МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра Комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
Аліна САВЧЕНКО
«_____» _____ 2024р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ **"БАКАЛАВР"**

Тема: "Вебсайт календаря виконання завдань на базі React"

Виконавець: студент групи УС-414 Демиденко Богдан Григорович

Керівник: к.т.н., доцент Харченко Олександр Григорович

Нормоконтролер: Олександр ШЕВЧЕНКО

Київ – 2024

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет комп’ютерних наук та технологій

Кафедра Комп’ютерних інформаційних технологій

Галузь знань, спеціальність, освітньо-професійна програма: 12 “Інформаційні технології”, 122 “Комп’ютерні науки”, “Інформаційні управляючі системи та технології”

ЗАТВЕРДЖУЮ
Завідувач випускової кафедри
_____ Аліна САВЧЕНКО
«____»_____2024р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Демиденка Богдана Григоровича

(прізвище, ім'я, по батькові)

1. Тема роботи: «**Вебсайт календаля виконання завдань на базі React**» затверджена наказом ректора від “05”квітня 2024 р. за № 517/ст.

2. Термін виконання роботи: 06.05.2024 – 13.06.2024

3. Вихідні дані до роботи: процеси створення веб сайтів; тестування відображення на різних пристроях, інтегроване середовище розробки Visual Studio Code

4. Зміст пояснювальної записки: вступ, теоретичні основи розробки веб додатків, поняття, етапи, стандарти, сучасні тенденції розробки веб сайтів, технології розробки веб-додатків, UI/UX, робота з даними, Git, технології серверного використання, безпека в веб додатках, проектування та реалізація веб-сайту календаря, аналіз вимог та проектування веб сайту, використання React, бази даних та структури зберігання даних

5. Перелік обов'язкового графічного матеріалу: Технології створення UI/UX,контроль версій Git, структура та дизайн веб-сайту календаря виконання завдань, форма реєстрації та логіну, поле для створення завдань

6. Календарний план-графік

№ п/п	Завдання	Термін виконання	Підпис керівника
1.	Отримання завдання на дипломну роботу про веб-календар виконання завдань на базі React,	06.05.2024 08.05.2024	
2.	Створення та узгодження календарного плану виконання дипломної роботи	08.05.2024 10.05.2024	
3.	Консультування з науковим керівником щодо особливостей виконання дипломної роботи.	11.05.2024 13.05.2024	
4.	4.Огляд і аналіз наукових матеріалів та складання першого розділу дипломної роботи.	14.05.2024 19.05.2024	
5.	5.Розробка другого розділу	19.05.2024 24.05.2024	
6.	Формування третього розділу дипломної роботи. Завершення написання пояснювальної записки до дипломної роботи.	24.05.2024 31.05.2024	
7.	Оформлення та друк пояснювальної записки.	01.06.2024 02.06.2024	
8.	Створення презентації та доповіді для захисту дипломної роботи, підготовка до цього процесу.	03.06.2024 05.06.2024	
9.	Підготовка матеріалів дипломної роботи для надання секретарю ДЕК	06.06.2024 08.06.2024	

7. Дата видачі завдання: «06» травня 2024 р.

Керівник дипломної роботи _____ Олександр ХАРЧЕНКО

(підпис керівника)

Завдання прийняв до виконання _____ Богдан ДЕМИДЕНКО

(підпис випускника)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Вебсайт календаря виконання завдань на базі React»: 56 с., 8 рис., 18 літературних джерела.

Об’єкт дослідження: архітектура і функціональність вебсайту, структуру коду, інтерактивність, реактивність, та як користувачі можуть ефективно використовувати цей сайт для управління своїми задачами.

Мета роботи: розробити вебсайт календаря виконання завдань, щоб полегшити процес планування та управління часом для користувачів, забезпечуючи ефективне використання їх часу.

Методи дослідження: теоретичний аналіз, практичне моделювання,тестування аналіз даних.

Отримані результати та їх новизна: створено вебсайт календаря виконання завдань на базі React, що дозволяє користувачам створювати, редагувати та видаляти завдання в календарному форматі. Інтегровано API для синхронізації даних з сервером, що дозволяє зберігати інформацію про завдання в безпечному місці та доступ до них з будь-якого пристрою з підключенням до інтернету. Реалізовано адаптивний дизайн, що забезпечує зручне користування вебсайтом на різних пристроях. Додано функціонал для призначення завдань на конкретні дати та час, з можливістю повторення цих завдань.

Результати кваліфікаційної роботи рекомендується використовувати в сфері менеджменту та організації для поліпшення планування та контролю за завданнями.

REACT, ВЕБСАЙТ,. ВИКОНАННЯ ЗАВДАНЬ, БАЗА ДАНИХ, JAVASCRIPT, FRONT-END РОЗРОБКА, UI/UX ДИЗАЙН, API, JSON,NODE.JS, CSS, HTML

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

HTML	HyperText Markup Language - мова гіпертекстової розмітки, використовується для створення структури веб-сторінок.
CSS	Cascading Style Sheets - каскадні таблиці стилів, використовуються для оформлення веб-сторінок написаних на HTML та/або XHTML.
JS	JavaScript - мова програмування, що використовується для надання інтерактивності веб-сторінкам.
REACT	JavaScript бібліотека для створення користувацьких інтерфейсів.
API	Application Programming Interface - набір процедур, функцій та/або методів, що дозволяють одній програмі взаємодіяти з іншою.
UI/UX	User Interface / User Experience - користувацький інтерфейс та користувацький досвід. Це поняття використовуються для опису взаємодії користувачів з продуктом та їхнього досвіду від використання продукту.
SPA	Single-Page Application - односторінкова програма, що працює в браузері і не потребує перезавантаження сторінки під час використання.
PWA	Progressive Web App - прогресивний веб-додаток, комбінація веб-сайту і мобільного застосунку.
GIT	система контролю версій, яка дозволяє кільком програмістам працювати над одним проектом.
CMS	Content Management System - система управління контентом, дозволяє створювати та керувати веб-контентом.

WCAG Web Content Accessibility Guidelines - рекомендації щодо доступності веб-контенту для людей з інвалідністю.

SQL Structured Query Language - структурована мова запитів, що використовується для роботи з базами даних.

SEO Search Engine Optimization - оптимізація для пошукових систем, набір заходів для покращення видимості веб-сайту в результатах пошуку.

SSL Secure Sockets Layer - протокол безпечних сокетів, що забезпечує шифроване з'єднання між веб-сервером і браузером.

ВСТУП

Під час розробки веб-додатку для планування, я зауважив, що багато відволікаючих факторів впливають на ефективність роботи, яку ми виконуємо. Це можуть бути непередбачувані технічні проблеми, непостійність вимог до проекту або навіть просто перевантаження інформацією.

Відволікання також можуть бути пов'язані з робочим середовищем. Шум, перебої в інтернет-з'єднанні, відсутність комфортних умов для роботи - все це може знизити продуктивність. Крім того, особисті фактори також відіграють велику роль. Стрес, недостатній сон, нездорове харчування, відсутність фізичної активності - все це може вплинути на вашу здатність зосереджуватися та ефективно працювати.

Щоб протидіяти цим відволікаючим факторам, важливо розробити стратегії управління часом, забезпечити комфортні умови для роботи, підтримувати здоровий спосіб життя та встановлювати реалістичні цілі та очікування.

Календарне планування є важливим інструментом для боротьби з відволікаючим факторами та підвищення ефективності роботи. Використовуючи календар для планування, ви можете систематично розподіляти свої завдання на конкретні часові інтервали. Це допомагає зосередитися на одному завданні за раз, зменшуючи можливість відволікання. Календар також дозволяє вам візуалізувати свій графік, що може сприяти кращому управлінню часом та пріоритетами. Ви можете легко бачити, як ваш час розподіляється, і робити необхідні корективи. Крім того, календарне планування допомагає вам створити режим роботи, що може допомогти зменшити стрес та підвищити продуктивність. Ви можете встановити конкретний час для роботи, перерв, відпочинку та інших особистих обов'язків.

Тема кваліфікаційної роботи є створення “Веб-сайт календаря виконання завдань на базі React”, який має на меті оптимізувати керування часом та покращити продуктивність користувачів.

Щоб досягти цієї мети, потрібно виконати такі завдання:

- Вивчити та проаналізувати існуючі методи та програми і рішення для ефективного керування часом. Це допоможе нам зрозуміти основні принципи планування часу, а також оцінити переваги та недоліки доступних на ринку рішень;
- Дослідити характеристики впровадження веб-додатків для планування та управління часом. З'ясую, які функції найбільше вимагають користувачі, і як можна втілити ці функції в нашому веб-додатку;
- Розроблю веб-додаток "Календар", який бере до уваги потреби користувачів і особливості управління часом. Використаю отримані в ході дослідження знання для створення ефективного та користувацького інтерфейсу;
- Проведу тестування та оптимізацію веб-додатку "Календар". Після того, як додаток буде розроблений, проведу тестування, щоб забезпечити його надійність та ефективність. Також зберу відгуки від користувачів, щоб внести необхідні покращення.

Веб-сайт "Календар" може виявитися неймовірно корисним інструментом для організації та планування вашого часу, незалежно від того, чи йдеться про окремий день, цілий тиждень, місяць або навіть весь рік. Цей інструмент відкриває широкі можливості не лише для особистого використання, наприклад, для планування відпочинку, зустрічей з друзями або сімейних свят, але і є незамінним помічником у професійному житті. Він допоможе вам слідкувати за строками виконання роботи, планувати бізнес-зустрічі, координувати проекти та контролювати виконання обов'язків.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ВЕБ-ДОДАТКІВ

1.1. Поняття веб-додатку та його класифікація

Веб-сайт представляє собою комплекс веб-сторінок, що містять гіперпосилання і розташовані на одному або кількох веб-серверах. Його можна знайти в інтернеті за допомогою унікальної адреси (URL). Важливо, що веб-сайт надавав чітку, корисну та актуальну інформацію для своїх відвідувачів. Також він повинен бути доступним для всіх користувачів, незалежно від їхнього технічного досвіду чи типу використовуваного пристрою. Він має стимулювати взаємодію з користувачами, дозволяючи їм коментувати, голосувати, залишати відгуки або зв'язуватися з власником сайту. Його структура навігації має бути чіткою та логічною, щоб користувачі могли легко знаходити потрібну їм інформацію.

Дизайн веб-додатку має бути привабливим та естетичним, який відповідає його темі та цільовій аудиторії. Швидкість завантаження веб-сайту має бути високою, щоб користувачі не витрачали час на очікування. Він повинен бути оптимізований для пошукових систем, щоб користувачі, які шукають інформацію в Інтернеті, могли легко його знайти. Це важливий інструмент в сучасному світі, який може бути різноманітним за формою та розміром, кожен з них має свою унікальну мету та функціональність. Вони можуть бути класифіковані за різними параметрами, зокрема за функціональністю, цільовою аудиторією, технологіями створення та іншими.

Щодо функціональності, веб-сайти можуть бути інформаційними, що надають корисну та актуальну інформацію з певної теми, інтерактивними, що сприяють взаємодії з користувачами, трансакційними, що дозволяють купувати товари чи послуги онлайн, сайтами для просування бренду або персональними веб-сайтами.

Згідно з цільовою аудиторією, веб-сайти можуть бути широкоформатними, нішевими, бізнес-орієнтованими, урядовими або некомерційними.

Відповідно до технологій створення, веб-сайти можуть бути статичними, динамічними, базуються на CMS, односторінковими або прогресивними веб-додатками.

Також вони можуть відрізнятися за типом домену, географічним розташуванням, мовою та платформою. Наприклад, вони можуть мати домени .com, .org, .net, .edu, .gov тощо; бути локальними, національними або міжнародними; бути англомовними або багатомовними; бути орієнтованими на мобільні пристрої, десктопи або бути адаптивними. Також є різні форми онлайн-присутності, кожна з яких має свої особливості.

