route || onClick ? styles.clickable : styles.navitem}${
        end ? ` ${styles.end}` : ""
      }${dropdown ? ` ${styles.dropdown}` : ""}`}
      onClick={typeof onClick === "function" ? onClick : navigate}
    >
      {children}
    </div>
  );
};

export default NavItem;
```

components/nav/Nav.module.scss

```
.navbar {
  display: flex;
  position: relative;
  top: 0;
  flex-direction: row;
  width: 100%;
  background-color: darkblue;
  align-items: center;
  color: aliceblue;
  font-size: 1.5em;
  font-weight: bold;
  transition: top 0.5s;
  height: fit-content;
}

.hide {
  top: -3em;
}

.navitem {
  position: relative;
  padding: 1em;
  width: auto;
  display: flex;
  flex-direction: row;
  align-items: center;
  cursor: default;
  white-space: nowrap;
}

.clickable {
  @extend .navitem;
  cursor: pointer;
}

.clickable:hover {
  backdrop-filter: brightness(50%);
}

.dropdown_container {
  visibility: hidden;
  display: flex;
  flex-direction: column;
  background-color: darkblue;
  position: absolute;
  top: 100%;
  right: 0;
}

.dropdown:hover {
  cursor: pointer;
  backdrop-filter: brightness(50%);

  .dropdown_container {
    visibility: visible;
  }
}

.end {
  margin-left: auto;
}
```

Create these files in the table folder

components/table/SortableTable.tsx

```
import React from "react";

interface SortableTableProps {
  headers: { key: string; label: string }[];
  data: any[];
}

const SortableTable: React.FC<SortableTableProps> = ({ headers, data }) => (
  <table>
    <thead>
      <tr>
        {headers.map((header) => (
          <th key={header.key}>{header.label}</th>
        ))}
      </tr>
    </thead>
    <tbody>
      {data.map((row, i) => (
        <tr key={i}>
          {headers.map((header) => (
            <td key={header.key}>{row[header.key]}</td>
          ))}
        </tr>
      ))}
    </tbody>
  </table>
);

export default SortableTable;
```

Create these files in the components folder

components/PopulatedNavBar.module.scss

```scss
.user_container {
  position: absolute;
  aspect-ratio: 1;
  backdrop-filter: brightness(50%);
  display: flex;
  align-items: center;
  justify-content: center;
  width: 2em;
  border-radius: 100%;
  left: 1em
}
```

components/PopulatedNavBar.tsx

```tsx
import { IoMdArrowDropdown } from "react-icons/io";
import NavBar from "./nav/NavBar";
import NavDropdown from "./nav/NavDropdown";
import NavItem from "./nav/NavItem";

const PopulatedNavBar = () => {
  return (
    <NavBar>
      <NavItem>SPEED</NavItem>
      <NavItem route="/" end>
        Home
      </NavItem>
      <NavItem dropdown route="/articles">
        Articles <IoMdArrowDropdown />
        <NavDropdown>
          <NavItem route="/articles">View articles</NavItem>
          <NavItem route="/articles/new">Submit new</NavItem>
        </NavDropdown>
      </NavItem>
    </NavBar>
  );
};

export default PopulatedNavBar;
```

components/SubmissionForm.tsx

```
import React from "react";
import { useForm } from "react-hook-form";

export default function SubmissionForm() {
  const { register, handleSubmit } = useForm();

  const onSubmit = (data: any) => JSON.stringify(data);

  return (
    <form onSubmit={handleSubmit(onSubmit)}>
      <input {...register("title")} placeholder="Title" />
      <p>
        <input {...register("authors")} placeholder="Authors" />
      </p>
      <p>
        <input {...register("source")} placeholder="Source" />
      </p>
      <p>
        <input {...register("pubyear")} placeholder="Publication Year" />
      </p>
      <p>
        <input {...register("doi")} placeholder="DOI" />
      </p>

      <select {...register("linked_discussion")}>
        <option value="">Select SE practice...</option>
        <option value="TDD">TDD</option>
        <option value="Mob Programming">Mob Programmin</option>
      </select>
      <input type="submit" />
    </form>
  );
}
```

Create these files in the **src/utils** folder

utils/table_functions.ts

```
/**
 * Function to sort data based on a sortKey, and whether the sorting should be reversed
or not.
 *
 * @param tableData The data to sort. This is an array of objects
 * @param sortKey The key to sort by.
 * @param reverse True if we should reverse the order of sorting (sorts ascending if
false, descending if true)
 * @returns
 */
export function sortData<T>(
  tableData: T[],
  sortKey: keyof T,
  reverse: boolean
): T[] {
  const sortedData = tableData.sort((a, b) => {
    return a[sortKey] > b[sortKey] ? 1 : -1;
  });

  if (reverse) {
    return sortedData.reverse();
  }

  return sortedData;
}
```

Create these files in the src/styles folder

styles/Form.module.scss

```
.form {
  display: flex;
  flex-direction: column;
  max-width: 30em;
}

.formItem {
  padding: 1em;
  margin: 1em 0;
  flex-grow: 1;
}

.formTextArea {
  @extend .formItem;
  min-height: 8em;
}

.arrayItem {
  display: flex;
  flex-direction: row;
  align-items: center;
  justify-content: stretch;
}

.buttonItem {
  width: 3rem;
  height: 3rem;
  flex-grow: 0;
  font-size: 1.5em;
}
```

styles/globals.scss

```
.form {
  display: flex;
  flex-direction: column;
  max-width: 30em;
}

.formItem {
  padding: 1em;
  margin: 1em 0;
  flex-grow: 1;
}

.formTextArea {
  @extend .formItem;
  min-height: 8em;
}

.arrayItem {
  display: flex;
  flex-direction: row;
  align-items: center;
  justify-content: stretch;
}

