**Bachelor of Computer and Information Sciences**

**Contemporary Issues in Software Engineering**

**Semester 2, 2023**

**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 6.**

**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 SEED 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.

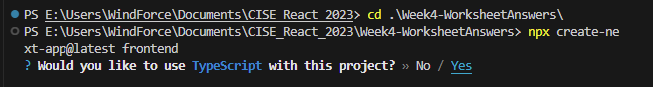
**Overview**

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. Extension deploy with Vercel
8. Open VSCode and Open the CISE\_REPOS folder
9. Create folder “cise\_ass1a\_worksheet4” under the CISE\_REPOS folder
10. In VS Code create folder “backend” under “cise\_ass1a\_worksheet4” folder
11. Open a new terminal (CLI) in folder in cise\_ass1a\_worksheet4” folder
12. In this folder:

> npx create-next-app@latest frontend

*(This assumes you have node.js installed on your machine from previous worksheets)*



You can test using the command – if this runs you are on the right track:

> npm run dev

1. Clean up our project to start from a clean slate. From inside your **frontend** folder, delete everything found inside your **src/app/page.tsx**. Once you've done this, change the page.tsx file that will serve as our app's starting point. Change the code in the **page.tsx** so it has the following contents:

export default function Home() {

  return (

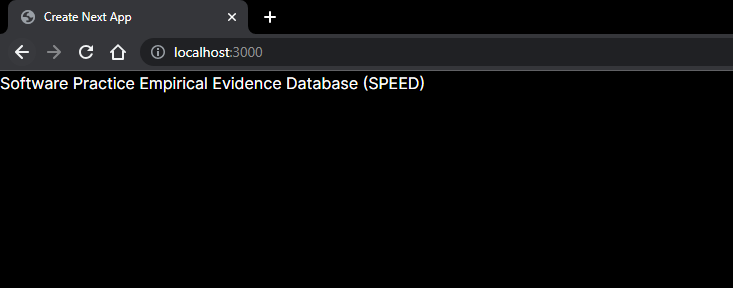
    <div>

      <h1>Software Practice Empirical Evidence Database (SPEED)</h1>

    </div>

  );

}

* The page should now look like this:  
    
  

1. Change the page.tsx file to the following:

import Link from "next/link";

export default function Home() {

  return (

    <div>

      <h1>Software Practice Empirical Evidence Database (SPEED)</h1>

      <br />

      <ul className="header">

        <li>

          <Link href="/">Home</Link>

        </li>

        <li>

          <Link href="/SEPractice">Select the Practice</Link>

        </li>

        <li>

          <Link href="/SubmitArticle">Submit an Article</Link>

        </li>

      </ul>

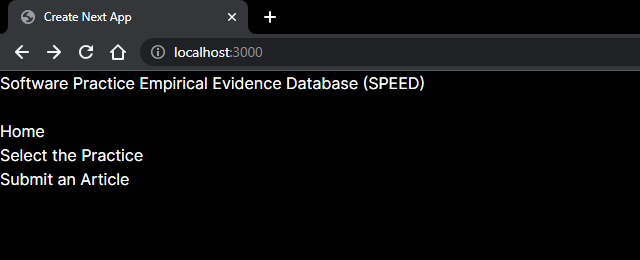
      <div className="content"></div>

    </div>

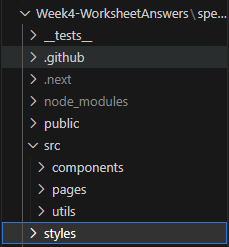
  );

}

You should see the following, but it does nothing yet, there are no routes setup yet!

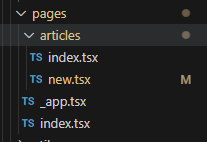


Now we have a basic page lets continue cleaning up the Next.js project. Please create the following so your file structure is also similar:



1. **Create some pages**

Create a “**pages**” folder files in the “**src**” folder and then create the files:



**Copy the following code into the retrospective 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.json";

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.articles.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>

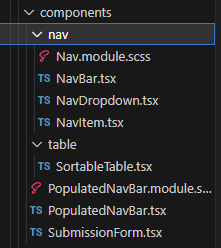
    </div>

  );

};

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)



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

**nav/NavBar.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;

**nav/NavDropdown.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;

**nav/NavItem.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;

**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**

**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**

**PopulatedNavBar.module.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

}

**PopulatedNavBar.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;

**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**](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 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.