Наприклад: веб-сайт - це центральний елемент онлайн-присутності бізнесу або особи. Він може містити детальну інформацію про продукти або послуги, блоги, контактні дані та інше. Веб-сайт дає можливість контролювати повністю весь контент і дизайн, що не завжди можливо в інших формах онлайн-присутності.

Соціальні мережі, такі як Facebook, Instagram, Twitter, LinkedIn, дозволяють вести двосторонню комунікацію з аудиторією, отримувати відгуки в реальному часі та швидко реагувати на них. Вони допомагають підтримувати активну присутність в інтернеті, залучати нових клієнтів та підтримувати відносини з існуючими.

Форуми - це місця, де користувачі можуть обговорювати певні теми, задавати питання та отримувати відповіді. Вони можуть бути корисними для отримання відгуків від користувачів, надання підтримки клієнтам або створення спільноти навколо бренду чи продукту.

Електронна пошта - це особистий та прямий канал комунікації з аудиторією. Вони може бути використана для розсилки новин, акцій, оновлень продуктів або

будь-якої іншої інформації. Електронна пошта також важлива для ведення бізнес-комунікацій, таких як відправка пропозицій, угод, рахунків-фактур тощо.

1.2. Етапи розробки веб-сайту

Розробка веб-сайту - це складний та багатоетапний процес, який вимагає великої уваги до деталей та співпраці між різними спеціалістами. Першим і одним з найважливіших етапів є аналіз та планування. На цьому етапі команда фахівців проводить замовником детальні консультації, визначає його потреби та очікування від майбутнього сайту. Тут також важливо встановити цілі проекту, визначити цільову аудиторію та зрозуміти основні завдання, які має вирішити веб-сайт.

Після аналізу настає етап дизайну, де візуальна концепція сайту набуває своєї форми. Дизайнери створюють макети, обирають кольорову палітру, шрифти та інші елементи, які визначатимуть зовнішній вигляд та користувацький досвід сайту. Важливо, щоб дизайн був не лише привабливим, але й зручним для користувачів, щоб вони легко знаходили необхідну інформацію та взаємодіяли з сайтом без зайвих перешкод.

Після затвердження дизайну розпочинається етап розробки, де програмісти переносять концепцію в живий код. Тут створюються всі необхідні функції, бази даних, а також забезпечується адаптивність сайту для різних пристроїв та браузерів. Кожен функціональний елемент та сторінка сайту ретельно розробляються з урахуванням кращих практик веб-програмування.

Після завершення розробки настає етап тестування, де проводиться комплексна перевірка всіх функцій сайту на наявність помилок та відповідність специфікаціям. Також важливо впевнитися, що сайт працює коректно на різних пристроях та в різних веб-середовищах. Тестування допомагає виявити та виправити будь-які проблеми перед запуском сайту.

І нарешті, останній етап - запуск сайту. Після успішного завершення всіх попередніх етапів сайт готовий до публікації та доступу для користувачів. Після

запуску важливо продовжувати вдосконалювати та підтримувати сайт, вносячи зміни та оновлення відповідно до потреб бізнесу та користувачів.

1.3. Основні технології розробки веб-сайтів

Мови розмітки HTML та CSS відіграють ключову роль у створенні веб-сайтів. HTML (HyperText Markup Language) використовується для створення структури та контенту веб-сторінок. Вона визначає, які елементи з'являються на сторінці, їх ієрархію та взаємозв'язки. HTML відповідає за текст, зображення, відео, посилання та інші елементи, які користувач бачить на веб-сторінці.

CSS (Cascading Style Sheets) використовується для визначення зовнішнього вигляду веб-сторінок. Вона дозволяє встановлювати кольори, шрифти, розміри, відступи та інші стилізовані елементи для кожного елементу на сторінці. CSS допомагає зробити веб-сайт привабливим та професійним, а також забезпечує його адаптивність для різних пристроїв.

HTML та CSS часто використовуються разом: HTML відповідає за структуру сторінки, а CSS - за її вигляд. Вони доповнюють один одного, створюючи зручне та естетичне веб-середовище для користувача. Знання цих мов є необхідним для веб-розробників, оскільки вони є основою будь-якого веб-сайту і визначають його зовнішній вигляд та функціонал.[2, с.126]

Також, крім мов розмітки HTML та CSS, важливими компонентами веб-розробки є мови програмування, такі як JavaScript, PHP та Python.

JavaScript - це мова програмування, яка використовується для реалізації інтерактивності на веб-сайтах. Вона дозволяє створювати анімацію, динамічно змінювати контент на сторінці, обробляти події користувача та багато іншого.

PHP - це мова програмування, яка використовується для створення динамічних веб-сайтів та роботи з базами даних. PHP використовується для обробки форм, взаємодії з базами даних, створення сесій та кукісів, а також для роботи з файлами на сервері. PHP дозволяє створювати функціональні веб-додатки з можливістю розширення та підтримки.

Python - це універсальна мова програмування, яка широко використовується для веб-розробки. Python має простий синтаксис та велику кількість бібліотек, що робить його популярним вибором для розробки веб-додатків. Python використовується для створення веб-серверів, веб-фреймворків, обробки даних та багатьох інших завдань веб-розробки.

Під час розробки веб-додатків розробники користуються різноманітними фреймворками та бібліотеками, які допомагають спростити процес створення і поліпшити функціональність додатків. Наприклад, для фронтенд розробки широко використовуються React.js, Angular та Vue.js, які дозволяють створювати інтерактивні інтерфейси. Для бекенду популярні фреймворки Django, Ruby on Rails та Express.js, які прискорюють розробку серверної частини додатків. У сфері CSS використовуються фреймворки Bootstrap та Foundation для швидкої розробки адаптивних веб-сайтів. Бібліотеки jQuery та D3.js допомагають взаємодіяти з DOM та створювати візуалізацію даних на веб-сторінках.

Ці інструменти значно полегшують роботу розробників, дозволяючи їм швидше створювати високоякісні веб-додатки з різноманітними можливостями. Використання фреймворків та бібліотек є важливою складовою успішної веб-розробки, оскільки вони надають готові рішення для багатьох задач та сприяють підвищенню продуктивності розробників.

Системи керування контентом (CMS) є також важливим інструментом для створення та управління веб-сайтами без необхідності глибоких знань програмування. CMS дозволяють користувачам легко додавати, редагувати та керувати контентом на своїх веб-сайтах. Ось деякі популярні CMS:

1. WordPress: Найпопулярніша CMS у світі, яка використовується для створення різноманітних веб-сайтів, від блогів до корпоративних порталів.

2. Joomla: Ще одна популярна CMS, яка володіє великою кількістю розширень та можливостей для розробки різноманітних веб-сайтів.

3. Drupal: CMS, яка використовується для створення складних та великих веб-сайтів з великою кількістю функціональних можливостей.

4. Magento: CMS, спеціалізована на створення інтернет-магазинів з широкими можливостями управління товарами, замовленнями та оплатою.

5. Shopify: Популярна CMS для створення за управління електронними комерційними платформами.

CMS дозволяють користувачам без технічних навичок ефективно управляти вмістом своїх веб-сайтів, додавати нові сторінки, публікувати контент та керувати дизайном. Вони також надають широкі можливості для розширення функціоналу за допомогою плагінів та тем оформлення. Вибір певної CMS залежить від конкретних потреб проекту та рівня технічних знань користувача.

1.4. Стандарти та норми розробки веб-сайтів

Доступність веб-сайтів - це принцип розробки, який спрямований на створення веб-ресурсів, що доступні для користувачів з різними фізичними та інтелектуальними обмеженнями. Один з основних стандартів доступності - WCAG (Web Content Accessibility Guidelines) - надає рекомендації щодо створення веб-сайтів, які будуть доступні для людей з різними видами обмежень, такими як відсутність зору, обмежена рухливість або вади слуху. Дотримання стандартів WCAG означає, що веб-сайт повинен мати можливість навігації за допомогою клавіатури, альтернативний текст для зображень, достатній контраст між текстом та фоном, можливість збільшення шрифту для полегшення читання, а також використання семантичних елементів HTML для кращого розуміння структури сторінки. Крім WCAG, існують інші стандарти доступності, такі як Section 508 у США або EN 301 549 в Європейському союзі. Дотримання цих стандартів важливо не лише з точки зору етики та соціальної відповідальності, але й може мати юридичні наслідки, особливо для урядових та комерційних веб-сайтів.

Забезпечення доступності веб-сайтів важливо не лише для покращення користувацького досвіду для всіх користувачів, але й для розширення аудиторії та підвищення конверсії. Розробники повинні уважно вивчати та впроваджувати принципи доступності в процесі створення веб-сайтів, щоб забезпечити їхню

доступність для всіх користувачів. Швидкість завантаження веб-сайту має прямий вплив на користувацький досвід. Користувачі очікують, що веб-сайт відкриється швидко і без затримок. Повільне завантаження може призвести до того, що користувачі втратять зацікавленість та покинуть сайт, що може негативно вплинути на конверсію та репутацію бренду.

Крім того, швидкість завантаження веб-сайту також впливає на SEO. Пошукові системи, такі як Google, враховують швидкість завантаження при ранжуванні сторінок у пошукових результатах. Швидкість завантаження може вплинути на позиції вашого сайту в пошукових системах, що робить її важливим фактором для успішного SEO стратегії.

Оптимізація швидкості завантаження веб-сайту є постійним процесом, оскільки з розвитком технологій та збільшенням обсягу контенту швидкість завантаження може змінюватися. Регулярний аналіз та вдосконалення елементів, що впливають на швидкість завантаження, допомагають забезпечити оптимальну продуктивність веб-сайту. Отже, швидкість завантаження веб-сайту є критично важливим фактором для забезпечення задоволення користувачів, покращення SEO та успішної роботи веб-сайту в цілому. Важливо приділяти увагу оптимізації швидкості завантаження як одному з пріоритетних завдань при розробці та підтримці веб-сайту.

Забезпечення безпеки веб-сайту - це складний та постійний процес, який включає в себе широкий спектр заходів і стратегій для захисту веб-ресурсу від різноманітних загроз. Однією з ключових аспектів безпеки є захист від хакерських атак, таких як SQL ін'єкції, Cross-Site Scripting (XSS), крадіжка ідентифікаторів сесій та інші форми зловмисного вторгнення. Крім захисту від хакерських атак, безпека веб-сайту також охоплює захист від вірусів, шкідливих програм та інших шкідливих впливів. Важливо встановлювати антивірусне програмне забезпечення на сервері та регулярно сканувати веб-сайт на наявність шкідливого коду. Також необхідно слідкувати за безпекою сторонніх компонентів, таких як плагіни, теми та бібліотеки, і вчасно оновлювати їх до останніх версій для усунення вразливостей.

Додатково, для забезпечення безпеки веб-сайту важливо використовувати механізми аутентифікації та авторизації, які дозволяють контролювати доступ

користувачів до різних частин сайту та даних. Регулярне моніторинг логів, виявлення підозрілих активностей та реагування на потенційні загрози - також важливі елементи ефективної стратегії безпеки веб-сайту. В цілому, безпека веб-сайту вимагає комплексного підходу, поєднуючи технічні, організаційні та процесуальні заходи для забезпечення надійності та захисту інформації. Правильно налаштована та підтримувана система безпеки допомагає запобігти можливим загрозам та забезпечити безпеку веб-сайту на довгострокову перспективу.

Респонсивність веб-сайтів - це властивість, яка означає, що дизайн та розмір веб-сайту автоматично адаптуються до розміру екрану пристрою, на якому він переглядається. Це означає, що веб-сайт буде виглядати та працювати оптимально на будь-яких пристроях, включаючи комп'ютери, планшети та смартфони, незалежно від їхнього розміру екрану та орієнтації.

Основні переваги респонсивного дизайну включають:

1. Кращий користувацький досвід: Користувачі можуть легко переглядати вміст сайту на будь-якому пристрої без необхідності масштабування або прокручування горизонтально.