.buttonItem {
  width: 3rem;
  height: 3rem;
  flex-grow: 0;
  font-size: 1.5em;
}
```

…AND BREATHE YOU DID IT; YOU SUCCESSFULLY COPIED AND PASTED ALL THIS CODE… A RITE OF PASSAGE FOR EVERY DEVELOPER. https://medium.com/@alejandro_duarte/copy-paste-based-development-1fc9070fb00f

As funny as that is, you won't learn anything and take away skills in the workforce you need without that desire to understand what you just copied. Here are a couple of resources I strongly suggest you read in your spare time; a great developer loves to struggle that's when we learn best!

- **Learn more about scss here:**

https://www.geeksforgeeks.org/what-is-the-difference-between-css-and-scss/

- **React useState:**

https://react.dev/reference/react/useState

- **Next.js App Router:**

https://react.dev/learn/start-a-new-react-project#nextjs-app-router

(*If it is not working ask a TA as they have access to a REPO that will be able to guide you further if you are struggling. Use this as a last resort remember… to struggle is to learn! Try debug what is wrong yourself.*)

Note:
This adds the <Router> tags around the outside <div> tags (it's actually using the BrowserRouter component). The BrowserRouter component provides the foundation for the navigation and browser history handling, that routing is made up of.

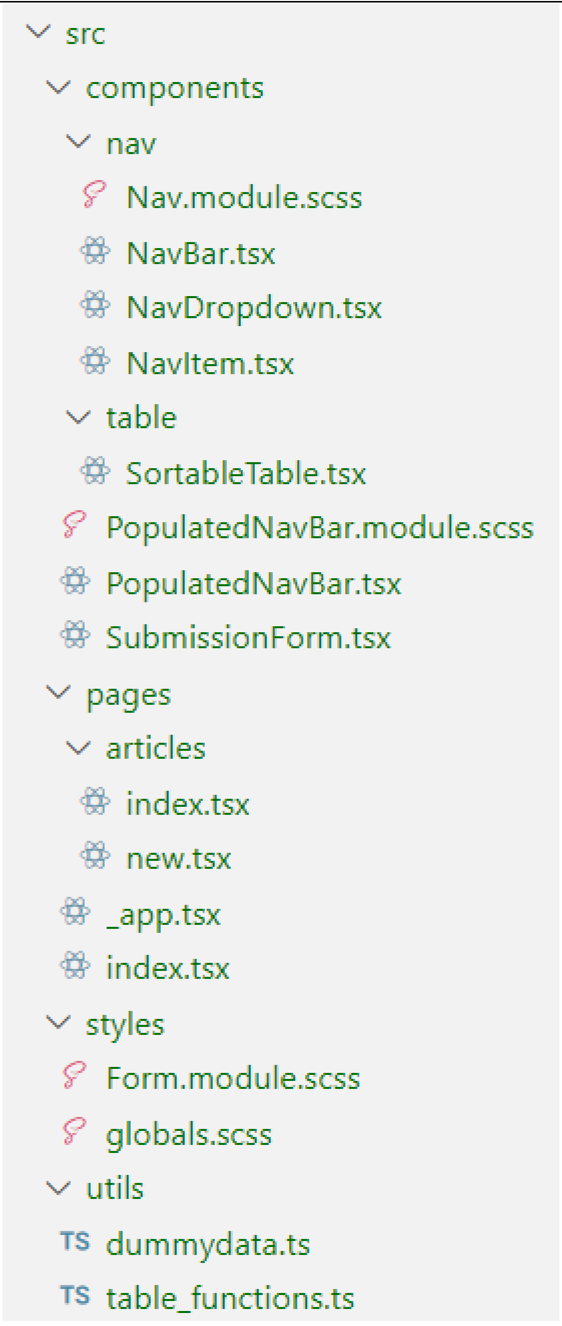
Next we define our navigation links. We already have list elements with each element defined. We will replace them with the more specialized NavLink component. These are between the **<ul className="header">** tags. We will turn this into a menu bar later using some css.

For each link, pay attention to the URL we are telling our router to navigate to. This URL value (defined by the ***to*** prop) acts as an identifier to ensure the right content gets loaded.

The way we match the URL with the content is by using a Route component – between the **<div classname=”content”>** tags.

The format for the Route component is:
<Route path = “*the URL in the browser***” component = {***the name of the component to render***}/>**

9. You should see & have the following file structure (**Create a file named dummydata.json that is empty for now**) for each of your pages (and links)



```
src
  components
    nav
      Nav.module.scss
      NavBar.tsx
      NavDropdown.tsx
      NavItem.tsx
    table
      SortableTable.tsx
    PopulatedNavBar.module.scss
    PopulatedNavBar.tsx
    SubmissionForm.tsx
  pages
    articles
      index.tsx
      new.tsx
    _app.tsx
    index.tsx
  styles
    Form.module.scss
    globals.scss
  utils
    dummydata.ts
    table_functions.ts
```

Double check this – are you missing any files? (Create a file named dummydata.ts that is empty for now)

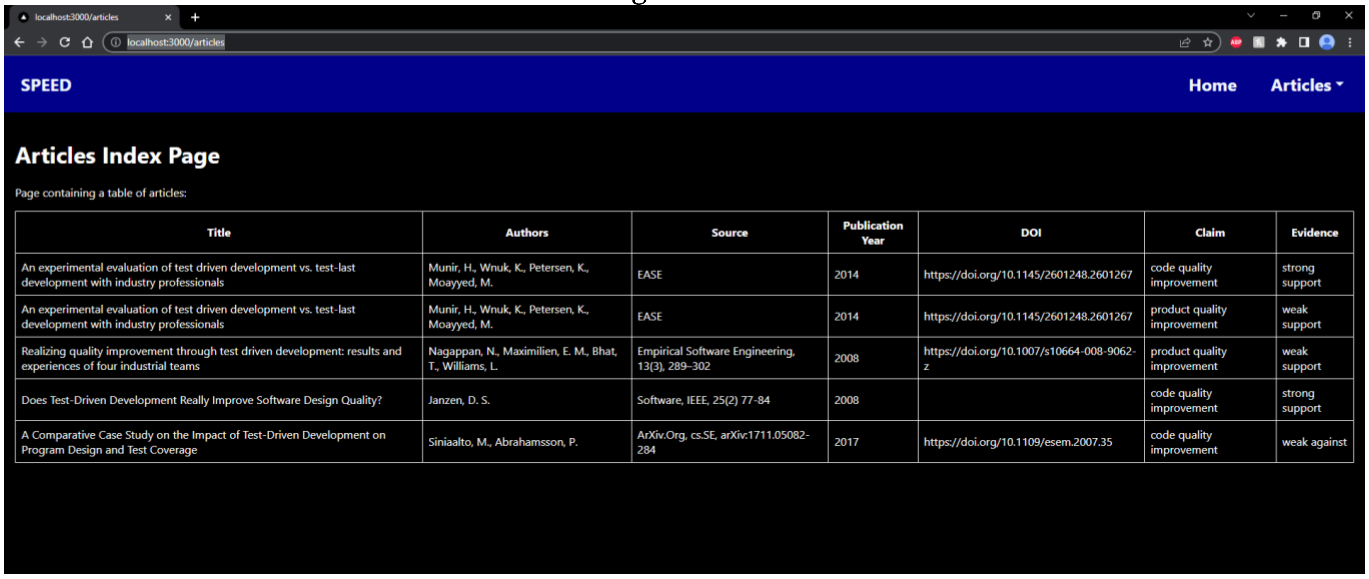
10. Now set up dummy article data to use on our front end

In the **src** folder create a file inside **utils** called **dummydata.ts** and copy this code into it.

