НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)

« »_____2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою “Інженерія програмного забезпечення комп'ютерних систем” спеціальності 121 “Інженерія програмного забезпечення”

на тему: Додаток для створення резюме

Виконав : студент 4 курсу, групи ІП-94
(шифр групи)

Бондік Кирило Геннадійович
(прізвище, ім’я, по батькові) (підпис)

Керівник ас. Валько.В.В.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант к.т.н., д.т.н. Волокита А.М.
(назва розділу) (посада,вченезвання,науковий ступінь,прізвищета ініціали) (підпис)

Рецензент к.т.н., д.т.н. Галушко Д.О.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2023

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

"Інженерія програмного забезпечення комп'ютерних систем"

спеціальності 121 "Інженерія програмного забезпечення"

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)

_ « »_____2023 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Бондіка Кирила Геннадійовича

1. Тема проєкту *Додаток для створення резюме*
керівник проєкту ас. Валько В.В.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

2. Термін здачі студентом закінченого проєкту *8 червня 2023 р.*

3. Вихідні дані до проєкту *технічна документація, теоретичні дані.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)

Розділ 1. Огляд існуючих аналогів.

Розділ 2. Аналіз інструментів для створення багатокористувацьких веб – додатків.

Розділ 3. Розробка додатку для створення резюме.

Розділ 4. Представлення роботи та графічного вигляду проекту.

5. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання Видав	Завдання Прийняв

6. Дата видачі завдання «10» грудня 2022 р.

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проекту	Примітки
1.	*Затвердження теми проекту*	*10.12.2022-15.12.2022*	
2.	*Вивчення та аналіз завдання*	*15.12.2022-15.03.2023*	
3.	*Розробка архітектури та загальної структури системи*	*15.03.2023-25.03.2023*	
4.	*Розробка структур окремих підсистем*	*25.03.2023-5.04.2023*	
5.	*Програмна реалізація системи*	*5.04.2023-15.04.2023*	
6.	*Оформлення пояснювальної записки*	*15.04.2023-11.06.2023*	
7.	*Захист програмного продукту*	*25.04.2023*	
8.	*Передзахист*	*08.06.2023*	
9.	*Захист*	*24.06.2023*	

Студент-дипломник Бондік Кирило
(підпис)

Керівник проекту Володимир Валько
(підпис)

АНОТАЦІЯ

В бакалаврському дипломному проекті створено веб – додаток для створення резюме.

Даний веб – додаток може бути використаний для створення резюме. Додаток дозволяє здійснювати реєстрацію та додавання особистої інформації для подальшого створення резюме. Обробка авторизаційних даних користувача, зберігання та завантаження даних (контактний номер, електронна адреса, освіта, загальна інформація і т.д.) відбувається в режимі реального часу.

Веб – додаток було створено за допомогою html, js, php, css, mysql на редакторі vscode.

ANNOTATION

A web application for creating a resume was created in the bachelor diploma project.

This web application can be used to create a resume. The application allows you to register and add personal information for further resume creation. Processing of user authorization data, storage and uploading of data (contact number, e-mail address, education, general information, etc.) takes place in real time.

The web application was created using html, js, php, css, mysql on the vscode editor.

Справки	Формат	*Значення*	*Найменування*	Кіл. листів	№ екземпля	*Додаток*
			Документація загальна			
			Знову розроблена			
	А4	***ІАЛЦ.467200.002 ТЗ***	Додаток для створення	3		
			Резюме			
			Технічне завдання			
	А4	***ІАЛЦ.467200.003 ПЗ***	Додаток для створення	66		
			Резюме			
			Пояснювальна записка			
	А4	***ІАЛЦ.4672008.005 Д1***	Додаток для створення	1		
			Резюме			
			Діаграма компонентів			
			(структурна схема)			
	А4	***ІАЛЦ.4672008.006 Д2***	Додаток для створення	1		
			Резюме			
			Діаграма використання			
			(функціональна схема)			
	А4	***ІАЛЦ.4672008.007 Д3***	Додаток для створення	1		
			Резюме			
			Алгоритм дій програмного			
			забезпечення			
			(принципова схема)			
	А4	***ІАЛЦ.467200.008 Д4***	Додаток для створення	13		
			Резюме			
			Лістинг коду			