2. Покращення SEO: Пошукові системи, такі як Google, віддають перевагу респонсивним веб-сайтам у пошукових результатах, що може позитивно вплинути на позиції веб-сайту.

3. Зменшення шкалювання: Респонсивний дизайн дозволяє показувати один і той самий контент на всіх пристроях, що спрощує управління сайтом та зменшує необхідність розробки окремих версій для різних пристроїв. [4, с.201]

Для досягнення респонсивності веб-сайту рекомендується використовувати гнучкі сітки (flexbox або grid), медіа-запити для адаптивного стилювання та ретельно тестувати веб-сайт на різних пристроях для переконання в правильності відображення на кожному з них.

Враховуючи зростання використання мобільних пристроїв для перегляду веб-сайтів, респонсивний дизайн стає все більш важливим для забезпечення успіху та конкурентоспроможності веб-проекту.

1.5. Сучасні тенденції розробки веб-сайтів

Односторінкові веб-сайти (SPA) є однією з сучасних тенденцій у веб-розробці, яка набула популярності через свою інноваційність та ефективність. SPA - це веб-сайти, які складаються з однієї HTML-сторінки, яка динамічно оновлюється без перезавантаження всієї сторінки при взаємодії користувача.

Основні переваги односторінкових веб-сайтів (SPA) включають:

- Швидкість: SPA завантажуються швидко, оскільки вони не потребують перезавантаження сторінки при переходах між різними розділами;
- Користувацький досвід: SPA надають плавну та інтерактивну взаємодію, що робить користувацький досвід більш приємним та зручним;
- Адаптивність: SPA легко адаптуються до різних пристроїв та розмірів екранів, що робить їх ідеальними для мобільних пристроїв;
- Простота у розробці: SPA дозволяють розробникам працювати з однією сторінкою, що спрощує управління кодом та зменшує час розробки.

Проте, існують деякі виклики та обмеження при розробці SPA, такі як SEO оптимізація, підтримка старих браузерів, управління станом додатку та безпека. Важливо враховувати ці аспекти при виборі SPA для вашого проекту. У сучасному веб-середовищі односторінкові веб-сайти (SPA) стають все популярнішими завдяки своїй інноваційності та можливостям. Вони дозволяють створювати сучасні та динамічні веб-додатки, які забезпечують високий рівень користувацького досвіду та ефективності[3, с.53].

Є такоє прогресивні веб-додатки (PWA) - це сучасна технологія розробки веб-додатків, яка поєднує в собі переваги веб-сайтів і мобільних додатків. Основна ідея PWA полягає в тому, щоб створити додаток, який може працювати як звичайний веб-сайт у браузері, але при цьому мати функціонал, що характерний для нативних мобільних додатків.

Основні переваги прогресивних веб-додатків включають високу швидкість завантаження, можливість роботи офлайн, незалежність від конкретної платформи (тобто PWA можуть працювати як на Android, так і на iOS), можливість отримувати

сповіщення і доступ до деяких функцій пристрою, таких як камера або геолокація. Для розробки PWA використовуються сучасні веб-технології, такі як Service Workers, Web App Manifest, HTTPS, що дозволяє забезпечити високу продуктивність і безпеку додатку. Важливою особливістю PWA є їх простота у встановленні для користувачів, оскільки не потрібно встановлювати їх через магазин додатків, а вони можуть бути додані на головний екран пристрою одним кліком.

Голосові інтерфейси та чат-боти є двома сучасними технологіями, які значно змінюють спосіб взаємодії користувачів з комп'ютерами і програмним забезпеченням. Голосові інтерфейси дозволяють користувачам взаємодіяти з пристроями за допомогою голосових команд. Це означає, що замість введення тексту або натискання кнопок, користувач може просто говорити з пристроєм, який розпізнає його голос та виконує відповідні дії. Голосові інтерфейси широко використовуються в особистих ассистентах, таких як Siri від Apple, Google Assistant від Google, а також у "розумних" домашніх пристроях, автомобілях та інших сферах.

Чат-боти, з іншого боку, це програмні агенти, які можуть спілкуватися з користувачами через текстовий інтерфейс, як правило, у месенджерах або на веб-сайтах. Чат-боти можуть виконувати різноманітні завдання, від відповіді на запитання користувачів до проведення операцій в онлайн-магазинах. Вони базуються на штучний інтелекті, машинному навчанні та обробці природної мови для розуміння та генерації текстових повідомлень. Обидві ці технології спрощують взаємодію користувачів з комп'ютерами, роблячи її більш інтуїтивною та ефективною. Голосові інтерфейси та чат-боти використовуються в різних галузях, включаючи клієнтське обслуговування, маркетинг, продажі та багато інших, демонструючи потенціал цих технологій для поліпшення користувацького досвіду та автоматизації бізнес-процесів.

Використання штучного інтелекту (ШІ) у веб-розробці є одним з найбільш інноваційних та перспективних напрямків сучасної технології. ШІ дозволяє створювати веб-додатки, які можуть аналізувати дані, робити прогнози, рекомендації та взаємодіяти з користувачами на більш інтелектуальному рівні. Одним з основних способів використання ШІ у веб-розробці є персоналізація контенту. Завдяки

алгоритмам машинного навчання, веб-сайти можуть адаптуватися до індивідуальних потреб користувачів, показуючи їм відповідний контент, рекомендації та пропозиції[5,с. 189].

Інший спосіб використання ШІ - це автоматизація процесів розробки. Наприклад, інструменти на основі штучного інтелекту можуть допомагати при аналізі коду, виявленні помилок, оптимізації веб-сторінок для кращої продуктивності та безпеки. Також ШІ використовується для покращення користувацького досвіду, наприклад, за допомогою чат-ботів для підтримки користувачів, систем автоматичного відповідання на запитання або голосових асистентів для навігації по сайту.

1.6. Адаптивність та кросбраузерність у веб-розробці

Адаптивність та кросбраузерність є основоположними принципами сучасної веб-розробки.

Адаптивність полягає в забезпеченні оптимального відображення та функціонування веб-сайту на різних пристроях: персональних комп'ютерах, ноутбуках, планшетах, смартфонах та ін. Це досягається за допомогою спеціальних технік CSS, як-от media queries, які дозволяють коригувати стилі веб-сторінки в залежності від характеристик пристрою.

Кросбраузерність веб-додатків означає, що вони правильно працюють і відображаються в різних веб-браузерах, таких як Google Chrome, Mozilla Firefox, Safari, Opera, Microsoft Edge та ін. Проблеми з кросбраузерністю виникають часто через різницю в інтерпретації стандартів веб-розробки різними браузерами.

Для вирішення цих проблем, розробники користуються спеціальними техніками і інструментами, включаючи поліфіли, які надають старішим браузерам підтримку новіших стандартів API, бібліотеки JavaScript для покращення сумісності між браузерами, а також різні сервіси для тестування кросбраузерної сумісності.

Врахування цих принципів під час розробки допомагає забезпечити доступність веб-додатків для широкого кола користувачів, незалежно від їхніх

пристроїв і вибору браузера, а також підвищує узагальненість та масштабованість розробленого продукту.

Для забезпечення адаптивності веб-додатків, розробники повинні враховувати різні аспекти дизайну і навігації. Адаптивний дизайн має бути гнучким, щоб розтягуватися і стискатися, в залежності від розміру екрана. Навігація має бути простою і зручною для користування, незалежно від пристрою. Окрім того, важливо перевірити, як веб-додаток відображається і працює на різних пристроях. Тестування може включати перевірку на мобільних телефонах, планшетах, настільних комп'ютерах з різними розмірами екрана, а також у різних орієнтаціях екрана.

Щодо кросбраузерності, треба забезпечити, що веб-додаток не просто виглядає однаково в усіх браузерах, але й працює однаково. Питання кросбраузерності особливо важливі для JavaScript, оскільки різні браузери можуть інтерпретувати код по-різному. Одним з популярних підходів до вирішення проблем кросбраузерності є прогресивне покращення (Progressive Enhancement), який полягає в розробці базової версії веб-додатка, яка буде працювати в усіх браузерах, а потім додаються додаткові функції і поліпшення для браузерів, які їх підтримують.

1.7. Впровадження веб-додатків: хостинг, домен, SSL

Одним з головних етапів розробки веб-додатку є його впровадження та розміщення в інтернеті. Цей процес включає кілька ключових компонентів: хостинг, домен та SSL сертифікат. Вершинною задачею хостингу є надання місця на сервері для розміщення веб-додатка. Хостинг може бути безкоштовним чи платним, при цьому якість та надійність послуги може значно відрізнятися. При виборі хостингу слід враховувати такі фактори, як вартість, надійність, швидкість доступу до додатка, об'єм доступного дискового простору, наявність технічної підтримки та автоматичного резервного копіювання данних.

Кожний веб-додаток повинен мати унікальну адресу в інтернеті - доменне ім'я. Домен слугує ідентифікатором веб-додатка в глобальній мережі, дозволяє користувачам легко знайти та відвідати сайт. У процесі реєстрації домену слід враховувати такі аспекти, як простота написання і запам'ятовування імені, його унікальність та відповідність контенту веб-додатка.

SSL сертифікат є необхідним для забезпечення безпечного з'єднання між сервером і клієнтом. Він шифрує дані, передані між веб-браузером користувача та сервером, запобігаючи можливому перехопленню або зміні цієї інформації зловмисниками. SSL сертифікат також підвищує довіру користувачів до веб-додатка, оскільки відображається в браузері як позначка безпеки.

Особливу увагу слід приділити вибору провайдера SSL сертифікатів, оскільки від цього залежить рівень безпеки веб-додатка. При виборі сертифіката слід враховувати такі параметри, як сумісність із різними браузерами та пристроями, що використовують користувачі, а також швидкість і якість технічної підтримки провайдера.

Впровадження веб-додатку не закінчується на етапі його розміщення в інтернеті. Потрібно постійно моніторингувати роботу додатка, вчасно оновлювати його, адаптувати до змінюваних вимог та очікувань користувачів. Крім того, регулярно слід перевіряти безпеку додатка, виявляти та усувати можливі вразливості. Процес впровадження веб-додатку є складним та багатоетапним, вимагає глибоких знань та навичок у сфері веб-технологій, але при правильному підході дозволяє створити надійний, ефективний та високоякісний продукт.

Конфігурація сервера є ще одним критичним компонентом під час розгортання веб-додатку. Це включає налаштування бази даних, інсталяцію необхідних бібліотек та модулів, налаштування середовища виконання для мови програмування, яка була використана для розробки додатку. Крім того,

важливо налаштувати файрвол та інші системи безпеки, щоб забезпечити захист від потенційних атак.

Іншим важливим аспектом є оптимізація продуктивності веб-додатку. Це може включати такі елементи, як кешування, мінімізація та стиснення файлів CSS та JavaScript, оптимізація зображень та інше. Оптимізація продуктивності може значно покращити швидкість завантаження веб-додатку та загалом покращити його роботу.

Також, при впровадженні веб-додатку, не забувайте про SEO (Search Engine Optimization). Правильна оптимізація для пошукових систем може допомогти збільшити видимість вашого веб-додатку в інтернеті та привести більше користувачів. Це включає використання ключових слів, налаштування метатегів, створення SEO-дружнього URL та інше.

Після впровадження веб-додатку, важливо проводити постійний моніторинг його роботи. Це допоможе вам вчасно виявити будь-які проблеми, а також допоможе вам краще зрозуміти, як користувачі взаємодіють з вашим додатком. Можна використовувати різні інструменти для моніторингу, такі як Google Analytics, New Relic, Pingdom та інші.

1.8. Висновки до 1 розділу

У цьому розділі ми детально розглянули поняття веб-додатків, їх класифікацію, етапи розробки, основні технології, стандарти та норми, а також сучасні тенденції веб-розробки, аспекти адаптивності, кросбраузерності і впровадження веб-додатків. Розглянуте введення охоплює широкий спектр знань, необхідних для успішного створення, оптимізації та підтримки веб-додатків у сучасному цифровому просторі.

