Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Використання Google Apps Script для розробки веб-застосунку календаря подій з інтеграцією Google Calendar

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Заказнюк Б. А.
(підпис) (прізвище та ініціали)

Керівник _____ Ясній О. П.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г. В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Заказнюк Богдан Анатолійович
(прізвище, ім'я, по батькові)

1. Тема роботи Використання Google Apps Script для розробки веб-застосунку календаря подій з інтеграцією Google Calendar

Керівник роботи Ясній Олег Петрович, доктор технічних наук, професор кафедри МН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*29*» *квітня* 2024 року № *4/7-470*

2. Термін подання студентом завершеної роботи 24 червня 2024р.

3. Вихідні дані до роботи Літературні та інтернет джерела про розробку веб-застосунків з використанням Google Apps Script

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. Розділ 1. Аналіз предметної області та постановка завдання. Огляд веб-застосунків та засобів їх розробки. 1.1 Поняття веб-застосунку. 1.2 Аналіз засобів для створення веб-застосунків. 1.3 Аналіз існуючих інструментів для управління подіями. 1.4 Висновок до першого розділу. Розділ 2. Проектна частина. Огляд Google Apps Script та його функціоналу. 2.1 Вибір технології для реалізації веб-застосунку. 2.2 Огляд Google Apps Script та його функцій. 2.3 Опис розробки веб-застосунків з використанням Google Apps Script. 2.4 Висновок до другого розділу. Розділ 3. Практична частина. Створення веб-застосунку календаря подій з використанням Google Apps Script. 3.1 Створення таблиць та календаря. 3.2 Створення функціоналу веб-застосунку. 3.3 Створення інтерфейсу користувача. 3.4 Висновок до третього розділу. Розділ 4. Безпека життєдіяльності, основи охорони праці. 4.1 Актуальність безпеки життєдіяльності людини. 4.2 Загальні вимоги безпеки з охорони праці для користувачів ПК. 4.3 Висновок до четвертого розділу. Висновки. Перелік джерел. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Слайди: Тема 1. Мета та завдання 2. Актуальність теми 3. Інструменти для керування подіями 4. Інструменти для розробки веб-застосунків 5. Google Apps Script та його функції 6, 7. Створення таблиці та календаря 8. Функціонал веб-застосунку 9. Створення та оновлення подій 10. Сторінка інтерфейсу 11. Сторінка редагування подій 12. Висновки 13.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В. С. к.т.н., доц.		

7. Дата видачі завдання 29 січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	30.01.2024	
2.	Підбір джерел про Google Apps Script та для розробки веб-застосунку	31.01.2024-03.02.2024	
3.	Опрацювання джерел по темі кваліфікаційної роботи	04.02.2024-06.02.2024	
4.	Виконання дослідження щодо використання Google	07.02.2024-11.02.2024	
	Apps Script для розробки веб-застосунку календаря		
	події з інтеграцією Google Calendar		
	Розроблення веб-застосунку календаря подій		
5.	Оформлення розділу «Аналіз предметної області та	03.06.2024-05.06.2024	
	постановка завдання. Огляд веб-застосунків та засобів		
	їх розробки»		
6.	Оформлення розділу «Проектна частина. Огляд	06.06.2024-08.06.2024	
	Google Apps Script та його функціоналу»		
6.	Оформлення розділу «Практична частина. Створення	09.06.2024-11.06.2024	
	веб-застосунку календаря подій з використанням		
	Google Apps Script»		
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	12.06.2024-13.06.2024	
8.	Виконання завдання до підрозділу «Основи охорони	14.06.2024-15.06.2024	
	праці»		
9.	Оформлення кваліфікаційної роботи	16.06.2024-17.06.2024	
10.	Нормоконтроль	18.06.2024-19.06.2024	
11.	Перевірка на плагіат	20.06.2024	
12.	Попередній захист кваліфікаційної роботи	21.06.2024	
13.	Захист кваліфікаційної роботи	25.06.2024	

Студент _____ Заказнюк Б. А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Ясній О. П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Використання Google Apps Script для розробки веб-застосунку календаря подій з інтеграцією Google Calendar // Кваліфікаційна робота освітнього рівня «Бакалавр» // Заказнюк Богдан Анатолійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2024 // С. – 66, рис. – 24, табл. – 0, слайд. – 13, додат. – 4, бібліогр. – 39.

Ключові слова: веб-застосунок, розробка, події, скрипт, google, javascript, html.

Кваліфікаційна робота присвячена дослідженню процесу розробки веб-застосунку календаря подій з використанням Google Apps Script.

В першому розділі кваліфікаційної роботи описано поняття веб-застосунку, висвітлено основні аспекти його функціонування та значення в сучасному цифровому середовищі. Розглянуто інструменти для управління подіями.

В другому розділі кваліфікаційної роботи проведено огляд технології Google Apps Script та його функцій. Досліджено можливості цієї технології для розробки веб-застосунків. Подано опис процесу розробки, включаючи створення скриптів, налаштування тригерів та інтеграцію з Google Calendar і Google Sheets.

В третьому розділі кваліфікаційної роботи описано практичну частину створення веб-застосунку календаря подій. Проаналізовано процес створення таблиць та календаря. Проведено розробку функціоналу веб-застосунку та створення користувацького інтерфейсу.

Об'єкт дослідження: процес розробки веб-застосунків для управління подіями.

Предмет дослідження: технології та інструменти для створення веб-застосунків, зокрема Google Apps Script, Google Sheets та Google Calendar.

ANNOTATION

The use of Google Apps Script for the development of a web application of the calendar of events with the integration of Google Calendar // Qualification work of the educational level «Bachelor» // Zakazniuk Bohdan // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-41 // Ternopil, 2024 // P. 66, fig. – 24, tabl. – 0, draw. – 13, annexes. – 4, references – 39.

Keywords: web application, development, events, script, google, javascript, html.

The qualification work is dedicated to researching the process of developing an event calendar web application using Google Apps Script.

In the first chapter of the qualification work, the concept of a web application is described, highlighting the main aspects of its functioning and significance in the modern digital environment. Tools for event management are reviewed.

In the second chapter of the qualification work, an overview of Google Apps Script and its features is conducted. The capabilities of this technology for developing web applications are explored. A description of the development process is provided, including the creation of scripts, setting up triggers, and integrating with Google Calendar and Google Sheets.

In the third chapter of the qualification work, the practical part of creating an event calendar web application is described. The process of creating tables and a calendar is analyzed. The development of the web application's functionality and the creation of the user interface are carried out.

Object of research: the process of developing web applications for event management.

Subject of research: technologies and tools for creating web applications, specifically Google Apps Script, Google Sheets, and Google Calendar.

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

API (англ. Application Programming Interface) – Інтерфейс програмування застосунків.

CSS (англ. Cascading Style Sheets) – Каскадні таблиці стилів.

DOM (англ. Document Object Model) – Об'єктна модель документа.

HTML (англ. HyperText Markup Language) – Мова розмітки гіпертексту.

ID (англ. Identifier) – Ідентифікатор.

PHP (англ. Hypertext Preprocessor) – Гіпертекстовий препроцесор.

SPA (англ. Single Page Application) – Односторінковий застосунок.

URL (англ. Uniform Resource Locator) – Універсальний вказівник ресурсу.

ПК – Персональний комп'ютер.

ЗМІСТ

ВСТУП

У сучасному світі, де швидкість та ефективність стали ключовими складовими успішного функціонування багатьох сфер діяльності, використання веб-застосунків для автоматизації та оптимізації робочих процесів стає необхідністю. Зокрема, управління часом та планування подій вимагають інструментів, які є зручними, функціональними та легко налаштовуваними відповідно до індивідуальних потреб користувачів.

У даному контексті використання Google Apps Script, як інструменту для розробки веб-застосунків, набуває все більшого значення. Google Apps Script – це потужний інструмент, який дозволяє створювати власні додатки та автоматизовані рішення, використовуючи інфраструктуру Google.

Одним із найбільш популярних та корисних сервісів в рамках Google Workspace є Google Calendar. Він став не тільки зручним інструментом для планування подій, але й потужним засобом для організації часу та спільної роботи. Однак, навіть з усією його функціональністю, іноді виникає потреба у створенні персоналізованих рішень, які відповідали б конкретним потребам користувача.

Актуальність обраної теми виявляється в її потенційній здатності вирішувати потрібні проблеми планування часу та управління подіями, які є важливими як для особистого, так і для професійного життя. Розроблений веб-застосунок буде сприяти підвищенню продуктивності та ефективності користувачів, допомагаючи їм зосередитися на найважливіших завданнях та досягати поставлених цілей.

Об’єкт дослідження – Google Apps Script, зокрема його можливості для розробки веб-застосунків та автоматизації робочих процесів. Дослідження вивчає, як Google Apps Script можна використовувати для створення веб-застосунків, аналізуючи його можливості, переваги та недоліки. Метою є виявлення того, наскільки добре Google Apps Script інтегрується з іншими сервісами Google, та розробка веб-застосунків, що відповідають конкретним потребам користувачів.

Метою цієї дипломної роботи є дослідження можливостей використання Google Apps Script для розробки персоналізованого веб-застосунку для управління календарем подій з інтеграцією з Google Calendar. На сьогоднішній день Google Calendar є одним із найпопулярніших інструментів для планування подій та спільної роботи над ними, тому інтеграція з ним надасть розробленому додатку значного практичного значення.

Завдання дослідження:

1. Провести аналіз інструментів розробки веб-застосунків та інструментів для управління подіями
2. Дослідження функціональних можливостей Google Apps Script, включаючи його інтеграцію з іншими сервісами Google та створення веб-застосунків.
3. Створення веб-застосунку за допомогою Google Apps Script, який буде забезпечувати можливість додавання, редагування та видалення подій та завдань через веб-інтерфейс.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ. ОГЛЯД ВЕБ-ЗАСТОСУНКІВ ТА ЗАСОБІВ ЇХ РОЗРОБКИ

1.1 Поняття веб-застосунку

Веб-застосунок є невід'ємною частиною сучасного інтернет-середовища та веб-технологій. За своєю суттю, веб-застосунок є програмним забезпеченням, яке виконується на сервері та забезпечує інтерактивний та функціональний інтерфейс для користувача через веб-браузер. Основна відмінність веб-застосунків від традиційних десктопних програм полягає у тому, що вони доступні через Інтернет та не потребують спеціального встановлення на пристрої користувача.

Веб-застосунки можуть мати різні призначення та функціональні можливості, включаючи управління даними, спільну роботу, комунікацію, розваги та багато іншого. Вони можуть бути реалізовані у вигляді різноманітних сервісів та інструментів, таких як електронна пошта, соціальні мережі, онлайн-редактори, системи управління відносинами з клієнтами (CRM) та інші.

Важливим аспектом веб-застосунків є їхня доступність та легкість використання, оскільки вони можуть бути запущені на будь-якому пристрої з доступом до Інтернету та сумісному з веб-браузером. Крім того, веб-застосунки часто надають можливість роботи у спільному режимі та синхронізації даних між різними пристроями, що забезпечує зручність та продуктивність користувача.

Поняття веб-застосунку веде до розуміння відмінностей між ним та веб-сайтом. Основна відмінність полягає в їхній функціональності та призначенні.

З точки зору користувача, різниця між веб-сайтами та веб-застосунками – це функціональність. Веб-сайти призначені для інформування, а веб-застосунки для допомоги. Вміст на веб-сайті можна переглядати, читати чи слухати, але користувач не може ним маніпулювати. І навпаки, вміст у веб-застосунках не тільки можна переглянути, але містить інтерактивні елементи. Веб-застосунки дозволяють користувачам маніпулювати даними [1].

Веб-застосунки є складнішими, в порівнянні з веб-сайтами. Веб-сайти – це тільки колекція статичних веб-сторінок. Нинішні стандарти розробки веб-сайтів зробили їх більш інтерактивними, але вони все ще відносно прості у створенні порівняно з веб-застосунками. Веб-застосунки не тільки мають надавати послугу та функціонувати безперервно, але вони потребують бекенд-сервісів, різного рівня користувачів та можливостей обробки даних. Додатково, захист веб-додатків потребує більш вдосконалених вирішень [1].

Загальний доступ є відмінною ознакою веб-сайтів, але майже всі веб-застосунки потребують реєстрації та автентифікації. У більшості випадків незареєстровані відвідувачі веб-сайтів мають такий самий доступ до функцій, як і зареєстровані користувачі. З іншого боку, веб- застосунки майже завжди вимагають аутентифікації користувачів, оскільки вони надають послуги, налаштовані під вимоги користувача [1].

Веб-застосунки не потрібно завантажувати, оскільки вони доступні через мережу. Користувачі можуть отримати доступ до веб- застосунків через веб-браузер, наприклад Google Chrome, Safari або Mozilla Firefox.

Враховуючи швидкий розвиток портативної електроніки та мобільних гаджетів, мобільні додатки відіграють важливу роль у сучасних технологіях [2]. Для ефективної роботи такі додатки повинні синхронізувати свої дані з іншими ресурсами, використовуючи backend-сервери. Це забезпечує зручний доступ до актуальної та захищеної інформації, що є критично важливим для функціонування мобільних веб-застосунків [3].

Основний принцип роботи веб-застосунків базується на зв’язку між клієнтською та серверною частинами програми. При взаємодії з веб-застосунком, клієнтська частина, яка знаходиться на боці користувача, взаємодіє з сервером, де зберігаються дані та виконуються основні обчислення.

Клієнтська частина веб-застосунку, яка зазвичай складається з веб-сторінок та скриптів, відповідає за інтерактивність та візуальний інтерфейс. Користувач взаємодіє з клієнтською частиною через веб-браузер, надсилаючи запити та отримуючи відповіді від сервера.

Серверна частина веб-застосунку відповідає за обробку запитів користувача, доступ до баз даних, виконання логіки програми та надсилання

відповідей на клієнтську частину. Вона забезпечує централізоване керування даними та логікою додатку, забезпечуючи узгодженість та надійність роботи веб-застосунку.

При взаємодії з веб-застосунком, користувач надсилає запити на сервер, які обробляються серверною частиною, після чого результати обробки повертаються на клієнтську сторону та відображаються користувачеві через веб-браузер. Цей процес може включати в себе обмін даними у реальному часі, виконання операцій на стороні сервера та оновлення вмісту сторінок без перезавантаження.

Таким чином, основний принцип роботи веб-застосунків полягає у взаємодії між клієнтською та серверною частинами програми через мережу Інтернет, що дозволяє користувачам отримувати доступ до різноманітних функцій та послуг у веб-середовищі.

Існує кілька типів веб-застосунків, які відрізняються за своєю структурою, функціональністю та способом взаємодії з користувачем. До найпоширеніших типів веб-застосунків відносяться односторінкові застосунки (SPA), багатосторінкові застосунки (MPA) та прогресивні веб-застосунки (PWA).

Односторінкові застосунки відрізняються тим, що вони загружаються лише один раз, і весь контент змінюється динамічно без перезавантаження сторінки. Це дозволяє створювати більш інтерактивні та швидкі застосунки, які надають користувачам плавний і безшовний досвід. SPA часто використовуються у веб-застосунках, які вимагають великої взаємодії з користувачем та роботи в реальному часі, наприклад, соціальні мережі, електроні пошти, відео чи аудіоплеєри.

Багатосторінкові застосунки складаються з кількох веб-сторінок, які повністю перезавантажуються при переході між ними. Кожна сторінка відповідає окремій функціональній частині застосунка, і користувач переходить між ними, натискаючи на посилання або використовуючи меню навігації. Цей тип застосунків може бути корисним у випадках, коли потрібно розділити функціональність на окремі сторінки або забезпечити стабільну навігацію на сайті.

Прогресивні веб-застосунки сучасним типом веб-застосунків, які поєднують в собі найкращі характеристики веб-сайтів та додатків для мобільних пристроїв. Вони пропонують користувачам зручний інтерфейс, можливість роботи офлайн, покращену швидкість завантаження та інші переваги. PWA можуть бути встановлені на пристрої користувачів та працювати як незалежні додатки, але при цьому вони використовують технології веб-розробки.

Кожен з цих типів веб-застосунків має свої переваги та особливості, і вибір конкретного типу залежить від потреб та вимог конкретного проекту.

1.2 Аналіз засобів для створення веб-застосунків

Розробка веб-застосунків вимагає уваги до деталей на кожному етапі – від концепції та дизайну до тестування та оптимізації. Завдяки відповідним інструментам, цей процес може стати більш ефективним та продуктивним.

HTML та CSS

HTML і CSS є двома ключовими мовами для створення веб-сайтів. HTML відповідає за структурування веб-сторінок, а CSS відповідає за їхнє оформлення та візуальний вигляд.

HTML є мовою розмітки, яка визначає структуру веб-сторінки за допомогою різноманітних елементів і тегів. Основною одиницею в HTML є елемент, який представляє певний тип вмісту, такий як текст, зображення або посилання. Елементи HTML використовуються для організації вмісту веб-сторінки в логічні блоки, такі як заголовки, абзаци, списки, таблиці тощо.

HTML використовується для визначення структури веб-сторінки, вказуючи, які елементи розташовані де на сторінці, як вони пов'язані між собою та яким чином вони мають відображатися браузером.

CSS відповідає за вигляд і оформлення веб-сторінки. Вона використовується для задавання стилів, таких як кольори, шрифти, розміри, відступи, рамки та інші візуальні атрибути елементів HTML. CSS робить можливим відокремлення структури і змісту веб-сторінки від її оформлення, що дозволяє змінювати вигляд сторінки без зміни її структури [6].

Завдяки CSS можна забезпечити веб-сайт зручнішим і привабливішим дизайном, що відповідає сучасним тенденціям веб-розробки. Також CSS дозволяє створювати адаптивні та респонсивні веб-застосунки, які правильно відображаються на різних пристроях і розмірах екранів.

Загалом, CSS і HTML використовуються разом для формування структури та стилю веб-сторінок, надаючи розробникам можливість створювати естетично привабливі та функціональні веб-сайти [6].

JavaScript

JavaScript — це мова програмування, яка використовується для додавання додаткової функціональності та динаміки веб-сторінкам. Основні характеристики JavaScript включають можливість взаємодії з користувачем без потреби перезавантажувати сторінку, обробку подій, маніпулювання елементами HTML, а також взаємодію з сервером для отримання або відправки даних. JavaScript застосовується у широкому спектрі веб-додатків, від простих скриптів для перевірки форм до складних односторінкових додатків та ігор.

JavaScript дозволяє динамічно змінювати вміст веб-сторінок, створювати анімаційні ефекти, обробляти форми перед їхнім надсиланням на сервер, здійснювати валідацію введених даних, а також додавати інтерактивні елементи, такі як підказки, відкриття та закриття вікон і меню. Окрім того, JavaScript дозволяє робити серверні запити без необхідності перезавантаження сторінки, що дає змогу створювати динамічні веб-додатки [6].

JavaScript особливо часто використовується для створення односторінкових додатків, де вся взаємодія з користувачем відбувається на одній сторінці без переходу на інші URL-адреси. Це забезпечує створення зручніших та ефективніших користувацьких інтерфейсів.

В цілому, JavaScript є невід'ємною частиною сучасної веб-розробки, допомагаючи створювати інтерактивні, динамічні та продуктивні веб-додатки, які відповідають потребам користувачів та вимогам сучасного інтернету. Варто також згадати важливість JavaScript-бібліотек, таких як React, Angular та jQuery, які значно спрощують процес розробки веб-додатків [6].

React

React – це бібліотека JavaScript, що використовується для розробки інтерфейсів користувача. Вона використовується для побудови веб-застосунків з високою динамічністю та ефективною роботою з DOM.

