МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій
Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ
завідувач кафедри
мережевих та інтернет технологій
_____ Ю.В. Кравченко
«_____»_____20__ року

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

галузі знань 17 «Електроніка та телекомунікації»
за спеціальністю 172 «Телекомунікації та радіотехніка»

на тему:

РОЗРОБКА WEB-ДОДАТКА ДЛЯ TASK-МЕНЕДЖЕРУ

Виконала: студентка групи МІТ -41

Коломійцева Єлизавета Олегівна _____
(прізвище ім'я по-батькові) (підпис)

Керівник: асистент кафедри мережевих та інтернет технологій

к.т.н., асистент Махович О.І. _____
(посада, прізвище ім'я по-батькові) (підпис)

Київ 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій
Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ
завідувач кафедри
мережевих та інтернет технологій
_____ Ю.В. Кравченко
«_____»_____20__ року

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ

Здобувачу вищої освіти **Коломійцевій Єлизаветі Олегівні**
(прізвище ім'я по-батькові)

1. Тема роботи: Розробка web-сервісів та web-інтерфейсу для планера
затверджена на засіданні кафедри МІТ «___»_____20___р. протокол №____
2. Термін здачі закінченої роботи «20 » червня 2022 р.
3. Вихідні дані до проекту (роботи)
   Фреймворк мови програмування JavaScript React.js
   Проаналізувати сучасні шляхи та методи проектування web-додатків
   Розробити зовнішній вигляд додатка для планування "ToDo"
4. Зміст пояснювальної записки (перелік питань, що їх потрібно розробити, обсяг – 35-40 стор.)
   Проаналізувати основні поняття та види фреймворків
   Дослідити основи проектування та способи створення якісного інтерфейсу користувача
   Детально розглянути фреймворк React.js та створити на ньому приклад web-додатка
5. Перелік графічного матеріалу 12 слайдів

Дата видачі завдання _____
Керівник роботи _____ к.т.н., асистент Махович Олександр Іванович
(підпис) (посада, прізвище, ім'я, по батькові)

Завдання прийняла до виконання _____ Коломійцева Єлизавета Олегівна
(підпис) (прізвище, ім'я, по батькові)

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ РОБОТИ

Номер	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Підготовчий	28.01.2022	
2	Розділ 1	25.05.2022	
3	Розділ 2	6.07.2022	
4	Доповідь та слайди	16.07.2022	
5	Пояснювальна записка	20.07.2022	

Здобувач вищої освіти _____ Коломійцева Єлизавета Олегівна

(підпис) (прізвище, ім’я, по батькові)

Керівник _____ Махович Олександр Іванович

(підпис) (прізвище, ім’я, по батькові)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Розробка web-сервісів та web-інтерфейсу для планера» складається зі вступу, основної частини, що містить 2 розділи, висновків і списку літератури та джерел.

Загальний обсяг роботи – 48 сторінок

Об’єкт дослідження – сучасні технології розробки web-додатка.

Мета роботи – розкрити основні поняття та види фреймворків для JavaScript, дослідити сучасні технології розробки та способи створення web-додатків, розглянути фреймворк React.js та створити з його допомогою web-додаток.

Предмет дослідження - розробка web-сервісів та web-інтерфейсу для task-менеджера.

Метод дослідження – аналіз та порівняння фреймворків для JavaScript.

У ході роботи проведено аналіз сучасних технологій, що використовуються при проектуванні web-додатків.

Запропоновано розробити web-додаток для task-менеджера з використанням фреймворку React.js.

Визначено необхідний функціонал на основі потреб користувача.

Розроблено сучасний, зручний і зрозумілий у використанні додаток для планування справ та раціонального використання часу.

Практичне значення роботи полягає у створенні web-додатка за допомогою сучасних технологій для подальшого використання користувачами.

Результати здійснених у дипломній роботі досліджень можуть бути використані будь-яким пересічним громадянином.

Ключові слова web-додаток, фреймворк, React.js, таск-менеджер.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ –програмне забезпечення

JS - JavaScript

CSS- Cascade Style Sheets

HTML – HyperText Markup Language

API – Application Programming Interface

DOM – Document Object Model

ASCII - American Standard Code for Information Interchange

UI – User Interface

ВСТУП

Сьогоденний світ є стрімким потоком, за яким важко встигати. Кожна сучасна людина має безліч планів і задач великих та малих, на які треба знаходити час. Чудовим помічником у впорядкуванні часу є task-менеджер, основним функціоналом якого є запис необхідних до виконання справ та годинник за сучасною системою Pomadoro. Принцип системи роботи Pomadoro є чітко визначений період сесій для роботи та відпочинку, які змінюють одна одну.

Майже кожна людина у наш час вже опанувала інтернет технології і майже весь робочий і вільний час проводить, використовючи комп`ютер або телефон. Тому використання web-додатка для task-менеджера є дуже актуальною потребою кожної людини.

Чудовою технологією для розробки web-додатка є фреймворки популярної мови програмування JavaScript, що дозволяє значно розришити функціонал при розробці.

На сьогоднішній день фреймворки використовуються при розробці web-додатків різної складності, наприклад, корпоративний сайт, інтернет-портал чи web-сервіс. Це надає ряд переваг, таких як зручність для користувача, швидкість розробки, надійність роботи та ефективність використання ресурсів.

У даній роботі запропоновано варіант використання сучасних технологій розробки web-додатка на прикладі вище описаного task-менеджера. З цією метою проаналізовано сучасні методи побудови web-додатків, розглянуті найпопулярніші фреймворки та програмне забезпечення, що необхідне для створення web-додатків, проведено порівняльний аналіз можливостей різних фреймворків. З урахуванням отриманих результатів обрано найбільш оптимальний варіант моделі побудови web-додатка.

Практичне значення роботи полягає у тому, що web-додаток, створений за допомогою фреймворку React.js, надасть можливість користувачу підвищити рівень продуктивності та раціональності планування часу.

1. АНАЛІЗ МЕТОДІВ ТА ТЕХНОЛОГІЙ ПОБУДОВИ WEB-ДОДАТКІВ

1.1 Сучасні засоби і технології побудови web-додатків

Сучасна реальність характеризується широким впровадженням сучасних інформаційних технологій в усі сфери життя людини, діяльності держави, суспільства та бізнесу. Найшвидшим засобом для сприйняття або поширенням інформації є інтернет, чим можна пояснити його широку популярність та надзвичайну перспективність.

Сьогоднішня всесвітня мережа об’єднує в собі різноманітні системи комунікації та засоби зв’язку, що дозволяє передавати інформацію на будь-яку відстань та в будь-яку точку планети. Вона містить у собі нескінченну кількість web-додатків різного рівня якості та тематики, які доступні кожному користувачеві.

Написання будь якого web-додатка неможливе без HTML. HTML це мова розмітки гіпертексту, що призначена для створення документів у всесвітній павутині (World Wide Web), які можна переглянути в різних браузерах.

Перевагою цієї мови є можливість створення HTML документів у будь-якому текстовому редакторі, чи спеціалізованому ПЗ, а переглянути його можна у будь-якому браузері на різних платформах.

HTML-файл - це звичайний файл в кодуванні ASCII, в основі якого лежать теги для визначення правильного форматування даних, але сам код являє собою звичайний текстовий файл, що може бути написаний в стандартному блокноті Windows та збережений у форматі .html.

Основні переваги:

– висока швидкість завантаження;
– простота;

– невелика кількість займаної пам’яті;

– незначне серверне навантаження.

Основні недоліки:

– оновлюваний контент знаходиться безпосередньо у файлі розмітки HTML (потрібно редагувати всю сторінку);

– негнучкість (для додавання нового розділу чи сторінки потрібно також найняти розробника.

Задля реалізації UI частини розробки web-додатка, тобто задля реалізації дизайну, використовують CSS. CSS, розшифровується як Cascading Style Sheets (каскадні таблиці стилів) і значно полегшує створення сайтів в процесі роботи з шрифтами, полями, таблицями, відступами, зображеннями та ін. та надає набагато більше можливостей ніж просто HTML.

Головною особливістю CSS є те, що кожен дочірній елемент набуває властивості батьківського.

Також CSS значно покращує час завантаження та трафік, завдяки зберіганню інформації про форматування в окремому файлі конфігурації, який завантажується лише один раз, на відміну від сайту без CSS, тож і переформатувати документ HTML за допомогою CSS можна значно швидше, адже ми маємо змінити лише код CSS і усі документи HTML, що використовують дану таблицю стилів автоматично будуть переформатовані.

Будь яка сучасна web-сторінка та web-додаток потребує наявності інтерактивних елементів, що реагують на дії користувача. Для реалізаціїї цієї необхідності використовують JavaScript. Це об’єктно-орієнтована скриптова мова програмування, яка використовується зазвичай, як вбудована модель для програмного доступу до об’єктів додатків. За допомогою JavaScript стає доступною реалізація таких функцій:

- можливість змінювати сторінки браузерів;
- додавання чи видалення тегів;
- зміна стилів сторінки;

- збір інформації про дії користувача на сторінці;
- запит доступу до випадкової частини вихідного коду сторінки;
- внесення змін в цей код;
- виконання дій з cookie-файлами.

Область застосування цієї мова дуже обширна і нічим не обмежена: серед програм, які використовують JS, наявні і текстові редактори, і додатки(як для комп'ютерів, так і мобільні і навіть серверні), і прикладне ПЗ.

Переваги JavaScript:

- жоден сучасний браузер не обходиться без підтримки JavaScript;
- доволі простий у використані;
- корисні функціональні налаштування;
- мова, що постійно вдосконалюється;
- взаємодія з додатком може реалізуватись навіть через текстові редактори.

