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: Веб-застосунок для керування часом використання мобільного телефону

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

_____ Бондаренко В.С.
(підпис) (прізвище та ініціали)

Керівник _____ Дмитроца Л.П.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Литвиненко Я.В.
(підпис) (прізвище та ініціали)

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

Рецензент _____ Загородна Н.В.
(підпис) (прізвище та ініціали)

Тернопіль
2023

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)

«__» ____18 червня___ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

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

Студенту Бондаренко Владислав Сергійович
(прізвище, ім'я, по батькові)

1. Тема роботи Веб-застосунок для керування часом використання мобільного телефону
Керівник роботи Дмитроца Леся Павлівна, кандидат технічних наук, доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 7 « лютого 2023 року № 4/7-133 .
2. Термін подання студентом завершеної роботи 19 червня 2023р.
3. Вихідні дані до роботи Літературні та інтернет джерела щодо створення веб-застосунку

4. Зміст роботи (перелік питань, які потрібно розробити)
ВСТУП, РОЗДІЛ 1. АКТУАЛЬНІСТЬ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ,1.1 Актуальність проблеми,1.2 Основні поняття і визначення ,1.3 Органайзер для планування часу ,1.4 Фактори для вибору методу розробки, 1.5 Огляд існуючих рішень,1.6 Аналіз патернів проектування,1.7 Висновок до першого розділу, РОЗДІЛ 2. АРХІТЕКТУРА ВЕБ-ЗАСТОСУНКУ,2.1 Мікросервіси ,2.2 Процес переходу на мікросервіси ,2.3 Поняття Front-end розробки, 2.4 Підходи та вимоги до Front-end розробки,2.5 Тестування веб-застосунку, 2.6 Висновок до другого розділу, РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ХОРОНИ ПРАЦІ, 3.1 Актуальність безпеки життєдіяльності,3.2 Вимоги безпеки до робочих місць для виконання робіт,3.3 Висновок до третього розділу, ВИСНОВКИ, ПЕРЕЛІК ДЖЕРЕЛ, Програмний код.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Презентація додана

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	Завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О.Я. , кандидат технічних наук, доцент	05.06.2023	08.06.2023

7. Дата видачі завдання 23 січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	23.01.2023	*Виконано*
2.	Підбір джерел про розробку програми, щодо розробки веб-застосунку для керування часом мобільного телефону	24.01.2023-26.01.2023	*Виконано*
3.	Опрацювання джерел по темі кваліфікаційної роботи	27.01.2023-31.01.2023	*Виконано*
4.	Виконання дослідження щодо дослідження відповідних програмних засобів	01.02.2023-07.02.2023	*Виконано*
	Розроблення відповідного додатка		
5.	Оформлення розділу «РОЗДІЛ 1. АКТУАЛЬНІСТЬ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ»	08.02.2023-09.02.2023	*Виконано*
6.	Оформлення розділу «РОЗДІЛ 2. АРХІТЕКТУРА ВЕБ-ЗАСТОСУНКУ»	10.02.2023-12.02.2023	*Виконано*
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	05.06.2023-06.06.2023	*Виконано*
8.	Виконання завдання до підрозділу «Основи хорони	07.06.2023-08.06.2023	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	09.06.2023-11.06.2023	*Виконано*
10.	Нормоконтроль	12.06.2023-13.06.2023	*Виконано*
11.	Перевірка на плагіат	14.06.2023	*Виконано*
12.	Попередній захист кваліфікаційної роботи	15.06.2023	*Виконано*
13.	Захист кваліфікаційної роботи	19.06.2023	

Студент _____ Бондаренко В.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дмитроца Л.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Веб-застосунок для керування часом використання мобільного телефону // Кваліфікаційна робота освітнього рівня «Бакалавр» // Бондаренко В.С.// Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2023 // С.44, рис. –4 , табл. –3 , бібліогр. – 10.

Ключові слова: веб застосунок, додаток, мобільний телефон, економія часу, визначення потреб часу, час, проектування.

Кваліфікаційна робота присвячена дослідженню розробки веб-застосунку для керування часу використання мобільного телефону у спеціальному відділі ІТ-компанії, яка працює над важливим проектом. На період виконання проекту команда робітників, задіяних у проекті має певні обмеження у використанні мобільних телефонів. На їх особисті телефони встановлено веб-застосунок, який контролює час перебування власника у мережі інтернет та у соцмережах, інформацію по кожному телефону має можливість бачити керівник проекту. У веб-застосунку «Органайзер» розроблено спеціальний чат, в якому можуть спілкуватись усі учасники проекту без витрат часу на інші особисті діалоги. Обмеження на використання мобільного телефону знімається автоматично у неробочий час : 18.00-8.00.

В першому розділі кваліфікаційної роботи описано актуальність проблеми та постановка завдання дослідження. Розроблено технічні характеристики та забезпечення програми.

В другому розділі кваліфікаційної роботи досліджено архітектуру веб-застосунку. Розроблено інтерфейс, функціональні та операційні особливості.

До заповнення цієї сторінки: 12.06.2023р

ANNOTATION

Web application for managing mobile phone usage time// Qualification work of the educational level «Bachelor» // V.S. Bondarenko// Ternopil National Technical University named after Ivan Pulyu, facultycomputer and information systems and software engineering, department of computer sciences, group CH-41// Ternopil, 2023 // C.44, fig. –4, tab. –3, bibliography - 10.

Keywords:web application, application, mobile phone, saving time, determining time needs, time, designing.

The qualification work is devoted to research on the development of a web application for managing the time of mobile phone use in a special department of an IT company working on an important project. During the project implementation, the team of workers involved in the project has certain restrictions on the use of mobile phones. A web application is installed on their personal phones, which monitors the owner's time spent on the Internet and in social networks, the information on each phone is visible to the project manager. The web application «Organizer» has developed a special chat in which all project developers can communicate without wasting time on other personal dialogues. The restriction on the use of a mobile phone is removed automatically during non-working hours: 6:00 p.m. to 8:00 a.m.In the first section of the qualification paper, the relevance of the problem and the setting of the research task are described.

In the first section of the qualification paper, the relevance of the problem and the setting of the research task are described. The technical characteristics and support of the program have been developed.

In the second section of the qualification work, the architecture of the web application was investigated. The interface, functional and operational features have been developed.

Until this page is completed: 12.06.2023

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

MVC - Модель–Вид–Контролер

HTML - мова розмітки гіпертексту

CSS - каскадні таблиці стилів

XML - розширювана мова розмітки

UI – користувацький інтерфейс

API - інтерфейс програмування застосунків, інтерфейс прикладного програмування

ЗМІСТ

ВСТУП

На сьогоднішній день ІТ-компанії створюють комерційні проекти, в яких задіяна величезна кількість людських ресурсів. Звичайно ж, що сам процес розробки ділиться між членами команди. Для того, щоб проектний менеджер міг ефективно керувати проектом, йому необхідно знати на якому етапі знаходиться процес розробки того чи іншого модуля додатку. Перед початком розробки проекту, його керівники планують роботу таким чином, щоб усі, хто задіяні у процесі розробки, працювали разом та ефективно. Для того, щоб робота була ефективною, необхідно контролювати роботу співробітників, а так як деякі працюють дистанційно, то розробити і встановити на мобільні телефони кожного працівника спеціальний веб-застосунок, який буде контролювати час перебування власника у мережі інтернет та власне у самому телефоні. Веб-застосунок має вигляд та структуру «Органайзера», з багатьма додатковими функціями. Такі як: планування робочого часу, планування зустрічей та конференцій.

Персонал компанії повинен комунікувати між собою без перешкод. Тому у розробленому веб-додатку, можна використовувати альтернативні джерела зв’язку такі як: телефон, соціальні мережі, чи скайп. Але є значно простіший та зручніший спосіб – використання task tracking систем. Це коли співробітники спілкуються в одному робочому чаті і не мають можливості переписуватись з кимось іншим. У такому випадку керівник зможе з легкістю стежити за виконанням роботи певним працівником чи команди у цілому. Насамперед, призначення task tracking систем – це стеження за якістю продукту, адже більшість дефектів потрібно виявляти на етапі розробки, тим самим це дає можливість на етапі підтримки не виявити критичні баги. Тобто забезпечити комфортну комунікацію для усіх членів команди.

Web-застосунок «Органайзер» необхідний для оптимізації трудовитрат пов’язаних з організацією робочого часу.

Багато людей часто стикаються з проблемою зайвих витрат часу при організації свого часу. Будь то директор компанії, якому доводиться витрачати чимало часу на постановку завдань для співробітників, призначення часу

зустрічей та термінів проведення різних заходів або людина, яка займається якоюсь приватною діяльністю, для якої важливо мати можливість легко і швидко позначити список необхідних завдань і виділити час під необхідні заходи.

Цей застосунок призначений як для приватного, так і корпоративного використання. Програма дає можливість швидко і легко сформувати список завдань, розподілити завдання по співробітникам. Призначити час проведення зустрічей та заходів. Зручний інтерфейс програми дозволить, маючи лише доступ в Інтернет, завжди отримати доступ до плану роботи на день.

Керівникам часто доводиться індивідуально опрацьовувати завдання кожного співробітника. Витрачати особистий час, щоб довести інформацію про завдання та зустрічі до своїх працівників. Необхідність у цьому зникає з web-додатком «Органайзер».

Метою кваліфікаційної роботи є створення web-застосунку для керування часом використання мобільного телефону пов’язаного з організацією робочого процесу компанії.

Для досягнення поставленої мети необхідне вирішення наступних завдань:

– аналіз пропонованих програмних продуктів націлених на організацію особистого та робочого часу;
– проектування функціоналу web-застосунку;
– розробка дизайну веб-додатку;
– розробка алгоритмів реалізації функціоналу web-застосунку;
– проектування бази даних;
– реалізація адаптивного, інтуїтивно зрозумілого інтерфейсу.

Об’єкт дослідження: веб-застосунок для керування часом мобільного телефону.