Починаючи з базового визначення веб-сайтів і їх функціональності, ми перейшли до більш складних концепцій, як-от етапи розробки, включаючи аналіз, дизайн, розробки, тестування та запуск. Значна увага приділялась важливості

користувацького досвіду, швидкості завантаження, доступності та безпеці веб-додатків, акцентуючи на тому, що ці елементи є ключовими для задоволення потреб кінцевих користувачів та досягнення бізнес-цілей.

Особливий акцент зроблено на новітніх технологіях та методологіях, таких як HTML, CSS, JavaScript, PHP, Python, а також різноманітних фреймворках і бібліотеках, які значно спрощують процес розробки та дозволяють створювати високофункціональні та ефективні веб-додатки. Також було висвітлено важливість дотримання стандартів доступності, як-от WCAG, і норм розробки для забезпечення широкої доступності веб-ресурсів. Сучасні тенденції, такі як односторінкові додатки (SPA), прогресивні веб-додатки (PWA), штучний інтелект (ШІ) у веб-розробці, голосові інтерфейси та чат-боти, відкривають нові можливості для поліпшення інтерактивності та функціональності веб-додатків, а також автоматизації та персоналізації користувацького досвіду.

В контексті адаптивності та кросбраузерності я оглянув методи та підходи, які забезпечують оптимальний відгук та сумісність веб-додатків на різноманітних пристроях та веб-браузерах, що є критично важливим для досягнення найширшої аудиторії.

Завершальна частина, присвячена впровадженню веб-додатків, охоплює аспекти хостингу, реєстрації доменного імені та важливість SSL сертифікації для забезпечення безпеки та довіри користувачів. Наголошено на постійному моніторингу, технічній підтримці, SEO оптимізації та необхідності регулярних оновлень для відповідності поточним вимогам безпеки та технологій.

Цей огляд підкреслює, що успішний веб-додаток - це результат глибокого дослідження, ретельного планування, використання сучасних технологій та методологій, постійної взаємодії з користувачами та адаптації до змінюваних тенденцій і стандартів у світі веб-розробки.

РОЗДІЛ 2
ТЕХНОЛОГІЇ РОЗРОБКИ ВЕБ-ДОДАТКІВ

2.1. Технології створення UI/UX

UI (User Interface) та UX (User Experience) - це дві важливі складові, які визначають успіх будь-якого цифрового продукту. UI та UX допомагають забезпечити зручність та привабливість продукту для кінцевого користувача. UI, або інтерфейс користувача, стосується візуального оформлення програмного забезпечення або веб-сайту. Він включає в себе все, що користувач бачить і взаємодіє з на екрані: кнопки, текст, зображення, слайдери, форми тощо. Дизайнери UI використовують графічні інструменти, такі як Sketch або Adobe XD, для створення макетів сторінок та інтерактивних елементів.

UX, або досвід користувача, стосується загального досвіду користувача при використанні продукту. Він включає в себе емоційні відчуття, сприйняття продукту, фізичну та психологічну взаємодію. Дизайнери UX проводять дослідження користувачів, створюють персонажів та сценарії використання, розробляють карти потоку користувачів і проводять тестування. Процес створення UI/UX починається з визначення цілей бізнесу та потреб користувачів. На наступному етапі формуються персонажі та сценарії використання. Потім створюються провізорні макети або прототипи, які допомагають уявити, як буде виглядати кінцевий продукт. Після цього проводиться тестування з користувачами, щоб отримати зворотний зв'язок та внести необхідні зміни. Фінальний етап - це створення високоякісних макетів для розробників. Це включає в себе візуальне оформлення, анімацію, взаємодію та адаптивний дизайн.

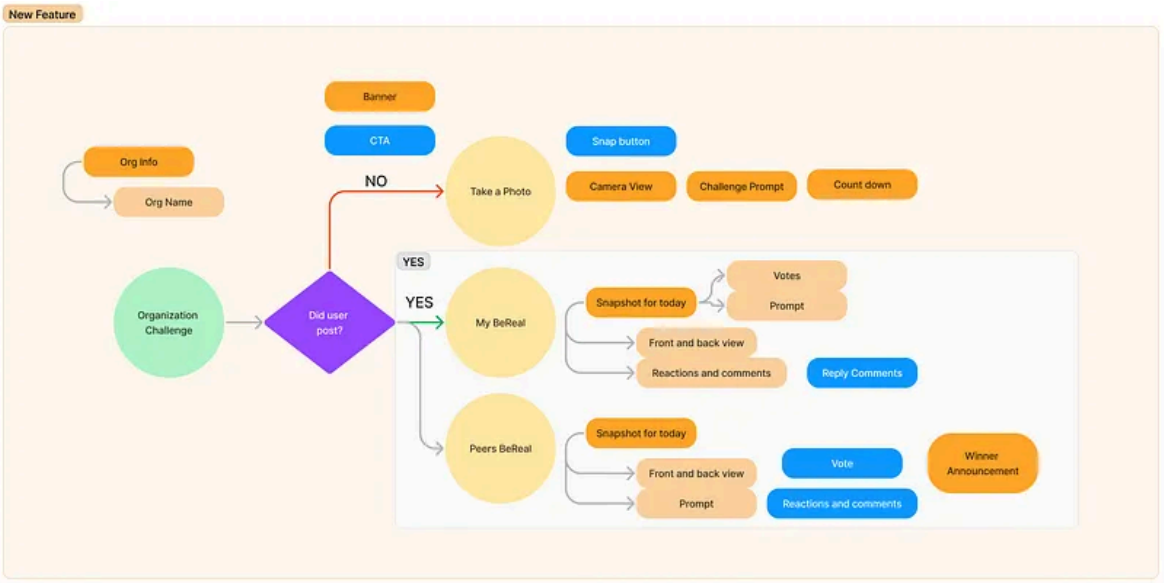

Рис. 2.1. Процес створення UI/UX

Важливо пам'ятати, що UI/UX - це не одноразовий процес. Після запуску продукту потрібно постійно аналізувати поведінку користувачів, отримувати зворотний зв'язок та робити вдосконалення. Останнім часом також набирає популярності концепція "Design Systems", яка включає в себе створення єдиної системи дизайн-компонентів, які можна використовувати для створення UI/UX в усіх продуктах компанії.

"Design Systems" - це колекція стандартів, компонентів та принципів, які допомагають підтримувати взаємодію між розробниками та дизайнерами. Вони дозволяють створювати послідовний та зрозумілий досвід для користувача на всіх платформах і в усіх продуктах компанії. Використання "Design Systems" може значно підвищити швидкість та ефективність розробки, а також забезпечити більшу послідовність в дизайні. Це забезпечує, що усі елементи інтерфейсу, від кнопок до модальних вікон, мають однаковий вигляд та відчуття незалежно від того, де вони використовуються . UI/UX дизайнери також використовують різні методи та технології для збору даних про користувачів і їхнє взаємодію з продуктом. Це можуть бути опитування, інтерв'ю, спостереження за користувачем, а також аналіз даних, отриманих з інструментів відстеження, таких як Google Analytics. Крім того, UI/UX дизайнери часто працюють в тісній взаємодії з командами розробників,

маркетингу, продажів та підтримки, щоб забезпечити, що продукт відповідає потребам всіх зацікавлених сторін.

Важливо пам'ятати, що UI/UX не відноситься лише до веб-сайтів та мобільних додатків. Це також стосується будь-якого іншого продукту або сервісу, з яким взаємодіє користувач - від автоматів з продажу напоїв до автомобілів.

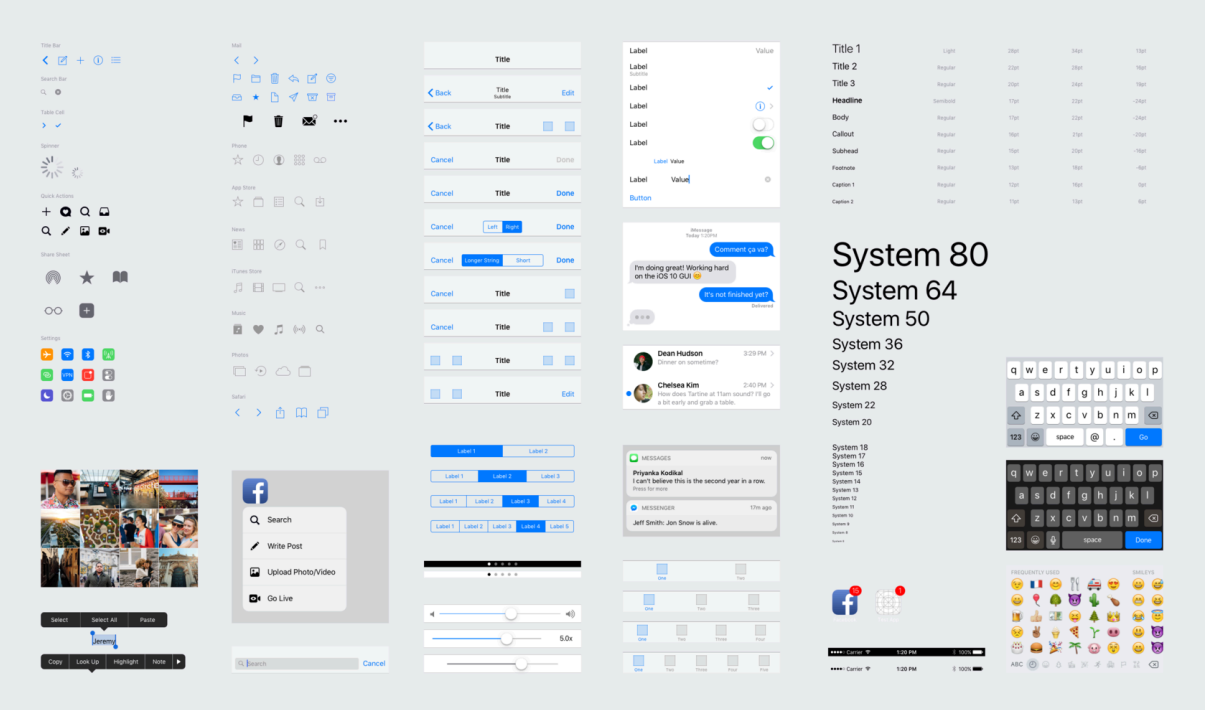

Рис. 2.2. Виділення компонентів сторінки

Важливим елементом роботи UI/UX дизайнера є постійне вивчення та освоєння нових технологій та трендів. Це може включати все, від останніх змін в програмному забезпеченні для дизайну до нових методів дослідження користувачів. Один з сучасних трендів в UI/UX дизайні - це використання штучного інтелекту та машинного навчання для створення більш персоналізованого досвіду користувача. Наприклад, системи можуть аналізувати поведінку користувача, щоб прогнозувати, які функції вони можуть шукати наступними, та відповідно адаптувати інтерфейс. Інший тренд - це використання голосових команд та голосових асистентів, які змінюють спосіб, яким користувачі взаємодіють з продуктами. UI/UX дизайнери

повинні розуміти, як створювати інтерфейси, які враховують цей новий тип взаємодії.

Також важливо зазначити, що хоча UI/UX дизайн часто асоціюється з технологіями, він також має глибокі корені в психології. Розуміння того, як люди сприймають інформацію, як вони приймають рішення та як вони взаємодіють з технологіями, є ключовим для створення продуктів, які вони будуть любити використовувати. Все це робить UI/UX дизайн захоплюючим та багатогранним напрямком, який вимагає не тільки технічних навичок, але й творчості, емпатії та стратегічного мислення.

2.2. Робота з даними

Робота з даними - це широкий термін, який включає в себе різноманітні процеси, методи, техніки і системи, що використовуються для отримання, обробки, аналізу, інтерпретації та візуалізації даних. Ця область стала надзвичайно важливою в сучасному світі, оскільки кількість доступних даних стрімко зростає, а компанії використовують ці дані для прийняття обґрунтованих рішень. Робота з даними може включати в себе різні етапи.