Недоліки JavaScript:

- низький рівень безпеки у зв'язку з популярністю використання та вільного доступу до вихідного коду популярних скриптів;
- велика кількість дрібних помилок на кожному етапі роботи. Більша частина з них легко виправляється, але їх наявність дозволяє вважати цю мову менш професійною, в порівнянні з іншими.

Багато веб-розробників визнають, що у JavaScript є недоліки та складні частини, але це все ще найпопулярніша мова програмування. В результаті проведеного на Stack Overflow у 2020 році опитуванні, результати якого наведені на рисунку 1.1, 69.7% з 47 184 опитуваних професійних розробників відповіли, що використовують у роботі JavaScript.

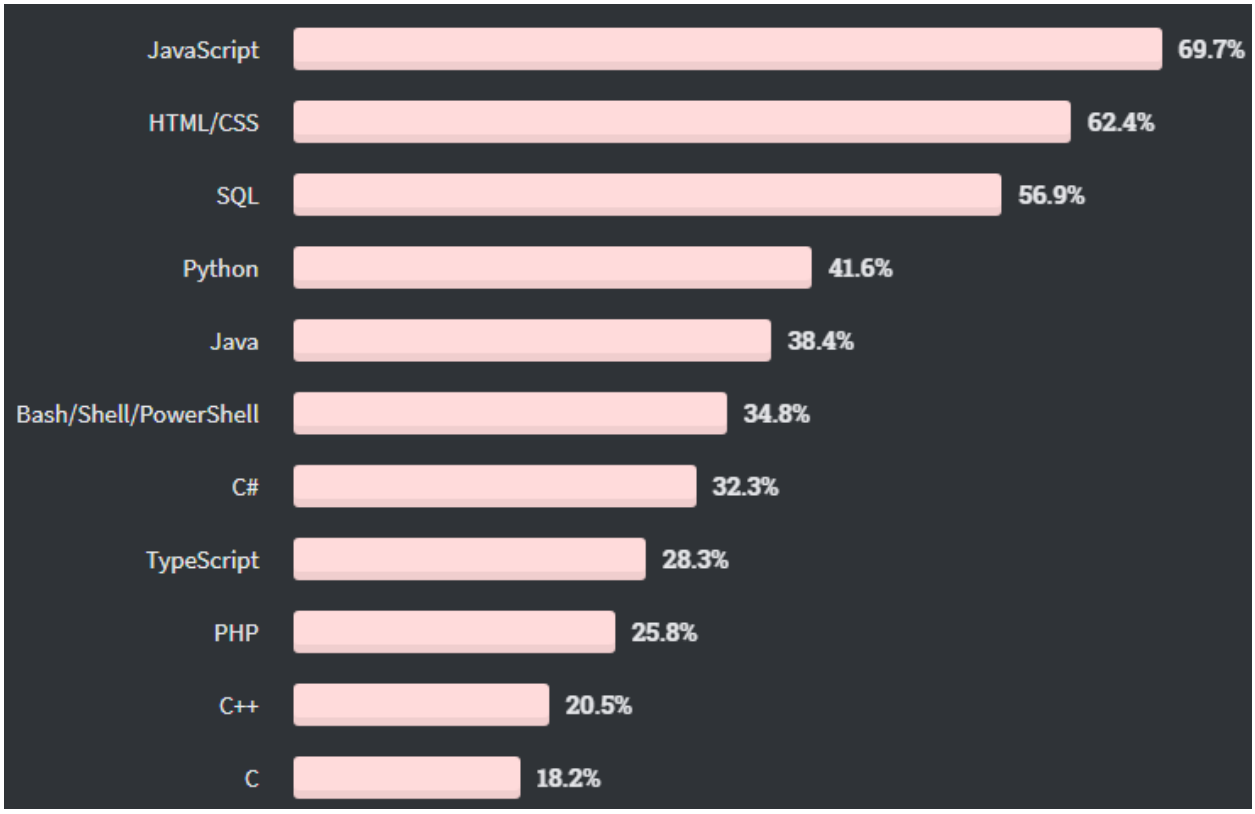

Рисунок 1.1 – Опитування результати опитування, яка мова програмування є найпопулярнішою

Недоліком використання при розробці лише HTML, CSS та JavaScript є необхідність весь час оновлювати сторінку та код HTML для додавання нового контенту, тобто сторінка є статичною. Для вирішення цієї проблеми використовують JS-фреймворки.

1.2 Огляд JS-фреймворків

JS-фреймворки це інструмент для побудови динамічних веб/мобільних додатків на мові JavaScript. Як і будь які інші інструменти, розробники використовують JS-фреймворки там, де неможливо або дуже складно або дуже довго виконувати задачу звичайними методами.

Переважно, фреймворки використовуються для написання, так званих, Single Page Applications, тобто тих додатків, де усі дії користувач виконує на одній сторінці без прямого переходу з неї. З допомогою фреймворків можна розробляти як повноцінні сайти, так і функціональні модулі.

Фреймворки надають чітку структуру додатка і реалізуються з використанням патернів проектування. Найбільш широко поширені наступні патерни: MVC (Model-View-Controller), MVP(Model-View-Presenter) та MVVM(Model-View-ViewModel).

Переваги побудови web-додатка за допомогою JS-фреймворку:

- простота реалізації Single Page Applications;
- чітка структура додатка;
- коду стає помітно менше, що позитивно впливає на швидкість розробки, а також при підтриманні та усуненні помилок в коді додатка;
- наявність структури забезпечує модульність додатка, що це дає можливість простіше працювати над додатком декільком розробникам одночасно;
- можливість швидко створити мобільний додаток чи настільний додаток, що сумісний з усіма браузерами, з веб-версії за допомогою систем типу PhoneGap чи Apache Cordova.

Із значних недоліків можна виділити лише тимчасово неповну

підтримку пошуковими системами, але ця задача рідко співпадає з задачою по реалізації Single Page Applications, тим паче, що найпопулярніші пошукові системи (як мінімум Google), вже практично повністю вирішили цю проблему.

Існує багато JS-фреймворків, серед найпопулярніших слід визначити такі:

- React.js;
- Angular;
- Vue.js;
- Svetle;
- Ember.js;
- Preact;
- Backbone.js.

Компанія «The Software House» регулярно публікує результати досліджень у сфері frontend-розробки «State of Frontend 2020». На рисунку 1.2 продемонстровані результати опитування, який з фреймфорків використовувався розробниками найбільше впродовж року.

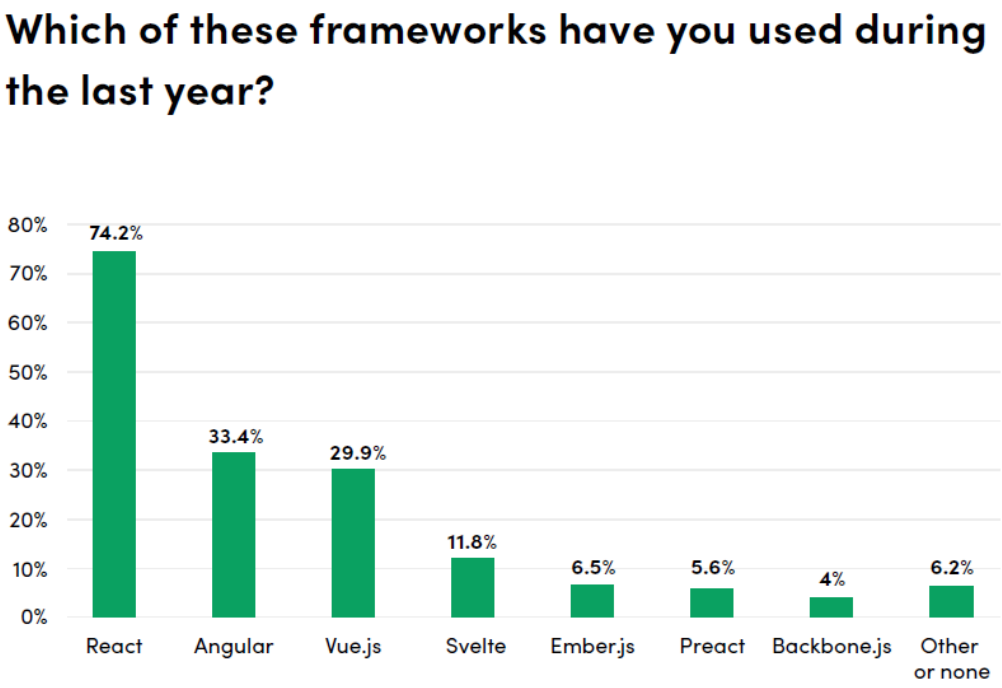

Рисунок 1.2 – результати опитування серед розробників

Розглянемо детальніше деякі з наведених вище фреймворків.

1.2.1 React

У JavaScript-розробці React – беззаперечний лідер. У цьому фреймворку використовуються ідеї реактивного програмування. Реактивне програмування — це парадигма програмування, побудована на потоках даних і розповсюдженні змін. Це означає, що у мовах програмування має бути можливість легко виразити статичні чи динамічні потоки даних, а реалізована модель виконання буде автоматично розсилати зміни через потік даних. При використанні, React надає можливість застосовувати багато додаткових інструментів. Ось, наприклад, далеко не вичерпаний список інструментів, представлений бібліотеками, які можна використовувати спільно з React. Це- Redux, MobX, Fluxy, Fluxible, RefluxJS. В React-розробці, крім того, можна використовувати jQuery AJAX, Superagent, Axios і Fetch API.