```
const data = [
  {
    id: "1",
    title: 'An experimental evaluation of test driven development vs. test-last development with industry professionals',
    authors: "Munir, H., Wnuk, K., Petersen, K., Moayyed, M.",
    source: "EASE",
    pubyear: "2014",
    doi: "https://doi.org/10.1145/2601248.2601267",
    claim: "code quality improvement",
    evidence: "strong support",
  },
{
  _id: "2",
    title: 'An experimental evaluation of test driven development vs. test-last development with industry professionals',
    authors: "Munir, H., Wnuk, K., Petersen, K., Moayyed, M.",
    source: "EASE",
    pubyear: "2014",
    doi: "https://doi.org/10.1145/2601248.2601267",
    claim: "product quality improvement",
    evidence: "weak support",
},
  {
    _id: "3",
    title: 'Realizing quality improvement through test driven development: results and experiences of four industrial teams',
    authors: "Nagappan, N., Maximilien, E. M., Bhat, T., Williams, L.",
    source: " Empirical Software Engineering, 13(3), 289–302",
    pubyear: "2008",
    doi: "https://doi.org/10.1007/s10664-008-9062-z",
    claim: "product quality improvement",
    evidence: "weak support",
  },
  {
    _id: "4",
    title: "Does Test-Driven Development Really Improve Software Design Quality?",
    authors: "Janzen, D. S.",
    source: "Software, IEEE, 25(2) 77-84",
    pubyear: "2008",
    doi: "",
    claim: "code quality improvement",
    evidence: "strong support",
  },
  {
    _id: "5",
    title: "A Comparative Case Study on the Impact of Test-Driven Development on Program Design and Test Coverage",
    authors: "Siniaalto, M., Abrahamsson, P.",
    source: "ArXiv.Org, cs.SE, arXiv:1711.05082-284",
    pubyear: "2017",
    doi: "https://doi.org/10.1109/esem.2007.35",
    claim: "code quality improvement",
    evidence: "weak against",
  },
];
  export default data;
```

[NOTE YOUR APPLICATION SHOULD BE RUNNING AT THIS POINT WITH npm run dev IF YOU ARE GETTING ERRORS PLEASE ASK FOR ASSISTANCE OR GO BACK THROUGH THE CODE]

11. Your browser should show the following for localhost:3000/articles



Articles Index Page

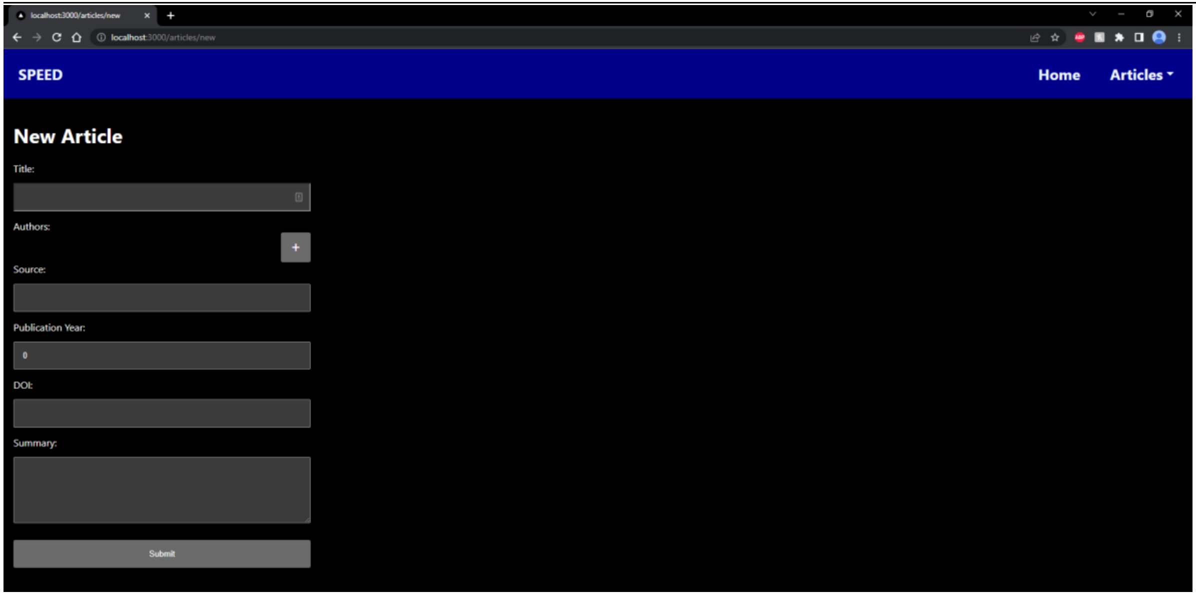
Page containing a table of articles:

Title	Authors	Source	Publication Year	DOI	Claim	Evidence
An experimental evaluation of test driven development vs. test-last development with industry professionals	Munir, H., Wnuk, K., Petersen, K., Moayyed, M.	EASE	2014	https://doi.org/10.1145/2601248.2601267	code quality improvement	strong support
An experimental evaluation of test driven development vs. test-last development with industry professionals	Munir, H., Wnuk, K., Petersen, K., Moayyed, M.	EASE	2014	https://doi.org/10.1145/2601248.2601267	product quality improvement	weak support
Realizing quality improvement through test driven development: results and experiences of four industrial teams	Nagappan, N., Maximilien, E. M., Bhat, T., Williams, L.	Empirical Software Engineering, 13(3), 289–302	2008	https://doi.org/10.1007/s10664-008-9062-z	product quality improvement	weak support
Does Test-Driven Development Really Improve Software Design Quality?	Janzen, D. S.	Software, IEEE, 25(2) 77-84	2008		code quality improvement	strong support
A Comparative Case Study on the Impact of Test-Driven Development on Program Design and Test Coverage	Siniaalto, M., Abrahamsson, P.	ArXiv.Org, cs.SE, arXiv:1711.05082-284	2017	https://doi.org/10.1109/esem.2007.35	code quality improvement	weak against

Also the data for the Dropdown list in SPEED and the table in SPEED should all come from a MongoDB using **axios** to communicate with Nest.js APIs which use mongoose to communicate with MongoDB Atlas. Also for you to work out (look at Worksheet 2 & 3)

The table could have been created with a number of other libraries. Here is a good summary and some code examples.
https://blog.bitsrc.io/top-5-react-table-libraries-170505f75da7

Your browser should show the following for localhost:3000/articles/new



Handling Error 404 – no such page error

For a website or a simple multi-page app, a 404 "page not found" is one of the obvious things to handle and know how to use if you decide to work with react-router. We set that up let's take a look at it:

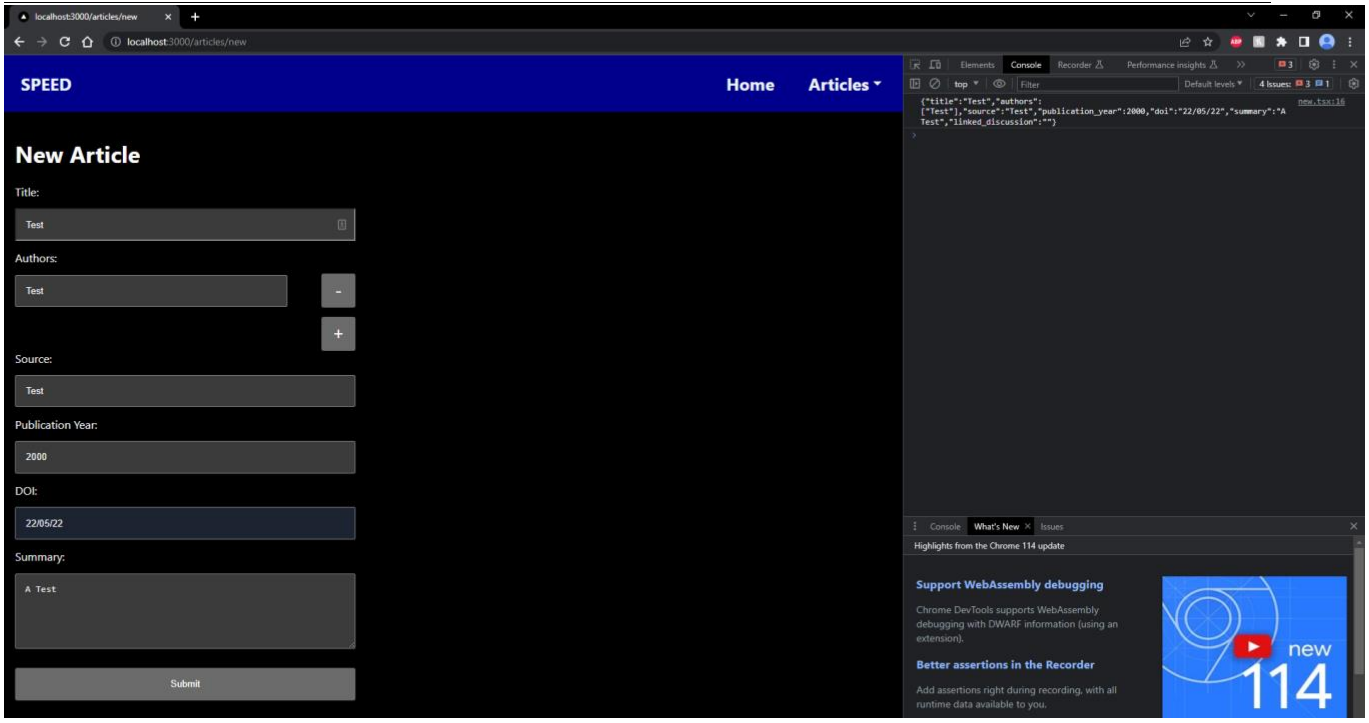
12. Try entering **localhost:3000/blabla** – you should be redirected to your 404 page (remember to start your application first: npm run dev, OR, npm run build THAN npm run start)

Notes for extending this to use in SPEED app.

If you enter some data and press "Submit" it will display the json in the console log which will be returned. In SPEED, this would then be sent to the backend and update the data in the MongoDB database. This is for you to work out for SPEED.

You can use the data in the TDD_Test_Data-3.docx file on Canvas to create some test data in MongoDB Atlas. (see worksheets 2 & 3 for MongoDB Atlas setup)

The frontend and server for SPEED would then be served to us using Vercel (see worksheet 3 for hints)


localhost:3000/articles/new
SPEED
Home
Articles
Elements
Console
Recorder
Performance insights
Default levels
4 Issues: 3 1
What's New
Issues
Highlights from the Chrome 114 update
Support WebAssembly debugging
Chrome DevTools supports WebAssembly debugging with DWARF information (using an extension).
Better assertions in the Recorder
Add assertions right during recording, with all runtime data available to you.
new
114

New Article

Title: Test

Authors: Test

Source: Test

Publication Year: 2000

DOI: 22/05/22

Summary: A Test

Submit