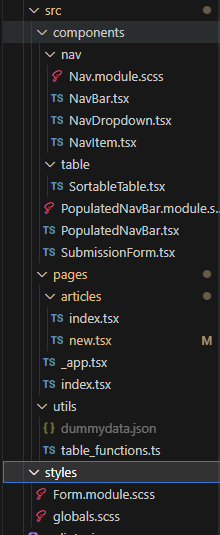
For each link, pay attention to the URL we are telling our router to navigate to. This URL value (defined by the ***to*** prop) acts as an identifier to ensure the right content gets loaded.

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***}/>**

1. 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)



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

[ 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 ]

**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:

1. Try entering **localhost:3000/blabla** – you should be redirected to your 404 page
2. Now set up dummy article data to use on our front end

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

const articles = [

{

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",

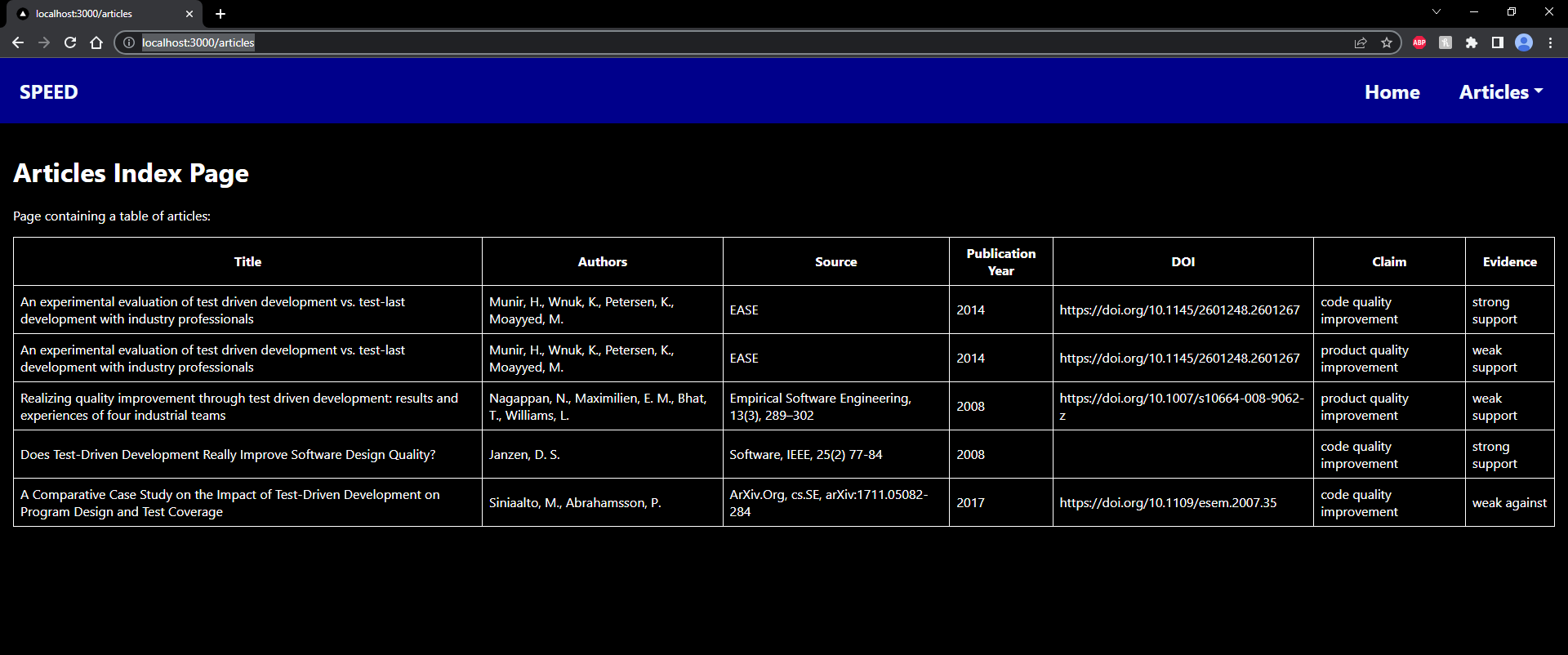
evidence: "weak against",

},

];

module.exports = articles;