					ІАЛЦ.467200.001 ОА	
Зм	***Лист***	***№ докум.***	***Підп***	***Дата***		
Розроб		Бондік К.Г.			*Додаток для створення резюме*	Літ. / Аркуш: 1 / Аркушів: 1
Перев		Валько В.В.				
					Опис альбому	*КПІ ім. Ігоря Сікорського,ФІОТ,ІП-94*

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Додаток для створення резюме*»

Київ – 2023

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на дипломний проект бакалавра за темою: «Веб – додаток для створення резюме». Може бути практично використаний користувачами для зручності створення резюме.

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи кваліфікаційно-освітнього рівня «бакалавр інженерії програмного забезпечення», затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського».

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даної розробки є створення веб – додатку із покращеним функіоналом вже існуючих рішень для легкого створення резюме користувачем.

4 ДЖЕРЕЛА РОЗРОБКИ

Основними джерелами розробки є публікації на дану тему в мережі Інтернет, загальнодоступна документація існуючих програм, наукові статті та науково-технічна література з теорії і практики програмування.

5 ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

Розроблена система має виконувати такі вимоги:

- Простий і інтуїтивно-зрозумілий функціонал.
- Надати можливість користувачам реєструватися в додатку.
- Надати можливість користувачам публікувати та редагувати власну інформацію про себе.
- Надати можливість користувачам переглядати інформацію про себе.
- Надати можливість користувачам звернутись в онлайн підтримку в будь – якому питанні.

- Надати вичерпну та зрозумілу документацію для розробленого продукту.

6 ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	10.12.2022-15.12.2022
Вивчення та аналіз завдання	15.12.2022-15.03.2023
Розробка архітектури та загальної структури системи	15.03.2023-25.03.2023
Розробка структур окремих частин системи	25.03.2023-5.04.2023
Програмна реалізація системи	5.04.2023-15.04.2023
Виправлення помилок	15.04.2023-20.05.2023
Оформлення пояснювальної записки	25.04.2023

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Додаток для створення резюме*»

Київ – 2023

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

VSCODE – Visual Studio Code

HTML -Мова гіпертекстової розмітки

ЦА – цільова аудиторія

ІТ – Інформаційні технології

БД – База даних

CSS –каскадні таблиці стилів

ВСТУП

Уже зараз створено досить багато різноманітних веб – додатків для створення резюме. Інформаційні технології змінили спосіб влаштування на роботу. Зараз будь – яка людина може з легкістю розказати про себе й свої навички не виходячи з дому. Наведемо статистику стосовно резюме, створених із належною увагою до загального дизайну, розташуванню елементів, логічності викладу: [1, 2]

1. На 70% більше переглядів.
2. У 2 рази більше запрошень на співбесіду.
3. На 50% менше часу, витраченого на пошук роботи.

Для того, щоб привернути увагу, створюють привабливі резюме за допомогою спеціальних сервісів онлайн, або ж використовують спеціальні онлайн та оффлайн програми для дизайну. Створення резюме за допомогою веб-додатка має кілька переваг, які зроблять цей процес зручнішим і ефективнішим. Ось деякі з них:

1. Зручний доступ: створювати резюме з будь-якого пристрою з підключенням до Інтернету, такого як комп'ютер, смартфон або планшет. Не потрібно встановлювати додаткове програмне забезпечення, що полегшує процес створення резюме.
2. Шаблони і готові форматування: надавати широкий вибір шаблонів, дизайнів та форматування. Це допоможе зекономити час і забезпечить професійний вигляд резюме.
3. Зручність редагування: надавати можливість зручного редагування. Можливість легко додавати, видаляти або змінювати розділи, деталі і форматування. Це робить процес створення резюме більш гнучким і дозволяє швидко вносити зміни в разі потреби.
4. Легке збереження: після створення резюме легко зберегти його в різних форматах.

5. Спрощений процес оновлення: веб-додатки можуть забезпечити легкий доступ до попередніх резюме. Швидко внести потрібні зміни і знову зберегти оновлену версію.