Основною концепцією React є компоненти. Кожен компонент є незалежним ізольованим елементом, який містить в собі як HTML-подібну розмітку, так і логіку JavaScript. Ці компоненти можна збирати в більш складні структури, які називаються деревами компонентів. React використовує спеціальний синтаксис JSX (JavaScript XML), що дозволяє описувати структуру компонентів, що нагадує HTML, але використовує JavaScript.

Однією з головних переваг React є віртуальний DOM. React підтримує внутрішню віртуалізацію DOM-структури, що дозволяє ефективно маніпулювати великими об'ємами даних шляхом зменшення кількості операцій з реальним DOM, що забезпечує покращену продуктивність додатків.

Ще однією важливою особливістю React є односторінкова архітектура додатків. Завдяки SPA, React додатки завантажуються один раз, а далі взаємодіють з користувачем без перезавантаження сторінки, що забезпечує більш зручний та швидкий досвід користувача.

Узагальнюючи, React дозволяє створювати потужні, ефективні та динамічні веб-застосунки шляхом зручного використання компонентного підходу та використання віртуального DOM.

Angular

Angular – це фреймворк для розробки веб-застосунків. Він використовується для побудови високопродуктивних та динамічних клієнтських застосунків, зокрема односторінкових додатків.

Основними концепціями Angular є модулі, компоненти та сервіси. Модулі дозволяють організовувати код додатку та розділяти його на логічні частини. Компоненти в Angular є будівельними блоками веб-інтерфейсу та містять в собі як HTML-подібну розмітку, так і логіку JavaScript або TypeScript. Сервіси використовуються для реалізації загальної логіки додатку та обробки даних.

Однією з відмінностей Angular від інших фреймворків є використання TypeScript. TypeScript дозволяє виявляти та усувати помилки на етапі розробки, що полегшує підтримку великих проектів.

Angular також надає вбудовану підтримку для різних функціональностей, таких як маршрутизація, валідація форм, обробка HTTP-запитів, анімація та інші. Він також має потужну систему завдань і плагінів для автоматизації процесів розробки та збірки додатку.

Загалом, Angular є потужним фреймворком для розробки веб-застосунків, що надає багато можливостей та зручний інструментарій для розробників.

jQuery

jQuery – це швидкий, легкий та потужний JavaScript-фреймворк, який спрощує взаємодію з HTML-документами, обробку подій, анімацію та AJAX-запити. Він забезпечує простий та зручний спосіб вибору DOM-елементів та взаємодії з ними.

Однією з ключових особливостей jQuery є його здатність працювати з CSS-селекторами для вибору елементів на сторінці. Це дозволяє легко здійснювати маніпуляції з DOM-елементами, такі як зміна їхнього вмісту, стилів або розташування [9].

jQuery також надає прості методи для обробки подій, таких як натискання кнопок, клікання миші або введення тексту. Це дозволяє створювати динамічні та інтерактивні веб-сторінки з легкістю.

Ще однією важливою функцією jQuery є підтримка AJAX-запитів. Вона дозволяє взаємодіяти з сервером без перезавантаження сторінки, що забезпечує плавний та швидкий досвід користувача.

Хоча в останні роки популярність jQuery трохи зменшилася через розвиток сучасних фреймворків та стандартів JavaScript, він все ще залишається корисним інструментом для швидкої та простої розробки веб-застосунків, особливо для проектів невеликої або середньої складності.

Python

Python — це потужна і динамічна мова програмування, що широко застосовується для створення веб-додатків. Простий та зрозумілий синтаксис

робить її популярним вибором для багатьох розробників. Крім того, Python має багато бібліотек та фреймворків, які полегшують розробку веб-додатків.

Одним з основних фреймворків для розробки веб-додатків на Python є Django. Django пропонує безліч вбудованих функцій, які полегшують створення високопродуктивних і безпечних веб-додатків. Він включає систему аутентифікації, управління базами даних, систему маршрутизації та інші інструменти, що допомагають швидко створювати повнофункціональні веб-додатки.

Крім Django, Flask є ще одним популярним фреймворком для розробки веб-додатків на Python. Flask більш легкий та гнучкий, ніж Django, і дозволяє розробникам створювати веб-додатки швидше із меншим обсягом коду. Flask також має багато розширень, які розширюють його функціональні можливості.

Крім цього, Python має безліч інших бібліотек та інструментів для розробки веб-додатків, таких як SQLAlchemy для роботи з базами даних, Celery для асинхронних завдань, і багато інших.

PHP

PHP — це універсальна мова програмування, яка в основному використовується для створення веб-додатків. Вона широко застосовується для розробки динамічних веб-сайтів та взаємодії з базами даних.

Однією з головних переваг PHP є її простота вивчення та використання. Синтаксис PHP легко сприймається багатьма розробниками, що робить її відмінним вибором для новачків. Крім того, PHP інтегрований у багато веб-серверів і має багатий набір документації та ресурсів для навчання та використання.

Однією з важливих особливостей PHP є її здатність взаємодіяти з HTML. Код PHP може бути безпосередньо вбудований у HTML-сторінку, що дозволяє створювати динамічні веб-сайти, які генеруються в реальному часі залежно від умов та даних із сервера.

Крім того, PHP має велику кількість додаткових модулів та бібліотек, які допомагають розробникам виконувати різноманітні завдання, такі як робота з базами даних, обробка зображень, обробка форм, автентифікація користувачів та багато іншого.

Web.com

Web.com є провідним постачальником послуг в галузі веб-хостингу та створення веб-сайтів, спеціалізується на наданні комплексних рішень для бізнесу та індивідуальних користувачів. Компанія пропонує широкий спектр послуг, які включають веб-хостинг, доменні імена, розробку та дизайн веб-сайтів, електронну комерцію, маркетинг в Інтернеті, а також інструменти для онлайн-продажу та управління бізнесом.

Основна мета Web.com – надати клієнтам усі необхідні інструменти та ресурси для успішного присутності в Інтернеті та розвитку їхнього бізнесу в онлайн-середовищі. Зокрема, компанія надає можливість швидко та легко створити професійний веб-сайт за допомогою свого конструктора веб-сайтів або скористатися послугами дизайнерів та розробників, які забезпечать індивідуальний підхід до потреб кожного клієнта.

Крім того, Web.com пропонує високоефективні хостингові рішення, що забезпечують надійну та безперебійну роботу веб-сайтів, а також інструменти та послуги для просування бізнесу в Інтернеті, такі як SEO, реклама в соціальних мережах, електронна комерція та інше.

Chrome DevTools

Chrome DevTools – це набір інструментів для розробників, який вбудований у веб-браузер Google Chrome. Він надає широкий спектр інструментів для аналізу, відлагодження та оптимізації веб-сторінок та веб-застосунків. Однією з ключових функцій Chrome DevTools є можливість перегляду та редагування HTML, CSS та JavaScript-коду прямо на сторінці, що дозволяє розробникам швидко та зручно вносити зміни у веб-сайт.

DevTools також надає різноманітні інструменти для аналізу профілювання, які допомагають виявляти та виправляти проблеми з продуктивністю веб-додатків. Наприклад, ви можете використовувати інструменти для аналізу завантаження сторінки, профілювання JavaScript-коду, аналізу використання пам'яті та інші.

Однією з корисних функцій Chrome DevTools є можливість відлагодження JavaScript-коду за допомогою вбудованого інструменту «Debugger». Можна встановлювати точки зупинки, виконувати код по кроках та відстежувати

значення змінних, що дозволяє ефективно виявляти та виправляти помилки в JavaScript-скриптах.

Ще однією корисною функцією є інструменти для аналізу мобільності, які дозволяють перевіряти, як веб-сайт виглядає та працює на різних пристроях та розмірах екрану. Це допомагає забезпечити оптимальний досвід користувача на різних пристроях та платформах.

CodePen

CodePen – це онлайн-платформа, яка спрощує процес розробки веб-застосунків і веб-сайтів, надаючи розробникам зручний та потужний інструментарій для написання та демонстрації фронтенд-коду. В основі CodePen лежить ідея створення віртуального пісочниці, де розробники можуть створювати, тестувати та демонструвати свої веб-проекти в реальному часі, без необхідності налаштовування власного середовища розробки.

Одним з ключових елементів CodePen є його інтерактивний редактор коду, який дозволяє розробникам працювати з HTML, CSS та JavaScript у зручному та інтуїтивно зрозумілому інтерфейсі. Редактор надає можливість писати код, переглядати зміни в реальному часі та взаємодіяти з ним без необхідності перезавантаження сторінки.

Крім того, CodePen має багато корисних функцій, таких як підтримка препроцесорів CSS (наприклад, Sass або Less), автозавершення коду, можливість додавання власних зовнішніх бібліотек та інструментів, а також можливість створення та використання шаблонів проектів.

CodePen також надає можливість зберігати, публікувати та вбудовувати свої проекти, що дозволяє розробникам легко ділитися своїми досягненнями та взаємодіяти зі спільнотою. Крім того, CodePen пропонує різноманітні суспільні функції, такі як коментування, лайки та репости, що дозволяє розробникам отримати фідбек та підтримку від інших учасників.

Загалом, при розробці веб-застосунків використовуються різні інструменти та технології. HTML та CSS відповідають за структуру та оформлення веб-сторінок, JavaScript – за надання їм інтерактивності та динамічності. Бібліотека jQuery спрощує роботу з JavaScript, а фреймворки Angular та React надають потужний функціонал для розробки веб-застосунків.

Мови програмування Python та PHP дозволяють створювати розширені веб-застосунки, а Web.com надає інструменти для їх розгортання та хостингу. Chrome DevTools та CodePen допомагають у відлагодженні та демонстрації веб-застосунків. Вибір інструментів залежить від конкретних потреб проекту та відповідних вимог розробки.

1.3 Аналіз існуючих інструментів для управління подіями

В сучасному світі ефективне управління подіями є важливою складовою успішного планування як у професійному, так і в особистому житті. Існує багато інструментів, які допомагають організаторам та користувачам керувати подіями, планувати зустрічі та відстежувати дедлайни. Нижче наведено декілька найпопулярніших інструментів.

Google Calendar

Google Calendar – це безкоштовний онлайн-календар, створений компанією Google, який дозволяє користувачам організовувати та керувати своїм часом, а також ділитися подіями з іншими користувачами. Основні особливості та переваги Google Calendar включають:

– Інтеграція з іншими сервісами Google: Google Calendar тісно інтегрований з іншими сервісами Google, такими як Gmail, Google Meet, Google Tasks та Google Keep. Це дозволяє автоматично створювати події з листів, додавати завдання та нагадування до календаря, а також запускати відеозустрічі безпосередньо з подій календаря.

– Спільний доступ до календаря: Користувачі можуть ділитися своїми календарями з іншими людьми, надаючи їм різні рівні доступу (перегляд, редагування тощо). Це особливо корисно для командної роботи, планування сімейних подій або координації з друзями.

– Мобільні додатки: Google Calendar доступний як у веб-версії, так і через мобільні додатки для Android та iOS, що дозволяє користувачам отримувати доступ до свого календаря з будь-якого місця та пристрою.

– Нагадування та сповіщення: Користувачі можуть налаштовувати сповіщення та нагадування для подій, які надходять через електронну пошту або як push-повідомлення на мобільний пристрій.

– Інтеграція з іншими календарями: Google Calendar підтримує імпорт та експорт подій, а також синхронізацію з іншими календарями.

Microsoft Outlook Calendar

Microsoft Outlook Calendar – це частина пакету Microsoft Office та сервісу Outlook.com, який забезпечує потужні інструменти для управління подіями та планування. Основні особливості та переваги Outlook Calendar включають:

– Інтеграція з іншими інструментами Microsoft: Outlook Calendar інтегрований з електронною поштою Outlook, Microsoft Teams, OneNote та іншими додатками Microsoft 365, що дозволяє користувачам створювати події з листів, додавати зустрічі до календаря з Microsoft Teams та синхронізувати завдання з OneNote.

– Потужні функції для бізнес-користувачів: Outlook Calendar пропонує розширені функції для бізнес-користувачів, такі як бронювання переговорних кімнат, управління спільними календарями, відстеження часу та ресурсів, а також планування повторюваних подій.

– Захист та безпека даних: Microsoft надає високий рівень захисту та безпеки даних, включаючи шифрування, багатофакторну аутентифікацію та можливості управління доступом, що особливо важливо для корпоративних користувачів.

– Крос-платформеність: Outlook Calendar доступний через веб-версію, настільні додатки для Windows та Mac, а також мобільні додатки для Android та iOS.

Apple Calendar

Apple Calendar (раніше відомий як iCal) – це стандартний календар для пристроїв на базі iOS та macOS, який пропонує простий у використанні інтерфейс та інтеграцію з іншими додатками Apple. Основні особливості та переваги Apple Calendar включають:

– Екосистема Apple: Apple Calendar інтегрується з іншими додатками Apple, такими як Mail, Contacts та Reminders, що дозволяє створювати події з листів, додавати контакти до подій та налаштовувати нагадування.

– Хмарне зберігання через iCloud: Календарі автоматично синхронізуються через iCloud, що дозволяє користувачам отримувати доступ до своїх подій з будь-якого пристрою Apple та здійснювати резервне копіювання даних.

– Зручний інтерфейс: Інтерфейс Apple Calendar відзначається інтуїтивністю та простотою використання, дозволяючи легко створювати, редагувати та переглядати події.

– Сумісність з іншими календарями: Apple Calendar підтримує імпорт та експорт подій у форматах iCal та ICS, а також синхронізацію з іншими календарями, такими як Google Calendar та Microsoft Outlook.

Trello

Trello – це інструмент для управління проектами, який використовує систему дошок, списків та карток для організації завдань та подій. Основні особливості та переваги Trello включають:

– Візуальна організація: Trello дозволяє користувачам візуально організовувати завдання за допомогою дошок, списків та карток, що робить управління проектами зрозумілим та наочним.

– Командна робота: Trello забезпечує спільний доступ до дошок, дозволяючи командам спільно працювати над завданнями, призначати виконавців, залишати коментарі та відстежувати прогрес.

– Інтеграція з іншими інструментами: Trello підтримує інтеграцію з багатьма популярними сервісами, такими як Google Drive, Slack, Jira, Microsoft Teams та іншими, що дозволяє легко поєднувати робочі процеси.

– Гнучкість та налаштування: Користувачі можуть налаштовувати дошки та картки відповідно до своїх потреб, додавати мітки, дедлайни, вкладення, контрольні списки та інші елементи.

Asana

Asana – це потужний інструмент для управління проектами, який дозволяє командам організовувати роботу, встановлювати дедлайни, призначати виконавців та відстежувати прогрес. Основні особливості та переваги Asana включають:

– Організація роботи за проектами: Asana дозволяє створювати проекти, розбивати їх на завдання та підзавдання, встановлювати пріоритети та дедлайни, що допомагає ефективно планувати та виконувати роботу.

– Інструменти для командної роботи: Asana підтримує спільний доступ до проектів, дозволяє призначати завдання конкретним користувачам, залишати коментарі, додавати файли та відстежувати статус завдань у реальному часі.

– Інтеграція з іншими сервісами: Asana інтегрується з багатьма популярними інструментами, такими як Slack, Google Drive, Microsoft Teams, Jira та іншими, що дозволяє легко інтегрувати її в існуючі робочі процеси.

– Звіти та аналітика: Asana пропонує потужні інструменти для звітності та аналітики, що дозволяють відстежувати прогрес проектів, аналізувати продуктивність та виявляти проблемні місця.

– Мобільні додатки: Asana доступна через веб-версію, а також мобільні додатки для Android та iOS, що дозволяє користувачам керувати завданнями та проектами з будь-якого місця.

1.4 Висновок до першого розділу

У першому розділі кваліфікаційної роботи було визначено поняття веб-застосунку, що є фундаментом для подальших досліджень і розробок. Проведено аналіз сучасних засобів для створення веб-застосунків, включаючи різноманітні платформи та фреймворки, що дозволяють ефективно вирішувати завдання розробки. Проаналізовано існуючі інструменти для управління подіями, таких як Google Calendar, Microsoft Outlook Calendar, Apple Calendar, Trello та Asana.

РОЗДІЛ 2. ПРОЕКТНА ЧАСТИНА. ОГЛЯД GOOGLE APPS SCRIPT ТА ЙОГО ФУНКЦІОНАЛУ

2.1 Вибір технології для реалізації веб-застосунку

При виборі технології для реалізації веб-застосунку календаря подій було розглянуто кілька важливих аспектів, таких як зручність використання, інтеграція з існуючими сервісами, можливості для розширення функціоналу та наявність документації і підтримки. Нижче наведено причини чому було обрано Google Apps Script для створення календаря подій.

Інтеграція з екосистемою Google

Google Apps Script дозволяє легко інтегрувати веб-застосунок з іншими сервісами Google, такими як Google Calendar, Google Sheets, Gmail та Google Drive. Це забезпечує зручне створення, зберігання та обробку даних подій, а також автоматизацію робочих процесів. Для календаря подій важливою є інтеграція з Google Calendar, що дозволяє користувачам створювати, переглядати та редагувати події безпосередньо в календарі Google, а також отримувати нагадування та сповіщення.

Простота та швидкість розробки

Google Apps Script – це високорівнева мова сценаріїв на базі JavaScript, яка не вимагає розгортання серверної інфраструктури. Всі сценарії виконуються в хмарі Google, що знижує витрати на розробку та підтримку. Інструмент має вбудовану інтеграцію з редактором коду, де можна писати та тестувати сценарії безпосередньо в браузері, що прискорює процес розробки.

Можливості автоматизації

Google Apps Script дозволяє автоматизувати багато рутинних задач, таких як створення подій з електронних листів, синхронізація даних між таблицями та календарем, надсилання сповіщень про майбутні події тощо.

Безпека та контроль доступу

Google надає високий рівень захисту даних та можливості для управління доступом до додатків і даних. Використання Google Apps Script дозволяє легко налаштовувати права доступу для користувачів та захищати конфіденційну

інформацію. Дані зберігаються у хмарі Google, що забезпечує їх надійне збереження та резервне копіювання.

Спільний доступ та колаборація

Google Apps Script підтримує створення та редагування спільних сценаріїв, що дозволяє декільком розробникам одночасно працювати над одним проектом. Це особливо корисно для командної роботи та швидкого вирішення проблем. Календар подій, створений за допомогою Google Apps Script, може бути легко поділений між користувачами, що забезпечує ефективну колаборацію та планування.

Документація та підтримка

Google Apps Script має обширну документацію та активну спільноту користувачів, що дозволяє швидко знаходити рішення для різноманітних задач та отримувати підтримку. Відкритий доступ до прикладів коду та навчальних матеріалів допомагає новачкам швидко освоїти основи роботи з Google Apps Script.

Зважаючи на всі перелічені переваги, для розробки веб-застосунку календаря подій у цій дипломній роботі було обрано Google Apps Script як основну технологію.

2.2 Огляд Google Apps Script та його функцій

Google Apps Script – це вбудований інструмент розробки, призначений для автоматизації різних завдань у Google-середовищі. Завдяки цьому інструменту розробники можуть створювати власні скрипти, які взаємодіють із продуктами Google, такими як Google Sheets, Docs, Calendar та іншими. За допомогою Apps Script можна автоматизувати процеси створення звітів, обробки даних, роботи з електронними таблицями та текстовими документами, а також створювати розширення і додатки для користувачів Google Workspace. Не потрібно встановлювати додаткове програмне забезпечення, оскільки редактор коду знаходиться безпосередньо у веб-оглядачі, а сам код виконується на серверах. Це надає широкі перспективи для створення власних рішень та підвищення ефективності роботи у екосистемі Google.