З 2019 року розробники React анонсували впровадження декількох технологій таких як, конкурентний режим та рендеринг лінивих компонентів, створених за допомогою React.lazy всередині компонетів React.Suspense. Обидві технології дозволяють зробити React-додатки більш чутливими за рахунок виконання рендерингу без блокування головного потоку. Це дозволяє React не затримувати обробку високо пріоритетних задач.

Технологія React.Suspense дозволяє покращити обробку асинхронного завантаження даних в React-додатках. Ця технологія дозволяє організувати очікування компонентом виконання деяких умов і при цьому не порушувати роботу усього додатка.

Також у React є технологія хуків (Hook), яка дозволяє розробнику використовувати важливі можливості React і при цьому обійтись без використання компонентів, які засновані на класах. Серед таких можливостей – управління станом компонента і робота з методами його життєвого циклу. React

містить декілька вбудованих хуків, при цьому дозволяє програмісту створювати власні хуки.

1.2.2 Angular

На конференції AngularConnect 2019 команда розробників Angular зробила заяви, які дозволяють вважати версію Angular 9 поворотною точкою в розвитку цього фреймворку. Компілятор Angular Ivy став стандартним засобом, доступним для усіх додатків, що дозволяє значно пришвидшити процес розробки, зменшити розмір додатків, підвищити їх продуктивність і надійність.

Сучасний Angular – це модульний фреймворк для фронтенд розробки. Раніше для підключення до строніки достатньо було лише додати в її HTML-код відповідний тег, але зараз розробник може імпортувати у свій проект необхідні йому модулі Angular.

Angular відомий своєю гнучкістю, саме з цієї причини, все ще актуальні версії Angular 1.x. Проте багато розробників використовують в наші дні Angular 2+ через MVC-архітектуру фреймворку, яка значно змінилась в сторону архітектури, заснованої на компонентах.

З цієї причини ті, хто хотів використовувати цей фреймворк, при його освоєнні, зіштовхнуться з деякими труднощами. Так, для створення Angular-додатків майже немає необхідності використовувати TypeScript, що ускладнює роботу з цим фреймворком, але надає декілька переваг. Наприклад, це підвищує надійність додатка за рахунок продвинутого контролю типів, що дає програмісту додаткові засоби для розробки.

Angular – це повноцінний фреймворк, що надає сучасному програмісту усе необхідне для продуктивної роботи. На Angular варто звернути увагу тим, кому хотілося б отримати в своє розпорядження великий набір стандартних засобів та звести до мінімуму використання сторонніх бібліотек.

1.2.3 Vue.js

Ідеї, які лежать в основі Vue.js, запозичені з Angular та React, але Vue.js, в декількох випадків, краще цих двох інструментів фронтенд-розробки. В цьому році Vue.js завантажили більше 40 мільйонів разів. У звіті Snyk JavaScript Frameworks Security, який був націлений на дослідження безпеки JS-фреймворків можна дізнатись про те, що відомо лише 4 безпосередні вразливості Vue.js, які були усунені.

При роботі з Vue.js логіка компонента, його макет та стилі зберігаються в одному файлі. Взаємодія компонентів у Vue.js забезпечується за допомогою об’єктів, які зберігають властивості і стан компонентів. Цей підхід, ще до того, як він з’явився в Vue.js, був використаним в React. Vue.js, що поріднює його з Angular, дозволяє змішувати HTML-розмітку та JavaScript-код.

Одна з причин, за якою треба використовувати Vue.js як достойну альтернативу React, полягає в тому, як у ньому організовано управління станом додатка. В React-проектах, при використанні зв’язки React+Redux, по мірі розширення додатка ускладнюються й процедури, необхідні для управління його станом. Це може привести до того, що програмісту, замість роботи над своїм додатком, необхідно буде витрачати немало часу на налаштування механізмів Redux. В Vue.js для управління станом використовується бібліотека Vuex, з якою набагато зручніше працювати.

Якщо робити вибір між Vue.js та Angular, то причини, за яких можна надати перевагу Vue.js, можна звести до того, що Angular, в порівнянні з Vue.js, виглядає занадто складним крупномасштабним проектом, в природі якого закладено прагнення обмежити розробника. Vue.js набагато простіший за Angular і не так сильно обмежує програмістів. Ще одною перевагою Vue.js над Angular та React полягає в тому, що для роботи з цим фреймфорком не потрібно вчити нову мову.

1.2.4 Svetle

Хотілося б окрему увагу приділити фреймворку Svetle, оскільки якщо звернути увагу на результати опитування, вище згаданою компанією «The Software House», який фреймворк розробники хотіли б використовувати або вивчити для своїх майбутніх проектів, то Svetle опинився у трійці лідерів.

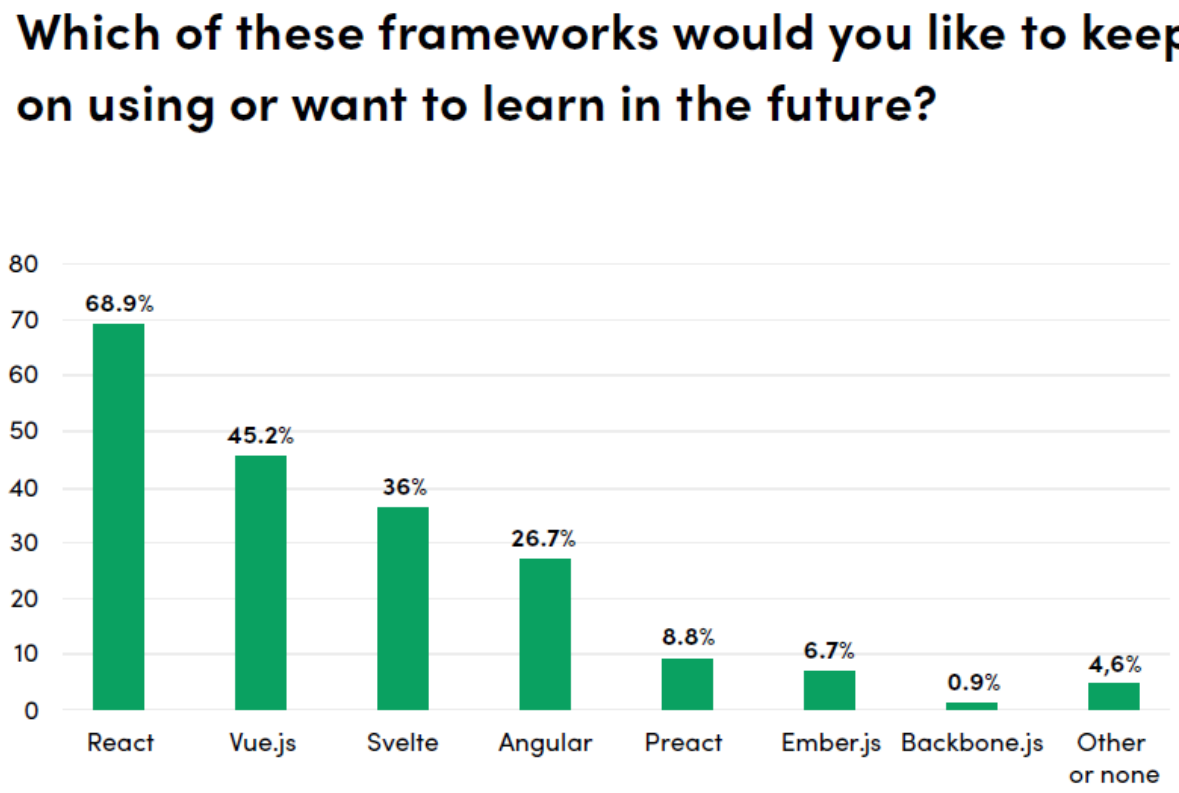

Рисунок 1.3 – результати опитування, який фреймворк ви б хотіли використовувати або опанувати

Svetle – це невеликий за вагою, компонентний фреймворк наступного покоління, написаний на TypeScript. Він надає новий спосіб створення високопродуктивних веб-додатків. На відміну від більш популярних React та Vue.js, які перетворюють код в Vanila JS під час використання, Svetle робить це під час компоновки. Тобто Svetle виконує роль компілятора, який дозволяє

запускати код у браузері без будь-якого рівня абстракції. Це підвищує рівень продуктивності додатка і забезпечує більшу задоволеність серед користувачів.

В результаті веб-додатки, створені за допомогою Svetle, с першого ж завантаження опиняються набагато швидшими, ніж додатки, розроблені на інших фреймворках. Для створення веб-додатків Svetle може використовуватись окремо або в поєднанні з цими фреймворками.

Можливо, зараз Svetle ще недостатньо розвинений аби створити конкуренцію таким гігантам, як React. Але в нього точно є потенціал для більш широкого використання у веб-додатках.

2.РОЗРОБКА WEB-ДОДАТКА «ToDo»

2.1 Вибір фреймворку

Для розробки web-додатка було обрано React, оскільки в порівнянні з іншими фреймворками він має ряд переваг.

React-додатки складаються з компонентів, які містять логіку роботи додатка і HTML-код для форматування інтерфейсу. Така структура надає можливість створювати код з невеликих елементів, додаючи тільки необхідний функціонал, не обтяжуючи код додатковими стандартними пакетами за замовчуванням, на відміну від Angular та Vue.js. Це значно збільшує швидкість розробки. Для того, аби покращити взаємодію між компонентами, розробник може використовувати Fluxy або схожу JavaScript-бібліотеку.