Предмет дослідження: розробка веб-застосунку для керування часом мобільного телефону.

РОЗДІЛ 1. АКТУАЛЬНІСТЬ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Актуальність проблеми

Проблема керування часом використання мобільних телефонів є актуальною і потребує ефективного рішення. На ринку існують різноманітні програмні продукти, спрямовані на організацію робочих процесів, планування

Таблиця 1.1 - Порівняння web-додатків та десктоп-додатків

Критерій	**Десктоп-застосунок**	**Web-застосунок**
Доступ до Інтернету	Не вимагається.	Потрібний. Як виняток, деякі додатки здатні деякий час працювати в автономному режимі.
Установка оновлення	Необхідне встановлення. Оновлення під час виходу нової версії.	Немає необхідності в установці і оновленнях.
Інтерфейс взаємодії	Стандартні інтерфейси, стандартна взаємодія.	Різноманітний інтерфейс взаємодії, різноманітні реалізації.
Сумісність з пристроями	Залежність від платформи.	Не залежить від платформи.
Анімація, графіка	Швидка анімація і відгук на події.	У зв’язку з необхідністю передачі даних по мережі, відносно повільний відгук.
Пошук за контентом	Неможливо, крім тих випадків, коли це реалізовано на рівні програми.	Є.
Розробка	Використання різних інструментів під різні платформи. Найчастіше доводиться писати свою версію під кожну версію платформи	Кросплатформно. Все виконується на сервері. Необхідний лише браузер. Всі інструменти та софт на сервері кросплатформні.

Реалізація проектів у вигляді web-додатку дозволила отримати кросплатформовий продукт зі зручним доступом на будь-якій платформі.

1.2 Основні поняття і визначення

«Клієнт-сервер» – мережева архітектура. Завдання у цій системі розподіляються між серверами та клієнтами. Програма-сервер очікує від клієнтських програм запити та надає їм свої ресурси у вигляді web-сторінок, оброблених баз даних, сервісних функцій та багато іншого.

Переваги:

– відсутність дублювання коду програми сервера програмами-клієнтами;

– всі обчислення відбуваються на сервері. Вимоги до комп'ютерів, де встановлено клієнт, дуже низькі;

– всі дані зберігаються на сервері, зазвичай сервер захищений набагато краще більшості клієнтів.

Клієнт-серверна програма дозволяє набагато простіше реалізувати контроль повноважень для клієнтів з різним рівнем доступу.

1.3 Додаток для планування часу

Model-View-Controller - MVC модель або Модель-Вид-Контролер. Дана модель поділяє дані програми, інтерфейсу користувача і керуючої логіки на три окремих копоненти:

– модель - компонент системи, що відповідає за вилучення даних, з подальшою їх маніпуляцією;

– подання - відповідає за відображення основної видимої структури та даних на сайті;

– контролер - здійснює управління та контроль за роботою моделей та уявлень.

Використання Model-View-Controller дає можливість змінювати кожен компонент незалежно один від одного та чітко розмежувати функціонал та завдання кожного елемента програми.

1.4 Фактори для вибору методу розробки

Для створення, зберігання та відображення гіпертексту у Всесвітньому павутинні традиційно використовується мова HTML.

Мова HTML - мова тегів. Теги формують структуру документа HTML. Теги оформлюються кутовими дужками, між якими прописується ім’я тега.

Теги розміщуються в HTML-документі відповідно до правил розмітки (порядок слідування, правило вкладення тегів), створюючи розділи майбутньої веб-сторінки. Крім того, HTML-документи можуть містити спеціальні символи.

Браузер переглядає (інтерпретує) HTML-документ, вибудовуючи його структуру та відображаючи її відповідно до інструкцій, включених до цього файлу (таблиці стилів, скрипти). Якщо правильна розмітка, то у вікні браузера буде відображена HTML-сторінка, що містить HTML-елементи - заголовки, таблиці, зображення і т.д.

Процес інтерпретації починається, перш ніж веб-сторінка повністю завантажена в браузер. Браузери обробляють HTML-документи послідовно, від початку, обробляючи CSS і співвідносячи таблиці стилів з елементами сторінки.

CSS (Каскадні таблиці стилів) - формальна мова опису зовнішнього вигляду документа, написаного за допомогою мови розмітки.

Переважно використовується як опис, оформлення зовнішнього вигляду веб-сторінок, написаних за допомогою мов розмітки HTML і XHTML, але може також застосовуватися до будь-яких XML-документів.

Каскадні таблиці стилів описують правила відображення елементів за допомогою властивостей та допустимих значень цих властивостей. Для кожного елемента можна використовувати обмежений набір властивостей, інші властивості не будуть впливати на нього.

Оголошення стилю складається з двох частин: елемента веб-сторінки - селектора, команди форматування - блоку оголошення. Селектор повідомляє браузеру, який саме елемент форматувати, а в блоці оголошення (код у фігурних дужках) перераховуються команди, що форматують властивості та їх значення.

Bootstrap 3 – це популярний фреймворк, що містить вільний набір інструментів для розробки адаптивних сайтів та мобільних web-проектів. Включає HTML і CSS шаблони оформлення для типографіки, веб-форм, кнопок, міток, блоків навігації та інших компонентів веб-інтерфейсу, включаючи JavaScript-розширення.

Основні переваги Bootstrap 3:

– економія часу - Bootstrap дозволяє заощадити час та зусилля, використовуючи шаблони дизайну та класи, і сконцентруватися на інших розробках;

– висока швидкість - динамічні макети Bootstrap масштабуються на різні пристрої та роздільну здатність екрану без будь-яких змін у розмітці;

– гармонійний дизайн - всі компоненти платформи Bootstrap використовують єдиний стиль та шаблони за допомогою центральної бібліотеки;

– дизайн та макети веб-сторінок узгоджуються один з одним;

– простота у використанні - платформа проста у використанні, користувач з базовими знаннями HTML та CSS може розпочати розробку з Twitter Bootstrap;

– сумісність із браузерами - Twitter Bootstrap сумісний із Mozilla Firefox, Google Chrome, Safari, Internet Explorer, Microsoft Edge та Opera;

– відкрите програмне забезпечення - особливість Twitter Bootstrap, яка передбачає зручність використання за допомогою відкритості вихідних кодів та безкоштовного завантаження.

Bootstrap спроектований для кращої роботи в нових браузерах, тобто старі браузери не завжди можуть правильно відображати стилі, хоча функціональні у візуалізації певних компонентів.

Далі наведено таблицю підтримки браузерами Bootstrap 3.

Таблиця 1.2 - Підтримка Bootstrap 3 браузерами

ОС	**Chrome**	**Firefox**	**Internet Explorer**	**Opera**	**Safari**
1	**2**	**3**	**4**	**5**	**6**
Android	+	-	N/A	-	N/A

1	2	3	4	5	6
iOS	+	N/A	N/A	-	+
MAC OS X	+	+	N/A	+	+
Windows	+	+	+	+	-

Мова JavaScript використовується для створення інтерактивних html-документів та виконання обчислень без необхідності звертатися до сервера. Код JavaScript інтерпретується браузером без необхідності компіляції в бінарний код.

Виконання JavaScript-коду починається відразу ж після виявлення його на сторінці, рядки коду виконуються послідовно.

Якщо на сторінці присутні кілька сценаріїв JavaScript, вони будуть виконуватися в порядку розташування в документі.

Javascript підключається безпосередньо до HTML-файлу. Найпростіший спосіб – це написати javascript-команди всередину тега <script> у тілі сторінки.

Коли браузер читає HTML-сторінку, і бачить <script> - він насамперед читає та виконує код, а лише потім продовжує читати сторінку далі.

Практично неможливо уявити собі гарний сайт без динамічних візуальних та функціональних ефектів.

Використання JavaScript дозволяє зробити web-застосунок сучасним та динамічним.

Для реалізації веб-застосунку для керування часом використання мобільного телефону, було вибрано використання бібліотеки jQuery та мови програмування PHP з певних причин.

Бібліотека jQuery є чудовим інструментом для взаємодії між HTML та JavaScript. Вона надає простий та швидкий спосіб отримувати доступ до елементів DOM, керувати їх атрибутами та вмістом. Завдяки jQuery, можна ефективно взаємодіяти з різними елементами сторінки, що дає зручність та швидкість в розробці функціональності з призначенням керування часом використання мобільного телефону.

Мова програмування PHP вибрана через її широке використання та спрямованість на розробку веб-програм. PHP дозволяє впроваджувати код безпосередньо в HTML, що робить процес розробки зручним та ефективним.

Важливою відмінністю PHP від JavaScript є те, що PHP-скрипти обробляються на сервері та генерують HTML-сторінку, яка передається клієнту. Це забезпечує безпеку та конфіденційність коду, оскільки клієнт отримує лише результат виконання скрипта і не може отримати доступ до вихідного коду.

Таке поєднання jQuery та PHP надає можливість створити потужний та безпечний веб-застосунок для керування часом використання мобільного телефону.

Розробка веб-додатку з використанням засобів фреймворку Yii2, дозволяє набагато швидше реалізувати весь функціонал web-додатку і створити конкурентноздатний продукт.

1.5 Огляд існуючих рішень

На ринку існує багато веб-застосунків, які пропонують різноманітні ідеї для керування часом використання мобільного телефону. Розглянемо деякі з них.

Web-застосунок "TimeTracker".

TimeTracker є потужним інструментом для контролю часу, призначеним спеціально для мобільних пристроїв. Він дозволяє користувачам встановлювати цілі для використання часу, відстежувати активність на різних додатках та веб-сайтах, а також отримувати детальну звітність про витрачений час. TimeTracker надає можливість налаштувати сповіщення та нагадування для ефективного управління часом.

Веб-додаток "FocusGuard".