```
{"title":"Test","authors":
["Test"],"source":"Test","publication_year":2000,"doi":"22/05/22","summary":"A
Test","linked_discussion":""}                                        new.tsx:16
```

Name: **Date:**

Worksheet Evidence:

This worksheet requires some answers to questions and **at least three selectively captured screenshots with an in depth reflection** as evidence. The aim is to be able to learn from the exercise, and evidence that.

For each of your three selected screenshots (or sequence of shots) in a brief paragraph or two reflect on:

- Why you have selected it?
- What have you learnt in this part of the worksheet?
- What was new or surprising?
- What useful external resource(s) did you consult and why? Provide a link(s) to the resource.

Evidence to Be uploaded to Canvas as well as Checked off by the Tutor (By end of Week 5)

Evidence	Check
1. You could explain the structure of the worksheet activity. You could paste a screenshot of your folders and files (Do you have the correct file structure set up?)	
2. You could discuss the branches you have created and how they have been used. A screenshot of your GitHub dashboard to show the branches, commits to support your answer may be suitable.	
3. You could discuss how the articles are created, and provide a screenshot of the final localhost:3000/articles/new page with some data in the form and submitted.	
4. You could discuss how the articles are stored, and perhaps provide a screenshot of the final localhost:3000/articles page with the table.	
5. How would you keep track of the status of a submitted article – whether it passed moderation or not, has been analysed and entered into the database or not.	
6. List three user stories for the Moderator	
7. Sketch a Design of the page that will address those user stories and take a photo and paste it here– *Note: – For example, the moderator page could display a selectable list of articles to be moderated and a form to check if the selected article is (a) Not a duplicate already in SPEED (b) is relevant -i.e. about empirical evidence of a claimed benefit for SE practice (c) is from a reputable source (peer reviewed). If it passes all three criteria, it is put into the Analyst's list.*	
8. [Optional] If you want to go further you could discuss how the Vercel deployment workflow is set up, and perhaps provide a screenshot of the SPEED app hosted on Vercel. (See the appendix below).	

Appendix

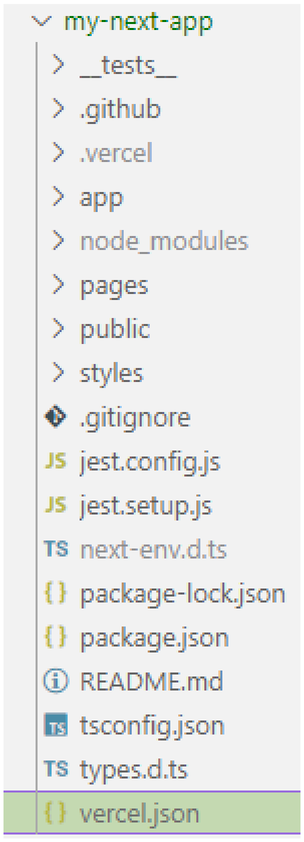
Deploy your app to Vercel and integrate with GitHub Actions for CI/CD

Deploy to Vercel via Vercel CLI

1. Install Vercel CLI if you haven't.

```
npm i -g vercel
```

2. Create a "vercel.json" config file in your app's root folder.



```
my-next-app
  __tests__
  .github
  .vercel
  app
  node_modules
  pages
  public
  styles
  .gitignore
  jest.config.js
  jest.setup.js
  next-env.d.ts
  package-lock.json
  package.json
  README.md
  tsconfig.json
  types.d.ts
  vercel.json
```

You can config your Vercel deployment in this config file. For example, this one is a NextJS app, and Vercel has built-in support for it. So in the config file, we put this:

```
{
    "framework": "nextjs"
}
```

So that Vercel knows this is a NextJS project, and you do not need to config things such as build command (you can if you want to customize it).

If it is something that Vercel does not have built-in framework support, such as NestJS, you will need to config it manually, like this (for NestJS):