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

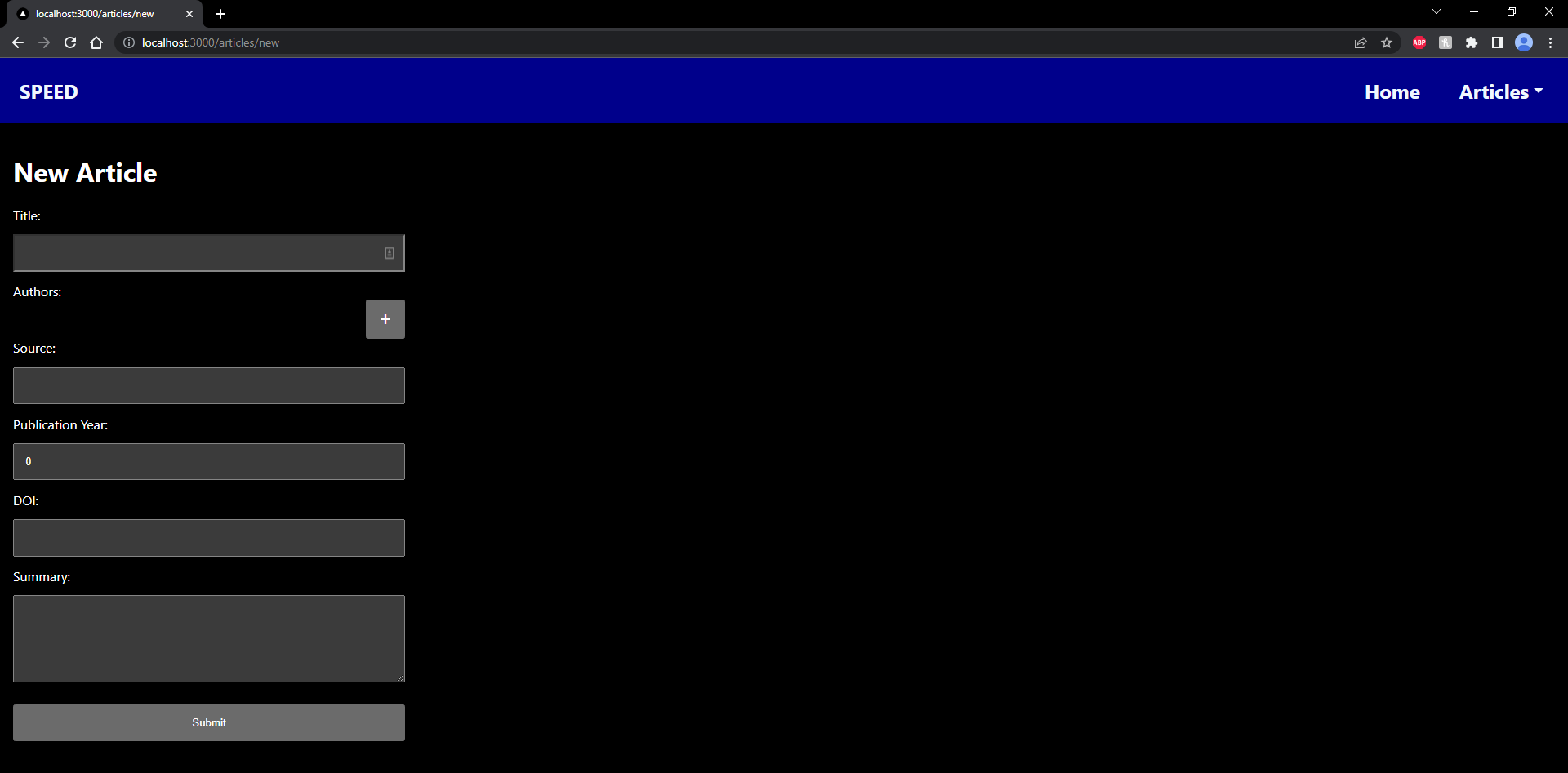


Also the data for the Dropdown list in SEED and the table in SEED should all come from a MongoDB using **axios** to communicate with Nest.js APIs which use mongoose to communicate with MongoDB Atlas. Also for you to work out (look at Worksheet 2 & 3)