FocusGuard є інноваційним рішенням для управління часом використання мобільних телефонів. Він пропонує ряд функцій, спрямованих на покращення продуктивності та уникнення відволікань. За допомогою FocusGuard, користувачі можуть встановлювати обмеження на використання певних додатків чи веб-сайтів, вмикати режим блокування на часи концентрації, а також налаштовувати періодичні паузи для відпочинку та розслаблення.

Веб-додаток "TimeMaster".

TimeMaster - це універсальний веб-застосунок, який дозволяє ефективно керувати часом використання мобільних телефонів. Він пропонує широкий спектр функцій, включаючи створення списків завдань, планування подій та зустрічей, відстеження часу, аналіз продуктивності та генерацію звітів. TimeMaster також підтримує синхронізацію даних між різними пристроями, що дозволяє користувачам отримувати доступ до своїх даних з будь-якого місця.

Веб-додаток "TimeSense".

TimeSense є простим, але ефективним веб-застосунком для керування часом використання мобільних телефонів. Він дозволяє користувачам встановлювати цілі, створювати розклади та плани, а також отримувати нагадування про пріоритетні завдання. TimeSense також надає звіти про використання часу та аналіз продуктивності, що допомагає користувачам покращити свою ефективність.

Кожен з цих веб-застосунків має свої переваги і може задовольнити різні потреби користувачів. При виборі рішення для керування часом використання мобільного телефону, важливо враховувати функціональність та зручність використання.

1.6 Аналіз патернів проектування

Патерни проектування є важливою складовою розробки веб-застосунків. Деякі патерни можуть бути особливо корисними для досягнення ефективності, розширюваності та зручності використання. Розглянемо кілька популярних патернів проектування, які можуть знайти застосування в даному контексті.

MVC (Model-View-Controller) або Модель-Вид-Контролер є одним з найпоширеніших патернів проектування. У веб-застосунку для керування часом використання мобільного телефону, Модель представляє дані про користувача, його цілі, завдання тощо. Вид відповідає за візуалізацію інтерфейсу, де користувач може взаємодіяти з додатком. Контролер обробляє запити користувача, взаємодіє з Моделлю та Видом, керує логікою застосунку. Використання MVC дозволяє забезпечити чистоту, розширюваність та легкість управління кодом.

Observer (Спостерігач). Патерн Observer використовується для реалізації механізму сповіщень та підписки на події. У контексті веб-застосунку для керування часом використання мобільного телефону, Observer може бути використаний для сповіщення користувача про досягнення певних цілей, нагадування про важливі події або зміни в розкладі. Коли стан об'єкта-спостерігача змінюється, всі підписники (наприклад, користувачі) отримують відповідні сповіщення.

Strategy (Стратегія). Патерн Strategy дозволяє замінювати алгоритми виконання певної функціональності без зміни клієнтського коду. У веб-застосунку, Strategy може бути використаний для реалізації різних стратегій нагадування користувачу, наприклад, через сповіщення на пристрої або електронною поштою. Клієнтський код може легко використовувати вибрану стратегію нагадування, а також замінювати її на іншу, якщо змінюються вимоги.

Singleton (Одиночка). Патерн Singleton дозволяє забезпечити, щоб клас мав тільки один екземпляр, до якого можна отримати доступ з будь-якого місця програми. У веб-застосунку, Singleton може бути використаний для створення центрального об'єкта, що керує відстеженням часу, обміном даними між різними компонентами тощо. Це дозволяє забезпечити єдину точку доступу та попереджати неправильне використання ресурсів.

Аналіз патернів проектування допомагає визначити оптимальну архітектуру веб-застосунку для керування часом використання мобільного телефону, забезпечити легкість розширення, розуміння та підтримки коду. Залежно від конкретних вимог і варіантів використання, можуть бути вибрані різні патерни проектування для досягнення оптимального рішення.

Патерн MVC (Model-View-Controller) було вибрано для розробки веб-застосунку з певних причин:

1. Розподіл відповідальності. MVC розділяє компоненти системи на три основні ролі - Модель, Вид та Контролер, кожен з яких виконує відповідні завдання. Це сприяє збереженню чистоти коду та відокремленню бізнес-логіки від представлення та управління.

2. Розширюваність та перевикористання. Завдяки розподілу відповідальності, відокремлення компонентів та інкапсуляції бізнес-логіки, MVC дозволяє легко вносити зміни у систему та розширювати її функціональність. Наприклад, можна легко замінити або модифікувати Вид без впливу на Модель або Контролер.

3. Тестування. Розділення Моделі, Виду та Контролера спрощує тестування системи. Модель може бути перевірена на правильність обробки даних, Вид - на відповідність дизайну та коректність візуалізації, а Контролер - на обробку запитів та взаємодію з різними компонентами системи.

4. Спрощення розробки і підтримки. Застосування MVC дозволяє розподілити роботу між розробниками та забезпечити їхню незалежність. Наприклад, дизайнери можуть працювати над Видом, програмісти - над Моделлю та Контролером, що сприяє ефективності розробки та зменшує залежність від конкретних людей.

5. Підтримка шаблонів проектування. MVC підтримує використання інших патернів проектування, таких як Спостерігач (Observer), Стратегія (Strategy), Одиночка (Singleton) тощо. Це дозволяє застосовувати найкращі практики проектування та реалізувати більш гнучку та розширювану систему.

Загалом, вибір патерна MVC для розробки веб-застосунку для керування часом використання мобільного телефону забезпечує структурованість, розширюваність та зручність управління кодом. Цей патерн є відповідним для веб-застосунків, де важливо відокремити бізнес-логіку від представлення та управління та забезпечити легкість розширення та підтримки системи.

1.7 Висновок до першого розділу

У першому розділі було проведено огляд теми створення веб-застосунку для керування часом використання мобільного телефону. Була показана актуальність проблеми та оглянуті існуючі рішення на ринку. Також був проведений аналіз патернів проектування, зокрема патерну MVC, та обґрунтований вибір цього патерну для розробки веб-застосунку.

Використання патернів проектування, таких як MVC, дозволяє досягти ефективності, розширюваності та зручності використання веб-застосунку. MVC розділяє компоненти системи на Модель, Вид та Контролер, що дозволяє зберегти чистоту коду, забезпечити розширюваність та перевикористання компонентів, спростити тестування та підтримку.

Далі в розділі було проведено аналіз патернів проектування, таких як Observer, Strategy, Singleton, які також можуть знайти застосуванняя у веб-застосунку для керування часом використання мобільного телефону. Ці патерни допомагають забезпечити гнучкість, розширюваність та оптимізацію роботи системи.

У підсумку, перший розділ надав загальний огляд теми та обґрунтував вибір патерну MVC для розробки веб-застосунку для керування часом використання мобільного телефону. Наступні розділи будуть присвячені більш детальному аналізу функціональності, архітектури та інших аспектів розробки цього веб-застосунку.

РОЗДІЛ 2. АРХІТЕКТУРА ВЕБ-ЗАСТОСУНКУ

2.1 Мікросервіси

Мікросервіси є підхідом до архітектури програмного забезпечення, в якому програмний додаток розбивається на невеликі незалежні компоненти, відомі як мікросервіси. Кожен мікросервіс виконує обмежену функціональність та має свою власну базу коду, базу даних та інфраструктуру. Вони взаємодіють один з одним за допомогою легковагих механізмів комунікації, таких як API.

В контексті веб-застосунку для керування часом використання мобільного телефону, мікросервісна архітектура може мати декілька переваг:

1. Розширюваність та масштабованість. Мікросервіси дозволяють масштабувати окремі компоненти системи залежно від навантаження. Це дозволяє гнучко реагувати на зміни обсягу роботи та забезпечує ефективне використання ресурсів.

2. Незалежна розробка та підтримка. Кожен мікросервіс може бути розроблений та підтримуваний окремою командою. Це дозволяє прискорити процес розробки та підтримки, забезпечуючи паралельну роботу над різними компонентами системи.

3. Відокремленість помилок та легка заміна. У разі помилки або несправності одного мікросервісу, це не впливає на решту системи. Крім того, можна легко замінити або оновити окремі мікросервіси без впливу на інші компоненти.

4. Гнучкість та технологічна свобода. Кожен мікросервіс може використовувати різні технології та мови програмування, залежно від вимог та потреб компонента. Це дає можливість вибирати оптимальні технології для кожного конкретного мікросервісу.

Проте, використання мікросервісної архітектури також має свої виклики:

1. Складність управління та координації. З кожним додатковим мікросервісом зростає складність управління та координації між компонентами системи. Потрібні відповідні механізми комунікації та моніторингу, а також стратегії для управління залежностями та синхронізації.

2. Збільшення складності тестування. Кожен мікросервіс потребує окремого тестування, а також тестування взаємодії між ними. Це може призвести до збільшення складності тестування системи в цілому.

3. Заплутаність мережі. Збільшення кількості мікросервісів може призвести до складності мережі комунікації та збільшення навантаження на неї. Потрібно ретельне проектування мережевої інфраструктури та механізмів комунікації.

Мікросервісна архітектура може мати ефективний підхід до розробки веб-застосунка для керування часом використання мобільного телефону, забезпечуючи гнучкість, масштабованість та незалежну розробку компонентів системи. Однак, перед використанням мікросервісів необхідно враховувати виклики, пов'язані з управлінням, координацією та тестуванням системи.

2.2 Процес переходу на мікросервіси

Перехід на мікросервісну архітектуру вимагає ретельного планування, аналізу та виконання кількох кроків. Ось основні етапи процесу переходу на мікросервіси:

1. Аналіз поточної системи. Перш ніж переходити на мікросервісну архітектуру, необхідно провести аналіз поточної монолітної системи. Це включає оцінку функціональності, взаємодії компонентів, проблем та викликів, з якими стикається система. Цей етап допомагає зрозуміти, які частини системи можна розбити на окремі мікросервіси.

2. Визначення границь мікросервісів. На основі аналізу поточної системи визначаються границі мікросервісів. Це означає ідентифікацію окремих функціональних блоків, які можуть бути розроблені та підтримувані незалежно. Кожен мікросервіс повинен мати чітку відповідальність та обмежений контекст.