В React програмуванні використовуються такі поняття як стан (state) та властивості (props) компонента. Вони представлені відповідними об’єктами. Їх використання дозволяє організувати зберігання даних в компоненті та обмін даними між компонентами. Наприклад, передачу даних із програмної логіки, яка реалізована компонентом, в інтерфейс додатка, або передачу даних від батьківських компонентів дочірнім компонентам.

Структура коду при розробці за допомогою React є модульною, що дозволяє створювати шаблони коду. Така властивість надає можливість значно оптимізувати код. Наприклад, при розробці інтернет-магазину зникає необхідність прописувати окремо картку для кожного товару у каталозі, оскільки можна зробити шаблон картки, яка кожен раз буде наповнюватись різним контентом.

Всередині React є механізм, який реалізовує алгоритм реконселяції, що дозволяє відслідковувати, які елементи додатка змінились, і поновити тільки їх, а не весь додаток повністю. Даний алгоритм значно підвищує швидкість роботи додатка.

React має засоби для роботи з Document Object Model. DOM – специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами, наприклад HTML. З точки зору об'єктно-орієнтованого програмування, DOM визначає класи, методи та атрибути цих методів для аналізу структури документів та роботи із представленям документів у вигляді дерева. Все це призначено для того, аби надати можливість комп'ютерній програмі доступу та модифікації структури, змісту та оформлення документа. Ця структура є дуже громіздкою, оскільки за замовчуванням кожен елемент має велику кількість властивостей. При створенні, структура DOM початково не була призначення для роботи з динамічними web-додатками. React містить технологію Virtual DOM, яка по суті є легкою копією DOM. При створенні елементів за допомогою Virtual DOM, вони мають значно менше властивостей за замовчуванням, що зменшує навантаження на браузер і підвищує швидкість завантаження web-додатка.

React дає доступ до управління низькорівневих взаємодій його функціональних компонентів, що підвищує гнучкість при розробці та надає можливість оптимізувати та підвищити рівень управління додатком.

Навколо фреймворку React склалась ціла екосистема, представлена різними допоміжними інструментами та бібліотеками. Ось деякі з них:

- Сам фреймворк React та React Router – засіб для управління маршрутами в додатка;
- Пакет react-dom, який створений для роботи з DOM;
- Інструменти розробника React для браузерів Firefox та Chrome;
- Препроцесор JSX – мова розмітки, яка дозволяє описувати HTML-елементи в JavaScript-коді;
- Засіб командного рядка create-react-app, який націлений на створення шаблонних React-проектів;

- Різноманітні допоміжні бібліотеки. Серед них, наприклад, можна завернути увагу на бібліотеку Redux, яка використовується для управління станом додатків, і бібліотеку Axios, яка використовується для обміну даними серверними API.

Фреймворк React розроблявся і підтримується до цього часу компанією Facebook. Тобто цей фреймворк весь час розвивається та модифікується однією з провідних IT-компаній. Розвиток React націлений на підтримку на сумісність з новими технологіями, що зменшує вірогідність його занепаду у найближчі роки і відповідно гарантує підтримку і актуальність розробленого web-додатка.

React є найпопулярнішим фреймворком для розробки web-додатків. У нього найбільша спільнота розробників, за рахунок чого набагато легше знайти готові рішення або відповіді на проблему, яка виникла.

Наведені вище властивості React підтверджують його пристосованість для розробки як і великих проектів, так і Single Page Applications.

2.2 Розробка web-додатка “ToDo”

Швидко розпочати роботу над web-додатком можна за допомогою введення у командний рядок команди npm create-react-app і назву папки, у якій буде зберігатись додаток. Виконання цієї команди реалізовано за допомогою npm (node package manager), що є стандартним менеджером пакетів, який використовується для завантаження пакетів з хмарного сервера npm, або для завантаження пакетів на ці сервера. Після усіх необхідних завантажень можна швидко розпочати роботу над додатком, написавши у командний рядок команду code . . Ця команда автоматично відкриє шаблонний React-додаток в середі розробки Visual Studio Code. На рисунку 2.1 представлена структура шаблонного React-додатка.

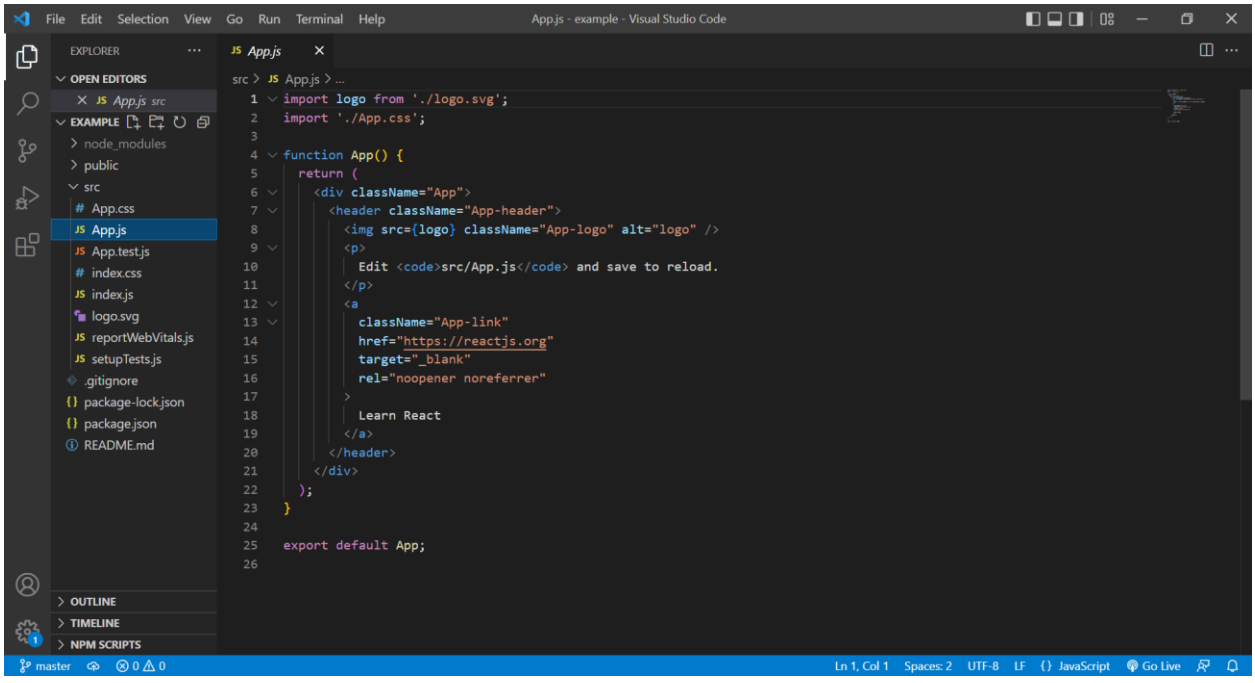

Рисунок 2.1 – структура шаблонного React-додатка

Якщо запустити додаток на локальному хості за допомогою команди npm start можна побачити, як виглядає стандартний React-додаток (рис. 2.2)

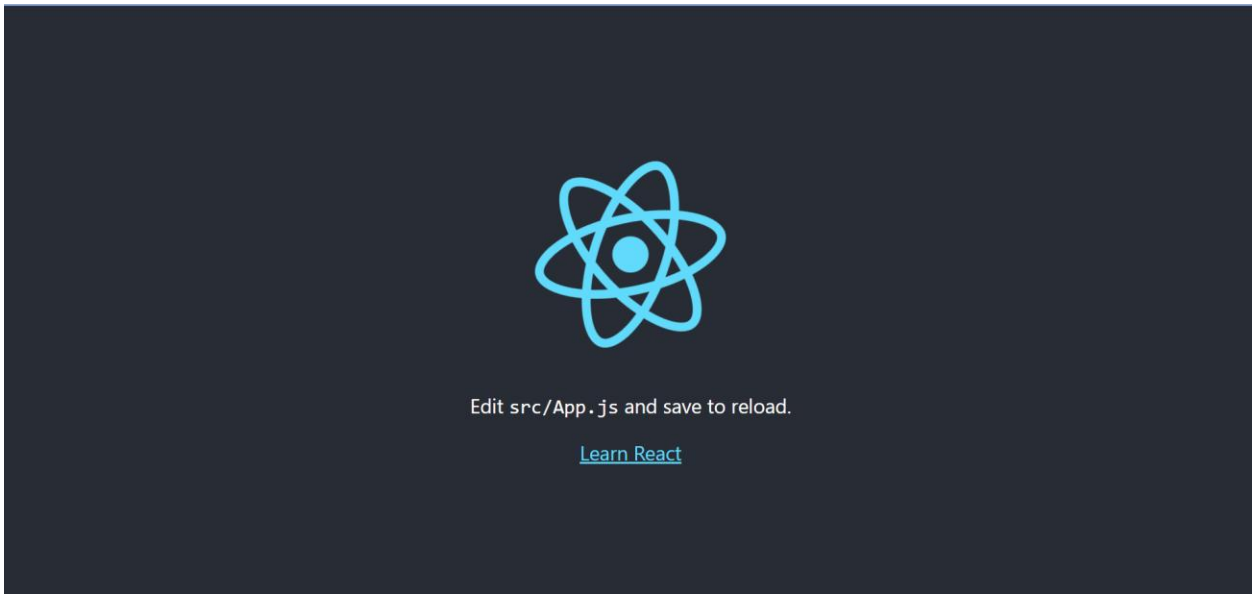

Рисунок 2.2 – стандартний React-додаток

Додаток “ToDo” складається з двох частин: task-менеджера та Pomadoro годинника. Task-менеджер призначений для запису, редагування та відмітки про виконання необхідних справ. Pomadoro годинник призначений для контролю продуктивності часу і містить у собі відлік за системою Pomadoro. Суть системи полягає у тому, що сесія роботи чергується з невеликою сесією відпочинку. Протяжність сесії встановлює користувач.

