Міністерство освіти і науки України
Національний університет “Львівська політехніка”

Кафедра прикладної математики

КУРСОВА РОБОТА

з курсу «Програмування web-додатків»

Виконав:
студент групи ПМ-33
Селіванов К.О.

Прийняв:
ст. викладач каф. ПМ
Топилко П.І.

Львів - 2022

Створений односторінковий (типу SPA) веб-застосунку типу планера на React з використанням Nexj.js. Реалізував навігаційну панель з переходом на сторінки сайту. Отримання та вималювання даних з json-файлу для імітації отримання даних з backend.

Created a one-way (SPA type) web application planner on React using Next.js. Implemented a navigation bar to go to the site pages. Providing and extracting data from json-file, imitating receiving data from backend.

Зміст

Вступ 4
Розділ 1 5
React 5
Bootstrap 6
TypeScript 7
Next.js 8
Розділ 2 9
Створеня проекта 9
Розбитя на компоненти 10
Компонента Home з усіма компонентами 11
Налаштування роутингу 17
Перевірка роутингу 17
Отримання даних з data.json 18
Перевірка 19
Візуальний вигляд сайту: 21
Висновок 24
Посилання 25

Вступ

Створити сайт певного планера (придумати назву та емблему). Сайт має бути типу SPA(односторінковий). Сайт має бути зверстаний відповідно до дизайну з .psd файлу. Сайдбар містить наступні кнопки, які забезпечують перехід на інші сторінки:

- Home
- Time Management
- Inbox
- Calendar
- Analytics
- Settings

Щоб перейти на сторінку 404, варто вказати шлях, який не є одним із вище вказаних.

Розділ 1

React

React — це декларативна, ефективна і гнучка JavaScript-бібліотека, призначена для створення інтерфейсів користувача. Вона дозволяє компонувати складні інтерфейси з невеликих окремих частин коду — “компонентів”. Розробляється Facebook, Instagram(Meta) і спільнотою індивідуальних розробників.

React дозволяє розробникам створювати великі вебзастосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC, таких як AngularJS. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови вебзастосунків. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з іншими бібліотеками, такими як Redux.

Особливістю бібліотеки є HTML-подібний синтаксис для створення власних компонент, який називається JSX.

React підтримує віртуальний DOM, а не покладається виключно на DOM браузера. Це дозволяє бібліотеці визначити, які частини DOM змінилися, порівняно (diff) зі збереженою версією віртуального DOM, і таким чином визначити, як найефективніше оновити DOM браузера. Таким чином програміст працює зі сторінкою, вважаючи що вона оновлюється вся, але бібліотека самостійно вирішує які компоненти сторінки треба оновити.

React не намагається надати повну «схему додатків». Він безпосередньо спрямований на побудову користувацьких інтерфейсів, і тому не включає в себе безліч інструментів, які деякі розробники вважають необхідними для створення програми. Це дозволяє вибрати будь-які бібліотеки, які розробник вважає за краще виконувати, щоб виконати певних завдань, таких як здійснення доступу до мережі або локальне зберігання даних.

Також бібліотека підтримує лише односторонню передачу даних, що означає, що змінні, оголошені в дочірній компоненті, не можуть бути використані в батьківській.

Існує два типи компонент:

- Класові
- Функційні

На сьогодні більш прийнятим є використання саме функційних компонент, які по суті є звичайними функціями, що вертають React вузли. Такий підхід є значно

зручнішим, адже є ближчим до самої суті JS, як функціональної мови програмування, та завдяки використанню спеціальних функцій – хуків, значно спрощує керування життєвим циклом компоненти.

Завдяки хукам зникає необхідність у використанні HOC(High Order Component) – компонент вищого порядку, якими огортають вже існуючі компоненти для надання їм додаткового функціоналу.

Основними хуками є

- useState
- useEffect
- useMemo
- useCallback
- useReducer
- useContext

На основі вбудованих хуків можна створити власні.

Для спілкування компонент між собою використовують атрибути – кожна створена компонента може мати власний набіл своїх атрибутів, подібних до тих, що існують у HTML-тегах.

Bootstrap

Bootstrap — це безкоштовний набір інструментів з відкритим кодом, призначений для створення вебсайтів та вебдодатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних вебсайтів і вебдодатків.

Bootstrap — це клієнтський фреймворк, тобто інтерфейс для користувача, на відміну від коду серверної сторони, який знаходиться на сервері. Репозиторій із цим фреймворком є одним із найпопулярніших на GitHub. Серед інших, його використовують NASA і MSNBC.

Bootstrap (початкова назва — Twitter Blueprint) був розроблений Марком Отто та Джейкобом Торнтоном як фреймворк для забезпечення однаковості внутрішніх інструментів Twitter. До появи Bootstrap у розробці інтерфейсу застосовувалися різні бібліотеки, що призводило до появи суперечностей та ускладнювало супровід.

Через кілька місяців до розробки рішення долучилося багато розробників компанії Twitter. Проект було перейменовано з Twitter Blueprint на Bootstrap. Реліз із відкритим сирцевим кодом вийшов 19 серпня 2011 року. Нині проект

підтримується групою розробників на чолі з Марком Отто та Джейкобом Торнтоном, а також широкою спільнотою прихильників.

Спеціально для React було створення React Bootstrap – фреймворк, звичайні елементи замінені JSX-елементами, що полегшує використання Bootstrap з React.

TypeScript

TypeScript — мова програмування, представлена Microsoft восени 2012; позиціонується як засіб розробки вебзастосунків, що розширює можливості JavaScript.

Розробником мови TypeScript є Андерс Гейлсберг (англ. Anders Hejlsberg), який створив раніше C#, Turbo Pascal і Delphi.

Код експериментального компілятора, котрий транслює код TypeScript в представлення JavaScript, поширюється під ліцензією Apache, розробка ведеться в публічному репозиторії через сервіс CodePlex. Специфікації мови відкриті і опубліковані в рамках угоди Open Web Foundation Specification Agreement (OWFa 1.0).

TypeScript є зворотньо сумісним з JavaScript. Фактично, після компіляції програму на TypeScript можна виконувати в будь-якому сучасному браузері або використовувати спільно з серверною платформою Node.js.

Переваги над JavaScript:

- можливість явного визначення типів (статична типізація),
- підтримка використання повноцінних класів (як в традиційних об'єктно-орієнтованих мовах),
- підтримка підключення модулів.