Google створив Apps Script як інструмент для автоматизації різних процесів та розширення можливостей своїх сервісів. Перші версії цього сервісу з'явилися наприкінці 2009 року. З часом інструмент постійно розвивався, надаючи розробникам все більше можливостей для створення скриптів і додатків. Відтоді Google Apps Script став невід'ємною частиною Google Workspace і отримав підтримку різних сервісів компанії. Сьогодні цей інструмент є потужним засобом для автоматизації робочих процесів та створення власних рішень в Google.

Google Apps Script має такі головні переваги:

– Інтеграція з Google Workspace: Google Apps Script дозволяє легко автоматизувати та розширювати функціональність додатків Google, таких як Google Sheets, Google Docs, Google Drive та інших сервісів Google Workspace.

– Простота використання: Google Apps Script базується на JavaScript, що робить його доступним для багатьох розробників, які вже знайомі з цією мовою програмування.

– Хмарна інфраструктура: Сценарії виконуються на серверах Google, що означає відсутність необхідності налаштовувати власну інфраструктуру або турбуватися про масштабування та продуктивність.

– Швидкий старт: Розробка сценаріїв може початися безпосередньо в браузері через редактор Google Apps Script, що дозволяє швидко створювати та тестувати сценарії без необхідності встановлення додаткового програмного забезпечення.

– Автоматизація завдань: Google Apps Script дозволяє автоматизувати рутинні завдання, такі як обробка електронної пошти, управління календарями, оновлення таблиць і багато іншого, що значно підвищує ефективність роботи.

– Інтеграція зі сторонніми сервісами: За допомогою Google Apps Script можна інтегрувати додатки Google з іншими сервісами через API, створюючи таким чином потужні рішення для бізнесу.

– Безкоштовний доступ: Для більшості користувачів Google Apps Script є безкоштовним, що робить його доступним інструментом для широкого кола завдань без додаткових витрат.

– Безпека та управління доступом: Google Apps Script пропонує можливості для управління доступом до сценаріїв та даних, забезпечуючи високу безпеку та контроль над інформацією.

– Активна спільнота та підтримка: Велика спільнота розробників і офіційна підтримка від Google забезпечують доступ до численних ресурсів, документації та прикладів коду, що спрощує навчання та розробку нових рішень.

– Гнучкість та налаштування: Google Apps Script дозволяє створювати кастомні рішення, що точно відповідають потребам бізнесу або особистим вимогам. Від створення спеціалізованих функцій до розробки складних додатків, ця платформа забезпечує високу гнучкість у налаштуванні та розширенні можливостей існуючих сервісів Google.

Google Apps Script надає широкий спектр функцій для створення сценаріїв, що можуть інтегруватися з різними сервісами Google. Дані сценарії можуть бути створені як інший файли в Google Диску, внутрішні частини документів або електронних таблиць, або навіть на Google Сайти [4].

Розроблений скрипт інтегрується безпосередньо в екосистему Google, що дає можливість використовувати його з практично будь-якими продуктами Google.

Онлайн-редактор Google Apps Script надає всі потрібні інструменти для зручної розробки. Його інтерфейс є дуже зручним та інтуїтивно зрозумілим, що дозволяє легко створювати, модифікувати та перевіряти ваші скрипти безпосередньо у веб-браузері.

Використовуючи Google Apps Script, можна розробляти повноцінні веб-застосунки та розширення, які працюють у браузерах, включаючи мобільні пристрої. Це надає широкі можливості для створення різних застосувань, які можуть оптимізувати робочі процеси та полегшити повсякденне життя.

Google Apps Script – це інтегрована платформа для розробки, що базується на JavaScript і призначена для автоматизації різних бізнес-процесів та розширення можливостей сервісів Google. Цей інструмент надає можливість створювати програмні коди, які можуть взаємодіяти з багатьма продуктами Google, включаючи Google Sheets, Gmail, Calendar, Drive та багато інших.

Завдяки Apps Script, користувачі можуть автоматизувати повторювані завдання, розширювати функціональність користувацького інтерфейсу та розробляти різні види додатків.

У Google Apps Script існують деякі обмеження на кількість запитів до API та обсяг використання ресурсів, відомих як «квоти». Наприклад, є ліміти на кількість запитів до зовнішніх сервісів, обсяг використання пам'яті, час виконання тощо. Ці квоти можуть впливати на продуктивність та швидкодію скриптів, тому важливо уважно керувати їхнім використанням та враховувати їх під час розробки.

Хоча Apps Script дозволяє без проблем створювати та виконувати скрипти, ефективність виконання може залежати від різних факторів, таких як обсяг оброблюваних даних, складність алгоритмів, оптимізація коду та використання ресурсів. Дотримання кращих практик програмування та уникання зайвого використання ресурсів може покращити продуктивність ваших скриптів.

Apps Script підтримує асинхронні операції, що дозволяє виконувати деякі операції паралельно та ефективно керувати часом виконання скриптів. Це особливо важливо при взаємодії з віддаленими сервісами або обробці великих обсягів даних.

Однією з потужних перспектив Apps Script є здатність реагувати на різні події, такі як надходження нових електронних листів, зміни в Google Документаї або таблицях, а також відкриття файлів у Google Drive. Це дозволяє автоматизувати реакцію на різні події та виконувати відповідні дії без необхідності ручного запуску коду.

У Apps Script також існують тригери, які дозволяють автоматично запускати скрипти відповідно до певних критеріїв або розкладу. Наприклад, створення тригеру, що запускає скрипт щогодини або відправляє електронний лист при певній події. Це дозволяє автоматизувати виконання скриптів та реагувати на різні події без необхідності постійного моніторингу чи ручного запуску коду.

У середовищі Apps Script, як і в багатьох інших реалізаціях JavaScript, є рушій JavaScript. Це середовище встановлює правила управління пам'яттю,

взаємодію платформи з системою та прийнятну побудову. Різні браузери мають своє унікальне середовище JavaScript [3].

Зазвичай Google Apps Script використовує інтерпретатор Rhino. Даний інтерпритатор забезпечував зручне виконання скриптів в Apps Script, але також обмежував його версією ES5. Через це розробники не мали можливості застосовувати новітні синтаксичні конструкції JavaScript у власних сценаріях, якщо вони використовували Rhino. Для того, щоб усунути дану проблему, Google Apps Script послуговує рушій V8, який уживається в Chrome та Node.js.

Сценарії, що виконуються на рушії V8, мають доступ до широкого спектра функцій і можливостей, які надає цей рушій JavaScript. Однією з основних переваг є можливість використання сучасних синтаксичних конструкцій і функцій JavaScript, що дозволяє розробникам писати більш чистий та ефективний код.

Завдяки використанню V8, скрипти можуть взаємодіяти з іншими сервісами та продуктами Google з великою швидкістю та продуктивністю. Це дозволяє розробникам створювати потужні та ефективні веб-застосунки, які швидко відповідають на запити користувачів та забезпечують зручний інтерфейс користувача.

Крім того, середовище виконання V8 надає можливість оптимізації продуктивності скриптів шляхом автоматичної оптимізації та кешування проміжних результатів обчислень. Завдяки цьому можна зменшити час виконання скриптів та підвищити їх ефективність [10].

Також, завдяки середовищу виконання V8, скрипти можуть використовувати розширені можливості, такі як робота з масивами та об'єктами, робота з файловою системою та мережевими запитами, що робить їх більш гнучкими та потужними [10].

Однією з головних переваг Apps Script є його зручний онлайн-редактор, який дозволяє зручно та легко створювати, редагувати та відлагоджувати скрипти без необхідності встановлення додаткових програм або середовищ для розробки, що особливо зручно тим, хто працює у веб-браузері, оскільки весь процес розробки відбувається прямо в середовищі Google. На рисунку 2.1 зображено редактор скриптів.

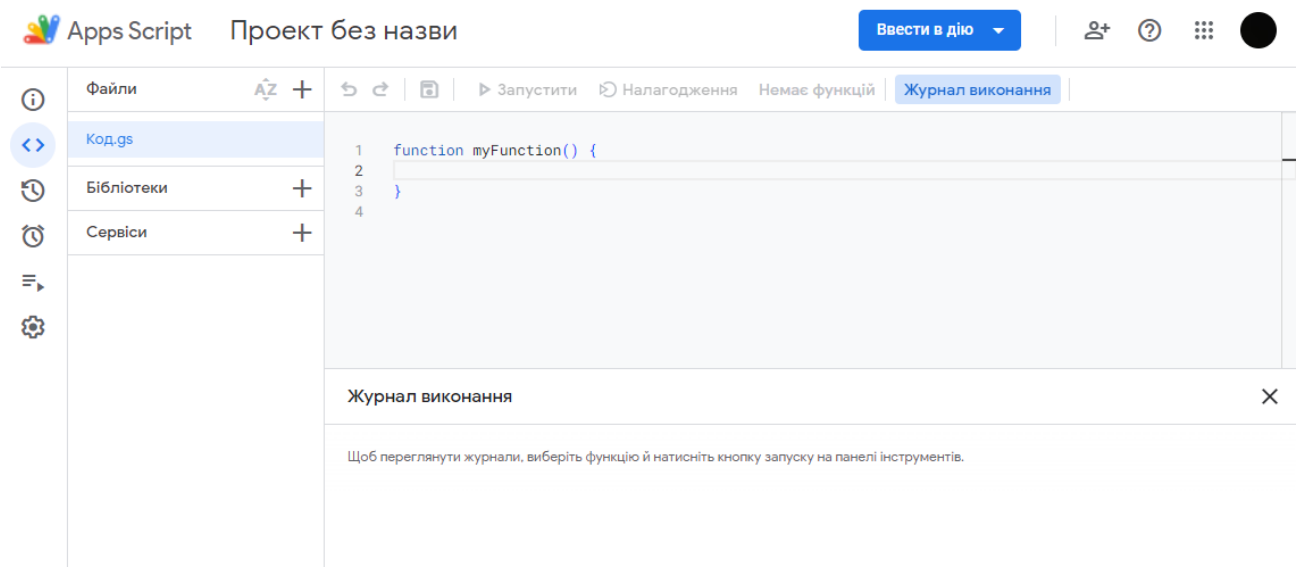


Рисунок 2.1 – Редактор скриптів

Основна частина інтерфейсу – це текстовий редактор, де можна писати та редагувати скрипти. Текстовий редактор має зручну підтримку автоматичного доповнення коду, підсвічування синтаксису та можливість робити швидкі перевірки наявних помилок у коді.

На верхній частині інтерфейсу розташоване меню з різноманітними опціями та інструментами для роботи зі скриптами. Тут можна створювати нові скрипти, відкривати та редагувати існуючі, керувати версіями та налаштуваннями проекту, а також запускати скрипти та переглядати журнали виконання.

Панель навігації дозволяє швидко переміщатися між різними файлами проекту та переглядати їх вміст. Тут можна легко керувати структурою проекту, створювати нові файли, переглядати та редагувати вже існуючі. Ця панель є важливим інструментом для розробників, які працюють з Google Apps Script, оскільки вона спрощує навігацію і керування великими проектами, дозволяючи зосередитися на важливих аспектах програмування і автоматизації без зайвих труднощів у керуванні файлами і структурою проекту. На рисунку 2.2 зображено навігаційну панель.

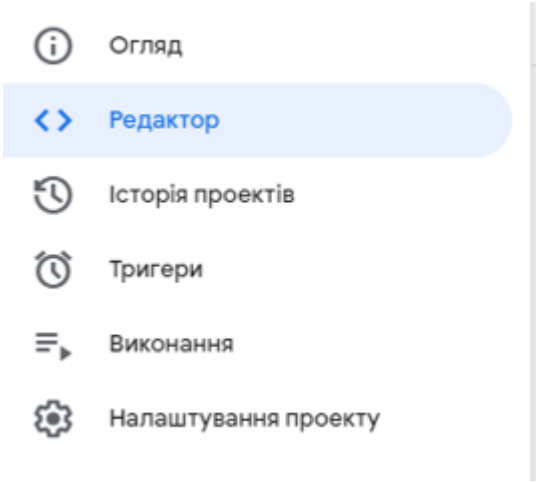


Рисунок 2.2 – Навігаційна панель

Додаткові вкладки в інтерфейсі дозволяють керувати бібліотеками скриптів, налаштуваннями середовища виконання, ресурсами проекту та іншими налаштуваннями. Також для роботи зі звітами та журналом виконання доступні інструменти, які дозволяють вам відстежувати та аналізувати роботу скриптів.

Редактор коду має вбудовану відлагоджувальну консоль, розташовану в нижній частині екрану. Ця консоль дозволяє відображати повідомлення, похибки та результати роботи скрипту. Вона є важливим інструментом для проведення налагодження та повного аналізу коду, дозволяючи встановлювати контрольні точки та слідкувати за динамікою значень змінних протягом виконання скрипту.

У Google Apps Script існують два основних типи скриптів: автономні та прив'язані. Автономні скрипти функціонують незалежно від будь-яких конкретних документів чи електронних таблиць. Їх можна використовувати для створення різноманітних веб-застосунків або автоматизації операцій в середовищі Google без прив'язки до певного документа. Ці скрипти можуть взаємодіяти з різними сервісами Google, такими як Gmail, Календар або Диск, та виконувати широкий спектр завдань за розкладом або за запитом користувача.

Прив'язані скрипти, навпаки, пов'язані з конкретними документами або електронними таблицями у Google Документах або Таблицях. Вони дозволяють автоматизувати операції, що виконуються у межах цих документів чи таблиць, наприклад, обробку даних, створення звітів або розрахунків. Прив'язані скрипти можуть реагувати на різні події у документі чи таблиці, такі як введення даних,

зміна значень або натискання кнопок, що забезпечує їхню велику гнучкість та використання у специфічних контекстах.

Для створення прив'язаного скрипту необхідно відкрити конкретний документ або електронну таблицю в Google Документах або Таблицях, потім перейти до розділу «Інструменти» та вибрати пункт «Редактор скриптів». Це відкриє онлайн-редактор коду Google Apps Script, де можна створити новий скрипт або редагувати вже існуючий.

Прив'язані скрипти можуть реагувати на різні події у документі чи таблиці, такі як введення даних, зміна значень або натискання кнопок. Для цього можна використовувати спеціальні обробники подій, які викликаються автоматично при виникненні певної події. Наприклад, створити скрипт, який автоматично відправляє електронний лист звіт з даними, якщо в конкретному рядку електронної таблиці змінюються значення.

Загалом, прив'язані скрипти дозволяють автоматизувати процеси роботи з документами та таблицями у Google Документах або Таблицях, що значно підвищує продуктивність та ефективність виконання завдань.

У сфері хмарних рішень для підвищення продуктивності Google Workspace вирізняється своєю універсальністю та інтеграційними можливостями. Важливим аспектом його адаптивності є можливість розробляти власні надбудови, інструменти, призначені для розширення та покращення основних функціональних можливостей додатків Google Workspace [6].

Доповнення є спеціалізованими програмами, які інтегруються з додатками Google Workspace для розширення їх функціональності. В основному, вони розробляються за допомогою Google Apps Script і призначені для додавання нових можливостей та автоматизації завдань у Google Workspace. Ці доповнення спрощують робочий процес, забезпечуючи доступ до сторонніх сервісів або допомагаючи в обробці та організації даних.

За допомогою доповнень можна створювати індивідуальні інтерфейси, які інтегруються безпосередньо з додатками Google Workspace, надаючи користувачам зручний доступ до інформації та керування. Вони дозволяють автоматизувати рутинні завдання та оптимізувати робочі процеси у Google

Workspace, включаючи автоматичне виконання дій, форматування даних і обмін інформацією між програмами [6].

Процес розробки доповнень для Google Workspace складається з кількох етапів, починаючи з визначення потреби або проблеми, яку буде вирішувати доповнення. Коли мета зрозуміла, розробники використовують Apps Script для створення функціоналу доповнення.

Google Apps Script дозволяє створювати два основні типи доповнень: доповнення для Google Workspace та доповнення для Google редактора. Доповнення для Google Workspace розширюють можливості декількох програм Google Workspace одночасно, дозволяючи користувачам налаштовувати індивідуальні домашні сторінки для кожної програми. Вони забезпечують засоби для створення уніфікованих інтерфейсів та інтеграції зі сторонніми сервісами [6].

З іншого боку, доповнення для Google редактора збільшують функціональність окремого додатка Google Редактора. Вони дозволяють автоматизувати обробку файлів та створювати користувацькі інтерфейси за допомогою HTML та CSS. Основною перевагою розробки користувацьких доповнень є можливість адаптувати додатки Google Workspace до унікальних бізнес-процесів і робочих процесів. Таке налаштування може призвести до:

– Підвищення продуктивності за рахунок автоматизації повторюваних завдань.

– Підвищення точності даних завдяки спеціальним інструментам перевірки та аналізу.

– Спрощена співпраця завдяки інтеграції додаткових функцій комунікації та управління проектами.

– Більшої гнучкості завдяки зв'язку інших платформам та сервісів з Google Workspace [6].

Створення власних надбудов для Google Workspace – це хороший спосіб розширити функціональність і без того надійного набору інструментів для підвищення продуктивності. Використовуючи Apps Script, розробники можуть розробляти індивідуальні рішення, які відповідають конкретним робочим потребам, підвищуючи ефективність і продуктивність бізнесу будь-якого

розміру. Оскільки все більше організацій прагнуть максимально ефективно використовувати Google Workspace, попит на спеціалізовані доповнення буде зростати, що відкриває широкі можливості для розробників і технічних фахівців.

Google Workspace Marketplace – це магазин додатків та доповнень, який надає доступ до широкого спектру рішень для розширення можливостей платформи Google Workspace. У цьому маркетплейсі користувачі можуть знайти та встановити різноманітні додатки, що інтегруються з різними продуктами Google, такими як Gmail, Google Drive, Google Календар та інші.

Навігація в Google Workspace Marketplace інтуїтивно зрозуміла та проста. Його зручний інтерфейс демонструє пропоновані та рекомендовані програми для Google Workspace прямо на головній сторінці. Але якщо потрібно щось конкретне, у верхній частині сторінки також є панель пошуку [7]. На рисунку 2.3 зображено головну сторінку Google Workspace Marketplace.

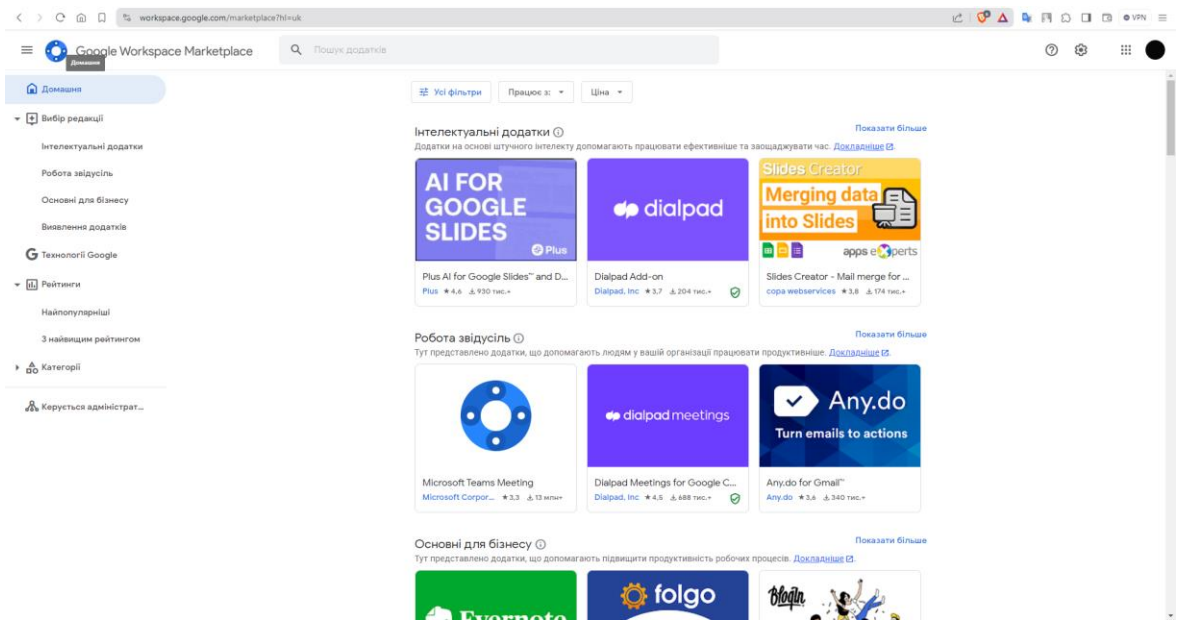


Рисунок 2.3 – Google Workspace Marketplace

Окрім панелі пошуку, є також різні категорії, які можна переглядати. Ці категорії варіюються від продуктивності та спілкування до продажів і освіти, що полегшує пошук програм, які відповідають потребам [7].