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

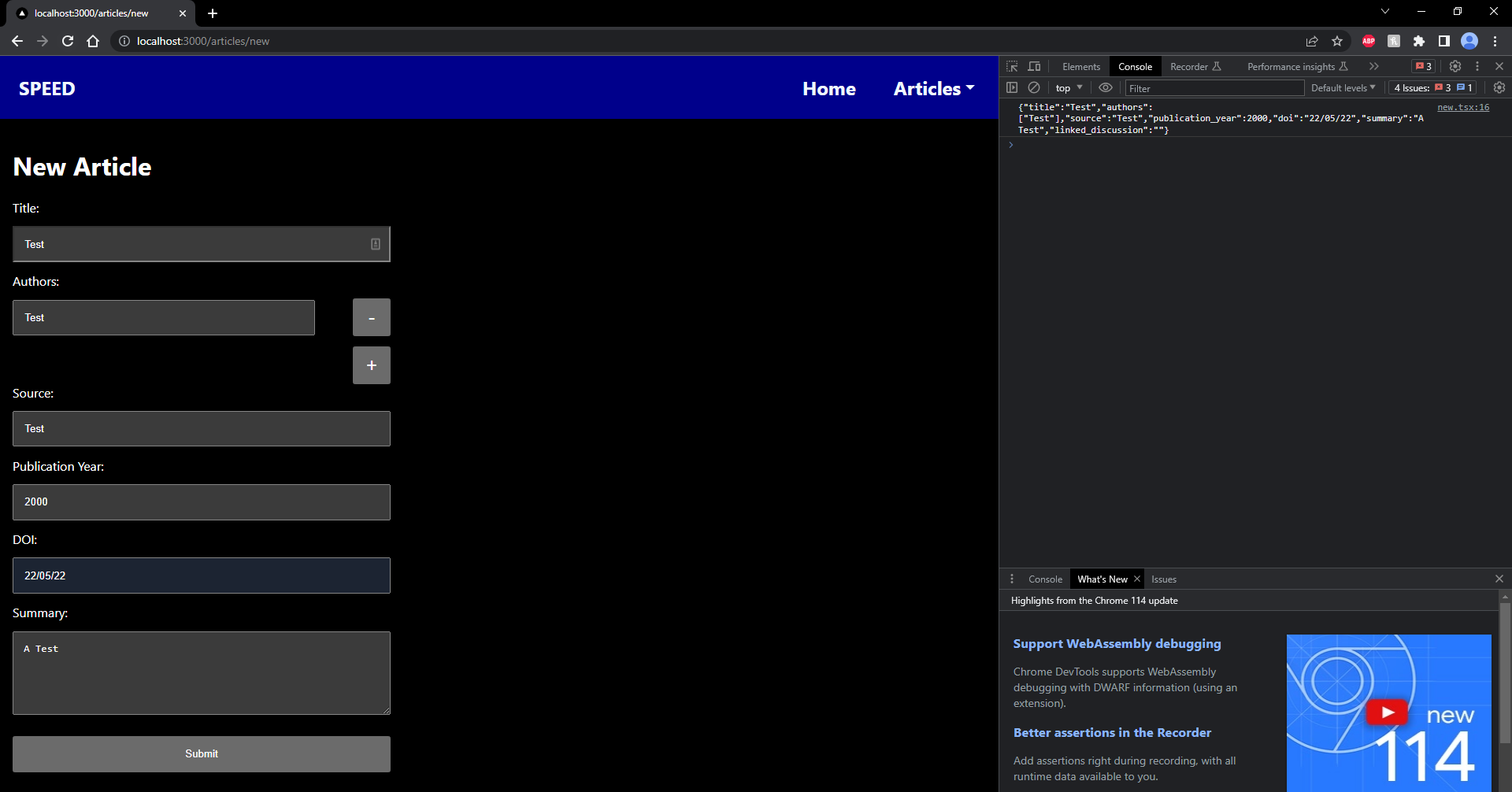


**Notes for extending this to use in SEED app.