```
{
    "version": 2,
    "builds": [
      {
        "src": "dist/main.js",
        "use": "@vercel/node"
      }
    ],
    "routes": [
      {
        "src": "/(.*)",
        "dest": "dist/main.js"
      }
    ]
}
```

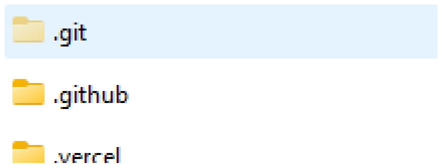
3. In your app’s root folder, run terminal command “vercel”, and follow the instructions to login to your Vercel account and setup your Vercel project. After that your app will be built and deployed to Vercel.

   An example output of the “vercel” command. Your options might be different, so do not follow my choice:

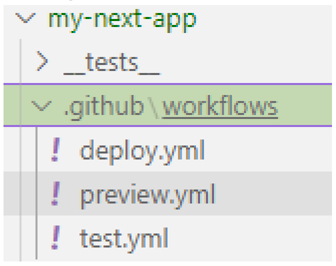
```
Vercel CLI 31.2.3
? Set up and deploy “~\Desktop\TA\New folder\test\my-next-app”? [Y/n] y
? Which scope do you want to deploy to? jcchenaut
? Found project “jcchenaut/my-next-app”. Link to it? [Y/n] n
? Link to different existing project? [Y/n] n
? What’s your project’s name? a-test-proj
? In which directory is your code located? ./
Local settings detected in vercel.json:
No framework detected. Default Project Settings:
- Build Command: `npm run vercel-build` or `npm run build`
- Development Command: None
- Install Command: `yarn install`, `pnpm install`, `npm install`, or `bun install`
- Output Directory: `public` if it exists, or `.`
? Want to modify these settings? [y/N]
```

Integrate with GitHub Actions

1. In your project’s root folder (the one “.git” sits in. In my case, it is the “my-next-app” folder) create a “.github” folder, then create a “workflows” folder inside .github folder.



2. Now we create three .yml files for test, preview deployment and production deployment. You can change the file’s name.



test.yml

```
name: GitHub Actions Test

on:
  push:
    branches: [ "main", "develop" ]
  pull_request:
    branches: [ "main", "develop" ]
  workflow_call:
```

```
  workflow_dispatch:

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - name: Install dependecies
        run: npm install

      - name: Run test
        run: npm run test:ci
```

preview.yml
(change the `jcchenaut/my-next-app/` to your GitHub account and repo)

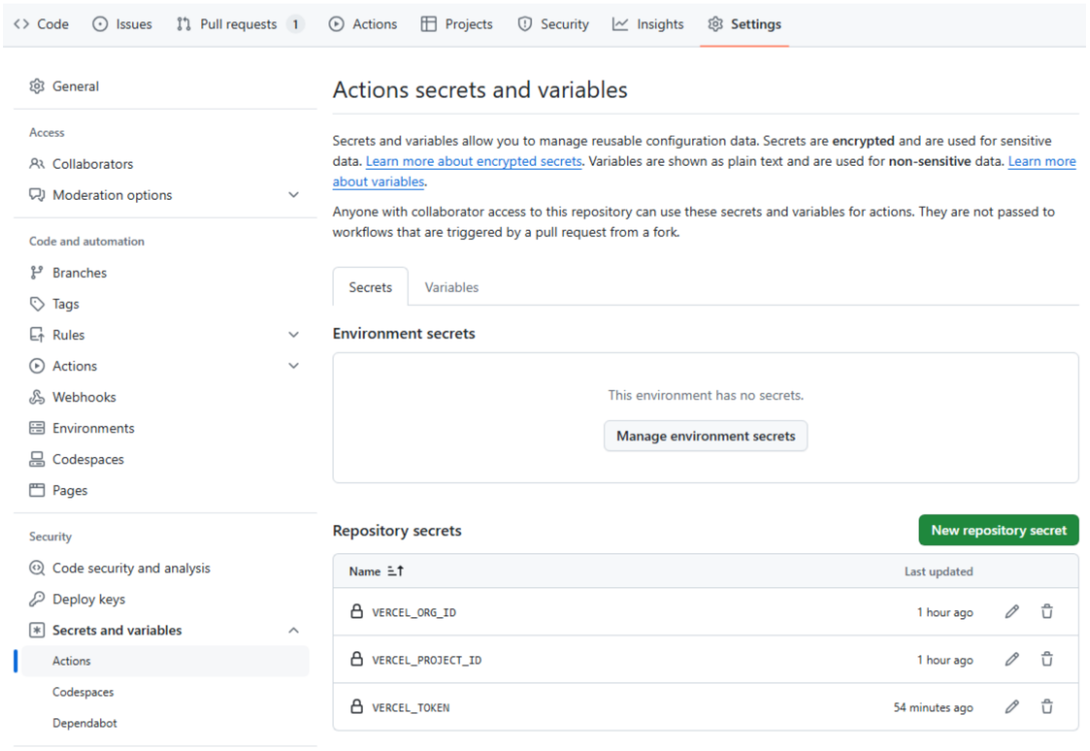
```
name: GitHub Actions Vercel Preview

env:
  VERCEL_ORG_ID: ${{ secrets.VERCEL_ORG_ID }}
  VERCEL_PROJECT_ID: ${{ secrets.VERCEL_PROJECT_ID }}
on:
  workflow_dispatch:
jobs:
  Test:
    uses: jcchenaut/my-next-app/.github/workflows/test.yml@develop

  Deploy-Preview:
    runs-on: ubuntu-latest
    needs: [test]
    steps:
      - uses: actions/checkout@v3
      - name: Install Vercel CLI
        run: npm install --global vercel@canary
      - name: Pull Vercel Environment Information
        run: vercel pull --yes --environment=preview --token=${{ secrets.VERCEL_TOKEN }}
      - name: Build Project Artifacts
        run: vercel build --token=${{ secrets.VERCEL_TOKEN }}
      - name: Deploy Project Artifacts to Vercel
        run: vercel deploy --prebuilt --token=${{ secrets.VERCEL_TOKEN }}
```

deploy.yml
(change the `jcchenaut/my-next-app/` to your GitHub account and repo)

```
name: GitHub Actions Vercel Deploy

env:
  VERCEL_ORG_ID: ${{ secrets.VERCEL_ORG_ID }}
  VERCEL_PROJECT_ID: ${{ secrets.VERCEL_PROJECT_ID }}
on:
  pull_request:
    branches: [ "main" ]
    types:
      - closed
jobs:
  Test:
    if: github.event.pull_request.merged == true
    uses: jcchenaut/my-next-app/.github/workflows/test.yml@develop