За задумом ці нововведення мають підвищити швидкість розробки, прочитність, рефакторинг і повторне використання коду, здійснювати пошук помилок на етапі розробки та компіляції, а також швидкодію програм.

Планується, що в силу повної зворотної сумісності адаптація наявних застосунків на нову мову програмування може відбуватися поетапно, шляхом поступового визначення типів. Підтримка динамічної типізації зберігається — компілятор TypeScript успішно обробить і не модифікований код на JavaScript.

Основний принцип мови — будь-який код на JavaScript сумісний з TypeScript, тобто в програмах на TypeScript можна використовувати стандартні JavaScript-бібліотеки і раніше створені напрацювання. Більш того, можна залишити наявні JavaScript-проекти в незмінному вигляді, а дані про типізації розмістити у вигляді анотацій, які можна помістити в окремі файли, які не заважатимуть розробці і прямому використанню проекту (наприклад, подібний підхід зручний при розробці JavaScript-бібліотек).

Next.js

Відкритий JavaScript фреймворк, створений як надбудова для React для створення веб-застосунків. Створений компанією Vercel. Фреймворк покликаний для вирішення проблеми рендерингу звичайного React на стороні серверу – SSR. Працює на сервері та у браузері.

Server Side Rendering дозволяє знизити навантаження на пристрій, що використовує застосунок, оскільки більшість операцій, пов’язаних з рендерингом, відбуваються на сервері, а не на пристрої користувача. Це може призвести до збільшення продуктивності веб-застосунку.

Окрім SSR, Next.js оптимузіє завантаження шрифтів, зображень, спрощує створення навігації по сайту, та дає можливість створення динамічний шляхів.

Розділ 2

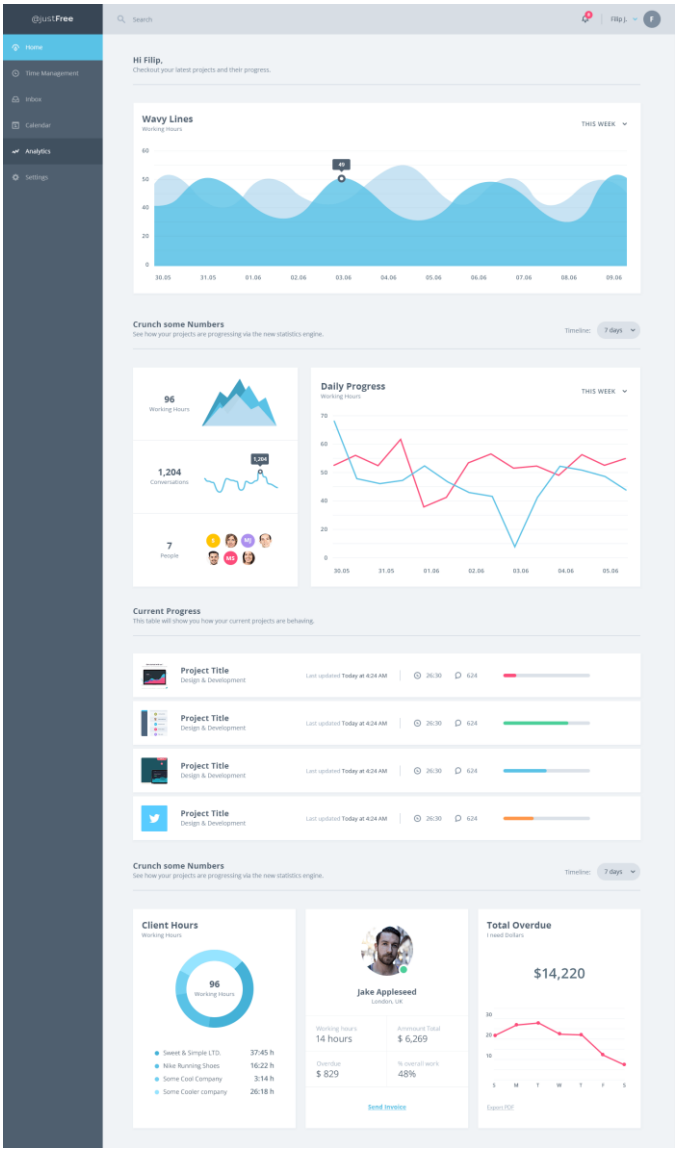

Рисунок 1 Макет дизайну web-сайту

Створеня проекта

```
$ npx create-next-app@latest --ts
```

```
$ npm i sass
```

```
$ npm run dev

> course-paper@0.1.0 dev
> next dev

ready - started server on 0.0.0.0:3000, url: http://localhost:3000
event - compiled client and server successfully in 1242 ms (130 modules)
```

Розбиття проекту

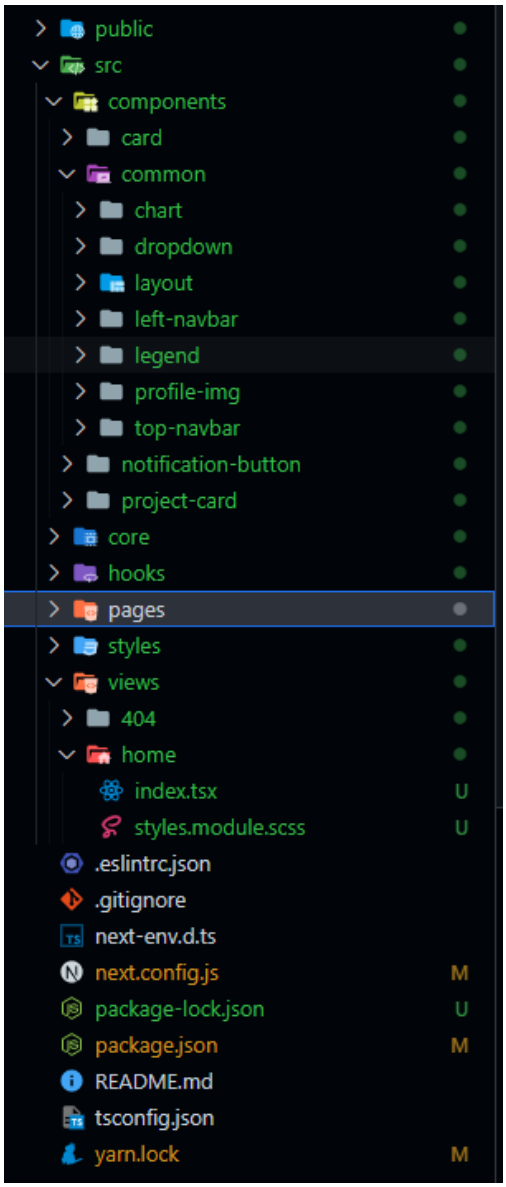

Рисунок 2 Розбиття проекту

Розбиття на pages та views дозволяє відділити вигляд сторінки та реалізацію функціоналу.