Коли ви клацнете програму, ви перейдете до детального перегляду програми. Ця сторінка містить вичерпний опис програми, її функції, оцінки користувачів і відгуки. Ця інформація має вирішальне значення, щоб допомогти

вам прийняти обґрунтоване рішення про те, чи підходить програма для ваших потреб [7].

Встановлювати додатки з Google Workspace Marketplace нескладно. Коли ви знайдете програму, яка вас цікавить, потрібно просто натиснути не неї, щоб переглянути докладнішу інформацію. На цій сторінці також можна встановити програму, натиснувши кнопку «Встановити». Залежно від програми може знадобитися надати їй певні дозволи на доступ до певної інформації з облікового запису Google Workspace [7].

Для розробників Google Workspace Marketplace відкриває можливості для просування своїх продуктів серед широкого кола користувачів Google Workspace та отримання додаткових можливостей для монетизації своїх рішень. Розробники можуть публікувати свої додатки у Marketplace, надаючи користувачам доступ до своїх рішень та отримуючи заробіток від їх використання.

Користувачі Google Workspace Marketplace мають можливість надавати відгуки, рецензії та оцінки для додатків, які вони використовують. Це забезпечує можливість отримати важливу інформацію про якість та корисність додатків перед їх встановленням.

2.3 Опис розробки веб-застосунків з використанням Google Apps Script

Якщо ви розробите користувацький інтерфейс для вашого коду, ви зможете перетворити його на веб-застосунок. Це може бути більш практичним, оскільки користувачі зможуть використовувати його безпосередньо зі своїх інтернет-браузерів. Доступ до цього застосунку здійснюється через спеціалізовану URL-адресу, розміщену в Google Cloud. Але для цього потрібно виконати деякі вимоги.

Згідно з цими критеріями, скрипт може бути відкритим для веб-доступу, якщо він містить функцію doGet() або doPost(), яка віддає об’єкт HtmlOutput або TextOutput. Це дозволяє розробляти веб-застосунки, доступні з будь-якого пристрою з інтернет-браузером, інтегруючи з іншими сервісами Google або надаючи користувачам спеціальні можливості.

Функції doGet() і doPost() у Google Apps Script використовуються для обробки HTTP-запитів GET і POST відповідно. Коли користувач або програма надсилає GET-запит до застосунку, Google Apps Script автоматично викликає функцію doGet(). Ця функція є точкою входу в застосунок і відповідає за завантаження інтерфейсу веб-застосунку. Вона також повідомляє сервер про початок виконання скрипту [11].

Для POST-запитів викликається функція doPost(). В обох випадках функція приймає аргумент, що містить інформацію про параметри запиту. Ці аргументи можна використовувати для отримання і обробки даних, надісланих користувачами через форми або інші методи.

Таким чином, функції doGet() і doPost() відіграють ключову роль в обробці HTTP-запитів у веб-застосунках, створених на основі Google Apps Script. Вони дозволяють застосунку взаємодіяти з користувачами та обробляти дані, надіслані на сервер.

Проектування є першим кроком у розробці веб-застосунків за допомогою Google Apps Script. Воно полягає у визначенні мети проекту, завдань або функцій, які має виконувати застосунок, а також виборі конкретних сервісів Google, таких як Google Sheets, Google Docs, Google Calendar тощо.

Після завершення проектування можна перейти до створення нового проекту в середовищі розробки Google Apps Script. Існує два способи зробити це. Перший спосіб – перейти на сайт script.google.com і натиснути кнопку «New Project» у верхньому лівому куті. Другий спосіб – створити автономний скрипт безпосередньо в Google Drive, відкривши Google Drive, натиснувши кнопку «Create» у верхньому лівому куті, потім натиснувши «More» і вибравши «Google Apps Script».

Далі можна перейти до наступного етапу – розробки коду, який починається зі створення користувацького інтерфейсу. HTML-сервіс Google Apps Script дозволяє додавати HTML-код безпосередньо в скрипт, тому ви можете створювати складні інтерфейси, використовуючи HTML і JavaScript. Це підхід, який дозволяє вам зв’язувати HTML-елементи зі скриптовим кодом для реалізації спеціальних функцій.

Загалом, Google Apps Script використовує файли з розширенням .gs (Google Script) для написання коду. Ви також можете створювати звичайні HTML-файли, використовуючи елементи, що зазвичай використовуються у сучасних веб-сторінках, такі як jQuery і CSS. Для створення HTML-файлу в Google Apps Script натисніть значок плюса поруч з елементом файлу і виберіть HTML. На рисунку 2.4 зображено створення файлу HTML.

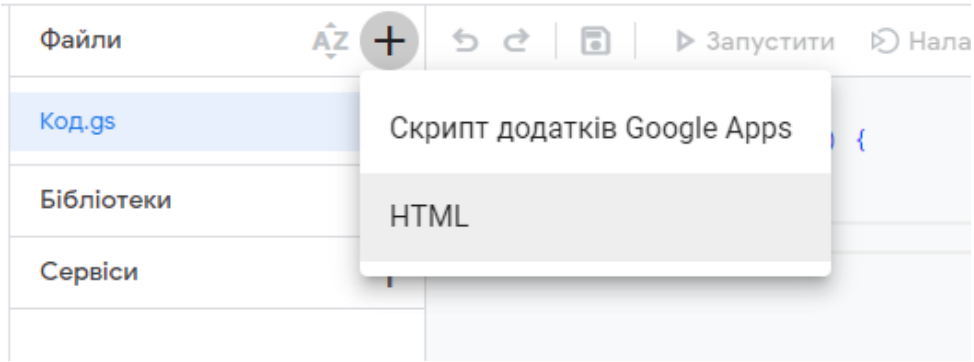


Рисунок 2.4 – Створення файлу HTML

Потім дайте ім'я створюваному файлу. З’явиться новий файл з розширенням .html. Цей файл вже містить основний код структури HTML-сторінки, включаючи заголовок і текстову частину. На рисунку 2.5 зображено створений HTML файл.

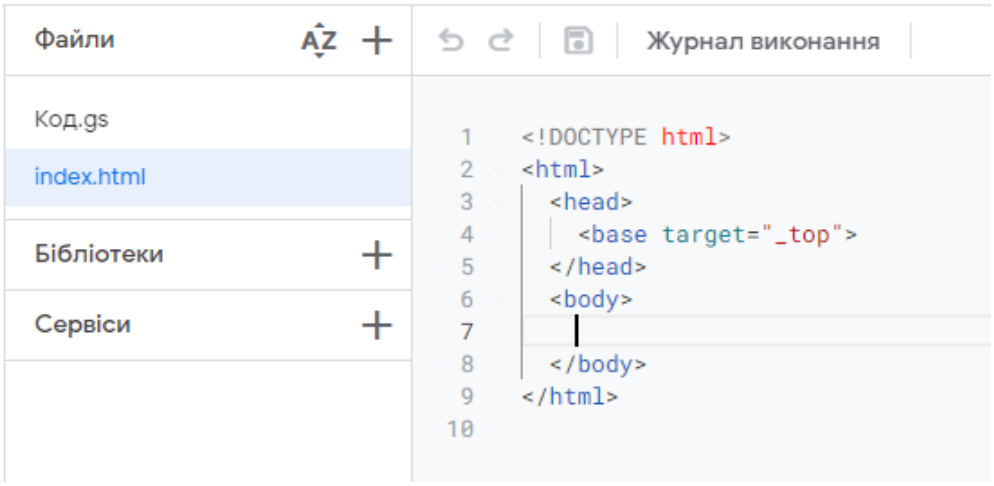


Рисунок 2.5 – Створений файл HTML

Google Apps Script використовує мову програмування JavaScript, хоча не найновішу її версію. Це означає, що ви можете писати JavaScript-код для

визначення функціональності створених застосунків. Ви можете також використовувати вбудовані сервіси Google Apps Script. Наприклад, можна використовувати сервіс SpreadsheetApp для роботи з Google Sheets, сервіс DocumentApp для роботи з Google Docs, і сервіс CalendarApp для роботи з Google Calendar. Ви можете використовувати також користувацькі методи і бібліотеки для спрощення і підвищення ефективності процесу розробки.

Google Apps Script надає розробникам інструменти та функції для тестування і відлагодження коду. У цьому середовищі можна відстежувати зміни, використовувати контрольні точки, перевіряти значення змінних і багато іншого. Розробники можуть покроково виконувати код, контролювати і відслідковувати виконання програми, перевіряти її поведінку на кожному етапі.

Google також перевіряє код на синтаксичні помилки кожного разу, коли скрипт зберігається. Редактор скриптів виділяє помилки червоним кольором у верхній частині сторінки і показує номер рядка, де сталася помилка. Однак важливо зазначити, що цей тест не виявляє всі можливі помилки, такі як відсутність оголошень змінних або неможливість доступу до веб-сервісів.

Скрипти Google Apps використовують оператори try/catch для обробки помилок під час виконання застосунку. Ці оператори дозволяють розробникам елегантно обробляти помилки і отримувати детальну інформацію про них.

Після збереження змін у редакторі Google Apps Script пропонує можливість попереднього перегляду змін у застосунку, використовуючи опцію «Test as add-on». Це дозволяє вам перевіряти внесені зміни перед тим, як їх впроваджувати. На рисунку 2.6 зображено введення в дію.

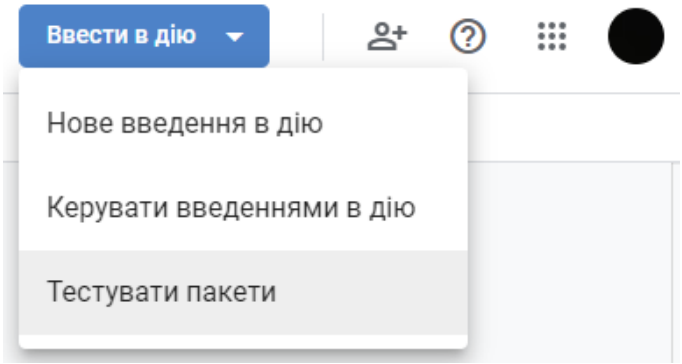


Рисунок 2.6 – Введення в дію

Далі відкриється нове вікно з URL-адресою веб-застосунку, яка закінчується на /dev. Це вказує на те, що ви переглядаєте версію для розробки, яка відображає поточний стан коду в редакторі. Щоразу, коли ви зберігаєте зміни, ця версія автоматично оновлюється. Щоб отримати тестову URL-адресу, необхідно запустити веб-застосунок хоча б один раз. Після цього ви отримаєте тестову URL-адресу, за якою можна перевіряти роботу застосунку перед його офіційною публікацією. На рисунку 2.7 зображено тестування пакету.

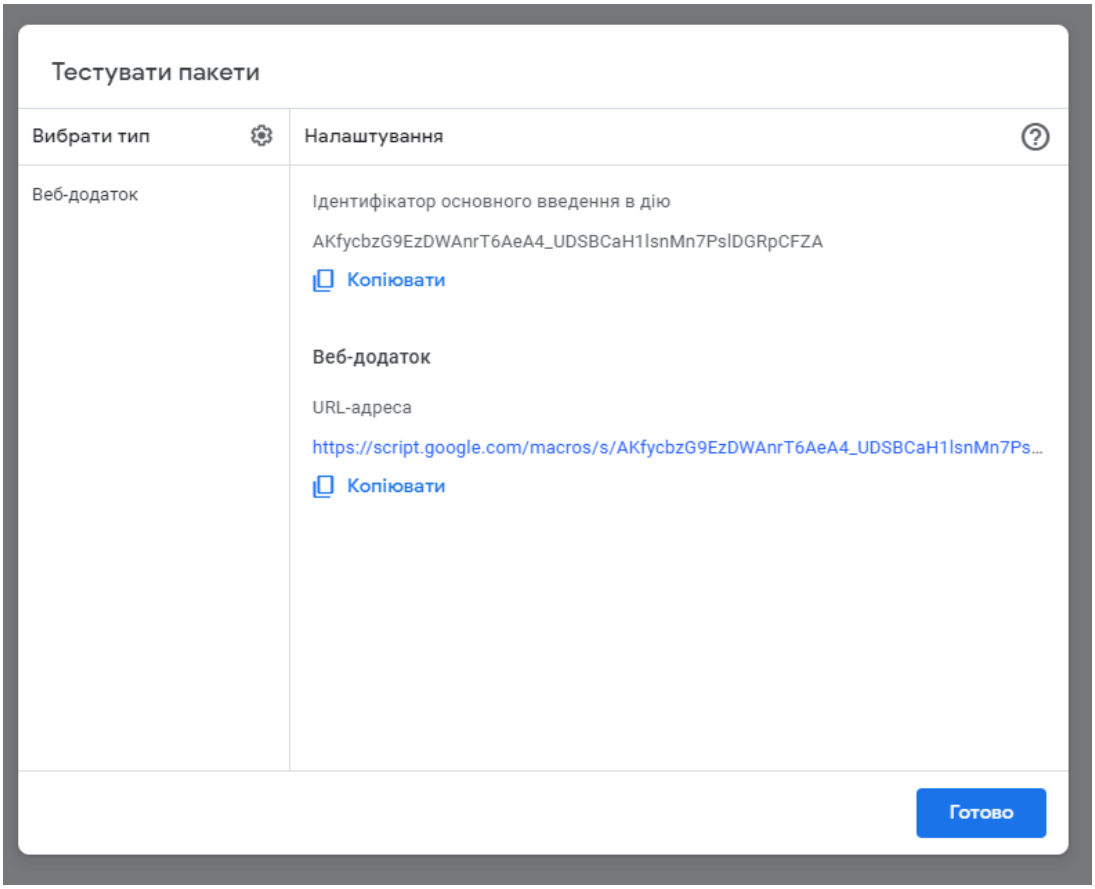


Рисунок 2.7 – Тестування пакету

Для публікації застосунку натисніть кнопку «Publish application» і виберіть опцію «New publication». Відкриється вікно «New deployment», де потрібно вибрати тип розповсюдження «Web application». Після написання скрипта його можна запускати вручну, автоматично за розкладом або реактивно на певні події. На рисунку 2.8 зображено типи введення в дію.

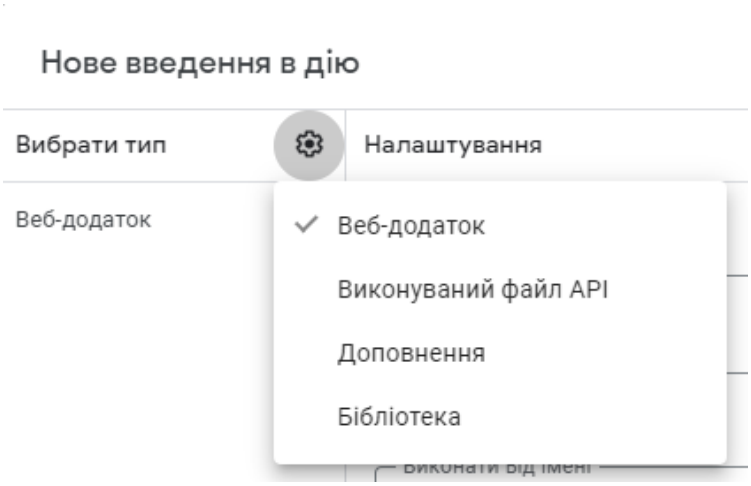


Рисунок 2.8 – Вибір типу введення в дію

У конфігураційному списку ви побачите кілька важливих параметрів. У полі «Description» можна вказати інформацію про поточну версію застосунку. Опція «Run as» дозволяє вибрати, від чийого імені буде запускатися веб-застосунок: розробника або користувача, який використовуватиме застосунок. На рисунку 2.9 зображено поле налаштувань.

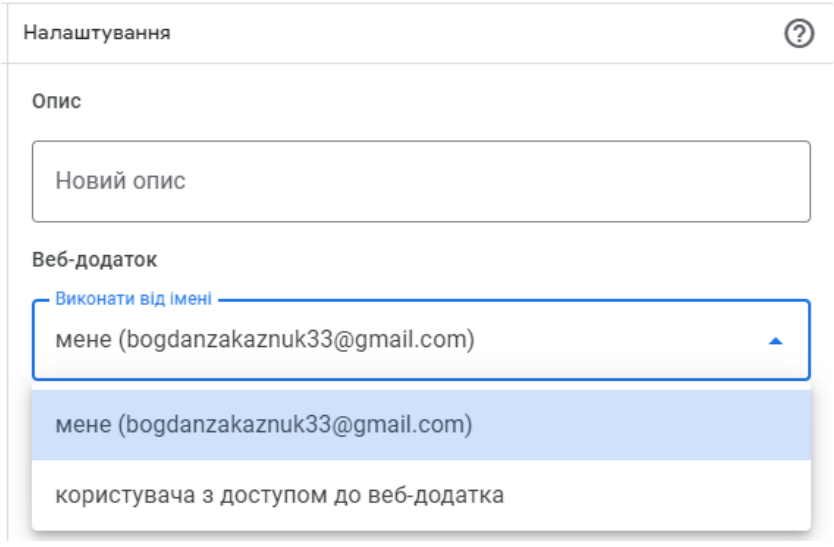


Рисунок 2.9 – Поле налаштувань

Якщо застосунок працює від імені розробника, він матиме доступ до даних розробника. Це зручно для тестування. Проте, якщо мета полягає в створенні застосунка для інших користувачів, краще налаштувати запуск від їх імені, щоб забезпечити доступ до їх особистих даних та забезпечити коректну роботу застосунку.

Запуск сценарію від імені автора може бути корисним для обмеження спільного доступу до електронної таблиці і контролю того, які дані бачить інтерфейс користувача. Це дає можливість користувачам вводити дані в таблицю, але з деякими обмеженнями. Однак цей підхід має обмеження, такі як відсутність прямого доступу до облікового запису користувача з вбудованих класів. Наприклад, якщо веб-застосунку потрібно отримати доступ до Google Контактів користувача, це не спрацює.

Наступним кроком є визначення того, хто може користуватися додатком. Дана функція має відношення тільки до веб-застосунку, а не скрипту. На рисунку 2.10 зображено налаштування доступу до застосунку.