2.2.1 Розробка task-менеджера

Розробка web-додатка починається з task-менеджера. Для початку створимо компонент TodoForm.js, функція якого полягає у введенні користувачем справи, яку він має виконати. Даний функціонал реалізований за допомогою форми з тегом <input> та кнопки відправки <button>. За допомогою функції Math.random() кожній справі присвоюється унікальний id. Результат реалізації форми представлений на рисунку 2.1.1.

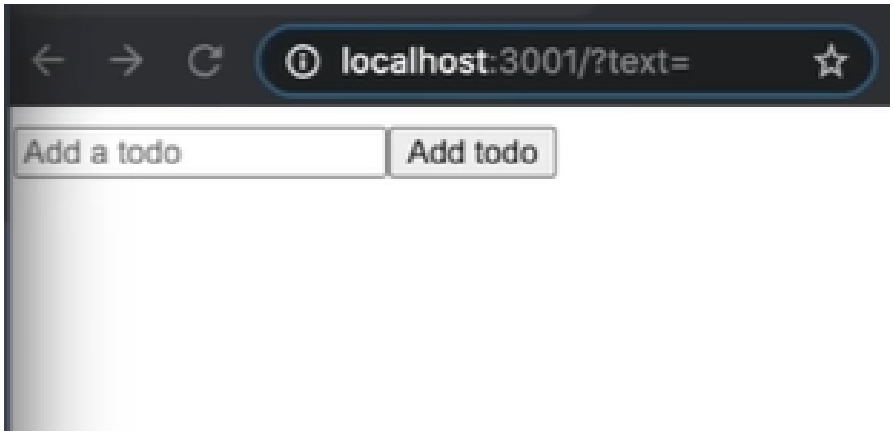

Рисунок 2.1.1 – реалізація форми

Наступним етапом є розробка компонента TodoList.js, який відповідає за збереження введених користувачем справ. Він містить у собі функції видалення списку справ та посилання на редагування справи. Попередньо усі справи зберігаються лише в консолі(рис 2.1.2).

Рисунок 2.1.2 – збережені у консолі справи

Надалі розробляється компонент ToDo.js, який відповідає за ряд функцій, перш за все, відображення справ на екрані, а не в консолі. Для відображення можливості редагувати та видаляти справи використовуються стандартні іконки з бібліотеки React, які завантажуються у проект за допомогою введення команди у командний рядок npm add react-icons. Після цього необхідні іконки імпортуються у проєкт за назвою (рис 2.1.3).

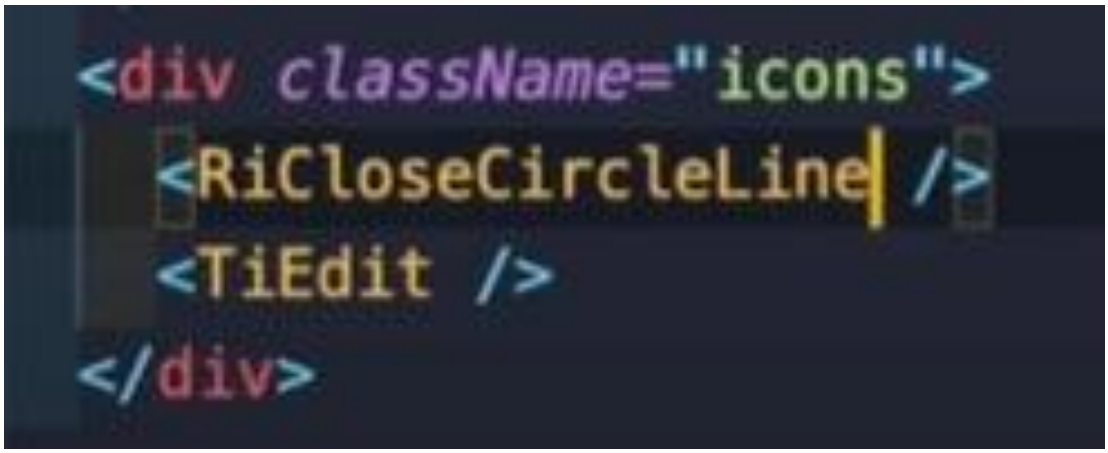

Рисунок 2.1.3 – імортування іконок

Надалі за іконками закріплюються події, які мають відбутися після натискання на них – редагування та видалення відповідно. Редагування реалізовано за допомогою ще однієї форми (рис 2.1.4).

Рисунок 2.1.4 – реалізація функції редагування

Останньою функцією, що виконує компонент ToDo.js є перевірка статусу виконання завдання. Ця функція забезпечує можливість викреслювати виконані справи не видаляючи їх.

Наступним етапом є додавання функції автоматичного фокусування на формі (рис. 2.1.5). Особливістю реалізації цієї функції є наступне: якщо користувач знаходиться у розділі редагування справи, то активною буде форма редагування.

Рисунок 2.1.5 – функція автоматичного фокусування на формі

Також була додана валідація у компонент TodoForm.js, яка відповідає за видалення зайвих пробілів при введені справи у форму користувачем. Повноцінне представлення функціоналу task-менеджера зображено на рисунку 2.1.6.

Рисунок 2.1.6 – Повноцінне представлення функціоналу task-менеджера

Останнім етапом розробки task-менеджера є реалізація власного дизайну за допомогою CSS. Фінальний вигляд task-менеджера представлено на рисунку 2.1.7 та рисунку 2.1.8

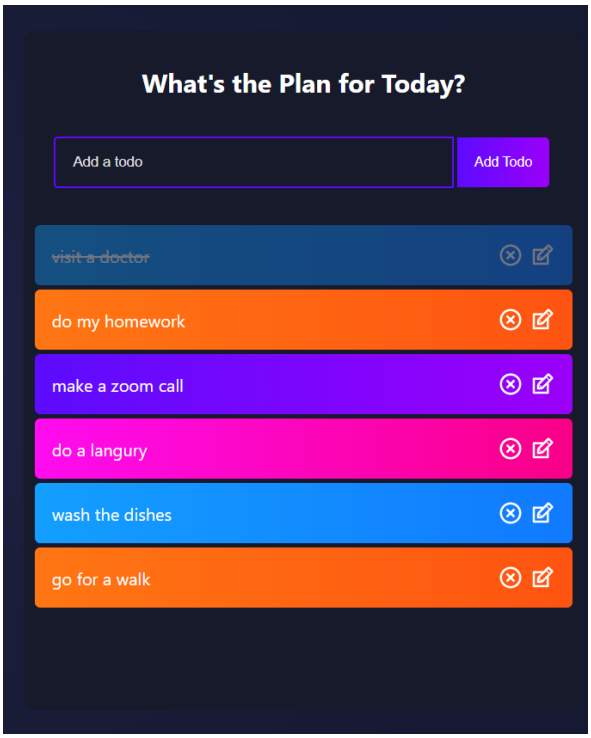

Рисунок 2.1.7 – фінальний вигляд task-менеджера

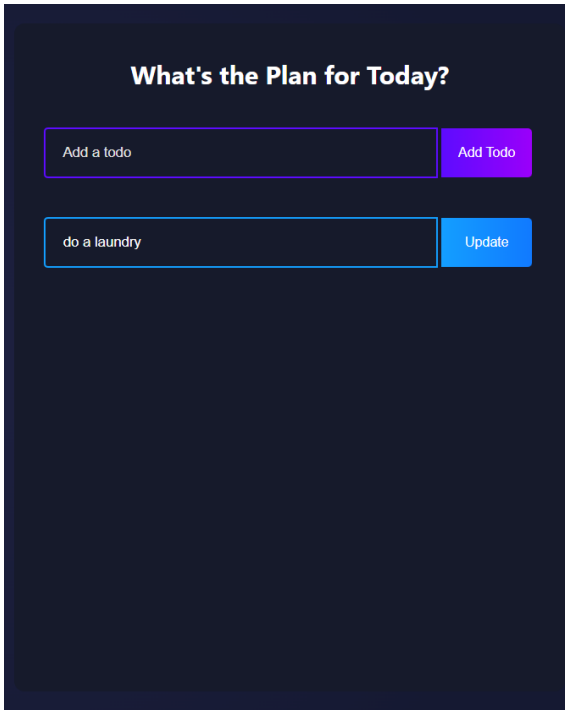

Рисунок 2.1.8 – фінальний вигляд task-менеджера

2.2.2 Розробка Pomadoro годинника

Першим етапом розробки Pomadoro годинника є створення компонента SetPomadoro.js, який відповідає за встановлення користувачем необхідну тривалість сесії роботи, сесії перерви та сесії відпочинку і збереження введених даних. Даний функціонал реалізований за допомогою форми та тегів <input> та <button>. Також важливою функціями цього компоненту є встановлення за замовчуванням, тривалість сесії роботи протяжністю 45 хвилин, тривалість сесії перерви протяжністю 15 хвилин і тривалість сесії відпочинку протяжністю 20 хвилин (рисунок 2.2.1).