Розбитя на компоненти

Рисунок 3 Компоненти

UML-діаграма

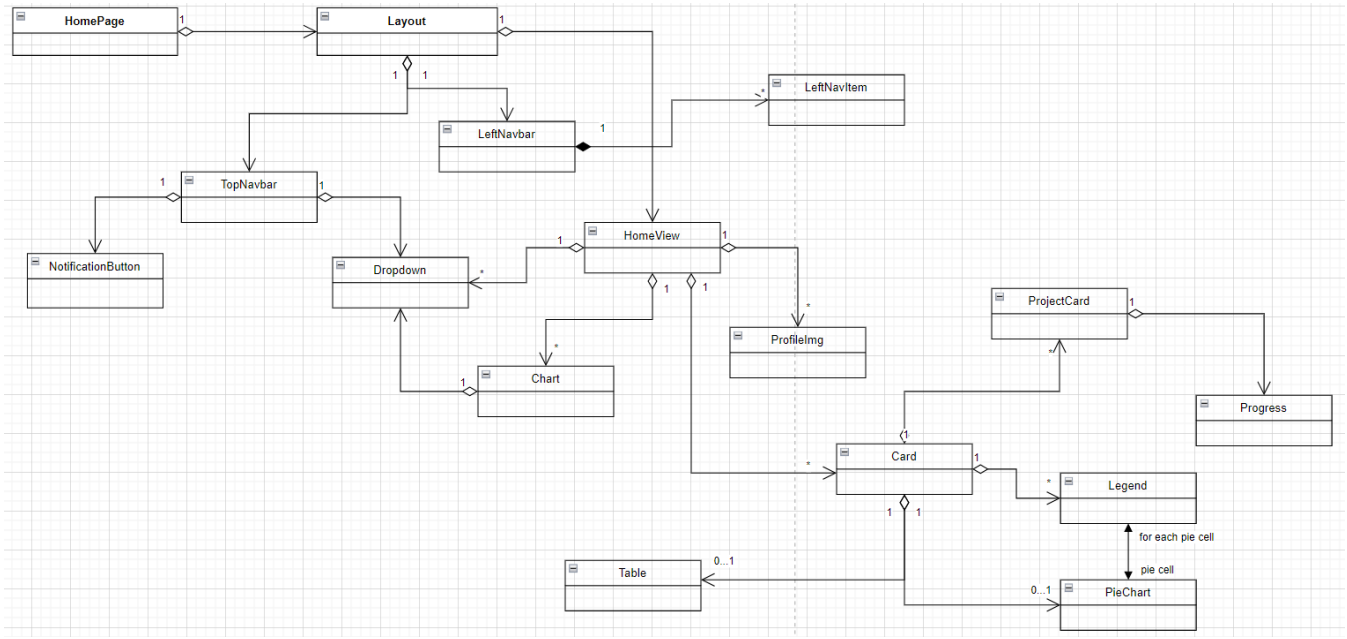

Рисунок 4 UML-діаграма

Компонента Home з усіма компонентами

```
const Home: React.FC<Props> = ({
  dataForGraph1,
  dataForGraph3,
  dataForGraph4,
  dataForLastGraph,
  clientHours,
  projects,
  totalInLastGraph,
  clientInformation,
  photoUrl,
```

```
}) => {
  const [pieColors, setPieColors] = useState<string[]>([]);

  useEffect(() => {
    setPieColors(clientHours.map(() => generateRandomColor()));
  }, [clientHours]);

  return (
    <div>
      <section className={styles.section}>
        <h2 className={styles["first-h"]}>Hi Fillip,</h2>
        <p className={styles["first-p"]}>
          Checkout your latest projects and their progress
        </p>
        <div className={styles.separator} />
        <Chart
          data={dataForGraph1}
          type="area"
          withDropdown
          title="Wavy Lines"
          subtitle="Working Hours"
          className={styles["first-chart"]}
        />
      </section>
      <section className={styles.section}>
        <div className={styles.flex}>
          <div className={styles.left}>
            <h2 className="h2">Crunch some Numbers</h2>
            <p className="p">
              See how your projects are progressing via the new statistics
              engine
            </p>
          </div>

          <div className={styles.right}>
            <span className={styles.label}>Timeline:</span>
            <Dropdown
              type="dropdown-timeline"
              onItemClick={() => {
                return;
              }}
            />
          </div>
        </div>
        {/* SEPARATOR */}
```

```
        <div className={styles.separator} />
        <div className={styles.container}>
          <div className={styles["left-graphs"]}>
            <Chart
              data={dataForGraph3}
              type="area"
              lineType="linear"
              title="96"
              subtitle="Working Hours"
              hideAxis
              hideGrid
              hideTooltip
              contentDirection="horizontal"
              className={styles["chart-small"]}
            />

            <Chart
              data={dataForGraph4}
              type="line"
              title="1,204"
              subtitle="Conversations"
              hideAxis
              hideGrid
              hideTooltip
              contentDirection="horizontal"
              className={styles["chart-small"]}
            />

            <div className={classNames(styles["chart-small"], styles.people)}>
              <div className={styles["text-container"]}>
                <h2>7</h2>
                <p>People</p>
              </div>
              <div className={styles["people-container"]}>
                <ProfileImg />
                <ProfileImg />
                <ProfileImg />
                <ProfileImg />
                <ProfileImg />
                <ProfileImg />
                <ProfileImg />
                <ProfileImg />
                <ProfileImg />
                <ProfileImg />
                <ProfileImg />
```

```
          </div>
        </div>
      </div>

      <Chart
        type="line"
        lineType="linear"
        data={dataForGraph1}
        title="Daily progress"
        subtitle="Working Hours"
        withDropdown
        className={styles.chart}
      />
    </div>
  </section>
  <section className={styles.section}>
    <h2 className="h2">Current Progress</h2>
    <p className="p">
      This table show you how your current projects are behaving
    </p>
    <div className={styles.separator} />
    <div className={styles["projects"]}>
      {projects.map((project, index) => (
        <ProjectCard key={index} data={project} />
      ))}
    </div>
  </section>
  <section className={styles.section}>
    <div className={styles.flex}>
      <div className={styles.left}>
        <h2 className="h2">Crunch some Numbers</h2>