Перший етап - це збір даних, який може включати в себе збір первинних даних (наприклад, через опитування або спостереження) або вторинних даних (наприклад, через використання вже існуючих наборів даних). Після збору даних вони повинні бути очищені і перетворені. Це може включати в себе видалення помилок або неповних даних, перетворення даних на потрібний формат або структуру, і нормалізацію даних для подальшого аналізу.

Наступний етап - це аналіз даних. Це може включати в себе статистичний аналіз, моделювання даних, машинне навчання, або будь-які інші методи, що допомагають зрозуміти тенденції, шаблони або взаємозв'язки в даних.

Останній етап - це візуалізація та комунікація результатів. Це може включати створення графіків, діаграм, інфографік або інших візуальних представлень даних, а також пояснення результатів іншим людям.Робота з даними вимагає знання різних

інструментів і технологій, включаючи мови програмування (наприклад, Python або R), системи управління базами даних (наприклад, SQL), та інструменти для візуалізації даних (наприклад, Tableau).

Важливо пам'ятати, що робота з даними - це не тільки про техніку. Це також про розуміння контексту даних, формулювання правильних питань, і інтерпретація результатів у спосіб, що має сенс для прийняття рішень. Після виконання аналізу й візуалізації даних, наступним кроком є інтерпретація отриманих результатів. Це включає у себе розуміння того, що саме показують дані, та як це стосується поставлених бізнес-цілей або дослідницьких запитань. Інтерпретація може також включати в себе співставлення результатів з іншими даними або дослідженнями, а також визначення можливих причин та наслідків виявлених тенденцій або шаблонів. На цьому етапі може бути корисним скористатися методами статистичного аналізу для перевірки гіпотез або визначення статистичної значущості результатів. Це допомагає забезпечити, що висновки, зроблені на основі даних, є надійними та обґрунтованими.

Після інтерпретації результатів важливо ефективно спілкуватись їх іншим учасникам процесу прийняття рішень. Це може включати в себе підготовку звітів або презентацій, створення інфографіки або дашбордів для візуалізації даних, а також проведення зустрічей або презентацій для обговорення результатів. У цілому, робота з даними - це ітеративний процес, який постійно вдосконалюється. З кожним новим набором даних або з кожним новим питанням, що виникає, можуть з'являтися нові можливості для аналізу та інтерпретації.

Важливо пам'ятати, що хоча технічні навички є важливими для роботи з даними, також важливо мати критичне мислення, аналітичні навички та здатність до розв'язання проблем. Це допомагає забезпечити, що дані використовуються ефективно та етично, а результати аналізу даних використовуються для підтримки обґрунтованих та інформаційних рішень.

2.3. Технології серверного використання

Технології серверного використання є ключовими у розробці сучасних програмних продуктів. Вони дозволяють взаємодіяти з інформацією та об'єднувати користувачів з різних країн та континентів. Особливо зазначимо таке поняття як серверне програмування. Це процес створення та підтримки серверного програмного забезпечення. Тут можна використовувати різноманітні мови програмування, залежно від потреб конкретного проекту. Наприклад, для веб-серверних застосунків часто використовуються PHP, Python, Ruby, Node.js, Java або ASP.NET.

Серверна сторона включає в себе бази даних, які слугують для зберігання, організації та обробки великих обсягів інформації. Найпопулярнішими СУБД для цього є MySQL, PostgreSQL, MongoDB, Oracle, Microsoft SQL Server. Слід зазначити, що в управлінні серверами існують і системи автоматизації. Це DevOps інструменти, такі як Docker, Kubernetes, Jenkins, Ansible, які допомагають автоматизувати розгортання, масштабування, тестування та моніторинг серверного програмного забезпечення.

Також необхідно пам'ятати про безпеку серверів. Застосунок firewalls, систем антивірусного захисту, SSL сертифікатів та інших методів захисту допомагає гарантувати надійність та цілісність даних.

Ще одним актуальним напрямком в сфері серверних технологій є хмарні рішення. Хмарні сервери, такі як Amazon Web Services (AWS), Google Cloud, Microsoft Azure, IBM Cloud, надають можливість зберігати та обробляти дані на віддалених серверах. Вони забезпечують гнучкість, масштабованість та економію коштів, звільняючи компанії від необхідності підтримувати власну серверну інфраструктуру. На сервері використовуються також різноманітні технології віртуалізації. Вони дозволяють імітувати роботу багатьох віртуальних машин на одному фізичному сервері. Такий підхід оптимізує використання ресурсів сервера та підвищує безпеку, оскільки кожна віртуальна машина ізольована від інших.

Системи управління контентом (Content Management Systems, CMS) також є важливою частиною серверних технологій. Вони дозволяють керувати контентом веб-сайтів та інтернет-магазинів без особливих навичок програмування. Деякі приклади CMS включають WordPress, Drupal, Joomla, Magento. Ще одним ключовим аспектом в серверному використанні є API (Application Programming Interface). API дозволяє різним програмам ефективно спілкуватися між собою, обмінюючись даними та функціоналом.Серверні технології використовуються також для створення і підтримки веб-застосунків. Такі технології, як Node.js, дозволяють розробникам писати серверний код на JavaScript, що спрощує процес розробки та підтримки веб-застосунків.

Останнім часом все більшої популярності набуває мікросервісна архітектура, яка передбачає поділ великих застосунків на невеликі, незалежні сервіси. Цей підхід дозволяє легко масштабувати та оновлювати окремі частини застосунку без впливу на інші. Важливу роль у сфері серверних технологій відіграють інструменти для моніторингу та аналізу. Вони допомагають слідкувати за станом сервера, виявляти і усувати проблеми, покращувати продуктивність. Професіонали в області IT часто користуються такими інструментами, як Grafana, Prometheus, Zabbix, Nagios.

Як бачимо, технології серверного використання є важливою частиною сучасного IT-світу. Вони включають велику кількість інструментів, рішень та підходів, що використовуються для створення, розгортання, підтримки та моніторингу серверного програмного забезпечення. Оскільки потреби користувачів та бізнесу продовжують розвиватися, важливо слідкувати за новітніми тенденціями в цій галузі, адаптувати нові технології і навички для постійного покращення сервісу.

2.4. Контроль версій Git

Git використовується в багатьох професійних середовищах для розробки програмного забезпечення, особливо там, де працюють команди програмістів. Ця система контролю версій допомагає зберігати цілісність та стабільність коду.

Окрім зазначених переваг, Git також має ряд інших ключових особливостей. Одна з них - це можливість відкату до попередньої версії коду. Якщо новий код спричиняє проблеми або не працює належним чином, ви можете легко повернутися до попереднього стану. Ви також можете порівнювати версії коду, щоб дивитися, що саме було змінено.

Ще однією корисною особливістю Git є те, що ви можете вести різні гілки роботи паралельно. Наприклад, одна гілка може бути присвячена розробці нової функції, в той час як інша гілка може бути використана для виправлення помилок в поточній версії продукту.

Крім того, Git добре працює з різними платформами та мовами програмування. Він має широкий спектр інструментів та утиліт, які допомагають автоматизувати та спрощувати процеси розробки.

Важливо пам'ятати, що навчитися користуватися Git може бути викликом. Це потужний інструмент з багатьма особливостями та опціями. Але на відповідних онлайн-платформах, таких як GitHub, Bitbucket чи GitLab, ви можете знайти багато навчальних посібників та ресурсів, які допоможуть вам освоїти цю систему.

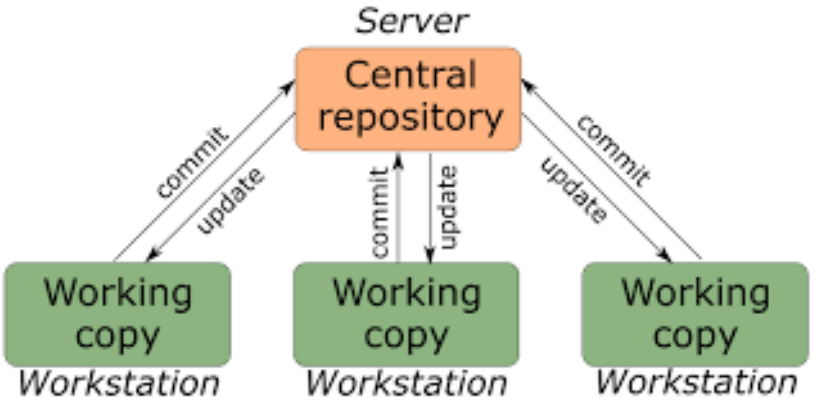

Рис. 2.3. Робота з Git репозиторієм

GitHub - це веб-сервіс, який базується на Git і дозволяє розробникам зберігати свої проекти в облачному середовищі. Він є одним з найпопулярніших сервісів для програмістів і використовується майже в усіх сферах програмування.

Одна з ключових переваг GitHub полягає в тому, що він сприяє співпраці і командній роботі. Розробники можуть створювати спільні репозиторії та працювати над проектами разом. GitHub також дозволяє відстежувати зміни в коді, що робить процес ревізії та аналізу значно простішим.

GitHub також служить головним майданчиком для відкритого програмного забезпечення. Тисячі проектів з відкритим вихідним кодом зберігаються на GitHub, і будь-який розробник може внести свій вклад до цих проектів, додавши нові функції або виправивши помилки.

Сервіс також має власну систему управління завданнями, яка дозволяє користувачам створювати та відстежувати завдання, пов'язані з проектом. Крім того, GitHub надає інструменти для автоматичної побудови та тестування коду, що допомагає забезпечити його якість. GitHub є безкоштовним для проектів з відкритим вихідним кодом, але для приватних репозиторіїв він пропонує платні плани.

2.5. Безпека в веб-додатках

Веб-додатки, такі як сайт календаря виконання завдань, є невід'ємною частиною сучасного числового ландшафту. Однак із збільшенням їх значення виникає гостра необхідність у гарантованій безпеці користувачів та захищеності їх даних.

Перш ніж заглибитися в специфіку безпеки веб-додатків, доречно зазначити, що безпека в веб-додатках - це процес та практика, що має на меті захист веб-додатків від нападників, які шукають вразливості для експлуатації. Існує багато можливих слабких місць веб-додатків, що можуть бути використані для несанкціонованого доступу, порушення даних та інших небажаних дій.

Простір веб-додатків сьогодні є частим місцем для всіляких зловмисних дій, починаючи від підробки ідентифікації до витоку даних. В результаті веб-додатки, такі як сайт календаря виконання завдань, повинні бути захищені на кількох рівнях: на рівні фронтенду (клієнт), на рівні бекенду (сервер) і на рівні бази даних.

По-перше, безпека на рівні клієнта є життєво важливою. Так як код JavaScript може бути видимим для користувача, він стає потенційною мішенню для нападників. Останні можуть використовувати крос-сайтовий скриптинг (XSS) для внесення шкідливих скриптів у веб-сторінки, що переглядаються користувачами.

Важливо зазначити, що поки код бекенду захищається від шкідливих дій, інформація, яка передається між сервером і клієнтом, також повинна бути захищена. Для цього використовують протокол HTTPS і SSL-сертифікати, які гарантують, що інформація передається безпечно.

На рівні баз даних, забезпечення безпеки також критично важливе. Зловмисники можуть використовувати тактики, такі як ін'єкція SQL, щоб отримати доступ до баз даних та викрасти цінну інформацію. Щоб запобігти цьому, важливо використовувати параметризовані запити або процедури, які допомагають уникнути ін'єкцій.

Крім того, автентифікація та авторизація є важливими компонентами безпеки веб-додатків. Вони гарантують, що тільки авторизовані користувачі мають доступ до конфіденційної інформації та можуть виконувати певні дії. Це можна зробити за допомогою сеансів, токенів, паролів або двофакторної аутентифікації.

Веб-сайт календаря виконання завдань, враховує всі вищезгадані елементи безпеки. Ми працюємо над тим, щоб гарантувати, що дані наших користувачів залишаються конфіденційними та безпечними від потенційних зловмисників.