Рисунок 2.2.1 – представлення функціоналу компонента SetPomadoro.js

Наступним кроком було створення компонента Button.js, у якому реалізовано шаблон кнопки з параметрами, які відповідають за посилання, на

яке веде кнопка при кліку(рис. 2.2.2) та назви класу, аби визначати, яке посилання зараз є активним(рис. 2.2.3).

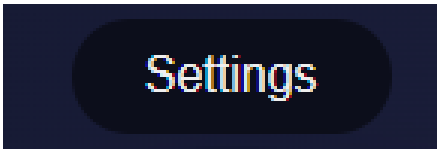

Рисунок 2.2.2 – кнопка, яка веде на налаштування тривалості сесій, зображених на рисунку 2.2.1

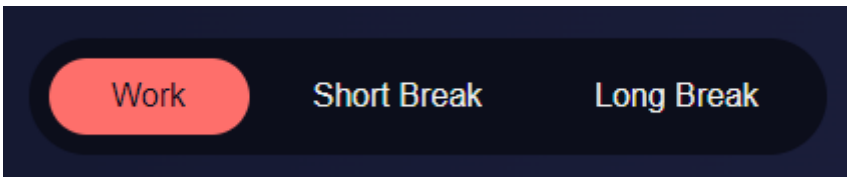

Рисунок 2.2.3 – кнопки, які відповідають за запуск відповідного типу таймера

Також було створено компонент SettingsContext.js, який відповідає за запуск та зупинку таймерів сесій роботи, перерви та відпочинку, відповідно до вказаних користувачем параметрів (рис. 2.2.4).

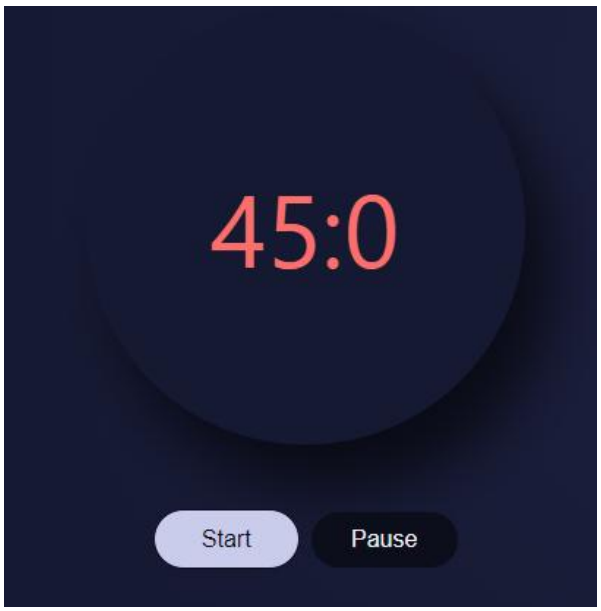

Рисунок 2.2.4 – представлення функціоналу компонента SettingsContext.js

Останнім етапом розробки Pomadoro годинника є створення компонента CountDownAnimation.js, який відповідає за анімацію відліку(рис. 2.2.5). Анімація реалізована за допомогою анімації react-countdown-circle-timer, завантаженої з бібліотеки стандартних анімацій React.

Рисунок 2.2.5 – реалізація анімації відліку

Фінальним етапом розробки web-додатка “ToDo” є збір усіх описаних вище компонентів в один проект у файлі App.js. Реалізація збірки проекту та його вигляд представлені на рисунку 2.2.6.

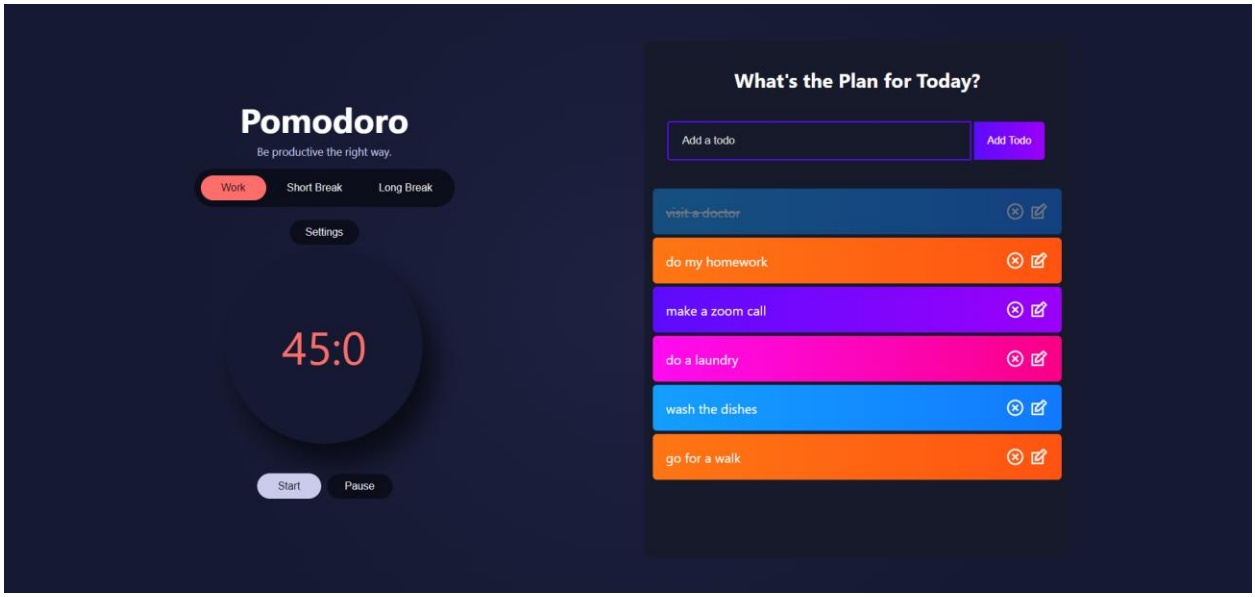

Рисунок 2.2.6 – web-додаток “ToDo”

2.3 Завантаження web-додатка на хостинг

Невід’ємним етапом розробки будь-якого web-додатка є завантаження його на хостинг. Хостинг (англ. *hosting*, від слова *host* — приймати гостей) — послуга надавання дискового простору, підключення до мережі та інших ресурсів для розміщення фізичної інформації на сервері, що постійно перебуває в мережі (наприклад інтернеті). Завантаження проекту на хостинг у інтернет є необхідним для того, аби будь-який клієнт міг отримати доступ до користування web-додатком. Різні види послуг хостингу працюють з різними напрямками. Деякі хостинги повністю розробляють веб-сайт, а деякі лише надають можливість розмістити його в інтернеті.

Для того щоб web-додаток відображався без проблем, необхідно обрати найкращу хостингову компанію. По-перше, треба визначити, яка пропускна здатність та скільки дискового простору вам знадобиться. Якщо веб-сайт містить багато графіки, десятки сторінок і велику кількість трафіку, необхідними будуть висока пропускна здатність і багато місця на диску. Якщо веб-сайт є базовим, то цими критеріями можна знехтувати.

Коли розробники намагаються знайти кращого хостингового провайдера, вони іноді забувають про тип операційних систем, який хостинг підтримує. На противагу, цей аспект є важливим для попереднього уточнення.

Надійність та доступність - важливі характеристики, які необхідно враховувати при виборі веб-хостигу. Значення доступності у кращих хостингових компаній, як правило, становить 98-99 процентів. Необхідно переконатись, що надана провайдером інформація є достовірною.

Безпека також є важливим аспектом. Недопустимим є вибір провайдера веб-хостингу, не враховуючи його функції безпеки. Наприклад, межмережеві екрани, щоденне резервне копіювання та перевірка користувача. Корисно також отримувати повідомлення про зміни, бо таким чином можна отримувати повідомлення про підозрілу активність.

Не менш важливим аспектом є час завантаження веб-сайту. Якщо сторінка буде завантажуватись довше, ніж у конкурентів, популярність через зниження комфорту користуванням web-додатоком може значно знизитись.

Хостинги є платними та безплатними. Найбільш популярні платні провайдери веб-хостингів:

- HostGator;
- Bluehost;
- A2 hosting.

Найпопулярніші безплатні провайдери веб-хостингів:

- Wix;
- InterServer;
- FreeHosting;
- GoogleHost;
- GitHub Pages.

З економічної точки зору безкоштовні веб-хостинги, звичайно, переважають плантні, але вони мають ряд недоліків, таких як:

- Деякі заповнюють веб-сторінку рекламою, через що комфорт користування значно знижується;
- Обмеження часу зберігання файлів;
- Низька швидкість завантаження;
- Низька продуктивність;
- Низьке забезпечення безпеки.

Серед представлених безкоштовних хостингів для завантаження web додатка “ToDo” було обрано хостинг GitHub Pages. Переваги використання GitHub Pages:

- Отримання безкоштовного домену;
- Безкоштовне використання з наданням усіх можливостей для невеликих проектів;
- Наявність тарифних планів для великих корпоративних проектів;
- Наявність системи контролю версій;
- Можливість розробляти проект сумісно з іншими розробниками;
- Наявність готових рішень для стилізації веб-сторінки.

Для того, аби завантажити React-проект на GitHub Pages було введено у командний рядок команду serve –s build. Після виконання цієї команди всередині проекту створилась папка build, яку було завантажено на загальнодоступний репозиторій GitHub(рис 2.3.1).