        <p className="p">
          See how your projects are progressing via the new statistics
          engine
        </p>
      </div>

      <div className={styles.right}>
        <span className={styles.label}>Timeline:</span>
        <Dropdown
          type="dropdown-timeline"
          onItemClick={() => {
            return;
          }}
        />
```

```
        </div>
      </div>
      <div className={styles.separator} />
      <div className={styles.cards}>
        <Card>
          <div className={styles["card-top"]}>
            <h3>Client Hours</h3>
            <p>Working hours</p>
          </div>
          <ResponsiveContainer width={181} height={181}>
            <PieChart>
              <Pie
                data={clientHours}
                dataKey="value"
                innerRadius={65}
                outerRadius={90.5}
              >
                <Label
                  content={(props) => {
                    const fontSize = 16;

                    const {
                      /*  @ts-ignore  */
                      viewBox: { cx, cy },
                    } = props;
                    const positioningPropsValue = {
                      x: cx,
                      y: cy - fontSize,
                      textAnchor: "middle",
                      verticalAnchor: "middle",
                    };

                    const positioningPropsText = {
                      x: cx,
                      y: cy + fontSize / 2,
                      textAnchor: "middle",
                      verticalAnchor: "middle",
                    };
                    return (
                      <>
                        {/*  @ts-ignore  */}
                        <Text {...positioningPropsValue}>{`${Math.round(
                          clientHours.reduce(
                            (prev, curr) => prev + curr.value,
                            0
```

```
                    )
                  )}`}</Text>
                  {/*  @ts-ignore  */}
                  <Text {...positioningPropsText}>Working Hours</Text>
                </>
              );
            }}
          />

          {clientHours.map((entry, index) => (
            <Cell key={`cell-${index}`} fill={pieColors[index]} />
          ))}
        </Pie>
      </PieChart>
    </ResponsiveContainer>
    <div className={styles.legends}>
      {clientHours.map((entry, index) => (
        <Legend
          key={entry.companyName}
          name={entry.companyName}
          value={entry.value}
          color={pieColors[index]}
        />
      ))}
    </div>
  </Card>
  <Card noSidePadding>
    <div className={styles["card-user-photo-container"]}>
      {photoUrl && (
        <Image src={photoUrl} width={121} height={121} alt=""></Image>
      )}
      <div
        className={classNames(
          styles.indicator,
          clientInformation.online ? styles.online : styles.offline
        )}
      />
    </div>
    <div className={styles["user-info"]}>
      <h3>{clientInformation.fullname}</h3>
      <p>{clientInformation.location}</p>
    </div>
    <Table clientInformation={clientInformation} />

    <button className={styles["send-invoice"]}>Send invoice</button>
```

```
          </Card>
          <Card>
            <Chart
              className={styles["last-graph"]}
              data={dataForLastGraph}
              type="line"
              lineType="linear"
              title="Total overdue"
              subtitle="I need dollars"
              strokeWidth={3}
              showDots
              someBigText={`${totalInLastGraph}$`}
              exportPdf
            />
          </Card>
        </div>
      </section>
    </div>
  );
};

export default Home;
```

Налаштування роутингу

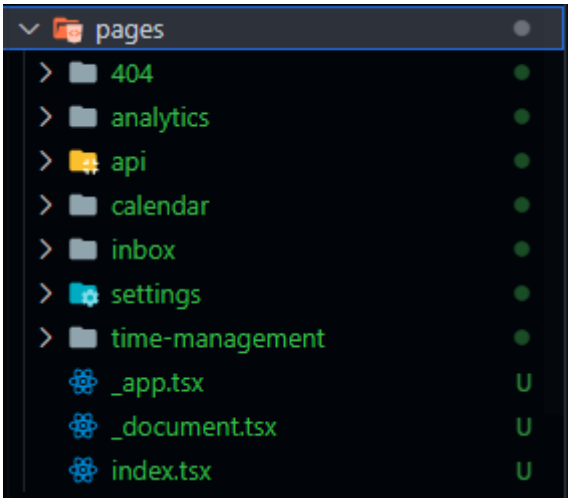

Рисунок 5 Шляхи вебсайту

Для налаштування роутингу достатньо просто створити папки з відповідною назвою в директорії /pages