Зрештою, безпека в веб-додатках - це не статичний процес. Вона вимагає постійного моніторингу, актуалізації та вдосконалення відповідно до еволюції загроз та вразливостей. Це вимагає інтеграції безпеки на кожному етапі процесу розробки, а також ставлення до безпеки як до важливого аспекту технологічної стратегії організації.

Деякі з найпоширеніших вразливостей включають:

1. SQL Injection: Це тип атаки, коли зловмисник використовує SQL запити для взяття контролю над базою даних веб-додатка. Це може призвести до витоку конфіденційної інформації або навіть до видалення даних.

2. Cross-Site Scripting (XSS): Вразливість, яка дозволяє зловмиснику вбудовувати скрипти на сторінці веб-додатка, що виконуються у браузері користувача. Це може призвести до крадіжки сесійних файлів, введення шкідливого коду або перенаправлення на інші сайти.

3. Cross-Site Request Forgery (CSRF): Ця атака використовує довіру користувача до сайту для виконання небажаних дій в їхньому обліковому записі. Наприклад, зловмисник може використовувати CSRF для відправлення шкідливих запитів від імені автентифікованого користувача.

4. Недостатня перевірка прав доступу: Ця уразливість виникає, коли веб-додаток не належним чином перевіряє права доступу користувачів до ресурсів. Це може призвести до незаконного доступу до конфіденційної інформації або функціональності.

5. Недостатня захист від переповнення буфера: Ця уразливість виникає, коли зловмисник вводить більше даних у буфер, ніж він може обробити, що може призвести до витоку даних або виконання шкідливого коду.

Щоб запобігти атакам на веб-додатки та захистити їх від вразливостей, розробники та адміністратори повинні вживати різноманітні заходи безпеки. Деякі з найефективніших способів запобігання вразливостей веб-додатків включають:

1. Валідація введених даних: Всі дані, які вводяться користувачами, повинні бути належним чином перевірені та валідовані перед їх обробкою. Це допоможе запобігти SQL ін'єкції та іншим атакам.

2. Застосування параметризованих запитів SQL: Замість складання SQL запитів з введених користувачем даних, слід використовувати параметризовані запити для запобігання SQL ін'єкціям.

3. Використання HTTPS та SSL/TLS: Для забезпечення конфіденційності та цілісності передачі даних між користувачем і сервером, слід використовувати шифрування за допомогою протоколів HTTPS та SSL/TLS.

4. Валідація файлів, завантажуваних користувачами: Перед збереженням або обробкою файлів, завантажених користувачами, слід виконати їх валідацію та перевірку на вміст шкідливого коду.

5. Обмеження доступу до ресурсів: Належне управління правами доступу користувачів до ресурсів веб-додатка допоможе запобігти несанкціонованому доступу.

6. Регулярні аудити безпеки: Регулярні аудити безпеки веб-додатків допоможуть виявляти потенційні уразливості та вчасно вживати заходи для їх усунення.

2.6. Висновки до 2 розділу

У цьому розділі розглянуто основні аспекти технологій розробки веб-додатків, охоплюючи створення UI/UX, роботу з даними, серверне використання, контроль версій за допомогою Git, а також безпеку веб-додатків.

Розробка веб-додатків є комплексним процесом, що вимагає інтеграції різноманітних технологій та практик для створення безпечних, користувач-орієнтованих та ефективно працюючих продуктів. Важливість глибокого розуміння UI/UX не може бути недооцінена, оскільки воно лежить в основі створення задовільного та зручного досвіду для кінцевих користувачів.

Робота з даними, включаючи їх збір, обробку, аналіз та візуалізацію, є ключовою для прийняття обґрунтованих рішень та підвищення вартості продукту. Технології серверного використання, зокрема управління базами даних, автоматизація DevOps та хмарні рішення, сприяють масштабуванню, стабільності та гнучкості веб-додатків.

Контроль версій, особливо за допомогою інструментів, таких як Git, є фундаментальним для координації роботи розробників, ефективного управління змінами та підтримки сталості кодової бази. Безпека веб-додатків займає центральне місце у процесі розробки, оскільки загрози й вразливості постійно еволюціонують.

РОЗДІЛ 3
ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-САЙТУ КАЛЕНДАРЯ

3.1. Аналіз вимог до веб-сайту календаря

Важливим аспектом є процес реєстрації та входу користувачів. Це означає, що користувачі повинні мати змогу створити свій власний обліковий запис, вводячи необхідну інформацію, таку як ім'я користувача, електронна пошта та пароль. Система, в свою чергу, має перевіряти автентичність їх облікових даних при вході.

Календар - це ще одна важлива функція, яку я планую використовувати. Користувачі повинні мати можливість переглядати календар на день, тиждень або місяць. Зрозуміло, що календар має відображати всі заплановані завдання для вибраного періоду.

Користувачі також повинні мати можливість створювати нові завдання, вказуючи назву завдання, дату та час. Важливо, щоб завдання відображалися в календарі на відповідний день і час. Крім того, користувачі повинні мати можливість переносити заплановані завдання з одного дня на інший.

Не менш важливою функцією є редагування та видалення завдань. Наші користувачі могли легко редагувати назву та час завдання або видаляти завдання зі свого облікового запису.

На веб-сайті буде простий та зрозумілий інтерфейс, що дозволяє користувачам легко розуміти, як виконати різні дії, такі як створення нового завдання або перегляд календаря. Він також буде надійним і стабільним, щоб користувачі могли покластися на його доступність, коли вони цього потребують.

Функціональні вимоги:

- Реєстрація та вхід користувачів: Користувачі повинні мати можливість створити власний обліковий запис, ввівши необхідну інформацію, таку як ім'я користувача, електронна пошта та пароль. Система повинна перевіряти автентичність облікових даних користувача при вході;
- Перегляд календаря: Користувачі повинні мати можливість переглядати календар на день, тиждень або місяць. Календар повинен відображати всі заплановані завдання для вибраного періоду;
- Створення задачі: Користувачі повинні мати можливість створювати нові завдання, вказуючи назву завдання, дату та час. Завдання повинні відображатися в календарі на відповідний день і час;
- Перенесення завдань: Користувачі повинні мати можливість переносити заплановані завдання з одного дня на інший. Система повинна автоматично оновлювати відображення календаря після перенесення завдання;
- Редагування та видалення завдань: Користувачі повинні мати можливість редагувати назву та час завдання або видаляти завдання.

Нефункціональні вимоги:

- Простота використання: Інтерфейс веб-сайту має бути простим та зрозумілим для користувачів. Користувачі повинні легко зрозуміти, як виконати різні дії, такі як створення нового завдання або перегляд календаря;
- Безпека: Веб-сайт повинен забезпечувати безпеку данах користувача. Особиста інформація користувача, така як електронна пошта та пароль, повинна бути захищена;
- Надійність: Веб-сайт повинен працювати стабільно і без помилок. Користувачі повинні мати можливість розраховувати на те, що веб-сайт буде доступний, коли вони його потребують;
- Сумісність: Веб-сайт повинен коректно відображатися і працювати в різних браузерах і на різних пристроях.

Вимоги до дизайну та взаємодії:

- Інтуїтивний дизайн: Дизайн веб-сайту повинен бути простим, чистим і інтуїтивно зрозумілим. Всі елементи керування повинні бути легко доступними і зрозумілими для користувачів;
- Адаптивність: Веб-сайт повинен мати адаптивний дизайн, який автоматично налаштовується під різні розміри екрану та орієнтації пристрою;
- Оптимізація для мобільних пристроїв: Веб-сайт повинен бути оптимізований для використання на мобільних пристроях. Всі функції та елементи керування повинні бути легко доступними і використовуватися на сенсорних екранах.

Вимоги до продуктивності:

- Швидкість завантаження: Сторінки веб-сайту повинні завантажуватися швидко, незалежно від швидкості Інтернет-з'єднання користувача;
- Стабільність роботи: Веб-сайт повинен працювати стабільно і без помилок, незалежно від числа користувачів, які використовують його одночасно.

Вимоги до сумісності:

- Сумісність з браузерами: Веб-сайт повинен коректно відображатися і працювати в усіх популярних веб-браузерах;
- Сумісність з операційними системами: Веб-сайт повинен коректно працювати на різних операційних системах.

Вимоги до безпеки:

- Захист даних: Веб-сайт повинен використовувати сучасні методи шифрування для захисту персональних даних користувачів;
- Безпека авторизації: Веб-сайт повинен використовувати надійні методи авторизації користувачів, щоб запобігти несанкціонованому доступу до облікових записів користувачів.

3.2. Проектування структури та дизайну веб-сайту календаря виконання завдань

Структура веб-сайту календаря виконання завдань повинна бути логічною та зручною для користувачів. Вона може включати наступні основні сторінки:

1. Головна сторінка: основна сторінка веб-сайту, яка відображає актуальний календар і список найближчих завдань. З цієї сторінки користувачі можуть перейти до інших розділів сайту.

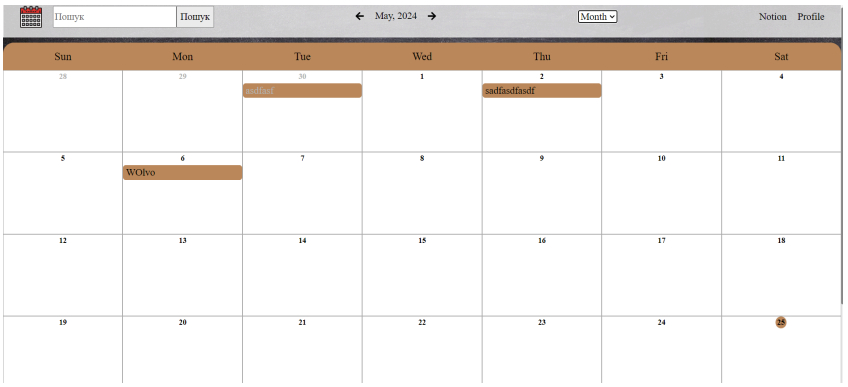

Рис. 3.1. Головна сторінка вебсайту календаря

2. Поле для створення завдання: сторінка, де користувачі можуть створювати нові завдання, вказуючи назву, дату, час та іншу необхідну інформацію.

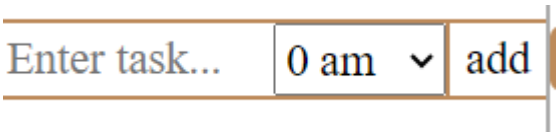

Рис. 3.2. Створення імені завдання та додавання часу

3. Сторінка профілю: сторінка, де користувачі можуть переглядати та редагувати свої особисті дані, змінювати пароль тощо.

Рис. 3.3. Сторінка профлію

4. Сторінка авторизації: сторінка для входу на сайт або реєстрації нового облікового запису.