  Deploy-Production:
    if: github.event.pull_request.merged == true
    runs-on: ubuntu-latest
    needs: [Test]
    steps:
      - uses: actions/checkout@v3
      - name: Install Vercel CLI
        run: npm install --global vercel@canary
      - name: Pull Vercel Environment Information
        run: vercel pull --yes --environment=production --token=${{ secrets.VERCEL_TOKEN
}}
      - name: Build Project Artifacts
        run: vercel build --prod --token=${{ secrets.VERCEL_TOKEN }}
      - name: Deploy Project Artifacts to Vercel
        run: vercel deploy --prebuilt --prod --token=${{ secrets.VERCEL_TOKEN }}
```

In my workflow, the "test.yml" will be triggered when a push or a pull request has been created to whether main or develop branch. The "preview.yml" can only be executed manually, since I do not want to make a preview deployment every time that I make some changes to the develop branch. The "deploy.yml" will be triggered only when the pull request to the main branch has been accepted, and new codes has been merged into the main branch. You can config these in your own way by modifying the .yml files.

3. In your GitHub repo's Settings page, add secrets:



Actions secrets and variables

Secrets and variables allow you to manage reusable configuration data. Secrets are **encrypted** and are used for sensitive data. Learn more about encrypted secrets. Variables are shown as plain text and are used for **non-sensitive** data. Learn more about variables.

Anyone with collaborator access to this repository can use these secrets and variables for actions. They are not passed to workflows that are triggered by a pull request from a fork.

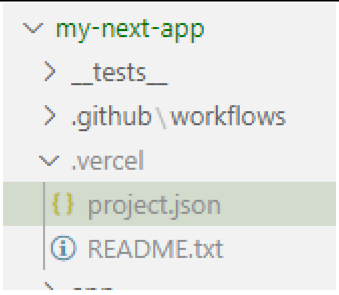
Environment secrets

This environment has no secrets.

Repository secrets

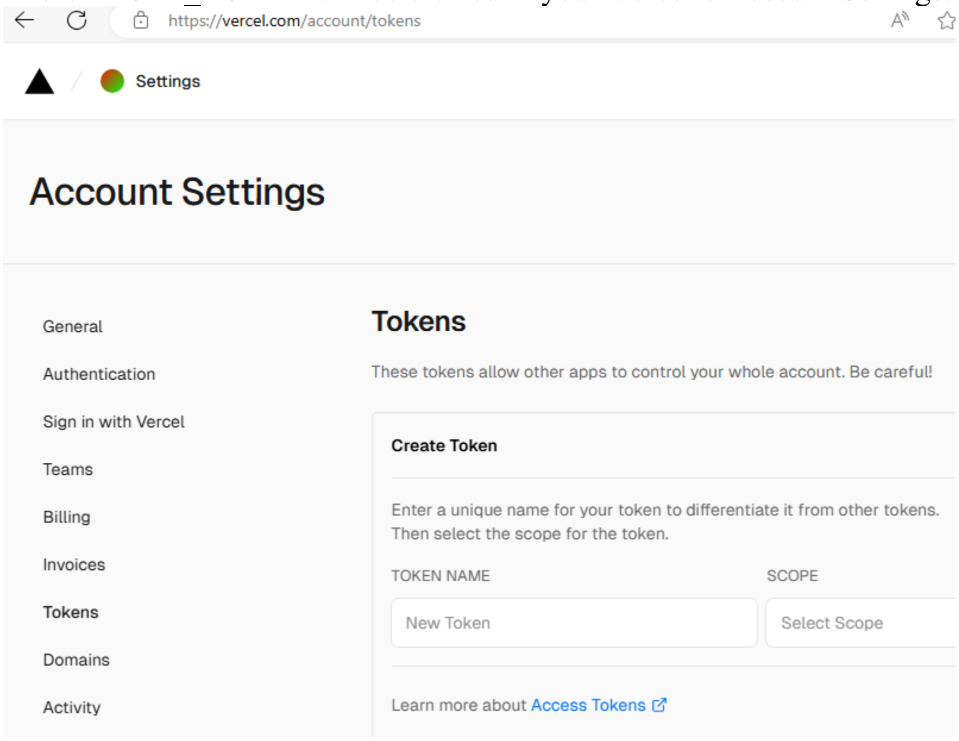
Name	Last updated
VERCEL_ORG_ID	1 hour ago
VERCEL_PROJECT_ID	1 hour ago
VERCEL_TOKEN	54 minutes ago

You can find your VERCEL_ORG_ID and VERCEL_PROJECT_ID in your app's root folder, /.vercel/project.json:



This folder will be created after running the “vercel” command in the previous step.

The VERCEL_TOKEN can be created in your Vercel’s Account Settings:


https://vercel.com/account/tokens

Settings

Account Settings

Tokens

These tokens allow other apps to control your whole account. Be careful!

Create Token

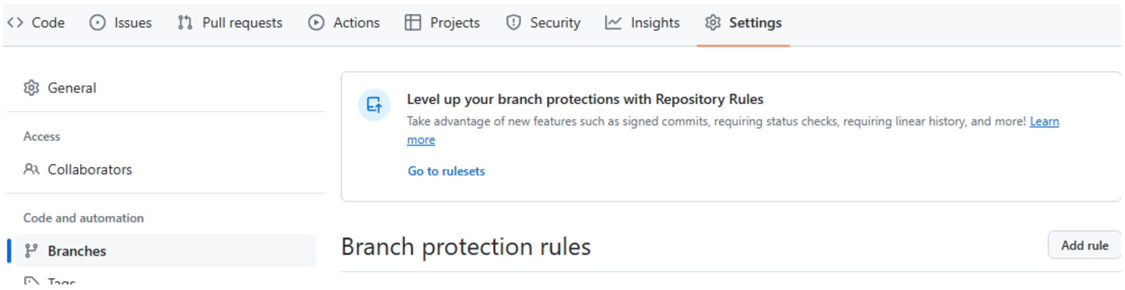