**

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

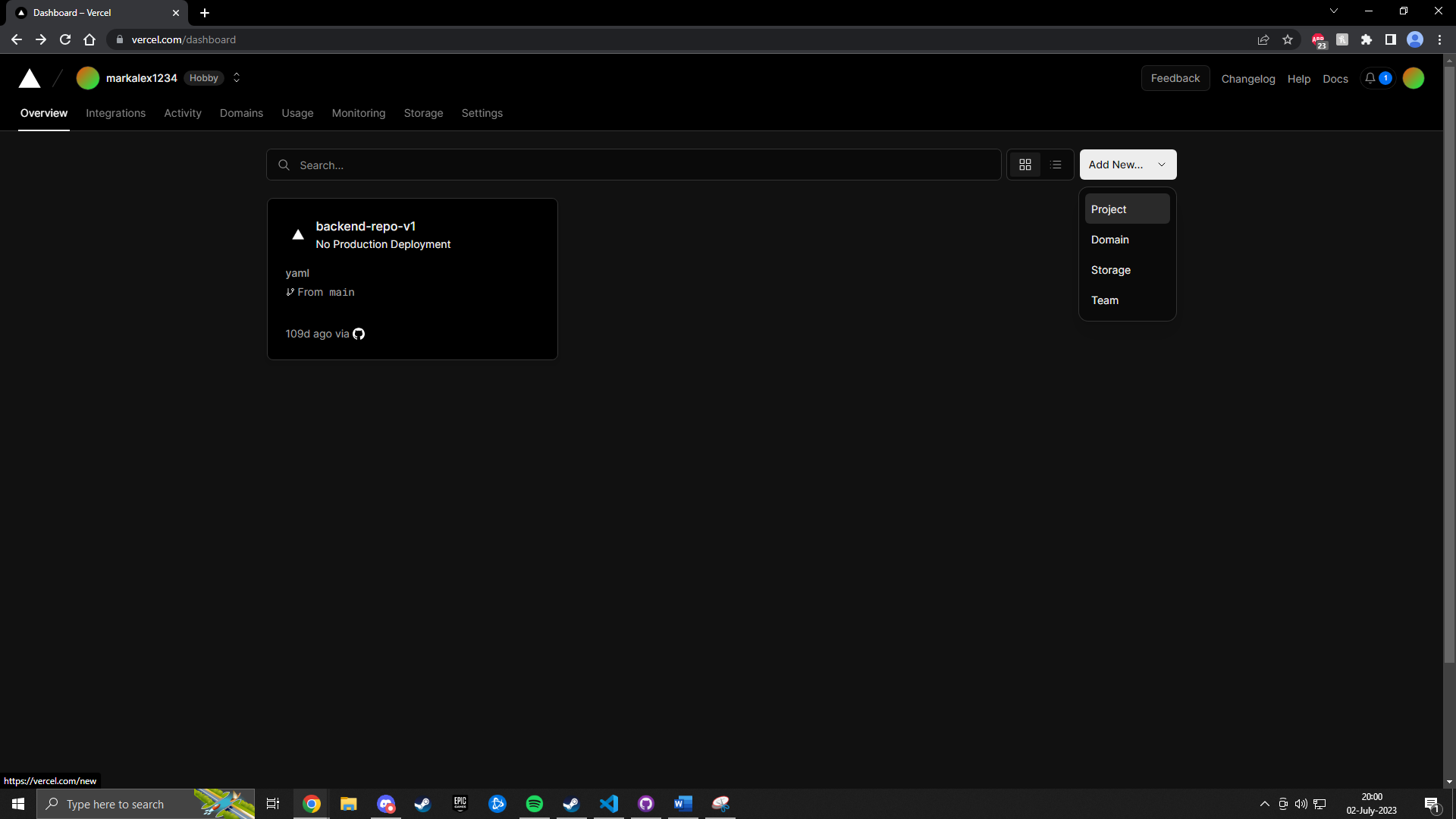
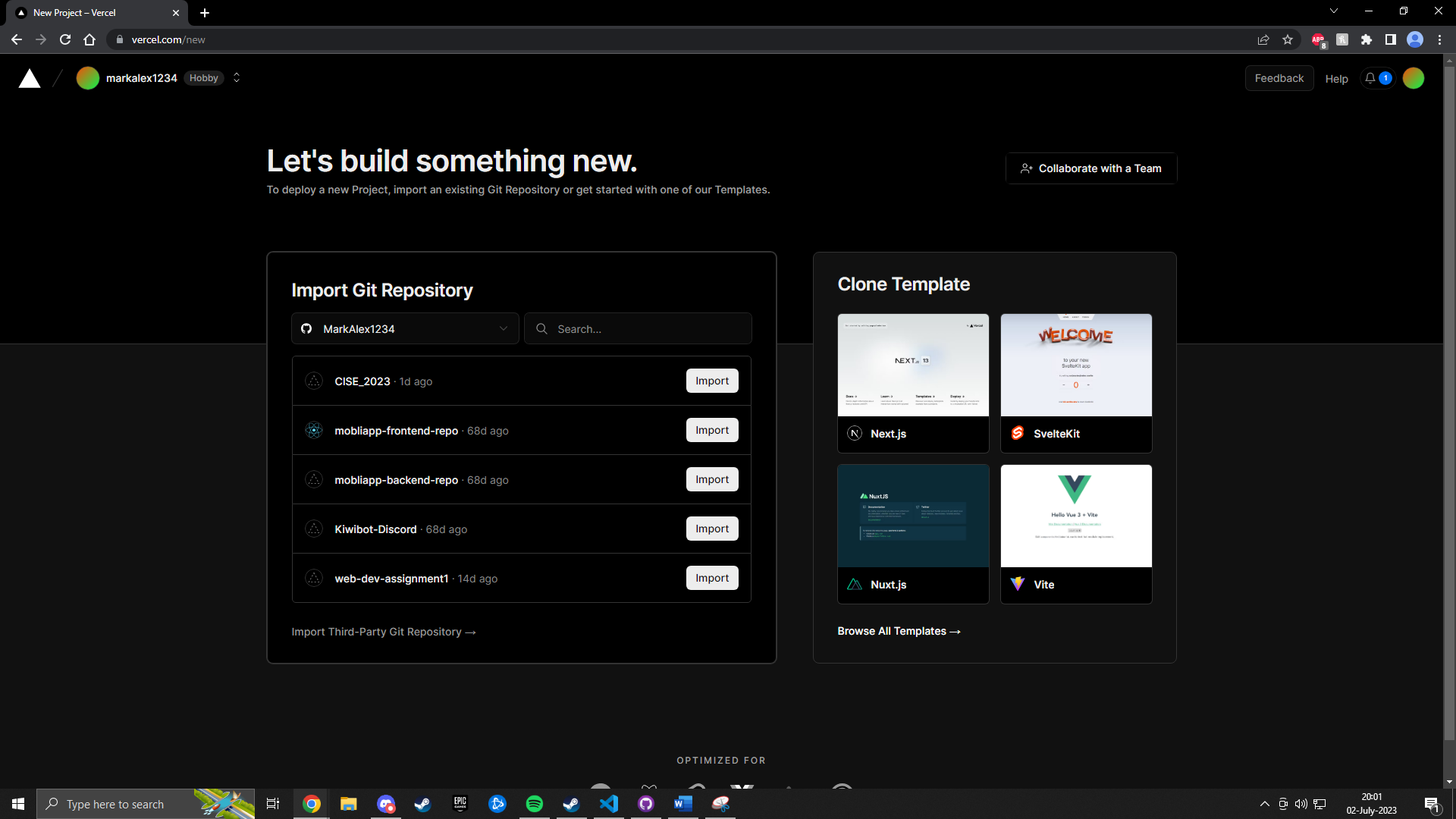