Перевірка роутингу

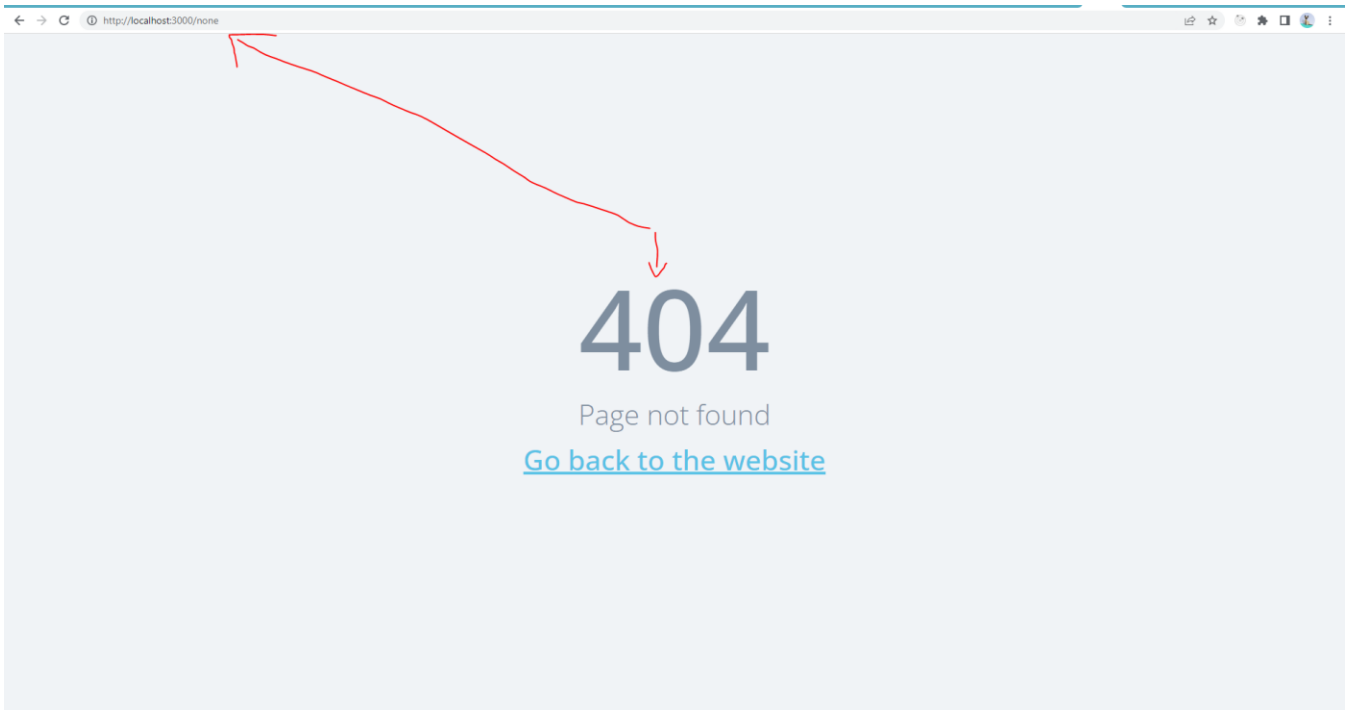

Рисунок 6 Шлях, що не існує (404)

Отримання даних з data.json

```
import type { NextPage } from "next";
import { useEffect, useState } from "react";

import Layout from "../components/common/layout";
import { pageData } from "../core/types";
import useInterval from "../hooks/use-interval";

import HomeView from "../views/home/index";

const Home: NextPage = () => {
  const [userPhotoUrl, setUserPhotoUrl] = useState("");

  const [data, setData] = useState<pageData | null>(null);

  useEffect(() => {
    fetch("https://random.imagecdn.app/121/121").then((res) =>
      setUserPhotoUrl(res.url)
    );

    fetchData().then((data) => setData(data));
  }, []);

  useInterval(async () => {
    const data = await fetchData();
    setData(data);
```

```
  }, 60000);

  const fetchData = async () => {
    const data = await fetch("./data.json", {
      headers: {
        "Content-Type": "application/json",
        Accept: "application/json",
      },
    }).then((res) => res.json().then((data: pageData) => data));

    return data;
  };

  return (
    <Layout>
      {data && (
        <HomeView
          dataForGraph1={data.dataForGraph1 || []}
          dataForGraph3={data.dataForGraph3 || []}
          dataForGraph4={data.dataForGraph4 || []}
          dataForLastGraph={data.dataForLastGraph || []}
          clientHours={data.clientHours || []}
          projects={data.projects || []}
          totalInLastGraph={data.totalInLastGraph || ""}
          clientInformation={data.clientInformation || null}
          photoUrl={userPhotoUrl}
        />
      )}
    </Layout>
  );
};

export default Home;
```

Для реалізації певного оновлення даних, дані отримуються кожну хвилину.

Перевірка

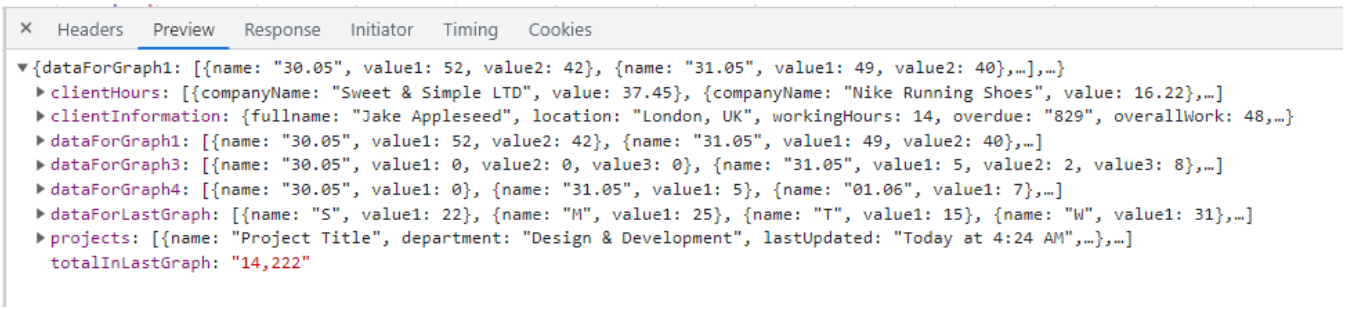

Рисунок 7 Дані, отримані з файлу

Посилання на GitHub:

https://github.com/kostvick/course-work

Візуальний вигляд сайту:

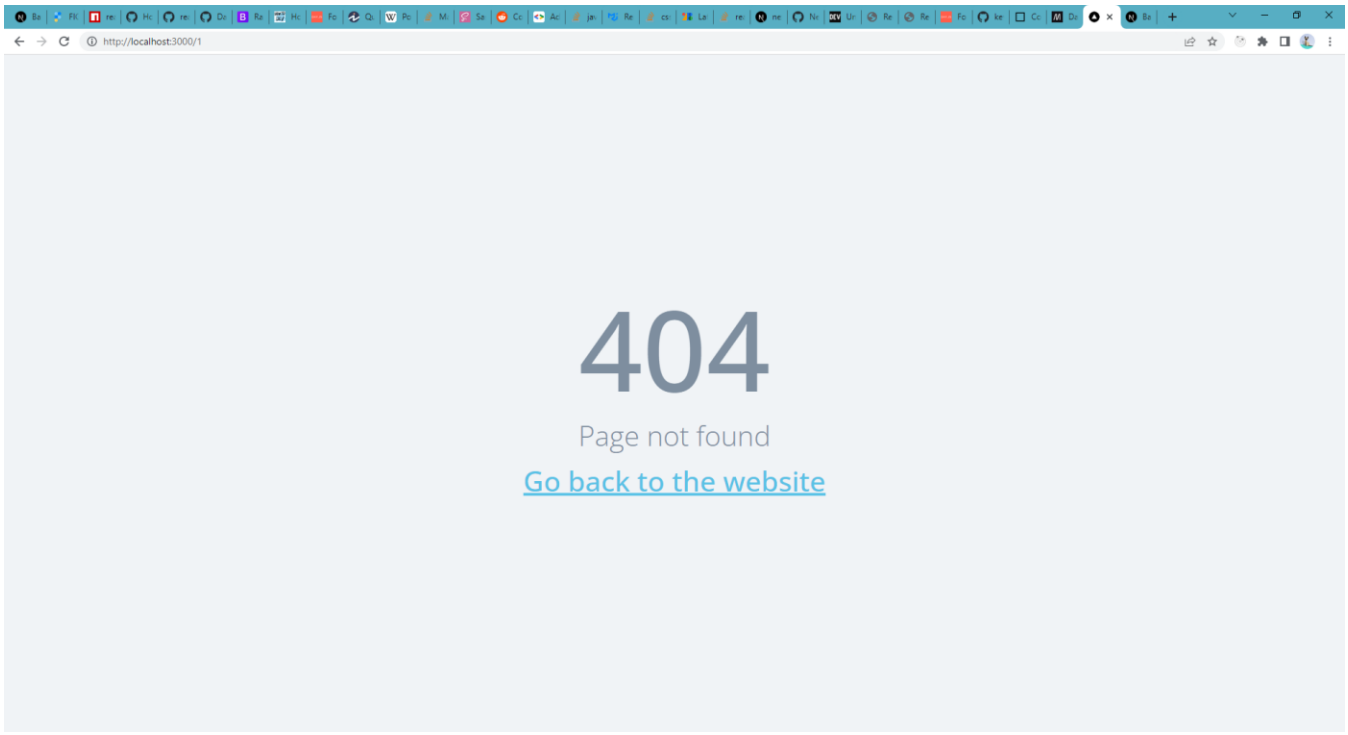

Рисунок 8 Сторінка 404

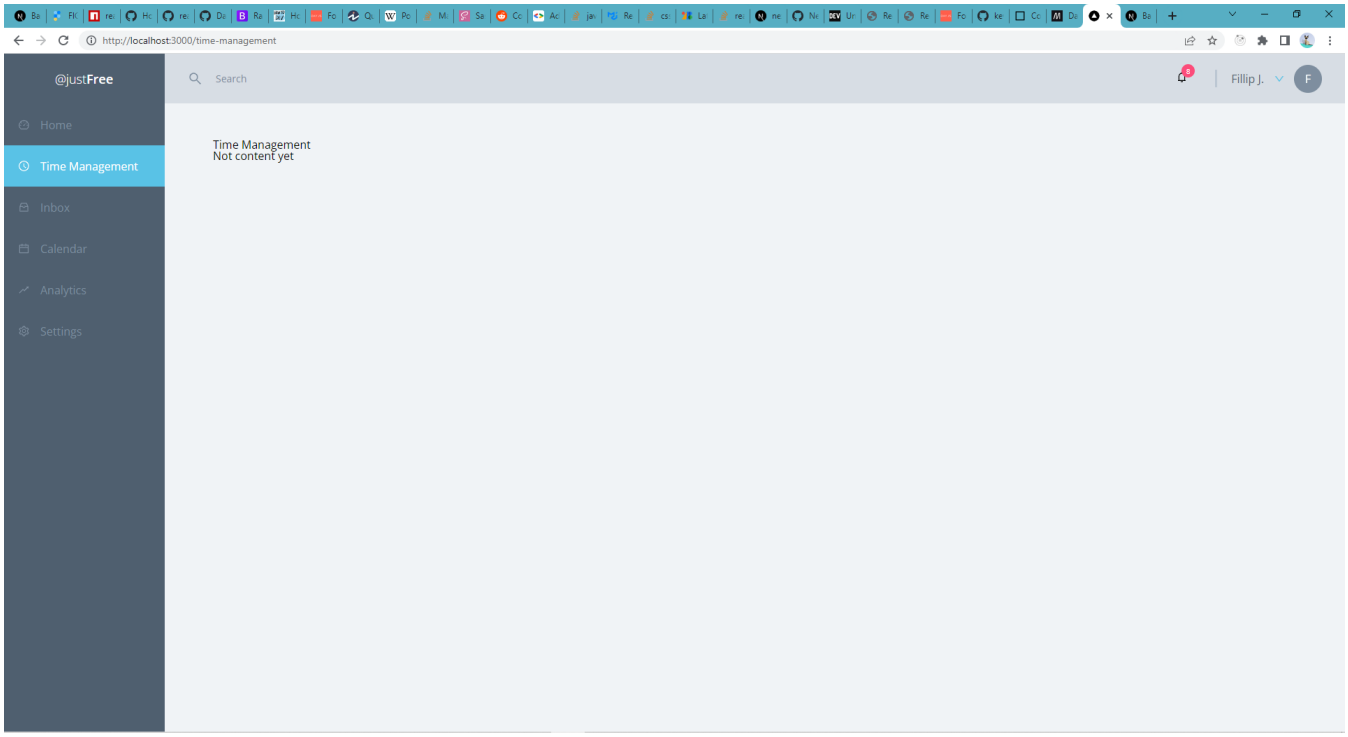

Рисунок 9 Сторінка Time Management

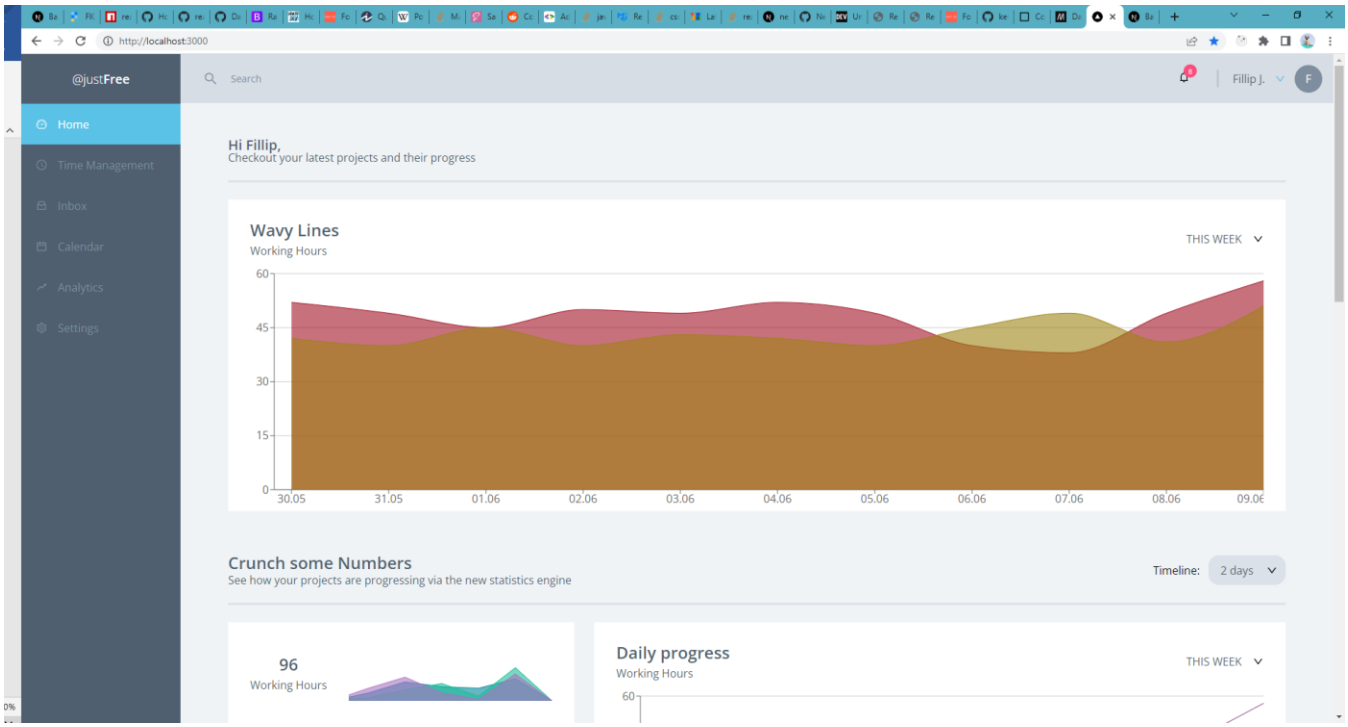

Рисунок 10 Головна сторінка

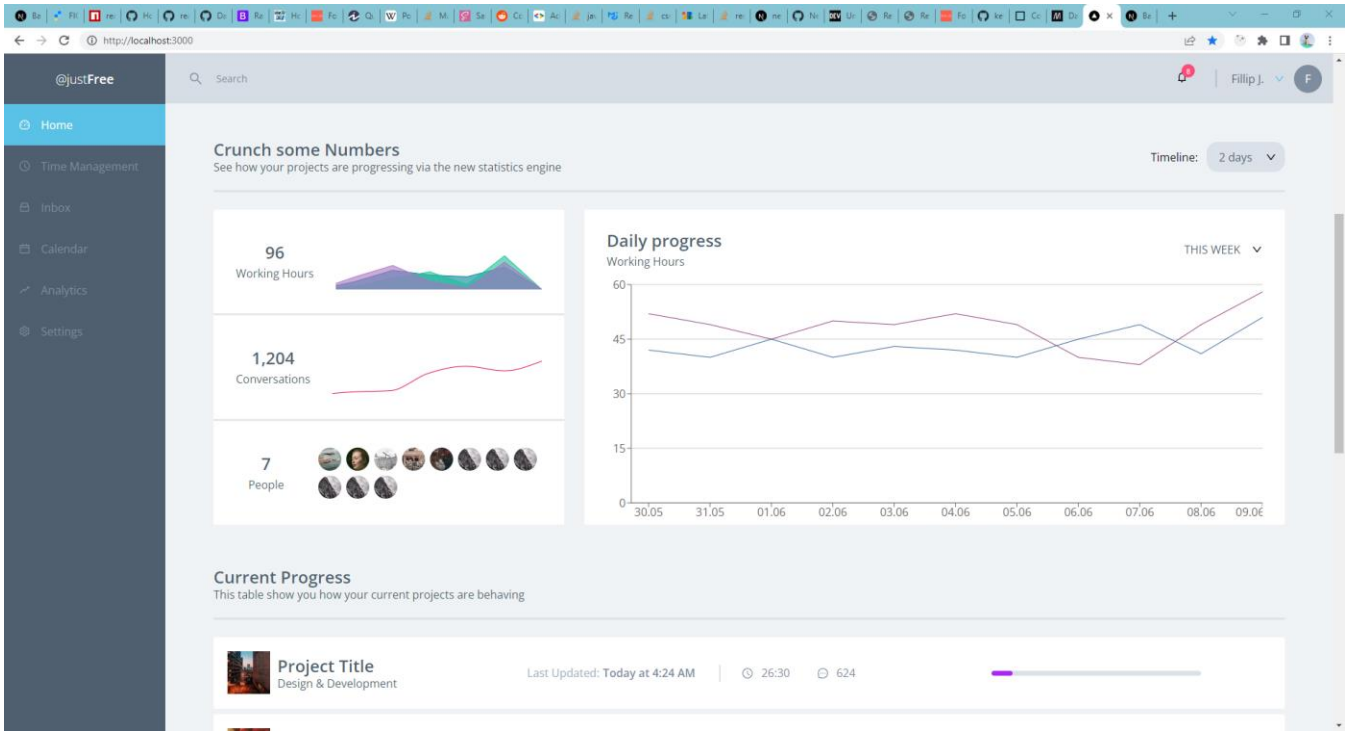

Рисунок 11 Головна сторінка

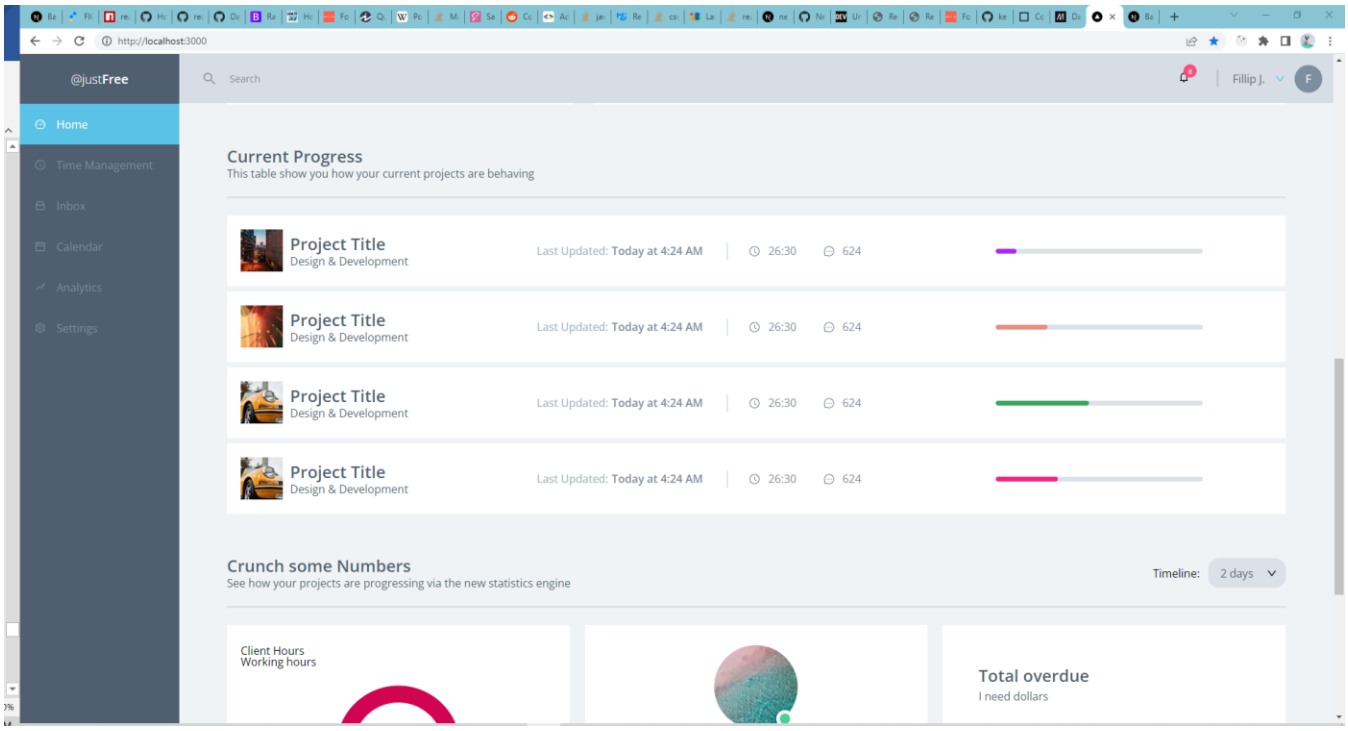

Рисунок 12 Головна сторінка

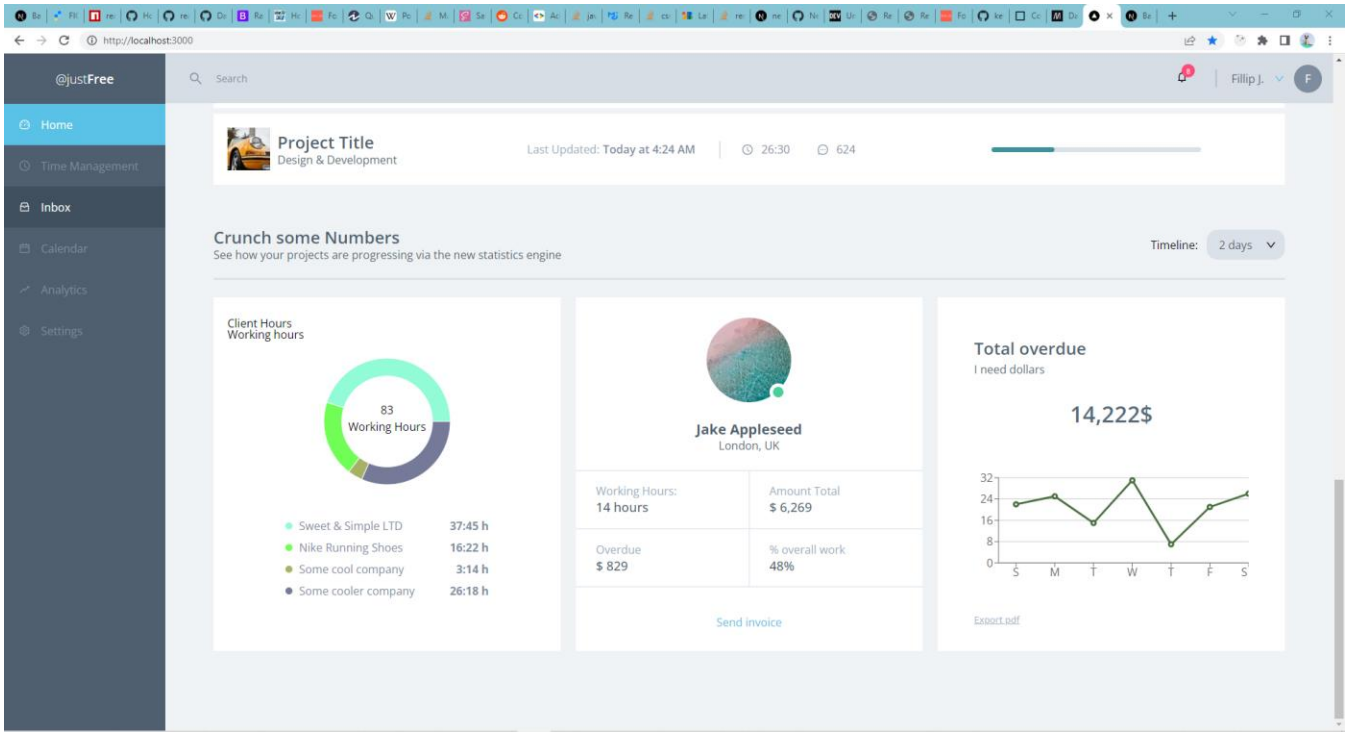

Рисунок 13 Головна сторінка

Висновок

Зверстав сайт відповідно до власного варіанту. Використав React, TypeScript, Next.js, React Bootstrap, SCSS. Використання TypeScript зумовлене підвищенням безпеки написання коду, завдяки типізації. Також завдяки типізації прискорюється процес написання коду, адже зв’являється підсвідка коду.

SCSS вдосконалює використання CSS, завдяки можливості створення міксинів, змінних, імпортування файлів, розширення класів і т.д.

Next.js полегшує роботу з роутингом та оптимізує картинки. Усі дані отримуються з json-файлу та перезавантажуються кожну хвилину. Випадаючі списки можуть бути активовані.

Зі шляхів покращення програми можна вказати використання бібліотеки dayjs для коректного відображення дат, але через незрозумілість як вона вираховується вона не була використана. Колів індикатора прогресу залежить від даних з json-файлу.

Посилання

1. Документація React та інша інформація зв’язана з цією бібліотекою: https://uk.reactjs.org/
2. Офіційний сайт TS: https://www.typescriptlang.org/
3. React Bootstrap: https://react-bootstrap.github.io/
4. Next.js

   https://nextjs.org/