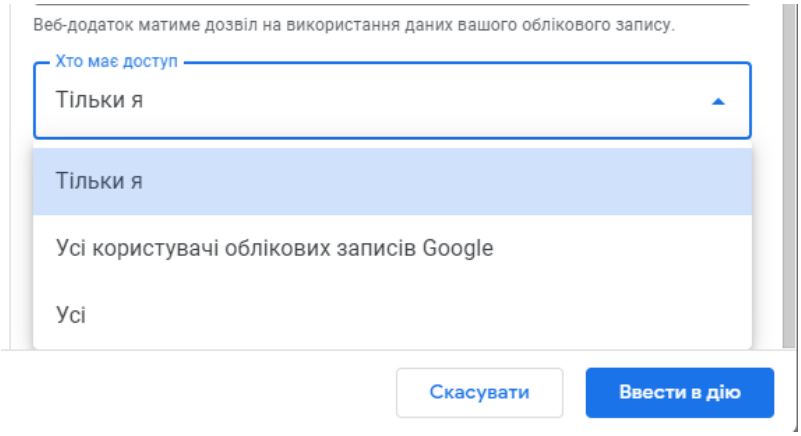


Рисунок 2.10 – Налаштування доступу

Якщо обрати «Усі» це дозволить користувачам працювати з веб-застосунком без потреби входу в обліковий запис Google.

У будь-який час можна змінити налаштування опцій «Виконати від імені» та «Хто має доступ».

Після проведення потрібних налаштувань веб-застосунку, потрібно натиснути на кнопку «Ввести в дію». Далі з'явиться нове вікно. На рисунку 2.11 зображено нове введення в дію.

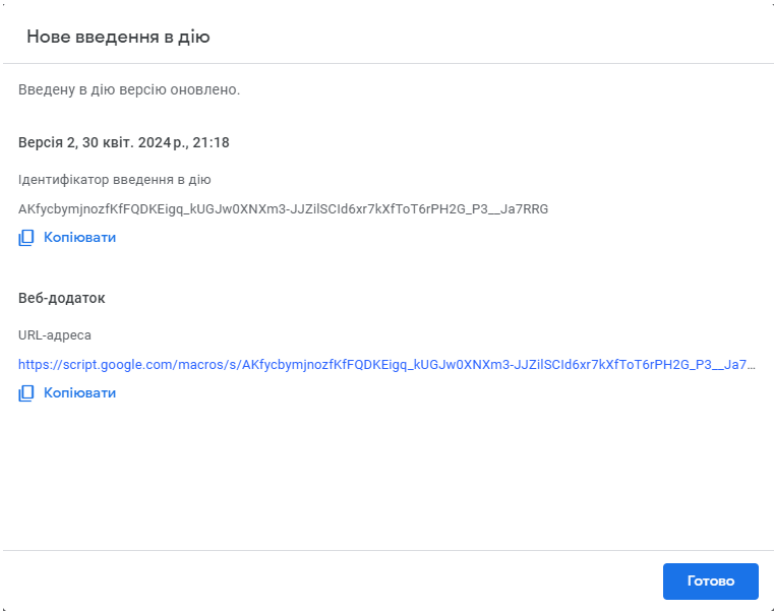


Рисунок 2.11 – Нове введення в дію

У цьому полі знаходиться поточна URL-адреса веб-застосунку, за якою можна перейти і переглянути поточну версію встановленого веб-застосунку. Важливо зауважити, що після публікації користувачі веб-застосунку не матимуть доступу до його коду, це можна контролювати налаштуваннями спільного доступу в редакторі сценаріїв або контейнері, якщо веб-застосунок публікується з нього. Це гарантує, що код залишається захищеним.

Також після розгортання веб-програми важливо підтримувати її та регулярно проводити оновлення відповідно до потреб користувача. Це може включати виправлення помилок, покращення функціональності, додавання нових функцій і захист веб-програми.

2.4 Висновок до другого розділу

У другому розділі кваліфікаційної роботи було вибрано технологію для реалізації веб-застосунку календаря подій. Основний вибір упав на Google Apps Script, що дозволяє швидко та ефективно створювати інтегровані веб-застосунки з використанням різноманітних сервісів Google. Google Apps Script є потужним інструментом для автоматизації робочих процесів, забезпечуючи тісну інтеграцію з Google Workspace. Це дозволяє створювати додатки, які можуть

взаємодіяти з Google Sheets, Google Calendar та іншими сервісами, що значно спрощує управління даними та подіями.

Було детально оглянуто можливості Google Apps Script, зокрема його здатність автоматизувати робочі процеси за допомогою скриптів та налаштування тригерів, які можуть запускати певні дії в заданий час або за певних умов. Це особливо корисно для створення веб-застосунків, які потребують регулярного оновлення даних або виконання автоматичних дій без втручання користувача.

Описано процес розробки веб-застосунків з використанням Google Apps Script. Створення скриптів, які визначають основні функції застосунку та їх налаштування. Використання Google Sheets і Google Calendar як основних інструментів для зберігання та управління даними було розглянуто як ефективне рішення, що забезпечує легкий доступ до даних та їх синхронізацію між різними сервісами.

Також розглянуто методи створення веб-інтерфейсів за допомогою Google Apps Script, що дозволяє користувачам взаємодіяти із застосунком через зручний та інтуїтивно зрозумілий інтерфейс. Використання HTML та CSS для створення користувацьких інтерфейсів у Google Apps Script забезпечує високу гнучкість у дизайні та функціональності застосунків. В результаті, було продемонстровано, як створення інтегрованих веб-застосунків з використанням Google Apps Script може значно покращити продуктивність та зручність управління завданнями і подіями.

РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА. СТВОРЕННЯ ВЕБ-ЗАСТОСУНКУ КАЛЕНДАРЯ ПОДІЙ З ВИКОРИСТАННЯМ GOOGLE APPS SCRIPT

У сучасному світі швидкого темпу ефективне управління часом і завданнями стає критично важливою навичкою для досягнення успіху, тому веб-застосунки для відстеження подій та задач набирають все більшої популярності.

Тому було вибрано розроблення веб-застосунку календаря подій з використанням Google Apps Script. Він показує перспективи та гнучкість Google Apps Script для розроблення власних веб-застосунків, які відповідають конкретним потребам користувачів.

Мета даного веб-застосунку – забезпечити корисний інструмент, який дозволить ефективно організовувати та управляти подіями та завданнями, що посприяє покращенню продуктивності та досягненню поставлених цілей. Завдяки функціоналу веб-застосунку, користувачі можуть мати можливість створювати нові завдання та встановлювати терміни їх виконання.

Одна з переваг веб-застосунку календаря подій це його можливості та гнучкість. Даний застосунок дає можливість користувачам адаптувати його під свої різноманітні потреби. Використовуючи Google Apps Script, можна в подальшому розширювати його можливості.

Для створення календаря подій було використано Google Календар, як основну платформу, оскільки вона надає зручний та масштабований інтерфейс для організації та зберігання подій. Завдяки інтеграції з Google Календарем, календар подій дозволяє зручно зберігати, оновлювати та відстежувати інформацію про події, забезпечуючи користувачам легкий доступ до актуальної інформації та можливість спільної роботи над проектами.

Також використано Google Таблиці для зберігання додаткових даних про події, що дозволяє зберігати додаткову інформацію та зручно відслідковувати статуси. Інтеграція з Google Поштою також забезпечує можливість надсилання повідомлень щодо подій їх виконавцям, що допомагає підтримувати зв'язок та забезпечує доставку інформації.

3.1 Створення таблиці та календаря

Спочатку було створено таблицю та календарю Для цього потрібно увійти в обліковий запис Google та зайти на Google Диск. Далі в Диску було створено таблиці, які містять інформацію про події. Далі було створено три аркуші.

Перший лист «access» відповідає за збереження користувачів, які мають доступ до управління подіями, він містить їхні електроні пошти. На рисунку 3.1 зображено перший аркуш таблиці «Calendar of events».

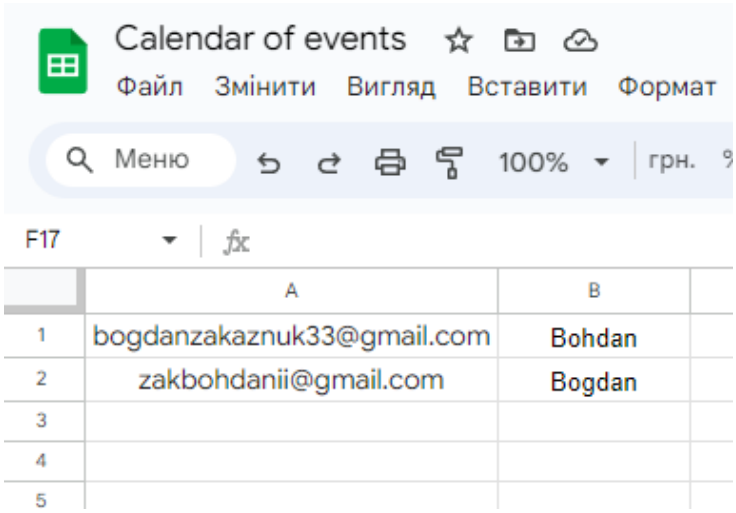


Рисунок 3.1 – Перший аркуш

Другий аркуш «tasks», він містить декілька колонок, в кожній з них є інформація про події: ID завдання, коли подія була створена, дедлайн, хто є виконавцем, деталі, статус виконання, хто створив подію, та ID події в календарі. На рисунку 3.2 зображено другий аркуш таблиці.

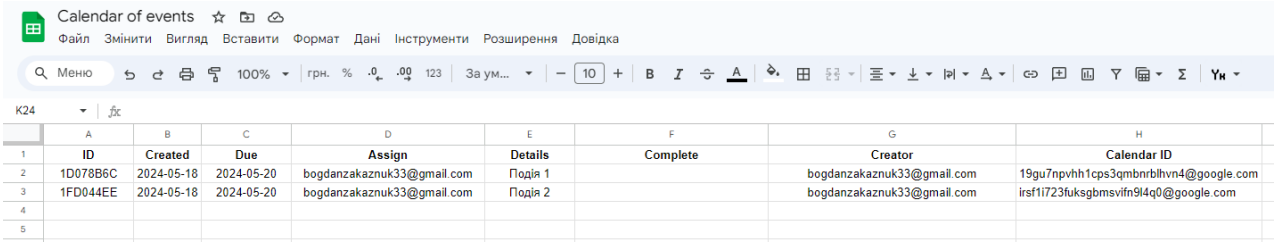
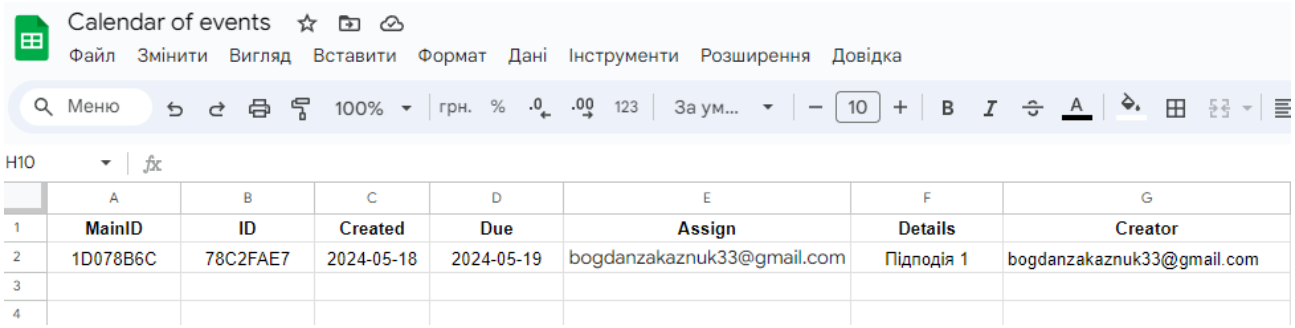


Рисунок 3.2 – Другий аркуш

Третій аркуш «subtasks», в ньому знаходиться інформація про підподії для подій, які містяться на аркуші «tasks», він має такі колонки: MainID – ID події до якої відносяться підподії, ID самої підподії, дедлайн, дата коли підподія була створена, її деталі, хто є виконавцем та автором підподії. На рисунку 3.3 зображено третій аркуш таблиці.



	A	B	C	D	E	F	G
1	MainID	ID	Created	Due	Assign	Details	Creator
2	1D078B6C	78C2FAE7	2024-05-18	2024-05-19	bogdanzakaznuk33@gmail.com	Підподія 1	bogdanzakaznuk33@gmail.com
3							
4							

Рисунок 3.3 – Третій аркуш

Далі було створено календар в якому будуть відображатися події. Для цього потрібно перейти в Google Calendar та створити новий календар з назвою «Calendar of events». Для його створення потрібно в полі «Інші календарі» натиснути на знак «+», «Створити новий календар» та задати параметри календаря. На рисунку 3.4 зображено створення календаря.

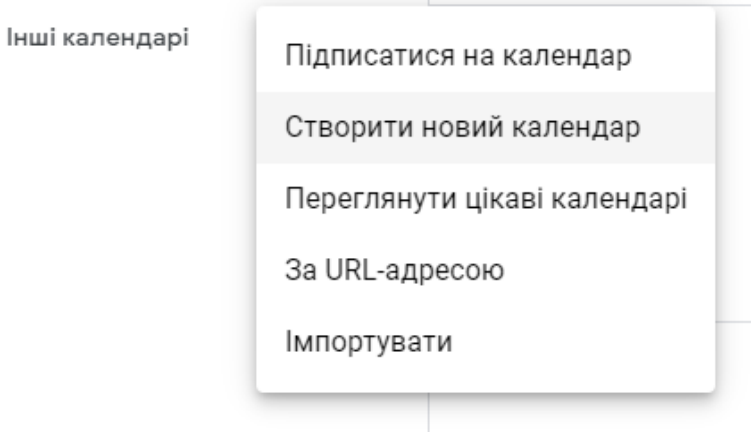


Рисунок 3.4 – Створення календаря

Після того як календар буде створено він буде відображатися зліва в полі «Мої календарі». На рисунку 3.5 зображено календарі користувача.

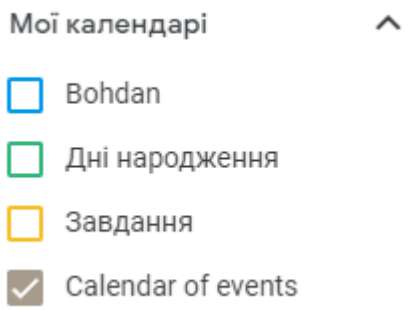


Рисунок 3.5 – Календарі

Таблиця та календар будуть зберігати інформацію про події, а також відображати її відповідно.

3.2 Створення функціоналу веб-застосунку

Після створення таблиці та календаря, наступним кроком є перехід до редактора скриптів для створення веб-застосунку. Файли user.html та edit.html відповідають за візуальний інтерфейс веб-застосунку, тоді як файл code.gs та calendar.gs містять код, що реалізує функціональність веб-додатку.

Файл code.gs включає код, який реалізує основні функції веб-додатку календаря подій, надаючи користувачам можливість перегляду, додавання та редагування подій та підподій (див. додаток А).

Константа «PROJECTSHEETID» зберігає ідентифікатор Google Sheet, де зберігаються дані для календаря подій. Цей ідентифікатор використовується для доступу до відповідного листа.

Функція «requestAccess» перевіряє, чи має поточний користувач доступ до редагування календаря подій. Вона відкриває лист із назвою «access» у вказаному Google Sheet, отримує всі дані з цього листа і перевіряє, чи є електронна адреса поточного користувача серед тих, хто має доступ.

«doGet» головна функція, яка обробляє GET-запити до веб-застосунку. Вона визначає, який HTML-шаблон (user або edit) завантажувати, залежно від параметрів запиту та доступу користувача. Функція також передає дані про користувача, основні події і підпідії у шаблон для подальшої візуалізації. У разі

виникнення помилки функція повертає JSON-відповідь з інформацією про помилку.

Функція «listSub» отримує всі підподії з листа «subtasks» в Google Sheet, формує з них масив об'єктів та повертає цей масив у вигляді JSON-об'єкта. Кожен об'єкт містить інформацію про підподії: його ID, ID основної події, рядок у таблиці, дату створення, дедлайн, виконавця та деталі підподії. На рисунку 3.6 зображено функцію «listSub».

```
function listSub() {
    var sheet = SpreadsheetApp.openById(PROJECTSHEETID).getSheetByName('subtasks');
    var sheetdata = sheet.getDataRange().getValues();
    var data = sheetdata.slice(1);
    var ep = [];
    for (var i = 0; i < data.length; i++) {
        var com = (!isNaN(data[i][6]) || data[i][6] == '') ? data[i][6] : 100;
        var obj = {
            id: data[i][1],
            mainId: data[i][0],
            row: i + 2,
            created: data[i][2],
            due: data[i][3],
            assign: data[i][4],
            details: data[i][5],
        };
        ep.push(obj);
    }
    var output = {
        status: 'status',
        data: ep
    }
    Logger.log(output);
    return output;
}

function updateCompleteStatus() {
    var sheet = SpreadsheetApp.openById(PROJECTSHEETID).getSheetByName('tasks');
    var sheetdata = sheet.getDataRange().getValues();
    var today = new Date();

    for (var i = 1; i < sheetdata.length; i++) {
        var dueDate = new Date(sheetdata[i][2]);
        var complete = (dueDate < today) ? "TRUE" : "FALSE";
        sheet.getRange(i + 1, 6).setValue(complete);
    }
}
```

Рисунок 3.6 – Функція «listSub»

Функція «listMain» отримує всі основні завдання з листа «tasks» в Google Sheet, формує з них масив об'єктів та повертає цей масив у вигляді JSON-об'єкта. Кожен об'єкт містить інформацію про основне завдання: його ID, рядок у таблиці, дату створення, дедлайн, виконавця, деталі та статус виконання. Дані записуються у «sheetdata», використовуючи метод getRange().getValues(), далі вони обробляються, створюючи об'єкти для підподії та додавання їх до масиву ep. Якщо подія позначена в таблиці як виконана, то вона не буде включена в

список завдань, який відображає веб-застосунок. Функція «updateCompleteStatus» перевіряє кожне завдання і оновлює стовпець Complete у Google Sheets.

Функції «updateSubTask» та «updateTask» оновлюють дані підподії та події в листі «subtasks» та «tasks». Вони знаходять підподії та події за їх ID, оновлюють інформацію про дедлайн, виконавця та деталі, і повертає статус операції.

Функція «addSubTask» додає нову підподію в лист «subtasks». Вона генерує унікальний ID для підподії, додає рядок у таблицю з інформацією про підподію та повертає об'єкт з даними про підподію. Функція «addTask» додає нову основну подію в лист «tasks». Вона генерує унікальний ID для події, додає рядок у таблицю з інформацією про подію та повертає об'єкт з даними про неї.

Нижче на рисунку 3.7 зображено функції «formatDate», «getRandom» та «createCalendarEvent».

```
function formatDate(date) {
    var d = new Date(date),
        month = '' + (d.getMonth() + 1),
        day = '' + d.getDate(),
        year = d.getFullYear();
    if (month.length < 2)
        month = '0' + month;
    if (day.length < 2)
        day = '0' + day;
    return [year, month, day].join('-');
}

function getRandom() {
    const uuid = Utilities.getUuid();
    const shortenedId = uuid.slice(0, 8);
    return shortenedId;
}

function createCalendarEvent(data) {
    var calendarId = "065182d29c6b49ab789d64b553337a0c6b5de2b43e28591d84872daef4dea285@group.calendar.google.com";
    var calendar = CalendarApp.getCalendarById(calendarId);
    var startDate = new Date();
    var endDate = new Date(data.due);

    if (startDate >= endDate) {
        Logger.log("Помилка: Дата початку події повинна бути раніше дати завершення.");
        return;
    }

    var event = calendar.createAllDayEvent(data.details, startDate, endDate);
    event.addGuest(data.assign);
    var newId = event.getId();

    var sheet = SpreadsheetApp.openById(PROJECTSHEETID).getSheetByName('tasks');
    var rowIndex = sheet.getLastRow();
    sheet.getRange(rowIndex, 8).setValue(newId);
    Logger.log(newId);

    sendEmailNotification(email, task, startDate, endDate);
}
```

Рисунок 3.7 – Функції «formatDate», «getRandom» та «createCalendarEvent»

Функція «deleteSubTask» видаляє підподію з листа «subtasks» за його ID. Вона знаходить підподію, видаляє відповідний рядок і повертає статус операції.