3. Дизайн та розробка мікросервісів. На цьому етапі розробляються та реалізуються окремі мікросервіси. Кожен мікросервіс може бути розроблений незалежно, використовуючи відповідні технології та інструменти. Для

забезпечення зручності використання та комунікації між мікросервісами, можуть бути використані стандартизовані API та протоколи.

4. Моніторинг та керування. У мікросервісній архітектурі важливо мати механізми моніторингу та керування мікросервісами. Це включає моніторинг стану мікросервісів, масштабування відповідно до навантаження, управління версіями та розгортанням.

5. Тестування та валідація. Кожен мікросервіс повинен бути тестований окремо, а також тестуватися взаємодіями з іншими мікросервісами. Важливо переконатися, що всі мікросервіси працюють належним чином та взаємодіють без помилок.

6. Розгортання та моніторинг. Після успішного тестування та валідації мікросервісів, їх можна розгорнути в продакшн середовище. Важливо мати механізми моніторингу, щоб відстежувати працездатність та продуктивність кожного мікросервісу.

Перехід на мікросервісну архітектуру - це складний процес, який вимагає ретельного планування, дизайну та розробки. Однак, правильно реалізована мікросервісна архітектура може принести багато переваг, таких як гнучкість, розширюваність та покращену розподілену обробку завдань.

2.3 Поняття Front-end розробки

Front-end розробка веб-застосунків охоплює процес створення користувацького інтерфейсу, з яким взаємодіє користувач на стороні клієнта (браузера). Це включає розробку та оформлення веб-сторінок, реалізацію функціональності та забезпечення позитивного досвіду користувача.

Основні поняття, пов'язані з front-end розробкою, включають:

1. HTML (HyperText Markup Language). HTML є основною мовою розмітки для створення структури веб-сторінок. Вона використовує теги для визначення різних елементів, таких як заголовки, параграфи, таблиці, форми тощо. [1]

2. CSS (Cascading Style Sheets). CSS використовується для оформлення та стилізації веб-сторінок. Він визначає зовнішній вигляд елементів, таких як кольори, шрифти, розташування, розміри, анімації та інші аспекти дизайну. [2]

3. JavaScript. JavaScript є мовою програмування, яка використовується для реалізації інтерактивності та динамічної поведінки веб-сторінок. Він дозволяє взаємодіяти з користувачем, маніпулювати DOM (Document Object Model), виконувати запити до сервера, робити асинхронні оновлення сторінок та багато іншого. [3]

4. Фреймворки та бібліотеки. Фреймворки та бібліотеки, такі як React, Angular, Vue.js, допомагають спростити та прискорити процес розробки фронт-енду. Вони надають готові компоненти, структури та інструменти для побудови складних веб-додатків.

5. Респонсивний дизайн. Респонсивний дизайн відноситься до підходу, при якому веб-сторінки адаптуються та відображаються оптимально на різних пристроях та розмірах екранів, включаючи комп'ютери, планшети та мобільні телефони.

Front-end розробка вимагає знань HTML, CSS, JavaScript та інших веб-технологій. Вона також включає розуміння принципів дизайну та взаємодії з користувачем. Впевнений front-end розробник може створювати привабливі та функціональні веб-застосунки, які відповідають потребам користувачів та бізнесу.

2.4 Підходи та вимоги до Front-end розробки

При розробці front-end частини веб-застосунка для керування часом використання мобільного телефону, важливо враховувати особливості та вимоги, що виникають з даної теми:

1. Респонсивний дизайн. Даний веб-застосунок буде використовуватися на мобільних телефонах, тому важливо забезпечити, щоб веб-інтерфейс був респонсивним. Це означає, що дизайн та розміщення елементів повинні адаптуватися до різних розмірів екранів, забезпечуючи комфортне користування навіть на невеликих дисплеях мобільних пристроїв.

2. Інтуїтивний інтерфейс. Для веб-застосунку, який допомагає керувати часом використання мобільного телефону, важливо мати інтуїтивний інтерфейс. Користувачі повинні легко розуміти, як взаємодіяти з додатком, встановлювати цілі, використовувати таймери та аналізувати свій час. Простота, зрозумілість та зручність використання повинні бути в центрі розробки інтерфейсу.

3. Оптимізація продуктивності. Веб-застосунок повинен працювати швидко та ефективно на мобільних пристроях, забезпечуючи плавну взаємодію та низький рівень навантаження пристрою. Важливо оптимізувати завантаження ресурсів, виконання запитів до сервера та ефективну роботу з пам'яттю, забезпечуючи швидкість та продуктивність додатку.

4. Безпека даних. Оскільки веб-застосунок має відношення до управління особистим часом користувача, безпека даних є критично важливою. Потрібно забезпечити захист персональної інформації користувачів, надійний механізм автентифікації та контролю доступу.

5. Інтеграція з мобільними функціями. Мобільні телефони мають різноманітні функціональні можливості, такі як сповіщення, геолокація, календарі, тощо. Важливо використовувати ці можливості та інтегрувати їх у веб-застосунок, щоб забезпечити більш багатогранний та функціональний досвід користувача.

Врахування цих підходів та вимог при розробці front-end частини веб-застосунка допомогла створити зручний, ефективний та безпечний продукт, який відповідає потребам користувачів у керуванні своїм часом.

2.5 Тестування веб-застосунку

Тестування веб-застосунку є важливою складовою процесу розробки, яка допомагає переконатися в якості та надійності програмного продукту. При створенні веб-застосунку для керування часом використання мобільного телефону, наступні види тестування можуть бути корисними:

1. Функціональне тестування. Перевірка функціональності веб-застосунку здійснюється для переконання, що всі функції та можливості працюють

належним чином. Це включає тестування функцій, таких як створення списків завдань, налаштування таймерів, аналітика витраченого часу тощо.

2. Користувацьке тестування. Включення реальних користувачів для випробування веб-застосунку може допомогти виявити проблеми та недоліки, які можуть бути незамітні розробникам. Користувачі зможуть оцінити зручність використання та надати цінні відгуки та пропозиції.

3. Тестування сумісності. Веб-застосунок повинен працювати на різних браузерах, операційних системах та пристроях. Тестування сумісності допомагає переконатися, що веб-застосунок правильно відображається та працює на різних платформах.

4. Тестування продуктивності. Важливо перевірити продуктивність веб-застосунку, зокрема швидкість завантаження сторінок, час відгуку та реагування на дії користувача. Тестування продуктивності дозволяє виявити можливі проблеми та забезпечити оптимальну продуктивність веб-застосунку.

5. Тестування безпеки. Зважаючи на те, що веб-застосунок може містити особисту інформацію користувачів, важливо провести тестування безпеки. Це включає перевірку на наявність потенційних уразливостей, перехоплення даних, аутентифікацію та авторизацію.

6. Тестування на різних пристроях. Оскільки веб-застосунок буде використовуватися на мобільних пристроях, важливо виконати тестування на різних моделях телефонів та планшетів з різними розмірами екранів та операційними системами.

Тестування веб-застосунку допоможе виявити та виправити проблеми, підвищити якість та надійність додатку, а також забезпечити задоволення користувачів від його застосування. Врахування різних видів тестування дозволить забезпечити якісний та функціональний веб-застосунок для керування часом.

2.6 Опис створення веб-застосунку

Процес створення веб-застосунку почався з вибору відповідних технологій. Було обрано мову програмування JavaScript для створення

інтерактивних HTML-документів. Це дозволяє виконувати обчислення на стороні клієнта без необхідності звертатися до сервера, що підвищує швидкість та ефективність веб-застосунку. Крім того, використовується бібліотека Bootstrap 3 для підтримки різних браузерів, що забезпечує широку сумісність веб-застосунку. Процес розробки зображено на рисунку 2.1

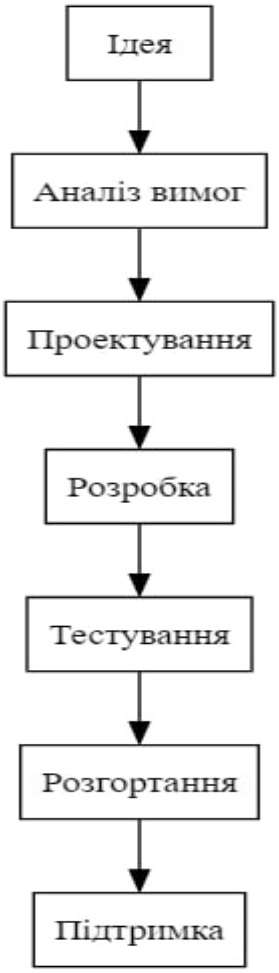

Рисунок 2.1 Процес створення додатку

2.7 Опис архітектури додатку

Веб-застосунок використовує мікросервісну архітектуру, що дозволяє розробляти, розгортати та масштабувати служби незалежно один від одного. Це допомагає забезпечити високу доступність та надійність веб-застосунку. Кожен мікросервіс відповідає за виконання конкретного набору завдань та може бути розгорнутий незалежно від інших, що дозволяє швидко вносити зміни та вдосконалення (рис. 2.2)

Рисунок 2.2 Архітектура додатку

2.8 Опис функціоналу додатку

Веб-застосунок надає різні функції, що допомагають користувачам керувати своїм часом на мобільних пристроях. Користувачі можуть встановлювати ліміти часу для різних додатків, отримувати повідомлення, коли вони перевищують ці ліміти, та відстежувати свій загальний час використання пристрою. Ці функції допомагають користувачу виконувати поставлені задачі вчасно (рис. 2.3)

Рисунок 2.3 Функціонал додатку

2.9 Процес розробки веб-додатку

Веб-застосунок для керування часом використання мобільного телефону є потужним інструментом, який допомагає користувачам контролювати та оптимізувати свій час, проведений з використанням мобільних пристроїв. Цей застосунок надає можливість встановлювати обмеження часу, визначати пріоритети та отримувати статистику щодо використання мобільного телефону.