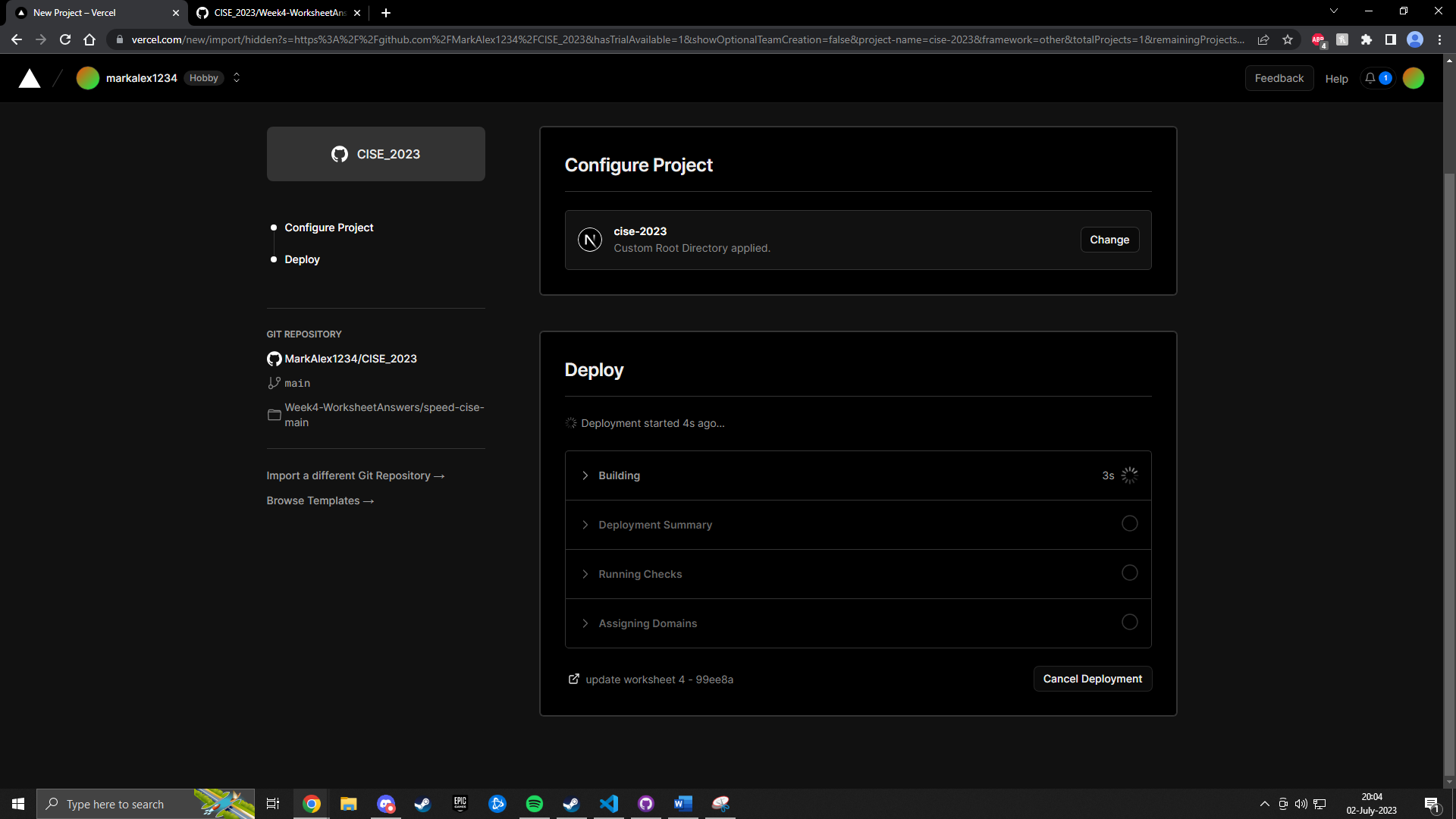
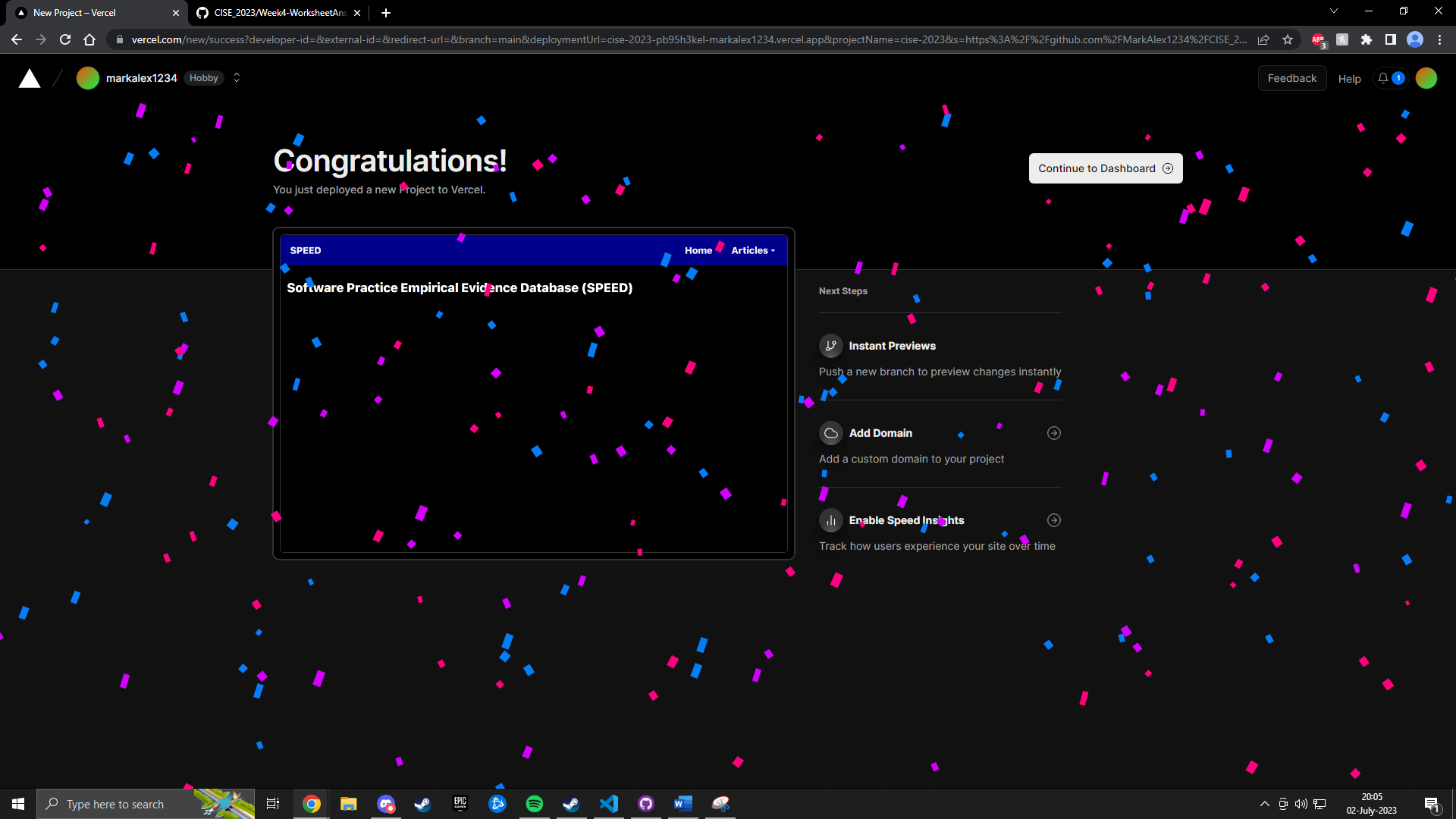
You can use the data in the [TDD\_Test\_Data-1.docx](https://canvas.aut.ac.nz/courses/13657/files/3542063?wrap=1) file on Canvas to create some test data in MongoDB Atlas. (see worksheets 2 & 3 for MongoDB Atlas setup)