Функція «formatDate» форматуватиме дату у форматі 'YYYY-MM-DD'. Функція «getRandom» генерує випадковий ID з 8 символів, використовуючи UUID.

Функція «createCalendarEvent» додає нову подію до Google Календаря з використанням наданих даних. Вона отримує календар за його ідентифікатором та створює подію з датами початку та кінця. Якщо дата початку дорівнює або перевищує дату завершення, функція завершує виконання з повідомленням про помилку. Подія створюється як «весь день», їй додається гість, а новий ідентифікатор події зберігається в таблиці Google Sheets у відповідному рядку. Після створення події відправляється сповіщення електронною поштою з деталями події.

Ці функції забезпечують основну функціональність для керування календарем подій, дозволяючи користувачам додавати, редагувати, переглядати та видаляти події та підподії.

За створення подій в Google Calendar відповідає файл calendar.gs (див. додаток Б).

Функція «createEventsFromSpreadsheet» зчитує дані з таблиці Google Sheets і створює або оновлює події в Google Calendar. Вона спочатку отримує доступ до календаря за його ідентифікатором та таблиці з назвою «tasks». Зчитує всі дані з таблиці у змінну «schedule», а потім видаляє перший рядок, який, ймовірно, містить заголовки стовпців. Далі, для кожного запису в таблиці, функція перевіряє наявність ідентифікатора події. Якщо ідентифікатор пустий, викликає «createNewEvent» для створення нової події, в іншому випадку викликає «updateExistingEvent» для оновлення існуючої події.

Функція «createNewEvent» створює нову подію в Google Calendar. Вона приймає запис з таблиці, об'єкт календаря, саму таблицю та список записів. Спочатку, вона створює об'єкти дат початку та завершення події, перевіряючи, чи дата початку не пізніше дати завершення. Якщо перевірка пройшла, створюється подія за допомогою методу «createAllDayEvent», вказуючи назву події, дату початку та кінця. Після створення події додається гість (email), ідентифікатор події зберігається у відповідний стовпець таблиці, а також

надсилається повідомлення на email гостя за допомогою функції «sendEmailNotification».

Функція «updateExistingEvent» оновлює існуючу подію в Google Calendar. Вона приймає запис з таблиці, ідентифікатор події та об'єкт календаря. Спочатку знаходить подію за її ідентифікатором, якщо подія не знайдена, виводить повідомлення про це. Якщо подія знайдена, перевіряє, чи дата початку не пізніше дати завершення. Якщо перевірка пройшла, оновлює назву події та час початку і завершення. Якщо подія є цілодобовою, видаляє всіх гостей і додає нового гостя.

Функція «sendEmailNotification» надсилає повідомлення на email гостя події. Вона приймає email, назву події, дату початку та дату завершення події. Створює тему повідомлення та тіло листа, де вказуються деталі події та дати. Використовує «GmailApp.sendEmail» для відправлення листа з вказаними параметрами. На рисунку 3.8 зображено функції «updateExistingEvent» та «sendEmailNotification».

```
function updateExistingEvent(entry, id, calendar) {
  let event = CalendarApp.getEventById(id);

  if (!event) {
    Logger.log("Подія з ID: " + id + " не знайдена");
    return;
  }

  Logger.log(event);
  let startDate = new Date(entry[1]);
  let endDate = new Date(entry[2]);

  if (startDate >= endDate) {
    Logger.log("Помилка: Дата початку події повинна бути раніше дати завершення.");
    return;
  }

  event.setTitle(entry[4]);
  event.setTime(startDate, endDate);

  if (event.isAllDayEvent()) {
    event.removeAllGuests();
    event.addGuest(entry[3]);
  }
}

function sendEmailNotification(email, task, startDate, endDate) {
  let subject = "Повідомлення про подію";
  let message = "Шановний(-а) " + email + "\n\n" +
    "Це повідомлення щодо події: " + task + "\n" +
    "Дата початку: " + startDate + "\n" +
    "Дата завершення: " + endDate + "\n" +
    "Успіхів!\n\n" +
    "З повагою,\n" +
    "Calendar of events";
  GmailApp.sendEmail(email, subject, message);
}
```

Рисунок 3.8 – Функції «updateExistingEvent» та «sendEmailNotification»

На рисунку 3.9 зображено повідомлення про подію яке відправляється на електрону пошту.

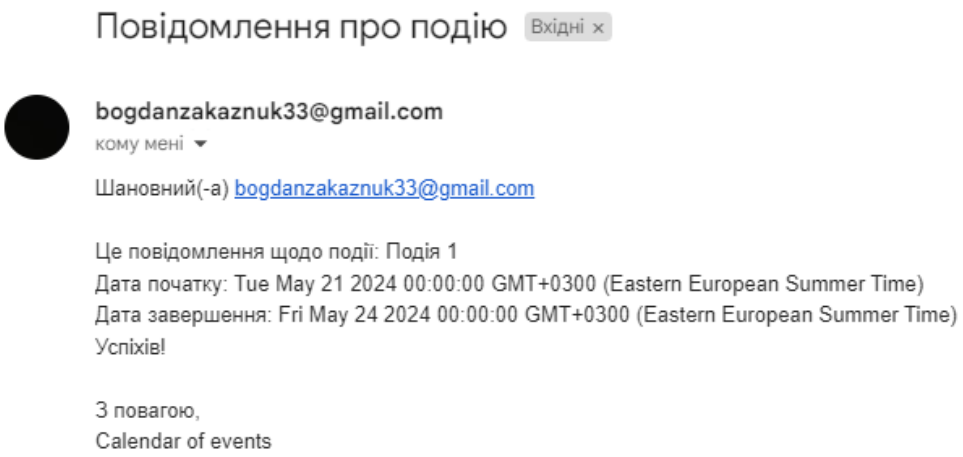


Рисунок 3.9 – Повідомлення про подію

На рисунку 3.10 зображено календар «Calendar of events» та як в ньому відображаються події.

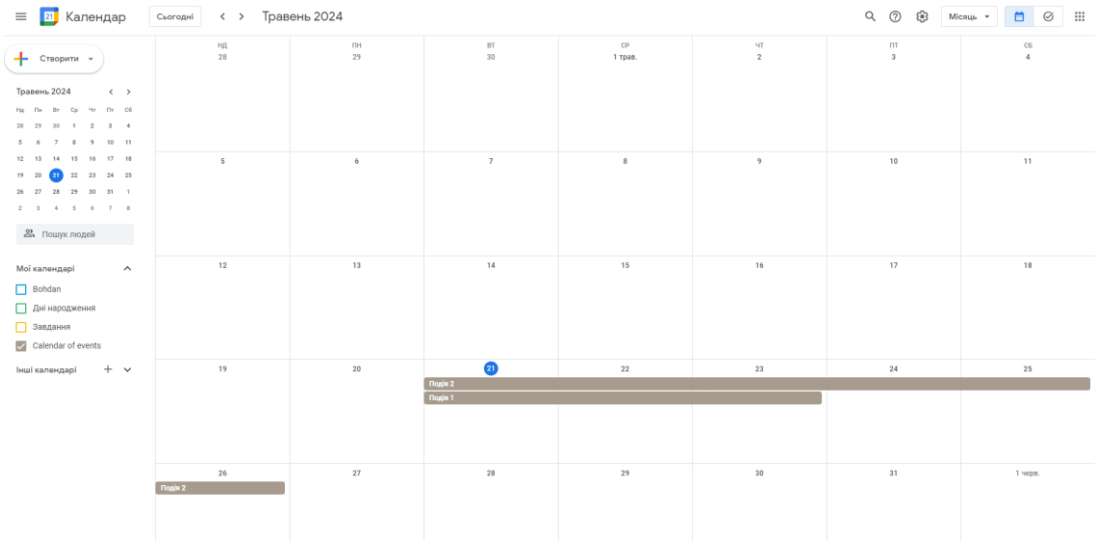


Рисунок 3.10 – Календар «Calendar of events»

Завдяки даним функціям користувачі зможуть взаємодіяти з календарем подій, керувати основними завданнями та підподіями, а також отримувати сповіщення про нові події в календарі.

3.3 Створення інтерфейсу користувача

За зовнішній вигляд веб-застосунку відповідають файли user.html (див. додаток В) та edit.html (див. додаток Г). Для розробки інтерфейсу користувача було використано Bootstrap. Це популярний фреймворк для розробки адаптивних веб-сайтів та веб-застосунків, що надає готові компоненти для створення дизайну інтерфейсу користувача.

Код user.html створює веб-сторінку для відображення календаря подій з використанням HTML, CSS і JavaScript. Він використовує фреймворк Bootstrap для стилізації та бібліотеки Moment.js і jQuery для роботи з датами та маніпуляцією DOM. У заголовку сторінки налаштовується стиль, а в тілі сторінки відображається список подій і підподій з даних, що надходять із змінної data. Події відображаються у вигляді сповіщень із відповідними датами та відповідальними особами, а підподії додаються як елементи списку всередині кожної події. На рисунку 3.11 зображено головну сторінку веб-застосунку.

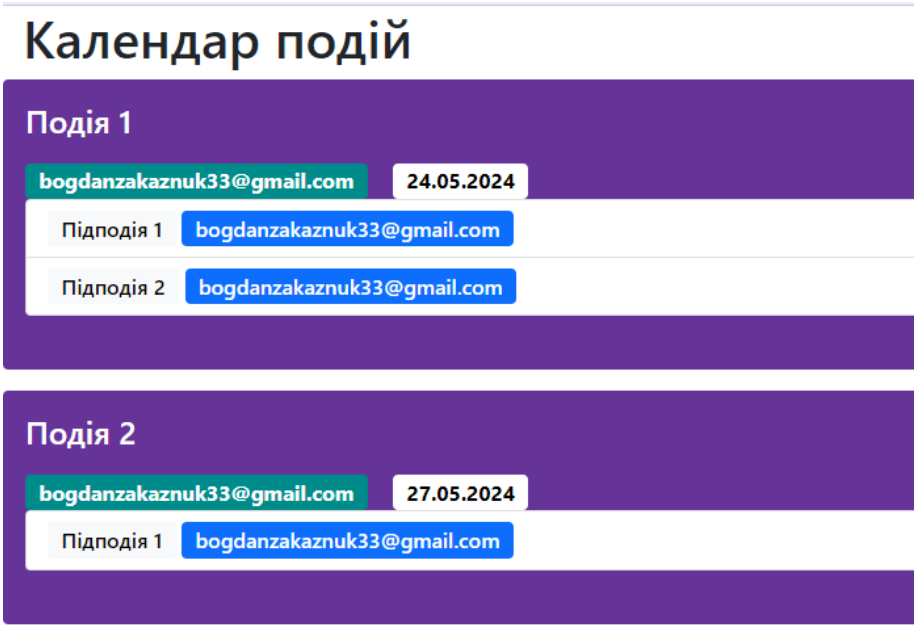


Рисунок 3.11 – Головна сторінка

Код edit.html створює веб-сторінку для управління подіями і підподіями. На сторінці є форма для додавання нових подій та підподій, що зберігаються та відображаються у вигляді списку. Події та підподії можна редагувати та

видаляти, і кожна дія супроводжується сповіщенням. Вся інформація обробляється через Google Apps Script для синхронізації з серверною частиною. На рисунку 3.12 зображено сторінку управління подіями.

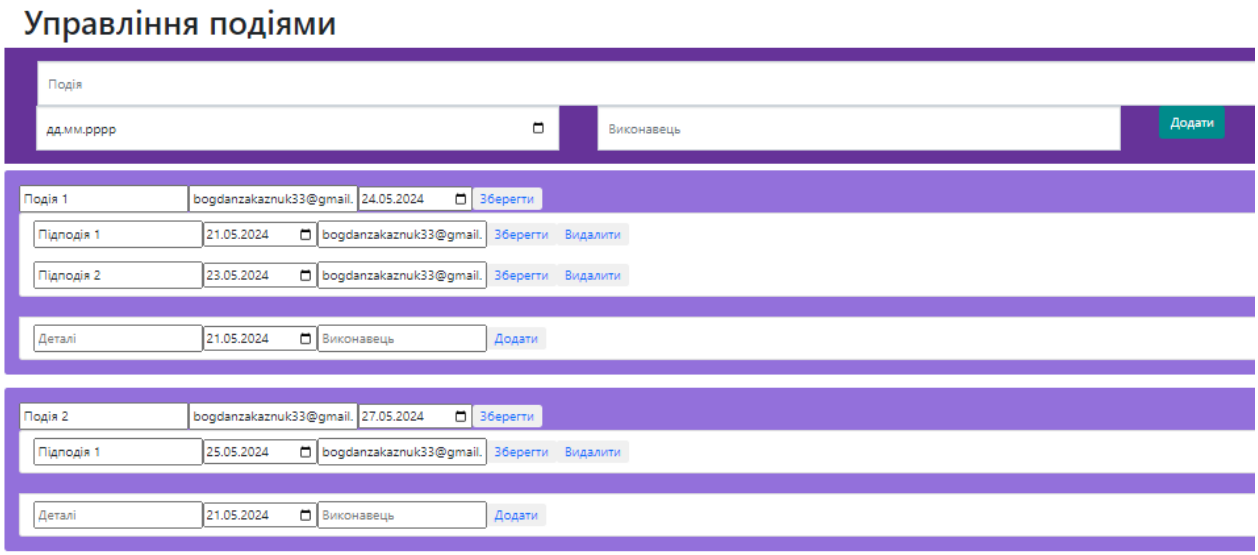


Рисунок 3.12 – Сторінка управління подіями

Після кожного внесення змін в події та підподії з’являється відповідні сповіщення. Це зображено на рисунку 3.13.

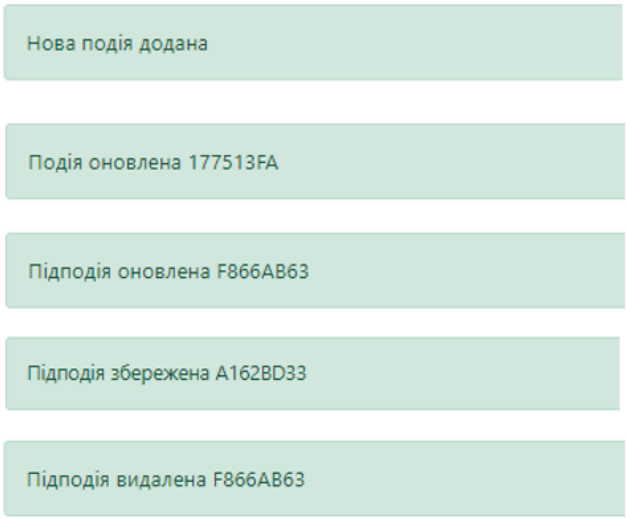


Рисунок 3.13 – Сповіщення про зміни

Для переходу на сторінку управління подіями, потрібно до адреси в пошуковому рядку додати «?p=edit».

3.4 Висновок до третього розділу

У третьому розділі кваліфікаційної роботи було розглянуто процес створення веб-застосунку для управління подіями з використанням Google Apps Script. Спершу було створено таблицю з трьома аркушами — «access», «tasks» та «subtasks», які слугують для зберігання даних про події та підподії. Також створено календар у Google Calendar під назвою «Calendar of events», який забезпечує візуалізацію та синхронізацію подій. Розглянуто створення функціоналу веб-застосунку, що охоплює написання скриптів code.gs та calendar.gs, які забезпечують основну логіку роботи з даними та інтеграцію з календарем. Далі описано розробку інтерфейсу користувача за допомогою HTML та CSS у файлах user.html та edit.html, що забезпечують зручне та інтуїтивне взаємодію користувача з системою.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Актуальність безпеки життєдіяльності людини

Безпека життєдіяльності (БЖД) є наукою, яка досліджує проблеми безпечного перебування людини в навколишньому середовищі під час різних видів діяльності. Діяльність, яка відрізняє людину від інших істот, є специфічною людською формою активності та необхідною умовою існування суспільства. Форми діяльності різноманітні, охоплюючи практичні, інтелектуальні та духовні процеси, які відбуваються в побуті, громадському, культурному, виробничому, науковому та інших сферах життя.

Усі люди, незалежно від віку, займаються діяльністю: діти, дорослі та люди похилого віку, тому питання безпеки діяльності стосується всього людства. З розвитком людства на основі сталого розвитку, який забезпечує його існування, актуальність цієї дисципліни зростає. Аксіома про потенційну небезпеку діяльності стверджує, що будь-яка діяльність є потенційно небезпечною. Це ж стосується і бездіяльності, оскільки бездіяльна людина ще більше залежить від діяльності інших.

Безпека людини є основною складовою сталого людського розвитку, який широко використовується ООН як основна характеристика гуманітарного прогресу суспільства. Сталий розвиток людства сприяє не лише економічному, а й соціальному та духовному зростанню, гуманізації національного менталітету та збагаченню позитивного загальнолюдського досвіду. Відмінною рисою сталого розвитку є відновлення природного та культурного довкілля, що не лише не знижує життєвий потенціал, а й підвищує соціальну відповідальність людей, гуманність у взаєминах, ставлення та реакції.

Основні завдання БЖД [35]:

– Аналіз та оцінка небезпек. Дослідження і аналіз різних видів небезпек, які можуть виникати у процесі діяльності людини. Це включає ідентифікацію потенційних ризиків, їх оцінку та класифікацію за ступенем небезпеки.

– Розробка і впровадження заходів безпеки. Створення системи захисних заходів і механізмів для запобігання небезпекам. Це може включати технічні рішення, організаційні заходи, інструкції та протоколи дій у надзвичайних ситуаціях.

– Навчання та підготовка населення. Проведення освітніх програм і тренінгів для різних груп населення з метою підвищення рівня обізнаності щодо безпеки життєдіяльності і навичок безпечної поведінки у різних умовах.

– Моніторинг і контроль. Постійний контроль і моніторинг стану безпеки у різних сферах діяльності, своєчасне виявлення та усунення порушень норм безпеки.

– Розробка нормативно-правової бази. Створення і вдосконалення законодавчої бази, що регулює питання безпеки життєдіяльності, а також контроль за її дотриманням.

Крім основних завдань, важливо враховувати такі аспекти, як психологічна безпека та соціальні фактори, що впливають на безпеку життєдіяльності. Наприклад, психологічна підготовка людей до дій у надзвичайних ситуаціях та розвиток культури безпеки в суспільстві є важливими елементами загальної стратегії забезпечення безпеки.

Актуальність питань безпеки життєдіяльності полягає у забезпеченні гармонійного розвитку людства та природи. Безпеку життєдіяльності можна охарактеризувати як багатопрофільну галузь знань про закони природозберігаючого формування техносфери планети та її збалансованого економічного і суспільного розвитку. Це передбачає інтеграцію різних аспектів діяльності людини з метою мінімізації ризиків і забезпечення здорового та безпечного середовища для всіх.

У сучасному світі значна увага приділяється також інформаційній безпеці, зокрема захисту особистих даних та конфіденційної інформації. Це пов'язано зі швидким розвитком інформаційних технологій і зростаючою кількістю кіберзагроз.