Веб-застосунок має наступні ключові функціональні можливості:

1. Обмеження часу. Користувачі можуть встановлювати обмеження на загальний час використання мобільного телефону або на певні додатки чи функції пристрою. Наприклад, вони можуть встановити, що час використання соціальних мереж не повинен перевищувати 1 годину на день.

2. Пріорітети. Користувачі можуть встановлювати пріоритети для різних типів додатків або завдань на мобільному телефоні. Наприклад, вони можуть встановити, що робочі додатки мають бути використані протягом робочого часу, а розважальні додатки - поза робочим часом.

3. Нагадування. Застосунок надсилає нагадування та сповіщення користувачам про наближення до обмежень часу або про виконання встановлених пріоритетів. Наприклад, користувач може отримувати

сповіщення, коли він витрачає багато часу на певний додаток або коли наближається до ліміту використання часу.

4. Статистика. Застосунок надає користувачам детальну статистику про використання їх мобільних телефонів. Вони можуть переглядати час, витрачений на різні додатки, загальний час використання за певний період, звіти про пріоритети та іншу корисну інформацію.

5. Персоналізація. Веб-застосунок надає можливість персоналізувати налаштування та обмеження використання мобільного телефону відповідно до потреб кожного. Користувачі можуть встановлювати свої власні правила і обмеження з урахуванням їхнього графіку, пріоритетів та інших факторів.

Веб-застосунок для керування часом використання мобільного телефону створений з урахуванням потреб сучасних користувачів, які бажають забезпечити баланс між продуктивністю та здоровим використанням технологій. Він допомагає користувачам бути свідомими щодо свого часу та ефективно їм управляти.

Процес розробки веб-застосунку був поділений на кілька етапів:

1. Аналіз вимог. На цьому етапі важливо ретельно проаналізувати вимоги користувачів і визначити основні функції та можливості, які має мати веб-застосунок. Вимоги можуть включати обмеження часу, пріоритети, нагадування, статистику та інші функції.

2. Проектування інтерфейсу. На цьому етапі створюється дизайн інтерфейсу користувача (UI) і визначається, як будуть взаємодіяти з веб-застосунком. Важливо забезпечити зрозумілість, легкість використання та привабливий вигляд інтерфейсу.

3. Розробка функціональності. На цьому етапі розробляється функціональність веб-застосунку, яка включає у себе обмеження часу, пріоритети, нагадування, зберігання статистики та інші функції. Використовуються веб-технології, такі як HTML, CSS та JavaScript, для створення динамічного інтерфейсу та взаємодії з сервером.

4. Тестування і налагодження. Після розробки функціональності веб-застосунку важливо провести тестування, щоб переконатися, що він працює належним чином і відповідає вимогам користувачів. Тестування може

включати функціональне тестування, тестування коректності даних, тестування взаємодії та інші види.

Після успішного тестування веб-застосунок готовий до розгортання. Він може бути розміщений на веб-сервері і доступний для користувачів через Інтернет. Крім того, важливо забезпечити підтримку та можливість оновлення веб-застосунку для вирішення можливих проблем та вдосконалення функціональності.

2.10 Результат розробки веб-додатку

Технічна розробка веб-застосунку для керування часом використання мобільного телефону включає в себе використання різних технологій та компонентів. Нижче наведено загальний опис технічних аспектів розробки:

1. Фронтенд розробка. Для створення користувацького інтерфейсу (UI) були використані веб-технології, такі як HTML, CSS і JavaScript. HTML використовується для створення структури інтерфейсу, CSS - для оформлення і стилізації, а JavaScript - для динамічної взаємодії з користувачем і обробки подій.

2. Бекенд розробка. Включає створення серверної частини веб-застосунку. Для цього використані мови програмування, такі як PHP, Node.js, а також бази даних, наприклад MySQL. [4]

3. Автентифікація та авторизація. Для забезпечення безпеки та контролю доступу до веб-застосунку використані механізми автентифікації та авторизації. Це системи входу за допомогою логіна та пароля, використання токенів аутентифікації.

4. API розробка. Для забезпечення взаємодії між фронтендом та бекендом розроблено API (Application Programming Interface). API дозволяє передавати дані між клієнтом (фронтендом) та сервером (бекендом) та виконувати різні операції, такі як отримання даних, оновлення, видалення, тощо.

5. Тестування та налагодження. Під час розробки було виконано тестування, яке допомагає виявляти та виправляти помилки. Тестування

включає модульне тестування. Налагодження (debugging) використовується для виявлення та виправлення помилок в коді.

6. Розгортання і підтримка. Після успішного тестування та налагодження веб-застосунок готовий до розгортання. Він розміщений на веб-сервері і доступний для користувачів через Інтернет. Крім того, важливо забезпечити підтримку та можливість оновлення веб-застосунку для вирішення можливих проблем та вдосконалення функціональності.

Користувач може ввести свої дані і система буде підраховувати час від початку виконання завдання, так само буде відображатися дата і час коли робота виконувалася.Також користувач по бажанню може змінювати колір теми інтерфейсу.

В результаті розробки було отримано такий веб-додаток (рис. 2.4)

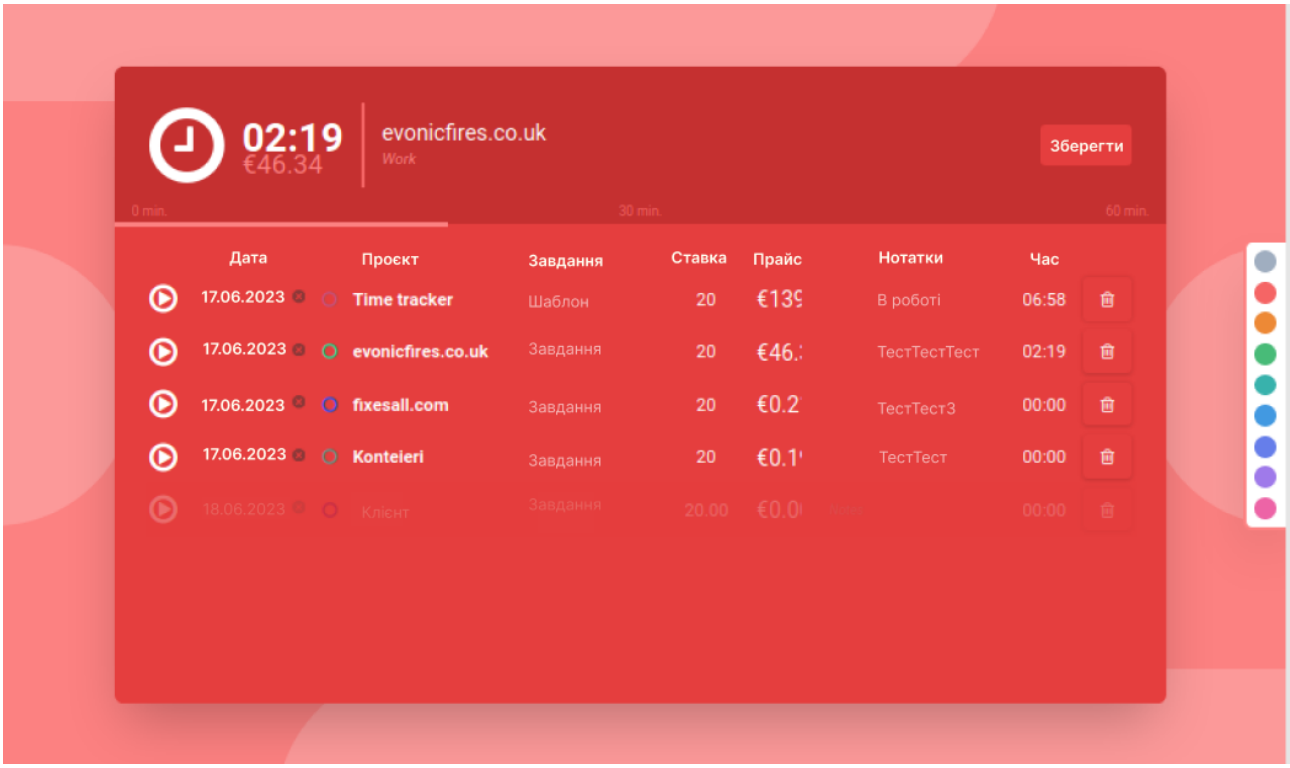

Рисунок 2.1 Інтерфейс веб-додатку

2.11 Висновок до другого розділу

У другому розділі було розглянуто важливі аспекти, пов'язані з розробкою веб-застосунку для керування часом використання мобільного

телефону. Було описано поняття мікросервісів та процес переходу на них, а також висвітлено основні принципи та вимоги до front-end розробки.

Мікросервісна архітектура є популярним підходом у сучасній розробці програмного забезпечення. Вона дозволяє розбити додаток на невеликі незалежні компоненти, що спрощує розробку, масштабування та підтримку системи. Процес переходу на мікросервіси вимагає аналізу архітектури, розбиття додатку на окремі сервіси, налагодження комунікації між ними та забезпечення високої доступності.

Front-end розробка має свої власні вимоги та підходи. Врахування респонсивного дизайну, інтуїтивного інтерфейсу, оптимізації продуктивності, безпеки даних та інтеграції з мобільними функціями є ключовими аспектами. Тестування веб-застосунку включає функціональне тестування, користувацьке тестування, тестування сумісності, тестування продуктивності та тестування безпеки.

У результаті виконання другого розділу було набуто розуміння того, що для успішної розробки веб-застосунку для керування часом використання мобільного телефону необхідно використовувати мікросервісну архітектуру, враховувати вимоги front-end розробки та проводити відповідне тестування. Ці кроки допомогли створити ефективний та надійний веб-застосунок, який забезпечить користувачам зручність у керуванні своїм часом на мобільних пристроях.

РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1 Актуальність безпеки життєдіяльності

Зусилля суспільства, створені задля розв'язання проблеми безпеки життєдіяльності нині, не дають результатів, адекватних розвитку проблеми. Негативні наслідки, особливо загибель людей та деградація навколишнього природного середовища, зростають із прискоренням та створюють загрозу подальшого розвитку світу. Серед різних напрямів вирішення проблеми БЗ найбільш яскраво виділяється проблема формування у нових поколінь людей ідеології безпеки життєдіяльності. Саме від ступеня усвідомлення проблеми БЗ людиною, прийняття ним обґрунтованих рішень щодо управління своїми діями та діями підлеглих, застосування ефективних рішень щодо забезпечення безпеки на різних рівнях залежить майбутнє суспільства. У цьому представляється особливо актуальною розробка сучасної методики викладання проблеми БЖД у вищій школі. Нормативна дисципліна «Безпека життєдіяльності» є інтегрованою дисципліною гуманітарно-технічного спрямування, яка узагальнює дані науково-технічної діяльності, формує понятійно-категорійний апарат, необхідний надалі для вивчення таких дисциплін як «Охорона праці», «Захист навколишнього середовища», «Громадянська ін, які вивчають конкретні шкідливі та небезпечні фактори життєдіяльності людини, способи захисту від них у конкретних окремих сферах діяльності людини. Предметом дисципліни «Безпека життєдіяльності» є вивчення закономірностей виникнення та розвитку небезпек, надзвичайних ситуацій, насамперед техногенного характеру, їх характеристики, можливий вплив на життя та здоров'я людини, системи заходів щодо захисту людини від небезпек та надзвичайних ситуацій у різних видах її життєдіяльності. Основними завданнями дисципліни є:

– формування у студентів світогляду про те, що в центрі уваги проблеми безпеки життєдіяльності стоїть людина як головна цінність держави;

– ідентифікація небезпечних та шкідливих факторів та створення безпечних умов життєдіяльності людей на території держави;

– оцінка технологічних процесів, виробництва, будівельних та інших об’єктів відповідно до сучасних вимог екологічної та безпеки, оцінка витрат на відновлення виробничих процесів, зупинених в умовах НС;

– прогнозування можливої обстановки на об’єктах в умовах НС.

Вироблення обґрунтованих рішень щодо захисту населення та персоналу об’єктів в умовах надзвичайних ситуацій. Розв’язання першого завдання, тобто. формування у студентів світогляду про пріоритет людини як головної цінності держави у проблемі безпека життєдіяльності на сьогоднішній момент є досить складним. Складність цього завдання обумовлюється багатьма причинами, серед яких слід зазначити такі:

– проблеми становлення ринкової економіки та, як наслідок, загострення соціальних протиріч у суспільстві, що заважають розвитку гуманітарних ідей;

– відсутність запитання фахівців із вищою освітою у системі національної економіки у кількості, пропорційному сукупному випуску таких фахівців державними та приватними вузами;

– низький рівень статистичної інформації про тенденції небезпек та надзвичайних ситуацій, їх наслідки, у курсах навчальних дисциплін вузів та засобах масової інформації;

– посилюється небезпечний техногенний вплив людини на природне середовище, що має в якості основи філософію споживання, культ речей, нестримну гонитву за благами життя.

Ідентифікація небезпечних та шкідливих факторів, надзвичайних ситуацій, є складним процесом у вигляді наступних причин:

– відсутність необхідного фінансування лабораторної бази ВНЗ з метою впровадження у навчальний процес систем приладів та вимірювальних пристроїв для ідентифікації небезпечних та шкідливих факторів, НС відповідно до світових стандартів;

– відсутність фундаментальних досліджень у галузі НС терористичного характеру, безпеки бізнесу та інших актуальних напрямів у сфері безпеки життєдіяльності.

У існуючих класифікаціях видів причин та видів НС не знайшли свого відображення соціально-політичні причини та відповідно соціально-політичні види НС, які у своїх сучасних проявах, у поєднанні з іншими видами НС, можуть істотно впливати на стабільність у суспільстві, психологічний та фізичний стан певних категорій населення, які приймаються рішення керівниками суб'єктів національної економіки. Разом з тим, Необхідно відзначити повну відсутність у сучасних підручниках та навчальних посібниках з проблем безпеки життєдіяльності аналізу взаємного впливу та підсумкового впливу на людину та довкілля НС різного характеру за їх одночасної дії. Необхідно відзначити, що за минулий період з 1995 по 2011 роки виникли такі тенденції, що загострили та ускладнили проблему безпеки життєдіяльності людини на сучасному етапі, які необхідно враховувати у процесі викладання предмета. 1. Прискорення розвитку екологічної кризи у світі та в тому числі в Україні. 2. Різке збільшення кількості надзвичайних ситуацій природного та техногенного характеру у світі та в Україні. 3. Збільшення кількості видів НС, ускладнення причин виникнення, посилення тяжкості наслідків. 2001 року постраждало 1013 осіб, з них 433 загинуло. [28] 4. Розвиток НС терористичного характеру для захоплення заручників, дестабілізації громадського порядку, спроби захоплення техногенно-небезпечних об'єктів, зброї, боєприпасів.

3.2 Вимоги безпеки до робочих місць для виконання робіт

Робоче місце має бути організоване відповідно до вимог стандартів, технічних умов та (або) методичних вказівок щодо безпеки праці. Воно має відповідати таким вимогам:

– забезпечувати можливість зручного виконання;

– враховувати фізичну тяжкість робіт;

– враховувати розміри робочої зони та необхідність пересування в ній працюючого;

– враховувати технологічні особливості процесу виконання робіт. Невиконання вимог до розташування та компонування робочого місця може призвести до отримання працівником виробничої травми або розвитку непрофесійного захворювання.Робоче місце програміста має відповідати вимогам ДСТУ 8604:2015. Конструкція обладнання та робочого місця при виконанні робіт у положенні сидячи повинна забезпечувати оптимальне положення працюючого, яке досягається регулюванням висоти робочої поверхні, висотисидіння, обладнанням простору для розміщення ноги заввишки підставки для ніг.

Мікроклімат у приміщенні є одним із найнеобхідніших для забезпечення сприятливих умов праці для працівників виробничим фактором, оскільки він дуже впливає на теплове самопочуття людини. Мікроклімат у виробничому приміщенні насамперед залежить від зовнішніх умов таких, як категорія робіт, період року, умови вентиляції, а також від особливостей самого технологічного процесу.

Можна виділити такі параметри, що характеризують мікроклімат у виробничих приміщеннях:

– температура повітря (t, ° C);

– температура поверхонь (t, ° C);

– відносна вологість повітря (?, %);

– швидкість руху повітря (?, м/с);

– інтенсивність теплового опромінення (I, Вт/м^2).

У виробничих приміщеннях при роботі з персональними обчислювальними машинами відбувається постійне виділення тепла обчислювальною технікою, засобами освітлення та іншими допоміжними приладами. І оскільки оператор ПК розташований у безпосередній близькості з джерелами виділення тепла, то висока температура повітря може сприяти швидкому перегріву програміста та швидкої втоми [32].

Робота програміста відноситься до категорії Іа, тобто вона провадиться сидячи та супроводжується незначною фізичною напругою. Інтенсивність енерговитрат організму для цієї категорії робіт становить до 120 ккал/год (до 139 Вт).

Оптимальні мікрокліматичні умови забезпечують загальне та локальне відчуття теплового комфорту при мінімальній напрузі механізмів терморегуляції, не викликають відхилень у стані здоров’я, створюють передумови для високого рівня працездатності та є кращими на робочих місцях. Оптимальні значення показників мікроклімату на робочих місцях виробничих приміщень згідно з ДСН 3.3.6.042-99 для категорії робіт Іа представлені у таблиці 3.1 [30].

Таблиця 3.1 - Оптимальні величини показників мікроклімату на робочих місцях виробничих приміщень

Період року	**Категорія робіт**	**Температура повітря, °С**	**Відносна вологість повітря, %**	**Швидкість руху повітря, м/с**
Холодний	Іа	22 – 24	60 – 40	0,1
Теплий	Іа	23 – 25	60 – 40	0,1

Згідно з вимогами ДСН 3.3.6.042-99, в кабінеті підтримується температура дорівнює 19-20 ° С, при відносній вологості в 55-58% [29]. Щоб досягти цього, необхідно проводити в приміщенні щоденне вологе прибирання та систематичне провітрювання. Недостатня освітленість робочої зони – шкідливий виробничий фактор, що регламентується СП 52.13330.2011.

3.3 Висновок до третього розділу

В більшості галузей промисловості науково-технічний прогрес викликає поліпшення умов праці, ліквідацію на багатьох підприємствах важкої ручної праці, запровадження нових ефективних засобів захисту на підприємстві. Інтенсивно розвивається інженерна психологія, що вивчає зв’язки конструкцій пультів управління важливими народногосподарськими об’єктами з особливостями сприйняття і переробки інформації операторами. Разом з тим, недостатнє використання можливостей науково-технічного прогресу, відсутність раціонального управління ним призводить у ряді випадків до погіршення умов праці. Застосування досягнень науки і техніки в промисловості, виробництві, за рахунок механізації, електрофікації та автоматизації виробничих процесів, застосування програмних пристроїв, рахунково-вирішуючих і електроннообчислювальних машин, автоматизованих систем управління (АСУ) змінюють умови і характер праці людини.

ВИСНОВОК

У цій роботі було розглянуто веб-застосунок для керування часом використання мобільного телефону. В першому розділі була описана актуальність проблеми, аналіз існуючих рішень та використання патерну проектування MVC. Було виявлено, що веб-застосунки є ефективним інструментом для організації робочого часу та досягнення більшої продуктивності.

У другому розділі було розглянуто перехід на мікросервіси, поняття front-end розробки та підходи до неї. Було з'ясовано, що мікросервісна архітектура дозволяє розбити додаток на невеликі компоненти, спрощуючи розробку та масштабування системи. Front-end розробка вимагає уваги до респонсивного дизайну, інтуїтивного інтерфейсу та оптимізації продуктивності.