Enter a unique name for your token to differentiate it from other tokens. Then select the scope for the token.

TOKEN NAME: New Token

SCOPE: Select Scope

Learn more about Access Tokens

4. To further protect our main branch from unwanted changes, we can add branch protection rules. Go to the Settings in your GitHub repo:



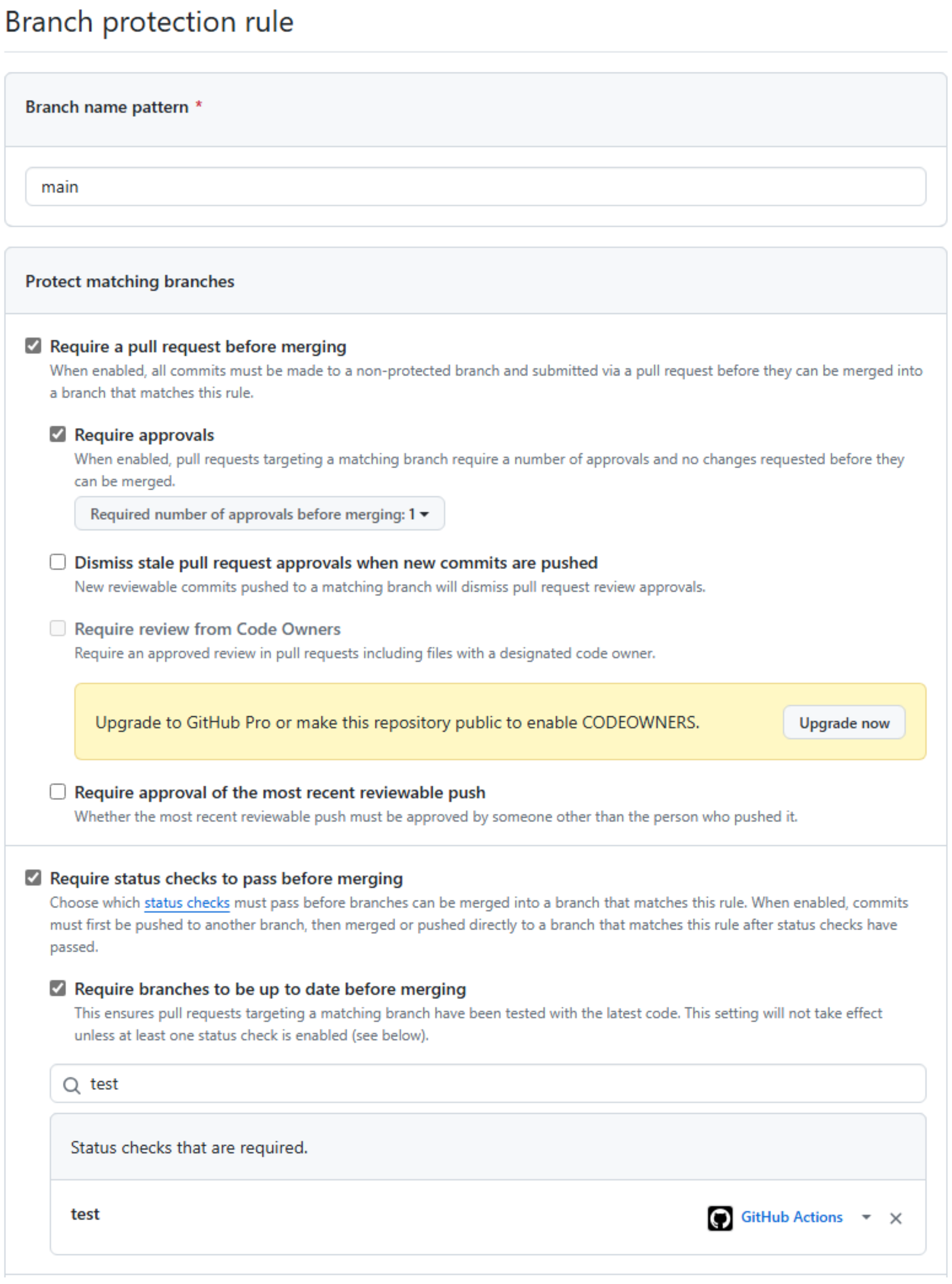
Level up your branch protections with Repository Rules

Take advantage of new features such as signed commits, requiring status checks, requiring linear history, and more! Learn more

Go to rulesets

Branch protection rules

Click “Add rule”, then you can config the protection rules:



Branch protection rule

Branch name pattern *

main

Protect matching branches

☑ **Require a pull request before merging**
When enabled, all commits must be made to a non-protected branch and submitted via a pull request before they can be merged into a branch that matches this rule.

☑ **Require approvals**
When enabled, pull requests targeting a matching branch require a number of approvals and no changes requested before they can be merged.
Required number of approvals before merging: 1

☐ **Dismiss stale pull request approvals when new commits are pushed**
New reviewable commits pushed to a matching branch will dismiss pull request review approvals.

☐ **Require review from Code Owners**
Require an approved review in pull requests including files with a designated code owner.

Upgrade to GitHub Pro or make this repository public to enable CODEOWNERS. Upgrade now

☐ **Require approval of the most recent reviewable push**
Whether the most recent reviewable push must be approved by someone other than the person who pushed it.

☑ **Require status checks to pass before merging**
Choose which status checks must pass before branches can be merged into a branch that matches this rule. When enabled, commits must first be pushed to another branch, then merged or pushed directly to a branch that matches this rule after status checks have passed.

☑ **Require branches to be up to date before merging**
This ensures pull requests targeting a matching branch have been tested with the latest code. This setting will not take effect unless at least one status check is enabled (see below).

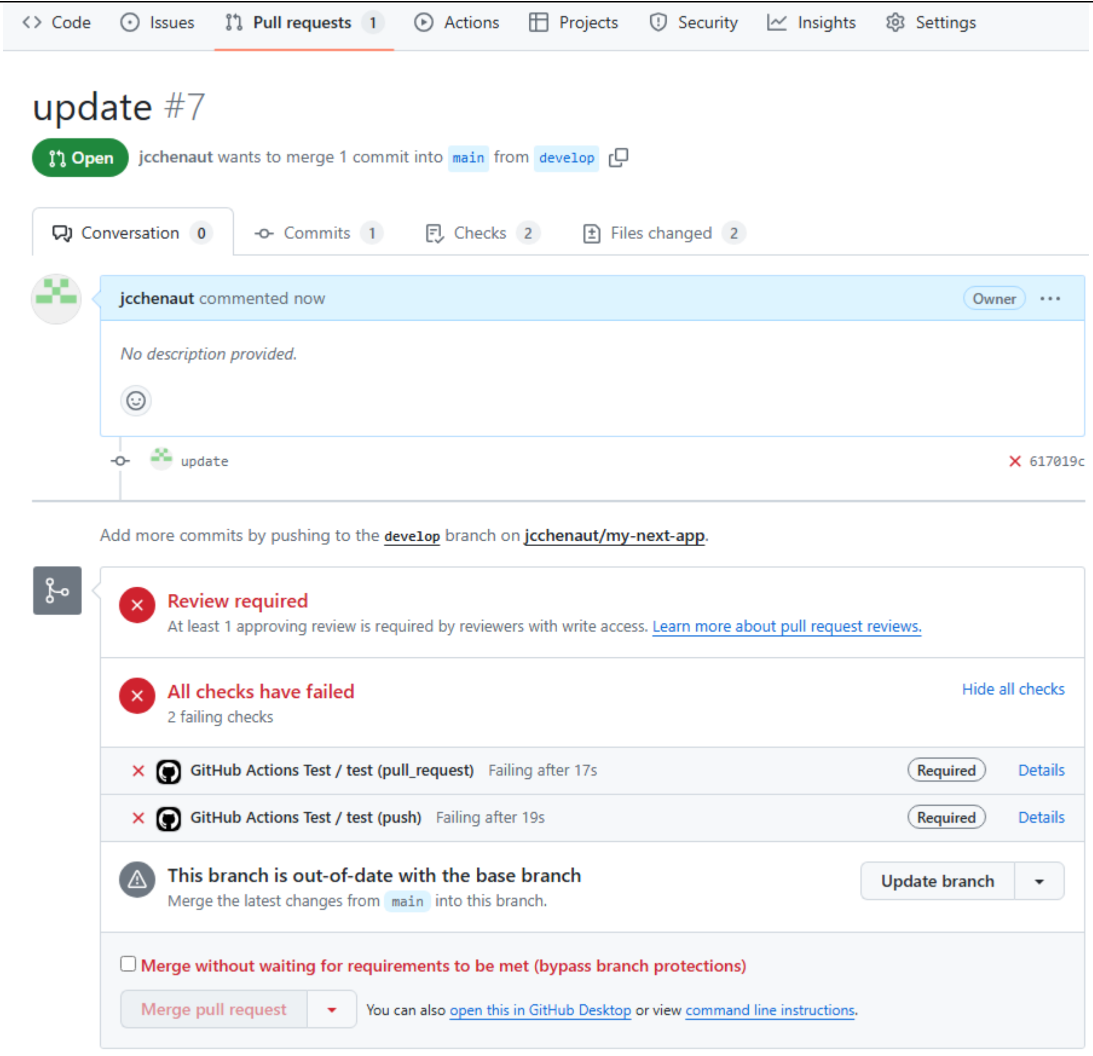
test

Status checks that are required.

test — GitHub Actions

Save the changes when it is done.