Для того, щоб дотримуватись принципів сталого розвитку, необхідно враховувати такі фактори, як екологічна стійкість, ефективне використання

ресурсів, соціальна справедливість і економічна стабільність. Це вимагає комплексного підходу до планування та впровадження заходів безпеки на всіх рівнях – від індивідуального до глобального. Таким чином, безпека життєдіяльності є комплексною наукою, яка об'єднує різні аспекти людської діяльності і спрямована на забезпечення безпечних умов для всіх членів суспільства. Вона включає технічні, соціальні, психологічні та правові заходи, що забезпечують гармонійний розвиток людства у взаємодії з природою.

4.2 Загальні вимоги безпеки з охорони праці для користувачів ПК

Основні вимоги безпеки з охорони праці для користувачів ПК є ключовими для створення безпечного та здорового робочого середовища. Як компанії, так і працівники повинні дотримуватись цих вимог, що дозволяє забезпечити безпеку, захист здоров'я та підвищити ефективність праці при роботі з комп'ютерами.

Роботодавець повинен дотримуватися чинних санітарних норм щодо освітлення, вимог до параметрів мікроклімату (температура, відносна вологість), рівнів вібрації, шуму та пожежної безпеки приміщення, а також враховувати вплив електромагнітного, ультрафіолетового та інфрачервоного випромінювання. Конкретні параметри цих санітарних норм викладені у ДСанПіН 3.3.2.007-98. Ці правила охоплюють умови та організацію праці при роботі з візуальними дисплейними терміналами (ВДТ) будь-яких типів, які виготовляються як в Україні, так і за кордоном, та використовуються в комп’ютерах колективного і персонального використання [36].

Наприклад, роботодавцям забороняється встановлювати комп’ютери у підвальних приміщеннях. Щоб запобігти аваріям та коротким замиканням, поруч із приміщеннями, де використовуються комп’ютери, не повинні проводитися роботи з надмірно вологими технологічними процесами. Приміщення повинні бути обладнані системами опалення, кондиціонування або вентиляції. У кожній кімнаті, де розташовані робочі місця з комп’ютерами, має бути природне і штучне освітлення. Вікна повинні мати регульовані жалюзі або штори для коригування рівня освітлення [37].

Для забезпечення безпечних умов праці приміщення повинні бути обладнані аптечками першої медичної допомоги, системами автоматичної пожежної сигналізації і вогнегасниками. У разі несправності комп'ютера, появи іскор, запаху гару чи ознак горіння необхідно негайно припинити роботу, вимкнути обладнання від електромережі та повідомити керівника або спеціаліста з ремонту.

У приміщенні з п'ятьма або більше комп'ютерами повинен бути встановлений службовий вимикач для повного відключення електроживлення в разі потреби. Роботодавець має забезпечити комфортні та безпечні умови праці для працівників, зокрема площа одного робочого місця повинна бути не менше 6 квадратних метрів. Якщо необхідно, суміжні робочі місця слід розділити перегородками висотою до 2 метрів [36].

Основним обладнанням робочого місця є ПК або ноутбук, монітор, клавіатура, робочий стіл і крісло. Розташування обладнання має забезпечувати оптимальні умови праці. На робочому столі можуть бути розміщені допоміжні пристрої (принтери, сканери), а також місця для зберігання документів, якщо це не обмежуватиме видимість екрану і не заважатиме працівнику. У разі надмірного шуму чи вібрації роботодавець повинен забезпечити працівників антивібраційними килимками. ПК повинен бути розташований на відстані не менше 1 метра від джерел тепла.

Монітор слід розташовувати так, щоб відстань від екрану до очей була 600-700 мм залежно від розміру екрана, при цьому екран має бути перпендикулярний до лінії зору. Екран монітора повинен розташовуватися так, щоб світло з вікна падало збоку, бажано зліва. Клавіатуру розміщують на робочому столі на відстані 100-300 мм від краю, кут нахилу залежить від побажань користувача (зазвичай 5-15°). Робоча поверхня столу повинна регулюватися у межах 680-800 мм, середня висота становить 725 мм. Робоче крісло має бути регульованим за висотою та забезпечувати підтримку спини і хребта [37].

Розетка біля ПК повинна бути в доступному місці для своєчасного відключення живлення у разі аварії. У приміщеннях з ПК необхідно щодня проводити вологе прибирання та регулярне провітрювання. Пил з екрану необхідно витирати не рідше одного разу на день.

Забороняється:

– Виконувати ремонт та технічне обслуговування комп’ютера безпосередньо на робочому місці працівника.

– Самостійно проводити ремонт або технічне налаштування комп’ютера без участі кваліфікованих спеціалістів.

– Зберігати на робочому місці зайві документи, деталі або інші предмети, які не потрібні для роботи.

– Використовувати монітори з розмитим зображенням або пошкодженим екраном.

– Працювати з матричним принтером без антивібраційного покриття або зі знятою кришкою.

До роботи з комп'ютером не допускаються особи, які не пройшли затверджений на підприємстві курс охорони праці.

Основні небезпечні та шкідливі фактори, які можуть впливати на користувача:

– Напруження зорових органів, що призводить до втоми і можливих побічних ефектів.

– Монотонність роботи та обробка великого обсягу інформації за короткий час.

– Довготривале статичне положення, що викликає застійні явища в організмі.

– Нервово-емоційне напруження під час роботи на ПК.

– Вплив різних видів випромінювання від моніторів на електронно-променевих трубках.

– Ризик ураження електричним струмом.

Згідно з ДСанПіН 3.3.2.007-98, особи повинні проходити медичні огляди: попередній під час прийому на роботу та періодичні протягом трудової діяльності. Обов’язковий медичний огляд раз на два роки включає перевірку терапевтом, невропатологом і офтальмологом, а за потреби – іншими спеціалістами [37].

Відповідно до п. 5 наказу Мінсоцполітики від 14.02.2018 р. № 207, роботодавець зобов'язаний організувати регламентовані перерви в роботі працівників за рахунок тривалості робочої зміни [38]:

Для технічних операторів – 15 хвилин кожні 2 години.

Для операторів комп'ютерного набору – 10 хвилин кожну годину роботи за комп’ютером.

Якщо виробничі умови не дозволяють застосувати регламентовані перерви, роботодавець повинен забезпечити такий графік роботи, щоб безперервна робота з комп’ютером не перевищувала 4 години.

Для підтримки здоров’я та професійної придатності працівників рекомендується облаштувати на підприємстві приміщення для відпочинку та зняття нервово-емоційного напруження.

4.3 Висновок до четвертого розділу

У розділі «Безпека життєдіяльності, основи охорони праці» були розглянуті питання актуальності безпеки життєдіяльності людини та загальні вимоги безпеки з охорони праці для користувачів ПК.

Актуальність безпеки життєдіяльності людини підкреслює необхідність розробки та впровадження заходів, які знижують ризики, пов'язані з різними видами людської діяльності. Забезпечення сталого розвитку, що враховує економічні, соціальні та екологічні аспекти, є ключовим елементом цієї дисципліни.

Загальні вимоги безпеки з охорони праці для користувачів ПК включають широкий спектр заходів, спрямованих на мінімізацію негативних впливів, таких як напруження зору, втома, монотонність роботи та інші фактори, що впливають на здоров'я працівників. Дотримання санітарних норм, правильна організація робочого місця, регулярні медичні огляди та внутрішні регламентовані перерви є важливими складовими забезпечення безпечних умов праці.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи освітнього рівня «Бакалавр» розроблено веб-застосунок календар подій. Основні особливості цього застосунку включають інтеграцію з Google Calendar для створення подій, використання Google Sheets для зберігання даних, а також зручний інтерфейс для додавання та редагування подій і підподій. Веб-застосунок може бути успішно використаний для ефективного управління подіями.

Для розробки веб-застосунку календаря подій в кваліфікаційній роботі освітнього рівня «Бакалавр»:

– сформульовано поняття веб-застосунку.

– проведено аналіз існуючих інструментів для управління подіями, включаючи Google Calendar, Microsoft Outlook Calendar, Apple Calendar, Trello та Asana.

– розглянуто та проаналізовано засоби для створення веб-застосунків, обґрунтовано вибір технології для розробки.

– проведено огляд технології Google Apps Script та її функціональних можливостей.

– обґрунтовано вибір Google Apps Script для реалізації веб-застосунку.

– описано процес розробки веб-застосунку з використанням Google Apps Script, включаючи створення скриптів, налаштування тригерів та інтеграцію з Google Calendar і Google Sheets.

– створено таблиці для зберігання даних у Google Sheets та налаштовано календар у Google Calendar.

– розроблено основний функціонал веб-застосунку, що реалізовано за допомогою скриптів code.gs та calendar.gs.

– створено інтерфейс користувача за допомогою HTML-кодів user.html та edit.html, забезпечуючи зручність у взаємодії з застосунком.

– було розглянуто актуальність безпеки життєдіяльності людини та описано загальні вимоги безпеки з охорони праці для користувачів ПК.

ПЕРЕЛІК ДЖЕРЕЛ

1. Gomez J. Website Vs. Web Application: Understanding the Differences. Koombea. URL: https://www.koombea.com/blog/website-vs-web-application/ (date of access: 26.05.2024).

2. Гарасимів Б. І. Розробка мобільного додатку кіберспортивного порталу з голосовим управлінням / Б. І. Гарасимів, О. П. Ясній // Збірник тез доповідей VI Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі сучасних технологій“, 16-17 листопада 2017 року. — Т. : ТНТУ, 2017. — Том 2. — С. 36. — (Комп’ютерно-інформаційні технології та системи зв’язку).

3. Ясній О. П. Основні особливості backend-серверів для розробки мобільних додатків / О. П. Ясній, А. Ю. Попель // Збірник тез доповідей VII Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі сучасних технологій“, 28-29 листопада 2018 року. — Т. : ТНТУ, 2018. — Том 2. — С. 199. — (Комп’ютерно-інформаційні технології та системи зв’язку).

4. Top Web Development Tools in 2023. BrowserStack. BrowserStack. URL: https://www.browserstack.com/guide/web-development-tools (date of access: 26.05.2024).

5. ProgIngContrSystems. ProgIngContrSystems. URL: https://pupenasan.github.io/ProgIngContrSystems/Лекц/gs.html#загальний%20огляд (date of access: 26.05.2024).

6. Duvall J. An Introduction to Google Apps Script. Zapier. Automate your work today, Zapier. URL: https://zapier.com/blog/google-apps-script-tutorial/ (date of access: 26.05.2024).

7. CHOUDHARY R. A. WEB DEVELOPMENT (HTML,CSS and JAVASCRIPT). Independently Published, 2022.

8. Super Scholar. Empowering Google Workspace with Custom Add-ons. LinkedIn: Log In or Sign Up. URL: https://www.linkedin.com/pulse/empowering-google-workspace-custom-add-ons-super-scholar-

hqqff?trk=organization_guest_main-feed-card_feed-article-content (date of access: 26.05.2024).

9. The Ultimate Guide To Using Google Workspace Marketplace. Patronum. URL: https://www.patronum.io/the-ultimate-guide-to-using-google-workspace-marketplace/ (date of access: 26.05.2024).

10. Difference between Google Script (.GS) and JavaScript (.js). Online Tutorials, Courses, and eBooks Library. Tutorialspoint. URL: https://www.tutorialspoint.com/difference-between-google-script-gs-and-javascript-js (date of access: 26.05.2024).

11. Apps Script. Google for Developers. Google for Developers. URL: https://developers.google.com/apps-script/ (date of access: 26.05.2024).

12. Google Calendar. Google for Developers. Google for Developers. URL: https://developers.google.com/calendar/ (date of access: 26.05.2024).

13. Building Google Calendar interfaces. Apps Script. Google for Developers. Google for Developers. URL: https://developers.google.com/apps-script/add-ons/calendar/building-calendar-interfaces (date of access: 26.05.2024).

14. Google Apps Script quickstart. Google Calendar. Google for Developers. Google for Developers. URL: https://developers.google.com/calendar/api/quickstart/apps-script#prerequisites (date of access: 26.05.2024).

15. Khadka N. How To Add Appointments In Google Calendar From Google Sheets. Medium. URL: https://medium.com/script-portal/how-to-create-web-app-with-google-apps-script-part-v-fd7b8e21470a (date of access: 26.05.2024).

16. Khadka N. How To Create Web App With Google Apps Script–Part I. Medium. URL: https://medium.com/script-portal/how-to-create-web-app-with-google-apps-script-part-i-008b0ffc1adf (date of access: 26.05.2024).

17. Bootstrap. Bootstrap The most popular HTML, CSS, and JS library in the world. URL: https://getbootstrap.com/ (date of access: 26.05.2024).

18. Start Bootstrap. Start Bootstrap. URL: https://startbootstrap.com/bootstrap-resources (date of access: 26.05.2024).

19. Bootstrap resources: the ultimate collection for your projects. web-eau.net créé votre site web Joomla et le rend visible. URL: https://web-eau.net/en/blog/bootstrap-resources (date of access: 26.05.2024).

20. Скідан В. В. Аналіз архітектурних стилів при розробці WEB-додатків / В. В. Скідан, Т. І. Демківська // Інформаційні технології в науці, виробництві та підприємництві : збірник наукових праць молодих вчених, аспірантів, магістрів кафедри комп’ютерних наук та технологій / за заг. наук. ред. В. Ю. Щербаня. – Київ : ТОВ "Фастбінд Україна", 2022. – С. 137-140.

21. Learn web development. MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Learn (date of access: 26.05.2024).

22. Web Application Development: Resources, Best Practices, and How to do it. TMS Outsource. URL: https://tms-outsource.com/blog/posts/web-application-development/ (date of access: 26.05.2024).

23. Types of Web Applications with Use Cases. Homepage. URL: https://www.spaceotechnologies.com/blog/types-of-web-applications/ (date of access: 26.05.2024).

24. (WAI) W. W. A. I. Introduction to Web Accessibility. Web Accessibility Initiative (WAI). URL: https://www.w3.org/WAI/fundamentals/accessibility-intro/ (date of access: 26.05.2024).

25. Minnick J. Responsive Web Design with HTML 5 & CSS / J. Minnick. – Boston: Cengage Learning, 2020. – 640 p.

26. Google Apps Script release notes. Google for Developers. Google for Developers. URL: https://developers.google.com/apps-script/docs/release-notes (date of access: 26.05.2024).

27. Google Apps Script: A Beginner's Guide To Learn Apps Script. Google Sheets and Apps Script Tutorials and Online Courses. URL: https://www.benlcollins.com/apps-script/google-apps-script-beginner-guide/ (date of access: 26.05.2024).

28. Getting Started to Learn Google Apps Script. Teacher Tech with Alice Keeler. URL: https://alicekeeler.com/2023/01/28/getting-started-to-learn-google-apps-script/ (date of access: 26.05.2024).

29. Duvall J. An Introduction to Google Apps Script. Zapier. Automate your work today | Zapier. URL: https://zapier.com/blog/google-apps-script-tutorial/ (date of access: 26.05.2024).

30. GitHub - PacktPublishing/Google-Apps-and-Google-Script-A-Content-System-Web-App-Project: Google Apps and Google Script - A Content System Web App Project, published by Packt. GitHub. URL: https://github.com/PacktPublishing/Google-Apps-and-Google-Script---A-Content-System-Web-App-Project (date of access: 26.05.2024).

31. Advanced Google services. Apps Script. Google for Developers. Google for Developers. URL: https://developers.google.com/apps-script/guides/services/advanced (date of access: 26.05.2024).

32. Libraries. Apps Script. Google for Developers. Google for Developers. URL: https://developers.google.com/apps-script/guides/libraries (date of access: 26.05.2024).

33. Пачковський Н. С. Дослідження принципів роботи VPN мережі / Н. Пачковський, О. Ясній // Матеріали VI науково-технічної конференції „Інформаційні моделі, системи та технології“, 12-13 грудня 2018 року. — Т. : ТНТУ, 2018. — С. 82.

34. Ясній О. П. Захист програмного забезпечення на апаратному та програмному рівнях / О. П. Ясній, В. І. Карплюк // Матеріали VIII науково-технічної конференції „Інформаційні моделі, системи та технології“, 9-10 грудня 2020 року. — Т. : ТНТУ, 2020. — С. 105.

35. Гандзюк, М. П. Основи охорони праці [Текст] : підручник / М. П. Гандзюк, Є. П. Желібо, М. О. Халімовський ; за ред. М. П. Гандзюка ; МОН України. – 4-е видання. – К. : Каравела, 2008. –С. 254 – 260. – ISBN 966-8019-01-6.

36. Охорона праці при роботі з комп'ютером / ПК. Довідник спеціаліста з охорони праці. URL: https://pro-op.com.ua/article/183-ohoron-prats-pri-robot-z-kompyuterom (дата звернення: 26.05.2024).

37. Інструкція з охорони праці при роботі з комп'ютером, принтером, ксероксом та іншою оргтехнікою. Інструкції для навчальних закладів України. Інструкції для навчальних закладів України | Інструкції з охорони праці, техніки

безпеки і пожежної безпеки. URL: https://osvita-docs.com/node/41 (дата звернення: 26.05.2024).

38. Інструкція з охорони праці при роботі на ПК / персональному комп'ютері. Довідник спеціаліста з охорони праці. URL: https://pro-op.com.ua/article/485-nstruktsya-z-ohoroni-prats-pri-robot-na-personalnomu-kompyuter (дата звернення: 27.05.2024).

39. Методичні вказівки для написання розділу «Безпека життєдіяльності, основи охорони праці» в кваліфікаційних роботах здобувачів освітнього рівня „бакалавр". Для студентів всіх форм навчання рівень вищої освіти перший (бакалаврський) / укл. : О. Я. Гурик , І. Б. Окіпний. – Тернопіль : ТНТУ імені Івана Пулюя, 2021. – 20 с.