Також було розглянуто підходи та вимоги до front-end розробки веб-застосунку для керування часом використання мобільного телефону. Було з'ясовано, що респонсивний дизайн, інтуїтивний інтерфейс, оптимізація продуктивності, безпека даних та інтеграція з мобільними функціями є ключовими аспектами розробки.

Крім того, було виявлено, що тестування веб-застосунку є важливим етапом розробки. Функціональне тестування, користувацьке тестування, тестування сумісності, тестування продуктивності та тестування безпеки допомагають забезпечити якість та надійність додатку перед його випуском.

З урахуванням усіх цих аспектів та підходів можна стверджувати, що розробка веб-застосунку для керування часом використання мобільного телефону є складним, але важливим процесом. Застосування мікросервісної архітектури, правильного підходу до front-end розробки та належного тестування дозволять створити ефективний та надійний веб-застосунок, який забезпечить користувачам зручність та ефективність у керуванні їхнім часом.

ПЕРЕЛІК ПОСИЛАНЬ

1. Документація HTML [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/HTML.

2. Документація CSS [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/CSS.

3. Документація JS [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

4. Документація PHP [Електронний ресурс] – Режим доступу до ресурсу: https://www.php.net/manual/en/.

5. jQuery [Електронний ресурс] – Режим доступу до ресурсу: https://api.jquery.com/.

6. JIRA [Електронний ресурс] – Режим доступу до ресурсу: https://support.atlassian.com/jira-software-cloud/docs/

7. MVC (Model-View-Controller): [Електронний ресурс] – Режим доступу доресурсу:https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller.

8. "Learning Web Design" [Електронний ресурс] – Режим доступу до ресурсу: https://learningwebdesign.com/.

9. "Web Application Architecture: Principles, Protocols, and Practices" [Електронний ресурс] – Режим доступу до ресурсу: http://bedford-computing.co.uk/learning/wp-content/uploads/2016/07/Web-Application-Architecture-Principles-Protocols-and-Practices.pdf.

10. "Designing Web Applications" [Електронний ресурс] – Режим доступу до ресурсу: https://nathanbarry.com/webapps/.

11. Beckman, Brian (1994), Theory of Spectral Graph Layout, Tech. Report MSR-TR-94- 04, Microsoft Research

12. Tytenko, S. V. INTERACTIVE CONCEPT MAPS IN ONTOLOGY-ORIENTED INFORMATION AND LEARNING WEB-SYSTEMS. KPI Science News, no. 2, pp. 24– 36, 2019. doi:10.20535/kpi-sn.2019.2.167515

13. S.V. Tytenko, “Construction of didactic ontology based on the analysis of

conceptthesis model elements", Naukovi Visti NTUU KPI, no. 1, pp. 82—87, 2010.

14. Nesbit, J. C., & Adesope, O. O. (2006). Learning With Concept and Knowledge Maps: A Meta-Analysis. Review of Educational Research, 76(3), 413–448. doi:10.3102/00346543076003413

15. S. Puntambekar et al., "Improving navigation and learning in hypertext environments with navigable concept maps", HumanComputer Interaction, vol. 18, no. 4, pp. 395—428, 2003. doi: 10.1207/S15327051HCI1804_3

16. E. Andres et al., "An adaptive Theory of Computation online course in ActiveMath", in Proc. 5th Int. Conf. Computer Science & Education, 2010, pp. 317—322. doi: 10.1109/ICCSE.2010.5593624

17. Хорець В.О. Мобільний додаток "Органайзер студента"/ Хорець В.О., Шушура О.М. // II Всеукраїнська науково-практична інтернет-конференція молодих вчених та здобувачів вищої освіти «СУЧАСНА МОЛОДЬ В СВІТІ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ» — Херсон, 2021.

18. Pro Git 2nd Edition [Електронний ресурс] / Scott Chacon, Ben Straub — 2014 — Режим доступу: https://git-scm.com/book/en/v2 .

19. Документація Docker Hub [Електронний ресурс] — Режим доступу: https://docs.docker.com/docker-hub/ .

20. Napoli M. Beginning Flutter: A Hand on Guide to App Development / Marco Na- poli, 2019. – 528 с.

21. Biessek A. Flutter for Beginners / Alessandro Biessek, 2019. – 512 с.

22. Flutter [Електронний ресурс] // flutter.github.io. - 2017 - Режим доступу до ресурсу: https://flutter.github.io/samples/#

23. Rap P. Beginning App Development with Flutter: Create Cross-Platform Mobile Apps / Rap Payne, 2019. – 309 с.

24. Moore K. Flutter Apprentice // K. Moore, M. Katz, V. Ngo, 2019. – 508 с.

25. Miola A. Flutter Complete Reference: Create beautiful, fast and native apps for any device // Miola Albero, 2020. – 408 с.

26. Windmill E. Flutter in Action // Eric Windmill, 2019. – 255 с.

27. Freitas E. Flutter Succinctly // Ed Freitas, 2019. – 129 с.

28. Mainkar P. Google Flutter Mobile Development Quick Start Guide: Get up and running with iOS and Android mobile app development // P. Mainkar, S.

Giordano, 2019. – 152 с.

29. Clow M. Learn Google Flutter Fast: 65 Example Apps // Mark Clow, 2019. – 476 с.

30. Zammetti F. Practical Flutter: Improve your Mobile Developemnt with Google’s Latest Open-Source SDK // Frank Zammetti, 2019. – 416 с.