5. Now, we can only change the main branch via pull request, and if the test fails, the merge won’t be allowed (your repository must be public or enterprise to enable force protection):



update #7

Open jcchenaut wants to merge 1 commit into main from develop

jcchenaut commented now — Owner

No description provided.

update — ✕ 617019c

Add more commits by pushing to the develop branch on jcchenaut/my-next-app.

Review required
At least 1 approving review is required by reviewers with write access. Learn more about pull request reviews.

All checks have failed — Hide all checks
2 failing checks

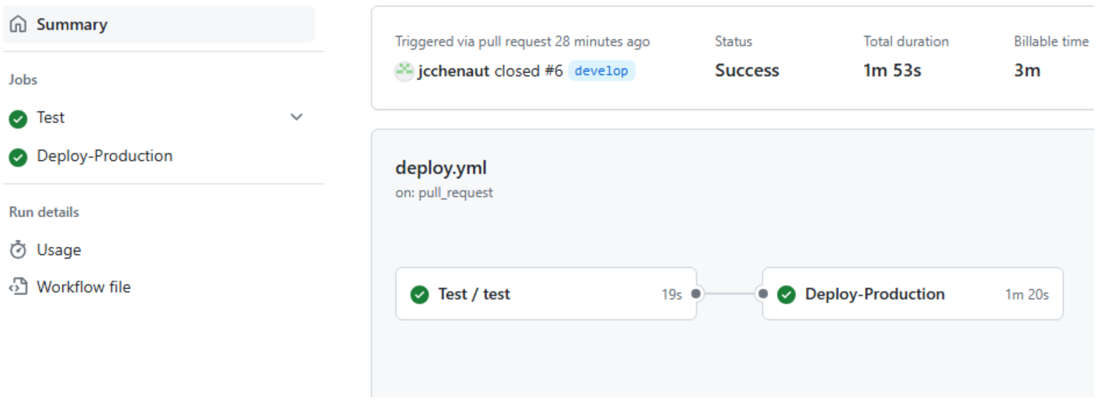
✕ GitHub Actions Test / test (pull_request) Failing after 17s — Required — Details
✕ GitHub Actions Test / test (push) Failing after 19s — Required — Details

This branch is out-of-date with the base branch — Update branch
Merge the latest changes from main into this branch.

Merge without waiting for requirements to be met (bypass branch protections)

You can also open this in GitHub Desktop or view command line instructions.

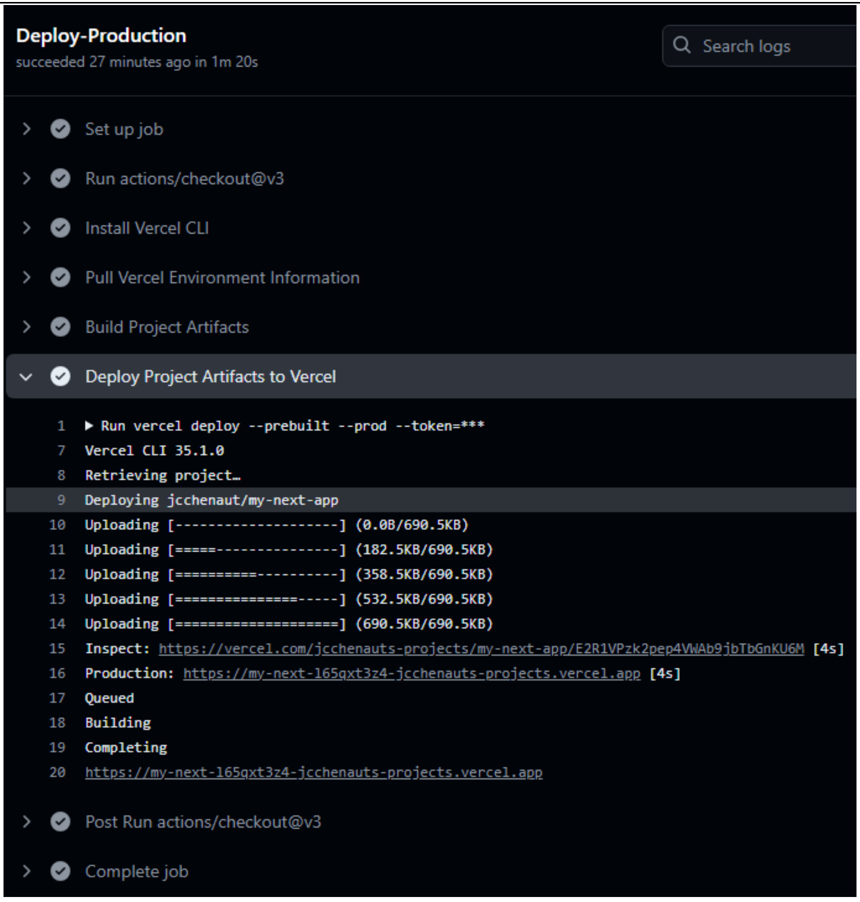
And if the changes have been approved and merged, our production deployment workflow will be triggered, and the app will be deployed to Vercel’s production environment:



Triggered via pull request 28 minutes ago	Status	Total duration	Billable time
jcchenaut closed #6 develop	Success	1m 53s	3m

deploy.yml
on: pull_request

Test / test 19s → Deploy-Production 1m 20s



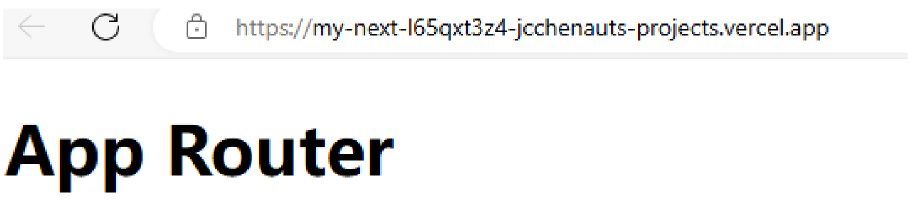
Deploy-Production

succeeded 27 minutes ago in 1m 20s

- Set up job
- Run actions/checkout@v3
- Install Vercel CLI
- Pull Vercel Environment Information
- Build Project Artifacts
- Deploy Project Artifacts to Vercel

```
▶ Run vercel deploy --prebuilt --prod --token=***
Vercel CLI 35.1.0
Retrieving project…
Deploying jcchenaut/my-next-app
Uploading [--------------------] (0.0B/690.5KB)
Uploading [=====---------------] (182.5KB/690.5KB)
Uploading [==========----------] (358.5KB/690.5KB)
Uploading [===============-----] (532.5KB/690.5KB)
Uploading [====================] (690.5KB/690.5KB)
Inspect: https://vercel.com/jcchenauts-projects/my-next-app/E2R1VPzk2pep4VWAb9jbTbGnKU6M [4s]
Production: https://my-next-165qxt3z4-jcchenauts-projects.vercel.app [4s]
Queued
Building
Completing
https://my-next-165qxt3z4-jcchenauts-projects.vercel.app
```

- Post Run actions/checkout@v3
- Complete job



https://my-next-l65qxt3z4-jcchenauts-projects.vercel.app

App Router