ДОДАТКИ

Додаток А

Код файлу code.gs

```
    const                        PROJECTSHEETID                        =
'1LsKUQThLwUkRH27X4Wd62vQHbGJXr_KVJQMytf62RpE';

    function requestAccess() {
        var                            sheet                             =
SpreadsheetApp.openById(PROJECTSHEETID).getSheetByName('access');
        var sheetdata = sheet.getDataRange().getValues();
        var user = Session.getActiveUser().getEmail();
        var access = false;
        for (var i = 0; i < sheetdata.length; i++) {
            if (sheetdata[i][0] == user) {
                access = true;
            }
        }
        return access;
    }

    function doGet(e) {
        updateCompleteStatus();
        var template = "user";
        var title = "Календар подій";
        var user = Session.getActiveUser().getEmail();
        if ('p' in e.parameters) {
            if (e.parameters['p'][0] === 'edit') {
                if (requestAccess()) {
                    template = 'edit';
                }
            }
        }
        var t = HtmlService.createTemplateFromFile(template);
        t.data = {
            user: user,
            tasks: listMain(),
            subtask: listSub()
        }
        try {
            var html = t.evaluate();
            html.setTitle(title);
            return html;
        } catch (e) {
            return
ContentService.createTextOutput(JSON.stringify({
                "error": e
            })).setMimeType(ContentService.MimeType.JSON);
        }
    }

    function listSub() {
```

```
        var sheet = SpreadsheetApp.openById(PROJECTSHEETID).getSheetByName('subtasks');
        var sheetdata = sheet.getDataRange().getValues();
        var data = sheetdata.slice(1);
        var ep = [];
        for (var i = 0; i < data.length; i++) {
            var com = (!isNaN(data[i][6]) || data[i][6] == '') ? data[i][6] : 100;
            var obj = {
                id: data[i][1],
                mainId: data[i][0],
                row: i + 2,
                created: data[i][2],
                due: data[i][3],
                assign: data[i][4],
                details: data[i][5],
            };
            ep.push(obj);
        }
        var output = {
            status: 'status',
            data: ep
        }
        Logger.log(output);
        return output;
    }

    function updateCompleteStatus() {
        var sheet = SpreadsheetApp.openById(PROJECTSHEETID).getSheetByName('tasks');
        var sheetdata = sheet.getDataRange().getValues();
        var today = new Date();

        for (var i = 1; i < sheetdata.length; i++) {
            var dueDate = new Date(sheetdata[i][2]);
            var complete = (dueDate < today) ? "TRUE" : "FALSE";
            sheet.getRange(i + 1, 6).setValue(complete);
        }
    }

    function listMain() {
        var sheet = SpreadsheetApp.openById(PROJECTSHEETID).getSheetByName('tasks');
        var sheetdata = sheet.getDataRange().getValues();
        var data = sheetdata.slice(1);
        var ep = [];
        var today = new Date();

        for (var i = 0; i < data.length; i++) {
            var dueDate = new Date(data[i][2]);
            var complete = (dueDate < today) ? true : false;
            if (!complete) {
                var obj = {
                    id: data[i][0],
                    row: i + 2,
```

```
                    created: data[i][1],
                    due: data[i][2],
                    assign: data[i][3],
                    details: data[i][4],
                    completed: complete
                };
                ep.push(obj);
            }
        }

        var output = {
            status: 'status',
            data: ep
        }
        Logger.log(output);
        return output;
    }

    function updateSubTask(data) {
        var                          sheet                          =
SpreadsheetApp.openById(PROJECTSHEETID).getSheetByName('subtasks')
;
        var sheetdata = sheet.getRange(2, 1, sheet.getLastRow(),
sheet.getLastColumn()).getValues();
        for (var i = 0; i < sheetdata.length; i++) {
            if (sheetdata[i][1] == data.taskID) {
                var    tempArr    =    [[data.due,    data.assign,
data.details]];
                var tempRange = sheet.getRange(i + 2, 4, 1, 4);
                tempRange.setValues(tempArr);
                return {
                    status: 'success',
                    message: tempArr
                };
            }
        }
        return data;
    }

    function updateTask(data) {
        var                          sheet                          =
SpreadsheetApp.openById(PROJECTSHEETID).getSheetByName('tasks');
        var sheetdata = sheet.getRange(2, 1, sheet.getLastRow(),
sheet.getLastColumn()).getValues();
        for (var i = 0; i < sheetdata.length; i++) {
            if (sheetdata[i][0] == data.taskID) {
                var    tempArr    =    [[data.due,    data.assign,
data.details]];
                var tempRange = sheet.getRange(i + 2, 3, 1, 3);
                tempRange.setValues(tempArr);
                return {
                    status: 'success',
                    message: tempArr
                };
            }
        }
```

```
        return data;
    }

    function addSubTask(data) {
        var sheet = SpreadsheetApp.openById(PROJECTSHEETID).getSheetByName('subtasks');
        var userEmail = Session.getActiveUser().getEmail();
        data.id = getRandom().toUpperCase();
        sheet.appendRow([data.taskID, data.id, formatDate(Date()), data.due, data.assign, data.details, userEmail]);
        data.rowValue = sheet.getLastRow();
        return data;
    }

    function addTask(data) {
        var sheet = SpreadsheetApp.openById(PROJECTSHEETID).getSheetByName('tasks');
        var userEmail = Session.getActiveUser().getEmail();
        data.id = getRandom().toUpperCase();
        sheet.appendRow([data.id,  formatDate(Date()),  data.due, data.assign, data.details, '', userEmail])

    createCalendarEvent(data);


        return data;
    }

    function deleteSubTask(subtaskID) {
        var sheet = SpreadsheetApp.openById(PROJECTSHEETID).getSheetByName('subtasks');
        var sheetdata = sheet.getRange(2, 1, sheet.getLastRow(), sheet.getLastColumn()).getValues();
        for (var i = 0; i < sheetdata.length; i++) {
            if (sheetdata[i][1] == subtaskID) {
                sheet.deleteRow(i + 2);
                return {
                    status: 'success',
                    message: 'Підподія успішно видалена'
                };
            }
        }
        return {
            status: 'error',
            message: 'Підподія не знайдена'
        };
    }

    function formatDate(date) {
        var d = new Date(date),
            month = '' + (d.getMonth() + 1),
            day = '' + d.getDate(),
            year = d.getFullYear();
        if (month.length < 2)
```

```
            month = '0' + month;
        if (day.length < 2)
            day = '0' + day;
        return [year, month, day].join('-');
    }

    function getRandom() {
        const uuid = Utilities.getUuid();
        const shortenedId = uuid.slice(0, 8);
        return shortenedId;
    }

    function createCalendarEvent(data) {
        var                          calendarId                          =
"065182d29c6b49ab789d64b553337a0c6b5de2b43e28591d84872daef4dea285@
group.calendar.google.com";
        var calendar = CalendarApp.getCalendarById(calendarId);
        var startDate = new Date();
        var endDate = new Date(data.due);

        if (startDate >= endDate) {
            Logger.log("Помилка: Дата початку події повинна бути
раніше дати завершення.");
            return;
        }

        var   event   =   calendar.createAllDayEvent(data.details,
startDate, endDate);
        event.addGuest(data.assign);
        var newId = event.getId();

        var                            sheet                            =
SpreadsheetApp.openById(PROJECTSHEETID).getSheetByName('tasks');
        var rowIndex = sheet.getLastRow();
        sheet.getRange(rowIndex, 8).setValue(newId);
        Logger.log(newId);

        sendEmailNotification(email, task, startDate, endDate);
    }
```

Додаток Б

Код файлу calendar.gs

```
function createEventsFromSpreadsheet() {
  let calendar = CalendarApp.getCalendarById("065182d29c6b49ab789d64b553337a0c6b5de2b43e28591d84872daef4dea285@group.calendar.google.com");
  let sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName('tasks');
  let schedule = sheet.getDataRange().getValues();
  schedule.splice(0, 1);
  schedule.forEach(function(entry) {
    var id = entry[7];
    Logger.log(id);
    if (id == "") {
      createNewEvent(entry, calendar, sheet, schedule);
    } else {
      updateExistingEvent(entry, id, calendar);
    }
  });
}

function createNewEvent(entry, calendar, sheet, schedule) {
  let startDate = new Date(entry[1]);
  let endDate = new Date(entry[2]);

  if (startDate >= endDate) {
    Logger.log("Помилка: Дата початку події повинна бути раніше дати завершення.");
    return;
  }

  let event = calendar.createAllDayEvent(entry[4], startDate, endDate);
  event.addGuest(entry[3]);
  var newId = event.getId();
  sheet.getRange(schedule.indexOf(entry) + 2, 8).setValue(newId);
  Logger.log(newId);

  sendEmailNotification(entry[3], entry[4], startDate, endDate);
}

function updateExistingEvent(entry, id, calendar) {
  let event = CalendarApp.getEventById(id);

  if (!event) {
    Logger.log("Подія з ID: " + id + " не знайдена");
    return;
  }

  Logger.log(event);
```

```
      let startDate = new Date(entry[1]);
      let endDate = new Date(entry[2]);

      if (startDate >= endDate) {
        Logger.log("Помилка: Дата початку події повинна бути раніше
дати завершення.");
        return;
      }

      event.setTitle(entry[4]);
      event.setTime(startDate, endDate);

      if (event.isAllDayEvent()) {
        event.removeAllGuests();
        event.addGuest(entry[3]);
      }
    }

    function   sendEmailNotification(email,   task,   startDate,
endDate) {
      let subject = "Повідомлення про подію";
      let message = "Шановний(-а) " + email + "\n\n" +
        "Це повідомлення щодо події: " + task + "\n" +
        "Дата початку: " + startDate + "\n" +
        "Дата завершення: " + endDate + "\n" +
        "Успіхів!\n\n" +
        "З повагою,\n" +
        "Calendar of events";
      GmailApp.sendEmail(email, subject, message);
    }
```

Додаток В

Код файлу user.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1, shrink-to-fit=no">
  <link                                          rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta2/dist/css/bootstrap.min.css">
  <style>
    .alert-custom {
      background-color: #663399;
      color: #fff;
    }
    .assign-badge-custom {
      background-color:  #008B8B;
      color: #fff;
      margin-top: 10px;
      margin-right: 15px;
      font-size: 1rem;
    }
    .badge-custom {
      background-color: #fff;
      color: #000000;
      font-size: 1rem;
    }
    .bg-light {
      color: #000000;
    }
    h1 {
      margin-left: 15px;
    }
    /* screen width < 576px */
    @media (max-width: 576px) {
      .h4 {
        font-size: 1rem;
      }
      .assign-badge-custom {
        font-size: 0.75rem;
      }
      .badge-custom {
        font-size: 0.75rem;
      }
    }
    /* screen width of 576px to 768px */
    @media (min-width: 576px) and (max-width: 768px) {
      .h4 {
        font-size: 1.25rem;
      }
```

```
            .assign-badge-custom {
              font-size: 0.9rem;
            }
            .badge-custom {
              font-size: 0.9rem;
            }
            .bg-light, .bg-success, .bg-primary {
              font-size: 0.9rem;
              font-weight: 500;
            }
          }
          /* screen width > 768px */
          @media (min-width: 768px) {
            .h4 {
              font-size: 1.5rem;
            }
            .assign-badge-custom {
              font-size: 1rem;
            }
            .badge-custom {
              font-size: 1rem;
            }
            .bg-light, .bg-success, .bg-primary {
              font-size: 1rem;
              font-weight: 500;
            }
          }
        </style>
        <script>
          const data = <?!= JSON.stringify(data) ?>;
          function goToEditPage() {
            const url = new URL(window.location.href);
            url.searchParams.set('p', 'edit');
            window.location.href = url.toString();
          }
        </script>
      </head>
      <body>
        <h1>Календар подій</h1>
        <div class="list"></div>
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.1/momen
t.min.js"></script>
        <script                src="https://code.jquery.com/jquery-
3.6.0.min.js"></script>
        <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.1/dist/umd/po
pper.min.js"></script>
        <script   src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta2/dist/js/bootstrap.min.js"></script>
        <script>
          console.log(data);
          const page = {};
          page.list = document.querySelector('.list');
          console.log(page);
          let total = 1;
```

```
          data.tasks.data.forEach(function (el) {
            console.log(el);
            const main = document.createElement('div');
            main.setAttribute('class', 'alert alert-custom');
            main.setAttribute('id', 'acc' + total);
            let creDate = el.created ?
moment(el.created).format('DD.MM.YYYY') : '';
            let dueDate = el.due ?
moment(el.due).format('DD.MM.YYYY') : '';
            main.innerHTML = '<h4 class="mb-2">' + el.details +
'</h4> ';
            main.innerHTML += ' <span class="badge assign-badge-
custom">' + el.assign + '</button>';
            main.innerHTML += ' <span class="badge badge-custom">' +
dueDate + '</span>';
            total++;
            page.list.append(main);
            const ul = document.createElement('ul');
            ul.classList.add('list-group');
            ul.classList.add('mb-4');
            main.append(ul);
            let taskid = el.id;
            data.subtask.data.forEach(function (subEle) {
              if (subEle.mainId == taskid) {
                addSubTask(ul, subEle, taskid);
              }
            })
          })
          function addSubTask(ele, subEle, taskid) {
            const li = document.createElement('li');
            li.classList.add('list-group-item');
            let creDate = subEle.created ?
moment(subEle.created).format('DD.MM.YYYY') : '';
            let dueDate = subEle.due ?
moment(subEle.due).format('DD.MM.YYYY') : '';
            li.innerHTML = `<span class="badge bg-
light">${subEle.details}</span> `;
            li.innerHTML += `<span class="badge bg-
primary">${subEle.assign}</span> `;

            console.log(subEle);
            ele.append(li);
          }
        </script>
      </body>
      </html>
```

Додаток Г

Код файлу edit.html

```
<!DOCTYPE html>
<html>

<head>
  <base target="_top">
  <link                                          rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstra
p.min.css">
  <link                                          rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootst
rap.min.css">
  <style>
    .custom-group {
      background-color: #663399;
    }

    .custom-alert {
      background-color: #9370DB;
    }

    .btn-custom {
      background-color: #008B8B;
      color: #fff;
    }

    h1 {
      margin-left: 15px;
    }
  </style>
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min
.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.
min.js"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.1/momen
t.min.js"></script>
  <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper
.min.js"                                       integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
crossorigin="anonymous"></script>
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/js/bootstra
p.min.js"></script>
</head>

<body>
  <div class="container">
```

```
        <h1>Управління подіями</h1>
        <div id="alerts"></div>
        <div class="manage">
          <div class="input-group control-group after-add-more p-3
mb-2 text-white custom-group">
            <div class="col-sm-12">
              <input  type="text"  name="temp_event"  class="form-
control" placeholder="Подія">
            </div>
            <div class="col-sm-4">
              <input  type="date"  name="temp_date"  class="form-
control">
            </div>
            <div class="col-sm-4">
              <input  type="text"  name="temp_assign"  class="form-
control" placeholder="Виконавець">
            </div>
            <div class="col-sm-4">
              <input type="button" class="btn btn-custom addTask"
value="Додати">
            </div>
          </div>
        </div>
        <div class="list"></div>
        <div id="task-list" class="panel-group"></div>
        <div id="subtask-list" class="panel-group"></div>
      </div>

      <script>
        function showAlert(cl, message) {
          $('#alerts').append('<div class="alert alert-' + cl + '
alert-dismissible fade show" >' + message + '</div>');
          $(".alert-dismissible").fadeTo(2000,   500).slideUp(500,
function () {
            $(".alert-dismissible").alert('close');
          });
        }

        const data = <?!= JSON.stringify(data) ?>;

        function formatDate(dateString) {
          var date = moment(dateString);
          return date.format('DD MMMM YYYY');
        }

        const page = {};
        page.list = document.querySelector('.list');
        page.tevent                                                =
document.querySelector('input[name="temp_event"]');
        page.tdate                                                 =
document.querySelector('input[name="temp_date"]');
        page.tassign                                               =
document.querySelector('input[name="temp_assign"]');
        page.taskAdderButton = document.querySelector('.addTask');
        page.taskAdderButton.addEventListener('click', function ()
{
```

```
            if (page.tevent.value.length > 3) {
              let temp = {
                due: page.tdate.value,
                assign: page.tassign.value,
                details: page.tevent.value
              };
              page.tevent.value = "";
              showAlert('success', 'Нова подія додана');

google.script.run.withSuccessHandler(success).addTask(temp);
            }
          });

          function success(data) {
            console.log(data);
            refresh();
          }

          console.log(data);
          let total = 1;
          data.tasks.data.forEach(function (el) {
            console.log(el);
            const main = document.createElement('div');
            main.setAttribute('class', 'alert custom-alert');
            main.setAttribute('id', 'acc' + total);
            let dueDate = el.due ? moment(el.due).format('YYYY-MM-
DD') : '';
            const input1 = document.createElement('input'); //Task Id
            input1.setAttribute('type', 'hidden');
            input1.value = el.id;
            main.append(input1);
            const input2 = document.createElement('input');
//Details
            input2.setAttribute('type', 'text');
            input2.value = el.details;
            main.append(input2);
            const input3 = document.createElement('input'); //Assign
            input3.setAttribute('type', 'text');
            input3.value = el.assign;
            main.append(input3);
            const input4 = document.createElement('input'); //Due
            input4.setAttribute('type', 'date');
            input4.value = dueDate;
            main.append(input4);
            const input5 = document.createElement('input'); //Button
            input5.setAttribute('type', 'button');
            input5.value = 'Зберегти';
            main.append(input5);
            input5.addEventListener('click', function (e) {
              let tempObj = {
                taskID: input1.value,
                details: input2.value,
                due: input4.value,
                assign: input3.value
              };
              showAlert('success', 'Подія оновлена ' + input1.value);
```

```
google.script.run.withSuccessHandler(success).updateTask(tempObj);
            console.log(tempObj);
          });
          total++;
          page.list.append(main);
          const ul = document.createElement('ul');
          ul.classList.add('list-group');
          ul.classList.add('mb-4');
          main.append(ul);
          let taskid = el.id;
          data.subtask.data.forEach(function (subEle) {
            if (subEle.mainId == taskid) {
              addSubTask(ul, subEle, taskid);
            }
          });
          addSubTaskAdder(main, el.id);
        });

        function addSubTaskAdder(mainEle, mainTaskID) {
          const li = document.createElement('div');
          li.classList.add('list-group-item');
          let dueDate = moment(new Date()).format('YYYY-MM-DD');
          const    input1    =    document.createElement('input');
//MainTask Id
          input1.setAttribute('type', 'hidden');
          input1.value = mainTaskID;
          li.append(input1);
          const    input2    =    document.createElement('input');
//SubTask Details
          input2.setAttribute('type', 'text');
          input2.setAttribute('placeholder', 'Деталі');
          input2.value = '';
          li.append(input2);
          const    input3    =    document.createElement('input');
//SubTask Due
          input3.setAttribute('type', 'date');
          input3.value = dueDate;
          li.append(input3);
          const    input4    =    document.createElement('input');
//SubTask Assign
          input4.setAttribute('type', 'text');
          input4.setAttribute('placeholder', 'Виконавець');
          input4.value = '';
          li.append(input4);
          const input5 = document.createElement('input'); //Button
          input5.setAttribute('type', 'button');
          input5.value = 'Додати';
          li.append(input5);
          input5.addEventListener('click', function (e) {
            showAlert('success',    'Підподія    оновлена    '    +
input1.value);
            let tempObj = {
              taskID: input1.value,
              details: input2.value,
              due: input3.value,
```

```
              assign: input4.value,
            };

google.script.run.withSuccessHandler(success).addSubTask(tempObj);
            console.log(tempObj);
          });
          mainEle.append(li);
        }

        function addSubTask(ele, subEle, taskid) {
          const li = document.createElement('li');
          li.classList.add('list-group-item');
          let dueDate = subEle.due ?
moment(subEle.due).format('YYYY-MM-DD') : '';
          const input1 = document.createElement('input');
//SubTask Id
          input1.setAttribute('type', 'hidden');
          input1.value = subEle.id;
          li.append(input1);
          const input2 = document.createElement('input');
//SubTask Details
          input2.setAttribute('type', 'text');
          input2.value = subEle.details;
          li.append(input2);
          const input3 = document.createElement('input');
//SubTask Due
          input3.setAttribute('type', 'date');
          input3.value = dueDate;
          li.append(input3);
          const input4 = document.createElement('input');
//SubTask Assign
          input4.setAttribute('type', 'text');
          input4.value = subEle.assign;
          li.append(input4);
          const input5 = document.createElement('input'); //Button
          input5.setAttribute('type', 'button');
          input5.value = 'Зберегти';
          li.append(input5);
          input5.addEventListener('click', function (e) {
            let tempObj = {
              taskID: taskid,
              subtaskID: input1.value,
              details: input2.value,
              due: input3.value,
              assign: input4.value,

            };
            showAlert('success', 'Підподія збережена ' +
input1.value);

google.script.run.withSuccessHandler(success).updateSubTask(tempOb
j);
            console.log(tempObj);
          });
```

```
        const input6 = document.createElement('input'); //Button
Delete
        input6.setAttribute('type', 'button');
        input6.setAttribute('value', 'Видалити');
        li.append(input6);
        input6.addEventListener('click', function (e) {
            if (confirm('Ви впевнені, що хочете видалити цю
підподію?')) {
                showAlert('success', 'Підподія видалена ' +
input1.value);

google.script.run.withSuccessHandler(success).deleteSubTask(input1
.value);
            }
        });
          ele.append(li);
        }


document.querySelectorAll('input[type="button"]').forEach(function
(button) {
       if (!button.classList.contains('addTask')) {
         button.classList.add('btn-outline-primary',     'rounded',
'shadow-none', 'border-0');
       }
      });



       </script>
     </body>

     </html>
```