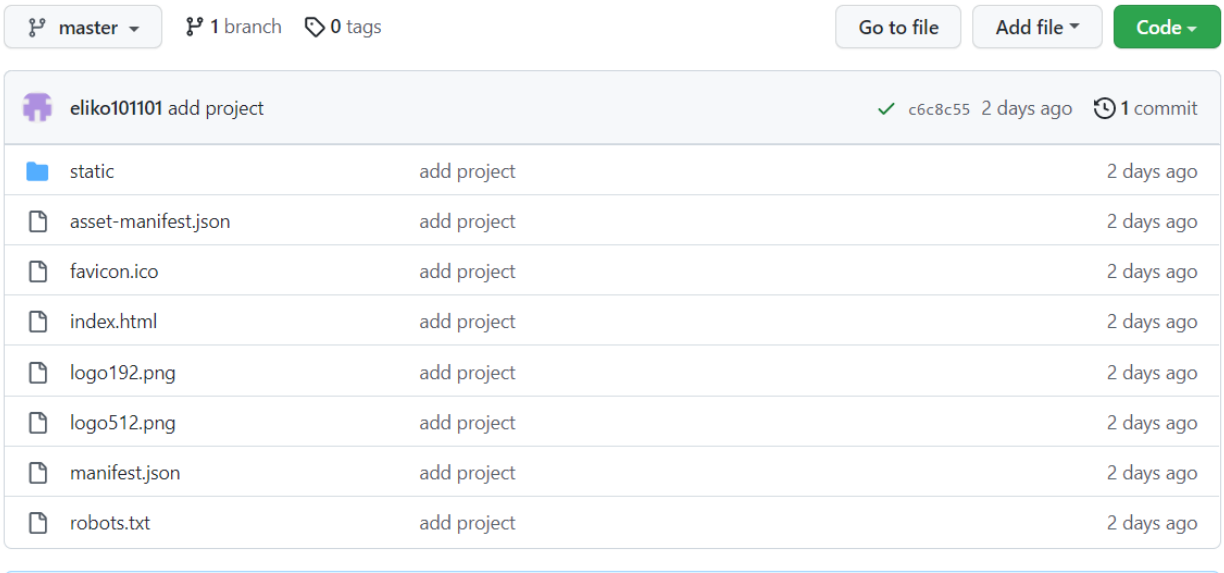

Рисунок 2.3.1 – структура проекту після збірки

Після завантаження зібраного проекту у налаштуваннях у розділі Pages(рис.2.3.2) було виконано усі необхідні налаштування, отримано безкоштовний домен і опубліковано web-додаток “ToDo” у інтернет для вільного користування.

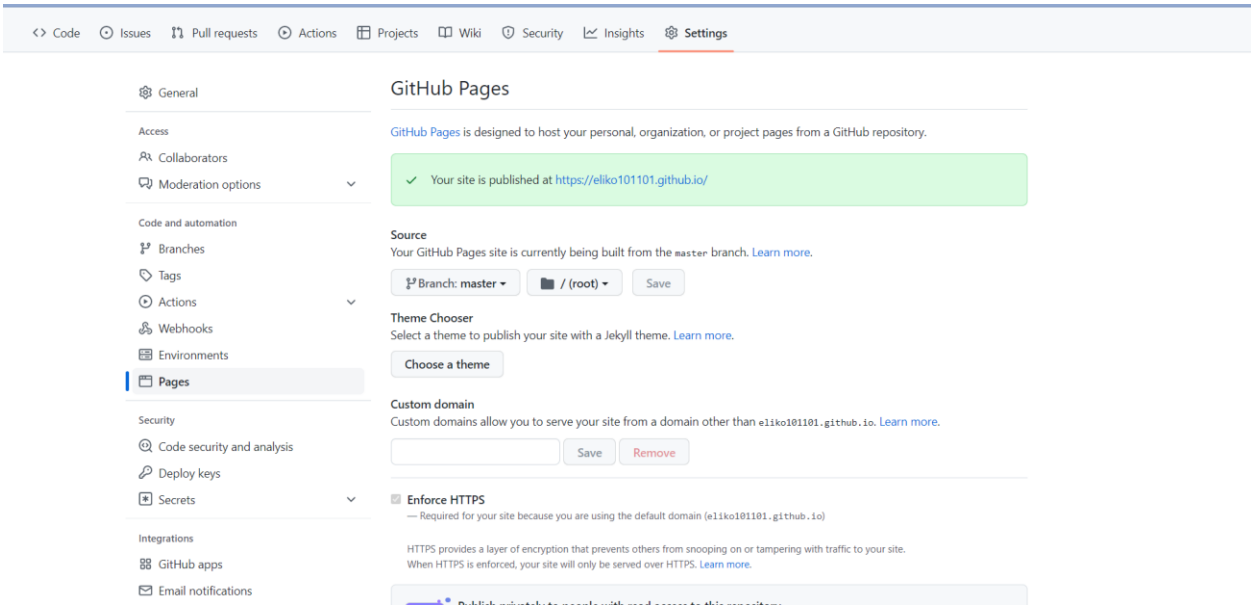

Рисунок 2.3.3 – розділ налаштувань Pages

ВИСНОВКИ

У цій дипломній роботі було описано та проаналізовано сучасні методи розробки web-додатків. Під час досліджень було розглянуто технології розробки web-додатків, визначено напрямок і сфери використання кожної з технологій. Детально описано мета і необхідність використання JS-фреймворків. Описано особливості, переваги та недоліки використання найпопулярніших JS-фреймворків, а саме React, Angular, Vue.js та Svetle. Було розглянуто тенденції розвитку серед JS-фреймворків.

В ході дослідження було визначено, що для розробки web-додатків найкраще підходить саме React через ряд переваг:

- чудово підходить для розробки як Single Page Applications, так і великих проектів;
- простий у вивченні;
- гарантія актуальності розробленого додатка, через підтримку компанією Facebook;
- велика кількість розробників, що використовують саме цей фреймворк, що збільшує кількість готових рішень;
- велика екосистема з додаткових засобів розробки;
- можливість застосовувати сторонні бібліотеки.

Для демонстрації переваг використання React було розроблено web-додаток “ToDo”. Додаток складається з двох частин: task-менеджеру та Pomadoro годинника. Першою функцією додатка є запис, видалення, редагування, відмічення виконання справ. Другою функцією додатка є підвищення продуктивності за рахунок наявності Pomadoro годинника, який працює за системою Pomadoro. У ході розробки було досліджено принципи розробки за допомогою React, способи реалізацій функцій та сфери застосування.

Також детально було розглянуто поняття хостингу як сучасної технології розробки web-додатків. У роботі було описано критерії для вибору хостингу. Проаналізувавши переваги та недоліки найпопулярніших хостингів, було опубліковано web-додаток “ToDo” у мережі інтернет за допомогою хостингу GitHub Pages, як фінальний етап розробки web-додатка.

ПЕРЕЛІК ПОСИЛАНЬ

1. Мова JavasScript та її можливості [Електронний ресурс] – Режим доступу: https://sites.google.com/site/webtehnologiietawebdizajn/mova-javascript-ta-ieie-mozlivosti
2. JavasScript фреймворки [Електронний ресурс] – Режим доступу: https://www.reclamare.ua/blog/javascript-frejmvorki/
3. Топ JavasScript фреймворків для Frontend-розробки у 2020 [Електронний ресурс] – Режим доступу: https://www.freecodecamp.org/news/complete-guide-for-front-end-developers-javascript-frameworks-2019/
4. Звіт про стан Frontend за 2022 [Електронний ресурс] – Режим доступу: https://tsh.io/state-of-frontend/#frameworks
5. Кращі JavasScript фреймворки і тенденції розробки у 2021 році [Електронний ресурс] – Режим доступу: https://bit.ly/3QAPv4Q
6. Об’єктна модель документа [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%B0_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0
7. React, Vue чи Angular: який фреймворк обрати для розробки інтерфейсу продукту в 2022 році[Електронний ресурс] – Режим доступу: https://bit.ly/3N76lFB
8. ASCII [Електронний ресурс] – Режим доступу: https://bit.ly/3y4PnDf
9. Що таке npm? [Електронний ресурс] – Режим доступу: https://nodejs.org/en/knowledge/getting-started/npm/what-is-npm/

10. Хостинг [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/%D0%A5%D0%BE%D1%81%D1%82%D0%B8%D0%BD%D0%B3
11. Як обрати необхідні послуги веб-хостингу? [Електронний ресурс] – Режим доступу: https://bit.ly/3tLwx1m
12. Варіанти хостингу [Електронний ресурс] – Режим доступу: https://bit.ly/3NaYiqZ

ДОДАТОК А

Код програми розробки web-додатка “ToDo”

```jsx
const Button = ({title,activeClass, _callback}) => {
  return (
      <button className={activeClass} onClick={_callback}>{title}</button>
  )
}
export default Buttons
```

Рисунок А.1 – компонент Button.js

```jsx
import { useContext } from 'react'
import {CountdownCircleTimer} from 'react-countdown-circle-timer'
import { SettingsContext } from '../context/SettingsContext'
const CountdownAnimation = ({item, timer, animate, children}) => {

  const { stopAimate } = useContext(SettingsContext)

    return (
      <CountdownCircleTimer
        item={item}
        isPlaying={animate}
        duration={timer * 60}
        colors={[
          ['#FE6F6B', 0.33],
          ['#FE6F6B', 0.33],
          ['#FE6F6B', 0.33],
        ]}
        strokeWidth={6}
        size={220}
        trailColor="#151932"
        onComplete={ () => {
          stopAimate()
        }}
      >
        {children}
      </CountdownCircleTimer>
    )
}

export default CountdownAnimation
```