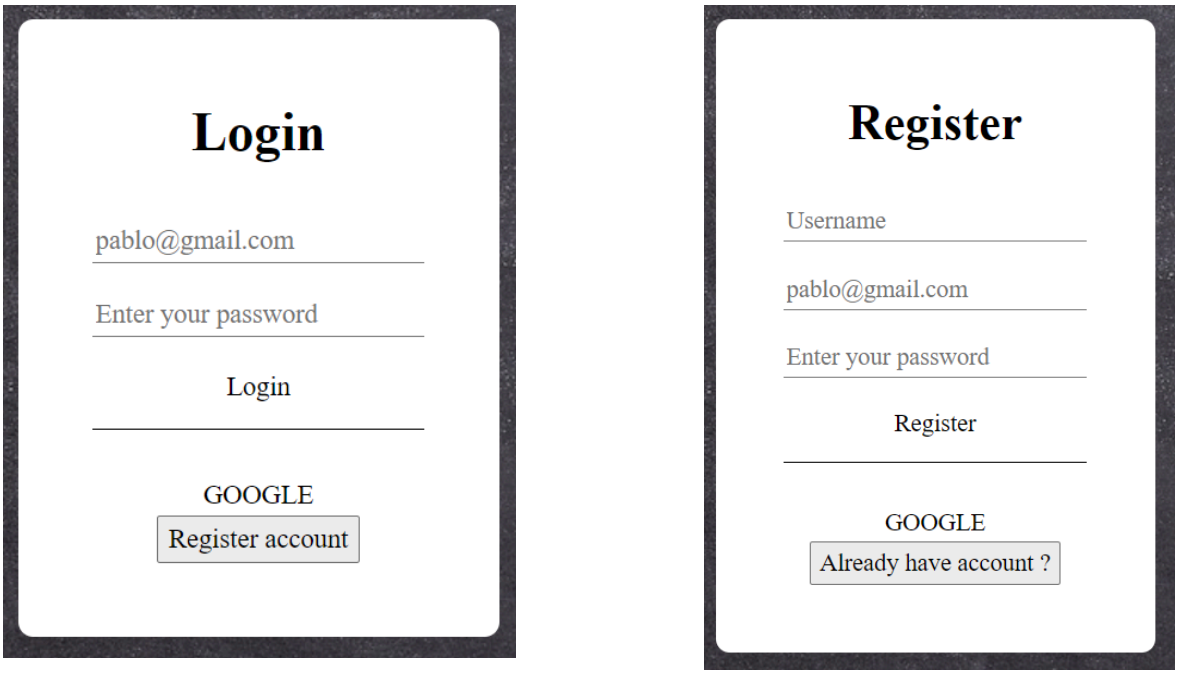

Рис. 3.1. Вигляд сторінки логіну і реєстрації

Щодо дизайну веб-сайту, він повинен бути простим, але елегантним, з легким для очей темним фоном та акцентами білих і жовтогарячих кольорів. Шрифти повинні бути чіткими та легко читабельним. Всі елементи керування повинні бути зручно розташовані та легко доступні. Важливо також забезпечити адаптивний дизайн, щоб сайт коректно відображається та працював на різних пристроях, включаючи настільні комп'ютери, ноутбуки, планшети та смартфони. Окрему увагу слід приділити безпеці: всі особисті дані користувачів повинні бути захищені, а процес авторизації - надійним і безпечним.

Детальніше розглянемо деякі аспекти проектування та розробки веб-сайту календаря виконання завдань:

1. Сценарії використання: перед початком проектування важливо визначити основні сценарії використання веб-сайту. Це можуть бути, наприклад, "Користувач хоче переглянути свій календар на наступний тиждень", "Користувач хоче створити нове завдання", "Користувач хоче перемістити завдання на інший день" тощо.

2. Діаграми потоку даних: для кожного сценарію використання можна створити диаграму потоку даних, яка показує, як користувач взаємодіє з веб-сайтом та як дані передаються між різними елементами системи.

3. Прототипування: на основі сценаріїв використання та діаграм потоку даних можна створити прототип веб-сайту, який демонструє основні елементи інтерфейсу та їх взаємодію. Прототип може бути створений за допомогою спеціалізованих інструментів, таких як Sketch, Figma або Adobe XD.

4. Тестування: прототип веб-сайту повинен бути протестований на різних користувачах, щоб отримати зворотний зв'язок та виявити можливі проблеми з використанням. Результати тестування потім можна використати для удосконалення дизайну та функціональності веб-сайту.

5. Розробка: після завершення проектування починається процес розробки веб-сайту. Він включає написання коду, інтеграцію з базою даних, налаштування сервера тощо.

6. Тестування та випуск: після завершення розробки веб-сайт повинен бути знову протестований, щоб переконатися, що всі функції працюють коректно. Після

успішного тестування веб-сайт може бути випущений і стати доступним для користувачів.

7. Підтримка та оновлення: після випуску веб-сайту необхідно забезпечити його підтримку, виправляти помилки, реагувати на зворотний зв'язок користувачів та регулярно випускати оновлення, щоб покращувати функціональність та безпеку веб-сайту.

3.3. Використання React для розробки функціоналу веб-сайту

React є одним з найпопулярніших JavaScript фреймворків для розробки інтерактивних веб-додатків, який використовується багатьма великими компаніями, такими як Facebook та Instagram. Він пропонує ряд переваг для розробки веб-сайту календаря виконання завдань.

Компонентний підхід: React дозволяє розбивати інтерфейс на незалежні, повторно використовувані компоненти, що спрощує процес розробки та підтримки коду. Наприклад, можна створити окремі компоненти для календаря, списку завдань, форми створення завдання тощо.

Virtual DOM: React використовує віртуальний DOM для оптимізації рендеринга, що підвищує продуктивність веб-сайту, особливо при частому оновленні інтерфейсу, наприклад, при додаванні або переміщенні завдань.

JSX: React використовує синтаксис JSX, який дозволяє писати HTML-код безпосередньо в JavaScript. Це робить код більш читабельним та зрозумілим.

Стан компонентів: React дозволяє керувати станом компонентів, що є корисним для зберігання даним, що змінюються, наприклад, списку завдань.

Екосистема: React має велику екосистему з численними бібліотеками та інструментами, які можна використовувати для розширення функціональності веб-сайту. Наприклад, можна використовувати React Router для реалізації навігації між сторінками або Redux для керування глобальним станом додатку.

Підтримка спільноти: React має велику та активну спільноту розробників, що означає, що знайти відповіді на питання або рішення проблем можна швидко та легко.

Я використав React у своєму веб-сайті для розробки календаря завдань. Для цього я створив декілька компонентів:

1. Компонент календаря:
    - Цей компонент відповідає за відображення календаря з днями та можливістю переходу між місяцями. Він може містити такі елементи, як назва місяця, кнопки для переходу до попереднього або наступного місяця, список днів та інші елементи.
    - Компонент отримує дані про завдання на кожний день (наприклад, кількість завдань, їх статус) та відображає їх у відповідних клітинах календаря. Це може бути представлено, наприклад, за допомогою кольорових маркерів або піктограм.
2. Компонент списку завдань:
    - Цей компонент відповідає за відображення списку завдань на обраний день. Він може містити назви завдань, їх опис, дедлайни, пріоритети та інші деталі.
    - Компонент отримує дані про завдання на обраний день та відображає їх у вигляді списку, що дозволяє користувачам швидко переглянути всі свої завдання на даний день.
3. Компонент форми додавання завдання:
    - Цей компонент дозволяє користувачам додавати нові завдання до календаря. Він може містити поля для введення назви завдання, опису, дати дедлайну, пріоритету тощо.
    - Після заповнення полів користувачем та підтвердження додавання, дані про нові завдання передаються до компонента календаря для відображення.

4. Компонент завдання:

- Цей компонент відповідає за відображення окремого завдання. Він може містити інформацію про завдання, таку як назва, опис, дедлайн, пріоритет, статус тощо.
- Компонент завдання може також містити кнопки для виконання чи видалення завдання, що дозволяє користувачам легко управляти своїми завданням.

Використання стейту та пропсів React дозволяє ефективно керувати даними та їх відображенням у компонентах. Бібліотека React Router допомагає забезпечити зручну навігацію між різними частинами календаря, що робить користування веб-сайтом більш зручним для користувачів.

3.4. Реалізація системи управління задачами на веб-сайті

Реалізація системи управління задачами на веб-сайті - це важливий етап розробки, який може бути оптимізований для покращення продуктивності та зручності користування. Давайте розглянемо, які технології можна використати для цього та як їх можна покращити:

1. Використання React: React є потужною бібліотекою для створення інтерактивних веб-сайтів. Використання React дозволяє створити швидкий та ефективний інтерфейс для управління завданнями. Для покращення можна використати бібліотеки стану, такі як Redux або Context API, для кращого управління станом додавання, редагування та видалення завдань.

2. Використання REST API: для зберігання та обміну даними між клієнтом та сервером можна використовувати REST API. Це дозволить ефективно обмінюватись даними про завдання та забезпечить консистентність даних між різними пристроями та сеансами користувачів.

3. Використання баз даних: для зберігання даних про завдання можна використовувати бази даних, такі як MySQL або MongoDB. Вони дозволяють

зберігати, оновлювати та видаляти дані про завдання, а також забезпечують швидкий доступ до них.

4. Аутентифікація та авторизація: для забезпечення безпеки та конфіденційності даних користувачів можна використовувати механізми аутентифікації та авторизації, такі як JWT або OAuth. Це дозволить захистити дані користувачів та забезпечити доступ тільки до необхідних функцій системи управління завданнями.

5. Мобільна адаптація: для зручності користувачів, що використовують мобільні пристрої, важливо забезпечити адаптивний дизайн веб-сайту. Використання CSS фреймворків, таких як Bootstrap або Material UI, допоможе підтримати оптимальний вигляд та функціональність на різних пристроях.

6. Тестування та оптимізація: після реалізації системи управління завданнями важливо провести тестування для перевірки функціональності та продуктивності. Також варто оптимізувати код для покращення швидкодії та ефективності роботи веб-сайту.

7. Реалізація функціоналу сповіщень: додавання функціоналу сповіщень дозволить користувачам отримувати повідомлення про нагадування про наближення дедлайнів або важливих подій. Для цього можна використати бібліотеки, такі як Firebase Cloud Messaging або Pusher, які дозволяють надсилати сповіщення на різні пристрої.

8. Використання Drag and Drop: додавання функціоналу перетягування та розміщення завдань за допомогою Drag and Drop може значно полегшити користування системою управління завданнями. Для цього можна використати бібліотеки, наприклад React Beautiful DND, які дозволяють впровадити цю функціональність з мінімальними зусиллями.

9. Аналітика та статистика: додавання функціоналу аналітики та статистики дозволить користувачам отримувати звіти про їхню продуктивність та ефективність управління завданнями. Можна використати бібліотеки, такі як Google Analytics або Mixpanel, для збору та аналізу даних про користувачів.

Загалом, поєднання сучасних технологій та функціональності допоможе покращити систему управління завданнями на веб-сайті, забезпечуючи користувачам зручний та ефективний інструмент для планування та виконання їхніх завдань.

3.5. Бази даних та структури зберігання даних

Оскільки моя робота - це розробка календаря для завдань, то ключовим аспектом є ефективне зберігання та управління даними. Використовуючи Jestify, я маю можливість працювати з моєю базою даних MongoDB в режимі реального часу, що дозволяє мені забезпечити надійність та ефективність мого додатка.

Jestify - це легка, гнучка та зручна використанні API платформа для розробки. Jestify надає повний набір RESTful API методів для роботи з даними в режимі реального часу, що дозволяє розробникам створювати повноцінні веб-додатки без необхідності власного сервера.

MongoDB - це високопродуктивна, відкрита, безсхемна база даних NoSQL, яка призначена для роботи з документами JSON-подібного формату. MongoDB працює по принципу "ключ-значення", дозволяє проводити пошук за декількома полями, розподіляти дані між різними серверами та ін.

Структура зберігання даних в MongoDB – це колекції документів. Вони подібні до таблиць у реляційних базах даних, але без жорсткого схематичного обмеження. Кожен документ відповідає одній записі або ряду і включає в себе одне або декілька полів з даними.Оскільки MongoDB це безсхемна база даних, мені не потрібно визначати структуру таблиці до того, як вставити документи. Я можу динамічно модифікувати схему без переривання служби. Використовуючи MongoDB, я можу легко масштабувати свій проект, оскільки MongoDB підтримує горизонтальне масштабування через розподіл даних по кластерах. Це означає, що при зростанні обсягу даних, я можу просто додати додаткові сервери в кластер, і MongoDB автоматично розподілить дані між ними. Такий підхід дозволяє витримувати великі обсяги даних і забезпечує високу доступність.

Jestify, в свою чергу, дозволяє мені працювати з базою даних MongoDB через RESTful API. Це означає, що я можу відправляти HTTP запити до сервера Jestify, і він буде виконувати потрібні операції з базою даних: отримання, вставка, оновлення та видалення даних. Такий підхід значно спрощує розробку, оскільки мені не потрібно прямо взаємодіяти з базою даних і писати SQL запити. Також Jestify надає автоматичну підтримку WebSocket, що дозволяє реалізувати функціонал реального часу в моєму додатку. Наприклад, коли користувач додає нове завдання в календар, інші користувачі, які в цей момент переглядають календар, можуть бачити це завдання відразу після його додавання, без потреби оновлювати сторінку.