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



**Further assistance using Vercel to deploy (optional) :**

So you want to do some extra work? Well done! Let’s deploy this thing to Vercel.

1. Head here and sign up: (I suggest using github as then we can easily select the repo we want to deploy!)  
   <https://vercel.com/login?next=%2Fdashboard>
2. Once in you should see the following, click on project -> add new:
3. Now select a repo you would like to deploy
4. After importing select the root of the project – for me its in week4 answers for you it may be different:
5. And boom! Deployed in 5 easy steps (you can check it out here <https://cise-2023.vercel.app/>):

**Extension Work**

1. Try moving the dummy datafiles to the server and setting up nest.js/node so that the pages still update
2. Try updating the data with a form that was submitted

**Worksheet Evidence Required to be Checked off and Uploaded to Canvas**

Cut and paste screenshots of your web browser showing the evidence needed. As in the previous worksheet include clearly highlighted screen shots of going through each step of the worksheet. Include some written notes beside the screenshots that explain what is being done and anything new you learned or any tricks or mistakes you made

1. Paste a screenshot of your folders and files (Do you have the correct file structure set up?)
2. Paste a screenshot of your GitHub dashboard to show the branches, commits
3. Paste a screenshot of the final localhost:3000/articles/new page with some data in the form and submitted
4. Paste a screenshot of the final localhost:3000/articles page with the table
5. What is the 404 page for?
6. 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.
7. List some user stories for the Moderator
8. Sketch a Design of the page that will address those user stories and take a photo and paste it here–

Note –Example the moderator page should 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 SEED (b) is relevant -About empirical evidence of a claimed benefit for SE practice (c) is from a reputable source (peer reviewed). If passes all three criteria, it is put into the Analyst’s list