ДОДАТКИ

Додаток А

```
<!DOCTYPE html>
<html lang="uk">
<head>
<meta charset="utf-8">
<title>Трекер часу</title>
<link rel="icon" href="favicon.ico" type="image/x-icon" />
<link rel="stylesheet" type="text/css" href="style.css">
<!-- DataTables CSS, jQuery, DataTables, jQuery UI, jQuery UI Lightness, etc. -->
<script type="text/javascript" charset="utf-8" src="//ajax.aspnetcdn.com/ajax/jQuery/jquery-1.8.2.min.js"></script>
<script src="//ajax.aspnetcdn.com/ajax/jquery.ui/1.9.0/jquery-ui.min.js"></script>
<link rel="stylesheet" type="text/css" href="//ajax.aspnetcdn.com/ajax/jquery.ui/1.9.0/themes/smoothness/jquery-ui.css">
<script type="text/javascript" charset="utf-8" src="globalize.js"></script>
<script type="text/javascript" charset="utf-8" src="globalize.culture.uk-UA.js"></script>
<script type="text/javascript" charset="utf-8" src="jquery.mousewheel.js"></script>
<script type="text/javascript" charset="utf-8" src="//ajax.aspnetcdn.com/ajax/jquery.dataTables/1.9.4/jquery.dataTables.min.js"></script>
<script type="text/javascript" charset="utf-8" src="script.js"></script>
<link rel="stylesheet" type="text/css" href="//ajax.aspnetcdn.com/ajax/jquery.dataTables/1.9.4/css/jquery.dataTables_themeroller.css">
</head>
<body>
<h1>Трекер часу</h1>
<?php
$username = $_SESSION["username"];
$month = $_SESSION["filterMonth"];
$year = $_SESSION["filterYear"];
?>
<div id="username">Привіт, <b><?php echo $username; ?></b>! (<a href="logout.php">Вийти</a>)</div>
<div id="time_range">
Фільтр:
    <select name="month" id="month" class="filter">
                <option <?php echo $month == 0 ? 'selected="selected"':"" ?> value="0">---</option>
                <option <?php echo $month == 1 ? 'selected="selected"':"" ?> value="01">01 Січень</option>
```

```
                        <option <?php echo $month == 2 ? 'selected="selected"':"" ?> value="02">02 Лютий</option>
                        <option <?php echo $month == 3 ? 'selected="selected"':"" ?> value="03">03 Березень</option>
                        <option <?php echo $month == 4 ? 'selected="selected"':"" ?> value="04">04 Квітень</option>
                        <option <?php echo $month == 5 ? 'selected="selected"':"" ?> value="05">05 Травень</option>
                        <option <?php echo $month == 6 ? 'selected="selected"':"" ?> value="06">06 Червень</option>
                        <option <?php echo $month == 7 ? 'selected="selected"':"" ?> value="07">07 Липень</option>
                        <option <?php echo $month == 8 ? 'selected="selected"':"" ?> value="08">08 Серпень</option>
                        <option <?php echo $month == 9 ? 'selected="selected"':"" ?> value="09">09 Вересень</option>
                        <option <?php echo $month == 10 ? 'selected="selected"':"" ?> value="10">10 Жовтень</option>
                        <option <?php echo $month == 11 ? 'selected="selected"':"" ?> value="11">11 Листопад</option>
                        <option <?php echo $month == 12 ? 'selected="selected"':"" ?> value="12">12 Грудень</option>
                    </select>
                    <select name="year" id="year" class="filter">
                        <option <?php echo $year == 0 ? 'selected="selected"':"" ?> value="0">---</option>
                        <?php for ($i = 2011; $i <= date('Y')+1; $i++): ?>
                            <option <?php echo $year == $i ? 'selected="selected"':"" ?> value="<?php echo $i ?>"><?php echo $i ?></option>
                        <?php endfor; ?>
                    </select>
                </div><br />
                <table id="times">
                    <thead>
                      <tr>
                                <th>День тижня</th>
                        <th>Дата</th>
                        <th>Тривалість</th>
                        <th>Коментар</th>
                      </tr>
                    </thead>
                </table>
                <p id="sum"></div>
                <script type="text/javascript">
                $(document).ready(function(){
                    $('#date').val("<?php echo date("d.m.y"); ?>");
                    $('#time').val("<?php echo date("H:i"); ?>");
```

```
});
</script>
<h3>Новий запис</h3>
<div id="new-entry">
<form id="form-new-entry">
    <label for="date">Дата:</label>
    <input type="text" id="date" class="textbox" maxlength="8" />
    <label for="time">Час:</label>
    <input type="time" id="time" maxlength="5" />
    <label for="duration">Тривалість:</label>
    <input type="text" id="duration" maxlength="5" />
    <label for="comment" id="label-comment">Коментар:</label>
    <input type="text" id="comment" class="textbox" maxlength="100" />
    <input type="button" id="save" value="OK" />
    <p id="saving"><img src="spinner.gif" /> Збереження...</p>
    <p id="saved"><b>Збережено!</b></p>
</form>
</div>
<div title="Помилка" style="display: none" id="dialog-error">
Введені дані некоректні.<br />Будь ласка, перевірте ваші дані.
</div>
</body>
</html>
?>
<!DOCTYPE html>
<?php
require_once "db.php";

$error_message = "";

if($_SERVER["REQUEST_METHOD"] == "POST")
{
    $username = mysql_real_escape_string($_POST["username"]);
    $password = $_POST["password"];

    $q = mysql_query("SELECT id, password FROM users WHERE username
= '$username'");
    if(mysql_num_rows($q) > 0)
    {
                $r = mysql_fetch_assoc($q);
                if(!bcrypt_check($password, $r["password"]))
                    $error_message = "Невірний пароль!";
                else
                {
                    if(isset($_POST["rememberme"])                    &&
$_POST["rememberme"] == "1")
                    {
```

```
                                        $token = md5(time());
                                        mysql_query("UPDATE users SET
token='$token' WHERE username='$username'"); //token зберегти в базі даних
                                        setcookie("timetracker_autologin", $token,
time()+60*60*24*30);
                                }
                                else
                                        mysql_query("UPDATE users SET
token=NULL WHERE username='$username'"); //видалити token

                                        makeLogin($username, $r["id"]);
                                }
                        }
                        else
                                $error_message = "Користувача $username не
знайдено!";
                }
                else if(isset($_COOKIE["timetracker_autologin"]))
                {
                        $cookie = $_COOKIE["timetracker_autologin"];
                        $q = mysql_query("SELECT id, username FROM users WHERE token =
'$cookie'");
                        if(mysql_num_rows($q) > 0) //token відповідає базі даних
                        {
                                $r = mysql_fetch_assoc($q);
                                makeLogin($r["username"], $r["id"]);
                        }
                }

                function makeLogin($username, $uid)
                {
                        session_start();
                        $_SESSION["logged_in"] = true;
                        $_SESSION["username"] = $username;
                        $_SESSION["uid"] = $uid;
                        $_SESSION["filterMonth"] = date("m");
                        $_SESSION["filterYear"] = date("Y");
                        header("Location: index.php");
                        die("Перенаправлення...");
                }
                ?>
                php
                Copy code
                <html>
                <head>
                <link rel="icon" href="favicon.ico" type="image/x-icon" />
                <link rel="stylesheet" type="text/css" href="style_login.css">
```

```
<title>Трекер часу :: Вхід</title>
</head>
<body>
<form action="login.php" method="post">
<div id="login_box">
    <h2>Трекер часу</h2>
    <div id="error"><?php echo $error_message; ?></div>
    <table>
    <tr>
                <td><label for="username">Ім'я користувача: </label></td>
                <td><input type="text" id="username" name="username"/></td>
    </tr>
    <tr>
                <td><label for="password">Пароль: </label></td>
                <td><input type="password" id="password" name="password"/></td>
    </tr>
    <tr id="login_remember_row">
                <td><label for="rememberme">Запам'ятати мене: </label></td>
                <td><input type="checkbox" id="rememberme" name="rememberme"/ value="1"></td>
    </tr>
    <tr>
                <td><input type="submit" value="Увійти" /></td>
    </tr>
    </table>
</div>
</form>
</body>
<?php

function bcrypt_check ($password, $stored)
{
   return crypt($password, $stored) == $stored;
}

?>
require_once "auth.php";
require_once "db.php";

$date = mysql_real_escape_string($_POST['date']);
$duration = mysql_real_escape_string($_POST['duration']);
$comment = mysql_real_escape_string($_POST['comment']);
```

```
list($day, $month, $year) = explode('.', $date);

$date = date('Y-m-d', mktime(0, 0, 0, $month, $day, $year));

$uid = $_SESSION["uid"];

mysql_query("INSERT INTO data (uid, date, duration, comment) VALUES ($uid, '$date', '$duration', '$comment')");

?>

 <?php
session_start();

if(!isset($_SESSION["logged_in"]))
{
    header("Location: login.php");
    die("Перенаправлення...");
}

require_once "db.php";

$username = $_SESSION["username"];
$uid = $_SESSION["uid"];
$month = $_SESSION["filterMonth"];
$year = $_SESSION["filterYear"];

$month_name = array(1 => "Січень", "Лютий", "Березень", "Квітень", "Травень", "Червень", "Липень", "Серпень", "Вересень", "Жовтень", "Листопад", "Грудень");

$filter = "";
if($month != 0 && $year != 0)
    $filter = " AND MONTH(date) = $month AND YEAR(date) = $year";
else if($month != 0)
    $filter = " AND MONTH(date) = $month";
else if($year != 0)
    $filter = " AND YEAR(date) = $year";

$q = mysql_query("SELECT id, date, duration, comment FROM data WHERE uid = $uid$filter ORDER BY date DESC");
$rows = array();
while($r = mysql_fetch_assoc($q)) {
   $rows[] = $r;
}
$data = array('username' => $username, 'month' => $month, 'year' => $year, 'data' => $rows);
```

```
echo json_encode($data);
?>
 <?php
session_start();

if(!isset($_SESSION["logged_in"]))
{
    header("Location: login.php");
    die("Перенаправлення...");
}

require_once "db.php";

$username = $_SESSION["username"];
$uid = $_SESSION["uid"];
$month = $_SESSION["filterMonth"];
$year = $_SESSION["filterYear"];

$month_name = array(1 => "Січень", "Лютий", "Березень", "Квітень", "Травень", "Червень", "Липень", "Серпень", "Вересень", "Жовтень", "Листопад", "Грудень");

$filter = "";
if($month != 0 && $year != 0)
    $filter = " AND MONTH(date) = $month AND YEAR(date) = $year";
else if($month != 0)
    $filter = " AND MONTH(date) = $month";
else if($year != 0)
    $filter = " AND YEAR(date) = $year";

$q = mysql_query("SELECT id, date, duration, comment FROM data WHERE uid = $uid$filter ORDER BY date DESC");
$rows = array();
while($r = mysql_fetch_assoc($q)) {
   $rows[] = $r;
}
$data = array('username' => $username, 'month' => $month, 'year' => $year, 'data' => $rows);
echo json_encode($data);
?>
<?php
session_start();

if(!isset($_SESSION["logged_in"]))
{
    header("Location: login.php");
    die("Перенаправлення...");
```

```
}

require_once "db.php";

$username = $_SESSION["username"];
$uid = $_SESSION["uid"];
$month = $_SESSION["filterMonth"];
$year = $_SESSION["filterYear"];

$month_name = array(1 => "Січень", "Лютий", "Березень", "Квітень", "Травень", "Червень", "Липень", "Серпень", "Вересень", "Жовтень", "Листопад", "Грудень");

$filter = "";
if($month != 0 && $year != 0)
    $filter = " AND MONTH(date) = $month AND YEAR(date) = $year";
else if($month != 0)
    $filter = " AND MONTH(date) = $month";
else if($year != 0)
    $filter = " AND YEAR(date) = $year";

$q = mysql_query("SELECT id, date, duration, comment FROM data WHERE uid = $uid$filter ORDER BY date DESC");
$rows = array();
while($r = mysql_fetch_assoc($q)) {
   $rows[] = $r;
}
$data = array('username' => $username, 'month' => $month, 'year' => $year, 'data' => $rows);
echo json_encode($data);
?>
<?php
session_start();

if(!isset($_SESSION["logged_in"]))
{
    header("Location: login.php");
    die("Перенаправлення...");
}

require_once "db.php";

$id = $_POST['id'];
$date = $_POST['date'];
$duration = $_POST['duration'];
$comment = $_POST['comment'];
```

```
mysql_query("UPDATE data SET date = '$date', duration = '$duration', comment = '$comment' WHERE id = $id");

?>
 <?php
session_start();

if(!isset($_SESSION["logged_in"]))
{
    header("Location: login.php");
    die("Перенаправлення...");
}

require_once "db.php";

$id = $_POST['id'];
$date = $_POST['date'];
$duration = $_POST['duration'];
$comment = $_POST['comment'];

mysql_query("UPDATE data SET date = '$date', duration = '$duration', comment = '$comment' WHERE id = $id");

?>
 <?php
session_start();

if(!isset($_SESSION["logged_in"]))
{
    header("Location: login.php");
    die("Перенаправлення...");
}

require_once "db.php";

$id = $_POST['id'];

mysql_query("DELETE FROM data WHERE id = $id");

?>
 <?php
session_start();

if(!isset($_SESSION["logged_in"]))
{
    header("Location: login.php");
    die("Перенаправлення...");
```

```
}

require_once "db.php";

$id = $_POST['id'];

mysql_query("DELETE FROM data WHERE id = $id");

?>
 <?php
session_start();

if(!isset($_SESSION["logged_in"]))
{
    header("Location: login.php");
    die("Перенаправлення...");
}

require_once "db.php";

$username = $_SESSION["username"];
$uid = $_SESSION["uid"];
$month = $_POST["month"];
$year = $_POST["year"];

$_SESSION["filterMonth"] = $month;
$_SESSION["filterYear"] = $year;
echo "OK";
?>
 <?php
session_start();
if(!isset($_SESSION["logged_in"]))
{
    header("Location: login.php");
    die("Перенаправлення...");
}
require_once "db.php";
$username = $_SESSION["username"];
$uid = $_SESSION["uid"];
$month = $_POST["month"];
$year = $_POST["year"];
$_SESSION["filterMonth"] = $month;
$_SESSION["filterYear"] = $year;

echo "OK";

?>
```