Рисунок А.2 – компонент CountDownAnimation.js

```
import React, { useContext, useState } from 'react'
import { SettingsContext } from '../context/SettingsContext'

const SetPomodoro = () => {

    const [newTimer, setNewTimer] = useState({
        work: 45,
        short: 15,
        long: 30,
        active: 'work'
    })

    const {updateExecute} = useContext(SettingsContext)

    const handleChange = input => {
        const {name, value} = input.target
        switch (name) {
            case 'work':
                setNewTimer({
                    ...newTimer,
                    work: parseInt(value)
                })
                break;
            case 'shortBreak':
                setNewTimer({
                    ...newTimer,
                    short: parseInt(value)
                })
                break;
            case 'longBreak':
```

Рисунок А.3 – компонент SetPomadoro.js

```
            case 'longBreak':
                setNewTimer({
                    ...newTimer,
                    long: parseInt(value)
                })
                break;
        }
    }
    const handleSubmit = e => {
        e.preventDefault()
        updateExecute(newTimer)
    }
    return (
        <div className="form-container">
            <form noValidate onSubmit={handleSubmit}>
                <div className="input-wrapper">
                    <input className="input" type="text" name="work" onChange={handleChange} value={newTimer.work} />
                    <input className="input" type="text" name="shortBreak" onChange={handleChange} value={newTimer.short} />
                    <input className="input" type="text" name="longBreak" onChange={handleChange} value={newTimer.long} />
                </div>
                <button type='submit'>Set Timer</button>
            </form>
        </div>
    )
}

export default SetPomodoro
```

Рисунок А.4 – компонент SetPomadoro.js

```jsx
import React, {useState} from 'react'
import ToDoForm from './ToDoForm'
import ToDoList from './ToDoList'
import { RiCloseCircleLine} from 'react-icons/ri'
import { TiEdit} from 'react-icons/ti'
function ToDo({todos, completeTodo, removeTodo, updateTodo}) {
    const [edit, setEdit] = useState({
        id: null,
        value: ''
    });
const submitUpdate = value => {
    updateTodo(edit.id, value)
    setEdit({
        id: null,
        value: ''
    })
};
if(edit.id){
    return<ToDoForm edit={edit} onSubmit={submitUpdate}></ToDoForm>
}
  return todos.map((todo, index) => (
    <div
    className={todo.isComplete ? 'todo-row complete' :'todo-row'}
    key={index}
    >
        <div key={todo.id} onClick={() => completeTodo(todo.id)}>
            {todo.text}
        </div>
        <div className='icons'>
            <RiCloseCircleLine
            onClick={() => removeTodo(todo.id)}
            className='delete-icon'
            />
            <TiEdit
            onClick={() => setEdit({id: todo.id, value: todo.text})}
            className='edit-icon'
            />
        </div>
    </div>
  ))

}

export default ToDo
```

Рисунок А.4 – компонент ToDo.js

```jsx
import React,{useState, useEffect, useRef} from 'react'

function ToDoForm(props) {
const [input, setInput] = useState(props.edit ? props.edit.value : '');

const inputRef = useRef(null)
useEffect(() => {
    inputRef.current.focus()
})

const handleChange = e =>{
    setInput(e.target.value);
};
const handleSubmit = e => {
    e.preventDefault();
    props.onSubmit({
        id: Math.floor(Math.random() *10000),
        text: input
    });
    setInput('');

};
```

Рисунок А.5 – компонент ToDoForm.js

```jsx
  return (
    <div>
        <form className='todo-form' onSubmit={handleSubmit}>
            {props.edit ? (
                <>
            <input
                type='text'
                placeholder="Update your item"
                value={input}
                name="text"
                className='todo-input edit'
                onChange={handleChange}
                ref={inputRef}
            />
            <button className='todo-button edit'>Update</button>
            </>
            ) :
            (
                <>
            <input
                type='text'
                placeholder="Add a todo"
                value={input}
                name="text"
                className='todo-input'
                onChange={handleChange}
                ref={inputRef}
            />
            <button className='todo-button'>Add todo</button>
            </>
            )}

        </form>
    </div>
  )
}

export default ToDoForm
```

Рисунок А.6 – компонент ToDoForm.js

```
import React, {useState} from 'react'
import ToDoForm from './ToDoForm'
import ToDo from './ToDo';
function ToDoList() {
  const[todos, setTodos] = useState([]);

  const addTodo = todo => {
      if(!todo.text || /^\s*$/.test(todo.text)){
          return
      }
      const newTodos = [todo, ...todos];
      setTodos(newTodos);
      console.log(todo, ...todos);
  };

const updateTodo = (todoId, newValue) => {
    if(!newValue.text || /^\s*$/.test(newValue.text)){
        return
    }

    setTodos(prev => prev.map(item => (item.id === todoId ? newValue : item)))
}

const removeTodo = id => {
    const removeArr = [...todos].filter(todo => todo.id !== id)
    setTodos(removeArr)
}
```

Рисунок А.7 – компонент ToDoList.js

```
  const completeTodo = id => {
      let updatedTodos = todos.map(todo => {
          if (todo.id === id){
              todo.isComplete = !todo.isComplete
          }
          return todo
      })
      setTodos(updatedTodos);
  };

   return (
   <div>
       <h1>What's the Plan for Today?</h1>
       <ToDoForm onSubmit={addTodo} />
       <ToDo
           todos={todos}
           completeTodo={completeTodo}
           removeTodo={removeTodo}
           updateTodo={updateTodo}
       />


   </div>
  )
}

export default ToDoList
```

Рисунок А.8 – компонент ToDoList.js

```
import { useState, createContext } from "react";

export const SettingsContext = createContext()

function SettingsContextProvider(props) {

    const [pomodoro, setPomodoro] = useState(0)
    const [executing, setExecuting] = useState({})
    const [startAnimate, setStartAnimate] = useState(false)

    function setCurrentTimer (active_state) {
        updateExecute({
            ...executing,
            active: active_state
        })
        setTimerTime(executing)
    }

    // start animation fn
    function startTimer() {
        setStartAnimate(true)
    }
    // pause animation fn
    function pauseTimer() {
    setStartAnimate(false)
    }
    // pass time to counter
    const children = ({ remainingTime }) => {
    const minutes = Math.floor(remainingTime / 60)
    const seconds = remainingTime % 60

    return `${minutes}:${seconds}`
    }
```

Рисунок А.9 – компонент SettingContext.js

```
    // clear session storage
    const SettingsBtn = () => {
        setExecuting({})
        setPomodoro(0)
    }

    const updateExecute = updatedSettings => {
        setExecuting(updatedSettings)
        setTimerTime(updatedSettings)
    }

    const setTimerTime = (evaluate) => {
        switch (evaluate.active) {
            case 'work':
                setPomodoro(evaluate.work)
                break;
            case 'short':
                setPomodoro(evaluate.short)
                break;
            case 'long':
                setPomodoro(evaluate.long)
                break;
                default:
                    setPomodoro(0)
                break;
        }
    }

    function stopAimate() {
        setStartAnimate(false)
    }
```

Рисунок А.10 – компонент SettingContext.js

```
    return (
        <SettingsContext.Provider value={{
            pomodoro,
            executing,
            updateExecute,
            startAnimate,
            startTimer,
            pauseTimer,
            children,
            SettingsBtn,
            setCurrentTimer,
            stopAimate
        }}>
            {props.children}
        </SettingsContext.Provider>
    )

}

export default SettingsContextProvider
```

Рисунок А.11 – компонент SettingContext.js

```
import React, { useEffect, useContext } from 'react'
import Button from './components/Button'
import CountdownAnimation from './components/CountdownAnimation'
import SetPomodoro from './components/SetPomadoro'
import { SettingsContext } from './context/SettingsContext'
import ToDoList from './components/ToDoList';


const App = () => {

  const {
    pomodoro,
    executing,
    startAnimate,
    children,
    startTimer,
    pauseTimer,
    updateExecute,
    setCurrentTimer,
    SettingsBtn } = useContext(SettingsContext)

    useEffect(() => {updateExecute(executing)}, [executing, startAnimate])

  return (
    <div className='app'>
      <div className="container">
        <h1>Pomodoro</h1>
        <small>Be productive the right way.</small>
        {pomodoro !== 0 ?
        <>
          <ul className="labels">
            <li>
              <Button
                title="Work"
                activeClass={executing.active === 'work' ? 'active-label' : undefined}
                _callback={() => setCurrentTimer('work')}
              />
            </li>
            <li>
              <Button
                title="Short Break"
                activeClass={executing.active === 'short' ? 'active-label' : undefined}
                _callback={() => setCurrentTimer('short')}
              />
            </li>
            <li>
```

Рисунок А.12 – компонент SettingContext.js

```
            <li>
              <Button
                title="Long Break"
                activeClass={executing.active === 'long' ? 'active-label' : undefined}
                _callback={() => setCurrentTimer('long')}
              />
            </li>
          </ul>
          <Button title="Settings" _callback={SettingsBtn} />
          <div className="timer-container">
            <div className="time-wrapper">
              <CountdownAnimation
                item={pomodoro}
                timer={pomodoro}
                animate={startAnimate}
              >
                {children}
              </CountdownAnimation>
            </div>
          </div>
          <div className="button-wrapper">
            <Button title="Start" activeClass={!startAnimate ? 'active' : undefined} _callback={startTimer} />
            <Button title="Pause" activeClass={startAnimate ? 'active' : undefined} _callback={pauseTimer} />
          </div>
        </> : <SetPomodoro />}
      </div>
      <div className="todo-app">
      <ToDoList />
      </div>
    </div>
  )
}

export default App
```

Рисунок А.13 – компонент SettingContext.js