Розробка календаря для завдань вимагає розуміння того, як ефективно зберігати, управляти та представляти дані. З цією метою, я використовую MongoDB для зберігання даних, Jestify для сервера та API, а також різноманітні JavaScript бібліотеки для фронтенд частини.

Один з головних принципів мого додатка – це простота та зручність використання. Щоб досягти цього, я працюю над тим, щоб користувачі могли легко і швидко додавати, редагувати та видаляти завдання. Також я розробляю систему сповіщень, яка буде нагадувати користувачам про майбутні завдання.

3.6. Висновки до 3 розділу

Проект розробки веб-сайту календаря для завдань включав детальний аналіз вимог до функціональності та нефункціональної характеристики, що склало основу для створення використовуваного та ефективного сервісу. Забезпечення інтуїтивно зрозумілого дизайну, високої безпеки, надійності та сумісності з різноманітними платформами було визначено як ключові напрями роботи. Реалізація веб-сайту базувалася на проектуванні структури та інтерфейсу, що передбачала логічне розташування сторінок і компонентів керування.

Як основа для фронтенду був обраний React - сучасний JavaScript фреймворк, який дозволив побудувати швидкий та реактивний інтерфейс з компонентним підходом до розробки. Система управління задачами була зосереджена на створенні

таких можливостей, як додавання, редагування, перенесення та видалення завдань, інтегровані з базою даних MongoDB через Jestify для ефективної взаємодії в режимі реального часу та забезпечення зручності користувачів. Продуманість архітектури, акцент на мобільній адаптованості та інтеграція інструментів для управління авторизацією та безпекою відіграли важливу роль. Також були враховані такі напрями, як масштабування продукту, розробка сповіщень для нагадування про завдання, додавання функції Drag and Drop та впровадження аналітики для підвищення продуктивності та управління завданнями.

У цілому, проект розробки веб-сайту виявився комплексним зусиллям, яке спрямовувалося на створення зручного у використанні, продуктивного та безпечного інструменту для планування та організації особистих або командних завдань. Вдалий вибір технологій та платформ забезпечив можливість подальшого розвитку та оптимізації веб-сайту.

ВИСНОВКИ

Розробка вебсайту календаря завдань на базі React виявилася ефективним рішенням, оскільки дана технологія забезпечує гнучкість, продуктивність та легкість у масштабуванні проекту. Використання Styled-components дозволило створити консистентний та адаптивний дизайн, який може бути легко перевикористаний. Авторизація користувачів, управління завданнями, та відображення інформації по днях успішно інтегрувалися, в ході чого були повністю використані переваги React.

Що стосується подальших напрямів розвитку, планується розширення функціоналу, зокрема імплементація планувальника подій, системи налаштування сповіщень, та можливість синхронізації з іншими сервісами. Такі додаткові функції не лише поліпшують користувацький досвід, а й зробили цей інструмент ще більш затребуваним серед широкого кола користувачів, допоможуть їм заощаджувати час та підвищувати їхню продуктивність.

Науково-дослідницька робота підкреслила значимість правильного вибору технологічного стеку та підходів до розробки для створення надійних та користувацьких привабливих вебсайтів. Доведено ефективність застосування React у високорівневих проектах, що забезпечують необхідність і постійному оновленні та вдосконаленні продукту у відповідь на змінні потреби користувачів, демонструючи значущість застосування сучасних веб-технологій у створенні ефективних інструментів для особистої продуктивності.

Завдяки використанню React та JSX, розробка інтерфейсу веб-сайту стала більш інтуїтивно зрозумілою та зручною для програмістів. Компонентний підхід є ключовим фактором, який сприяє швидким ітераціям розробки та легкості впровадження нових функцій. Подальше внесення оптимізацій та поліпшень буде спрямовано на забезпечення високого рівня доступності та користувацького досвіду на всіх типах пристроїв, включаючи мобільні.

Бібліотека Styled-components стала вдалим вибором для імплементації модульного та адаптивного CSS, що в свою чергу допомогло уникнути проблем із каскадуванням стилів та зробило масштабування дизайну простішим процесом.

Використання цієї бібліотеки гарантує консистентність стилів та полегшує їх перевикористання у різних частинах проекту.

З огляду на все вищевикладене, проект демонструє не тільки технічні успіхи у веб розробці, а й глибоке розуміння потреб кінцевого користувача. Надалі важливим кроком стане залучення користувачів до процесу тестування дя отримання якісного зворовного зв'язку, що дозволить виявити недоліки та вдосконалити існуччий функціонал. Також планування регулярного оновлення функціональності, забезпечення високого рівня системної інтеграції та впровадження аналітики користувацької поведінки дасть можливість постійно підтримувати актуальність продукту на динамічному ринку цифрових технологій.

Загалом, проект веб-сайту календаря завдань на базі React є прикладом продуманого підходу до розробки Складного додатку односторінкового застосунку (SPA), який ефективно сочетает в собі сучасні веб-стандарти, забезпечує відмінну продуктивність та пропонує значний потенціал для подальшого розширення й поглиблення взаємодії з користувачами.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Павленко І. В. HTML і CSS. Практичний курс : навч. посіб. / І. В. Павленко, В. М. Черненко, Є. Ю. Старостіна. – Київ : Професійна література, 2017. – 256 с. URL: http://elib.bsu.by/handle/123456789/154674 (дата звернення: 16.05.2024).
2. Кравець О. В. Основи роботи з Git: навч. посіб. / О. В. Кравець. – Львів : ЛНУ ім. Івана Франка, 2020. – 128 с. URL: http://eprints.zu.edu.ua/30733/ (дата звернення: 20.05.2024).
3. Шевченко А. І. MongoDB: підручник для вищих навч. закладів / А. І. Шевченко. – Київ : Академвидав, 2018. – 320 с.
4. Скляренко Г. В. SEO: оптимізація і просування сайтів / Г. В. Скляренко, О. В. Мельник. – Київ : Комп'ютерна література, 2016. – 512 с.
5. Лисенко В. В. TypeScript: програмування для веб-розробки / В. В. Лисенко, О. М. Петренко. – Київ : Професійна література, 2020. – 456 с.
6. Семенюк С. І. React.js: практичний курс / С. І. Семенюк. – Львів : Нова література, 2019. – 320 с.
7. Богданов С. В. Використання CSS3 в веб-дизайні: навч. посіб. / С. В. Богданов. – Одеса : ОНУ ім. Мечникова, 2018. – 128 с.
8. Головатий О. О. Git для початківців: навч. посіб. / О. О. Головатий. – Львів : ЛНУ ім. Івана Франка, 2020. – 128 с.
9. Кравченко О. В. MongoDB для веб-розробників / О. В. Кравченко. – Київ : Академвидав, 2019. – 320 с.
10. Скляренко Г. В. SEO: практичні рекомендації / Г. В. Скляренко, О. В. Мельник. – Київ : Комп'ютерна література, 2017. – 512 с.
11. Полякова Н.О. JavaScript ES6: нововведення / Н. О. Полякова. – Дніпро : Видавництво "Політехніка", 2020. – 256 с.
12. Бондаренко В.І. HTML та CSS: створення веб-сайтів / В. І. Бондаренко. – Харків : Видавництво "Фактор", 2018. – 350 с.

13. Захарчук М.О. React Native: розробка мобільних додатків / М. О. Захарчук. – Київ : Видавництво "Професійна література", 2021. – 400 с.
14. Литвиненко В.О. TypeScript для веб-розробки: підручник / В. О. Литвиненко. – Одеса : Одеський національний університет, 2021. – 450 с.
15. Олексієнко І.В. Git: керування версіями проектів / І. В. Олексієнко. – Київ : Видавництво "Комп'ютерна література", 2019. – 512 с.
16. Кравець О.В. MongoDB: використання в веб-розробці / О. В. Кравець. – Львів : ЛНУ імені Івана Франка, 2020. – 128 с.
17. Скляренко Г.В., Мельник О.В. SEO: аналіз веб-сайтів / Г. В. Скляренко, О. В. Мельник. – Київ : Видавництво "Комп'ютерна література", 2018. – 512 с.
18. Полякова Н.О. JavaScript ES7: нововведення / Н. О. Полякова. – Дніпро : Видавництво "Політехніка", 2021. – 256 с.

ДОДАТКИ

Додаток А

```
@injectable()
export class ServerLoader implements ApplicationLoader {
    public initialize = async () => {
        const app = express();
        this.loadUtilities(app);
        this.routesInit(app);
        this.listen(app);
        return app;
    };

    private loadUtilities = (application: Express) => {
        application.use(cors());
        application.use(express.json());
    };

    private routesInit = (server: Express) => {
        initializationRoutes.forEach((route) => route.initialization(server));
    };

    private listen = (server: Express) => {
        server.listen(process.env.PORT || 3999, () => console.log(`Server start
on port: ${process.env.PORT || 3999}`));
    };
}
```

Створення серверу, прослуховування на відповідному порту та підключення маршрутів

```
export const useDropHandle = () => {
    const { receiveTasks } = useTaskContex();

    return (e: React.DragEvent<HTMLDivElement>, day: DayParam, id: string,
hours?: string) => {
        e.preventDefault();

        const dataString = e.dataTransfer.getData('application/json');
        const parsedData = JSON.parse(dataString);
        if (hours !== undefined) {
            TasksProvider.myInstance
                .updateTaskDate(parsedData.id, { date:
`${day.currentDay}.${day.currentMonth}.${day.currentYear}`, hours: hours })
                .then(() => receiveTasks(id));

            (e.target as HTMLDivElement).style.boxShadow = 'none';
        } else {
            TasksProvider.myInstance
                .updateTaskDate(parsedData.id, { date:
`${day.currentDay}.${day.currentMonth}.${day.currentYear}` })
                .then(() => receiveTasks(id));

            (e.target as HTMLDivElement).style.boxShadow = 'none';
        }
    };
};
export const useDropHandleTask = () => {
    const { receiveTasks } = useTaskContex();

    return (e: React.DragEvent<HTMLParagraphElement>, day: DayParam, id: string,
task: TasksDTO, hour?: string) => {
        e.preventDefault();
        const dataString = e.dataTransfer.getData('application/json');
        const data = JSON.parse(dataString);
        if (data.id != task._id && hour) {
            TasksProvider.myInstance
                .updateTaskDate(data.id, { date:
`${day.currentDay}.${day.currentMonth}.${day.currentYear}`, hour: hour })
```

```
.then(() => receiveTasks(id));
        } else if (data.id != task._id) {
            TasksProvider.myInstance
                .updateTaskDate(data.id, { date:

`${day.currentDay}.${day.currentMonth}.${day.currentYear}` })
                .then(() => receiveTasks(id));
        }

        (e.target as HTMLParagraphElement).style.boxShadow = 'none';
    };
};

export const dragOverHandle = (e: React.DragEvent<HTMLDivElement>) => {
    e.preventDefault();
    if ((e.target as HTMLDivElement).className === 'calendar-task') {
        (e.target as HTMLDivElement).style.boxShadow = '0px 4px 5px gray';
    }
};

export const dragLeaveHandle = (e: React.DragEvent<HTMLParagraphElement>) => {
    (e.target as HTMLParagraphElement).style.boxShadow = 'none';
};
export const dragEndHandle = (e: React.DragEvent<HTMLParagraphElement>) => {
    (e.target as HTMLParagraphElement).style.boxShadow = 'none';
};

export const dragStartHandle = (e: React.DragEvent<HTMLParagraphElement>, task:
TasksDTO) => {
    const data = JSON.stringify({ id: task._id, date: task.date });
    e.dataTransfer.setData('application/json', data);
};
```

Створення Drag&Drop та його взаємодія з базою даних