Загалом, веб-додаток для створення резюме може спростити процес його створення, забезпечити широкий вибір шаблонів, легке редагування і зручність доступу до вашого резюме з будь-якого місця. Цей додаток дозволить ефективно представити професійну інформацію та зберегти її у зручних форматах для подальшого використання. Додатки для створення резюме постійно оновлюються, щоб відповідати потребам ринку праці. Додатки можуть включати актуальні розділи та формати, які популярні серед роботодавців і рекрутерів. Використовуючи такий додаток, можете бути впевнені, що резюме відповідає сучасним стандартам та вимогам. Також додаток надає структуру, швидкість, професійний вигляд та адаптацію до сучасних вимог працевлаштування. Використання такого додатку може полегшити процес створення резюме та покращити шанси на успіх у пошуку роботи.

Отже, метою цього проекту є створити веб – додаток із покращеним функціоналом існуючих рішень, який буде допомагати користувачам створювати резюме.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ АНАЛОГІВ

Завданням дипломної роботи є створення веб-додатку для створення резюме. Розробка веб-додатку для створення резюме - це створення сучасного маркетингового інструменту. Хороший веб-додаток дозволить працівникам привернути увагу більшої кількості роботодавців. Після того, як функціональні блоки додатку створені, можна приступати до роботи над контентом. Для основних розділів розробляється індивідуальний дизайн. Тому перед початком розробки проводиться ретельний аналіз, який допомагає виявити і створити унікальні переваги. Веб-додаток вимагає реєстрації та авторизації користувача. Він також дозволяє користувачам додавати та редагувати інформацію про себе. Додаток також пропонує онлайн-підтримку.

Порівнюючи інструменти, технології та методи розробки для вирішення завдання створення веб-додатку, слід враховувати основні нюанси завдання-додавання, редагування інформації і авторизація користувачів.

1.1 Порівняння аналогів

Cvmaker - це абревіатура від латинських слів "curriculum vitae", що перекладаються як "життєвий шлях". Професійне резюме дає короткий огляд і гарне уявлення про життя людини. Сайт допомагає користувачам створити резюме для подальшого працевлаштування. Резюме включає в себе освіту і кваліфікації, досвід роботи, навички та важливі якості.

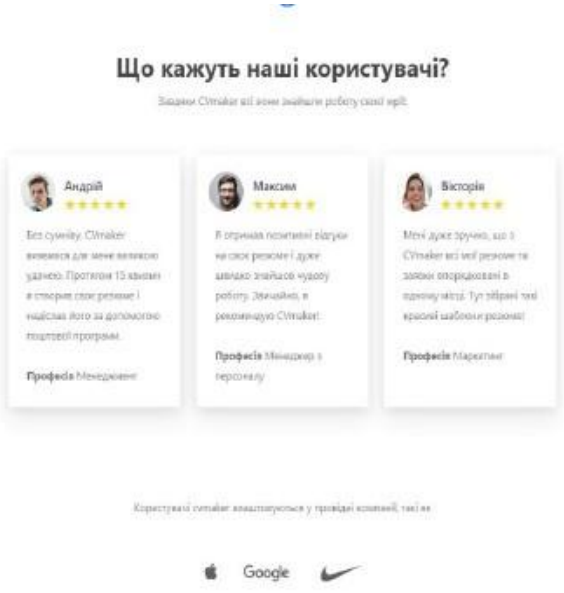


Рисунок. 1.1 Відгуки клієнтів даного веб – сайту

За допомогою резюме потенційний роботодавець зможе швидко отримати чітке уявлення про навички, досвід роботи і знання, і вирішити, чи варто запрошувати на співбесіду.

Рисунок. 1.2 Резюме,що виділяється

Сервіс Cvmaker допомагає створити резюме, яке виділяється. Це допомагає користувачам швидко працевлаштуватися. Переваги даного веб – сайту:

За допомогою CV maker, професійне резюме завжди під рукою. Незалежно від того, що вирішите: швидко і легко створити його самостійно за допомогою конструктора резюме чи замовити його у професіоналів - резюме завжди буде виділятися серед інших.[2]

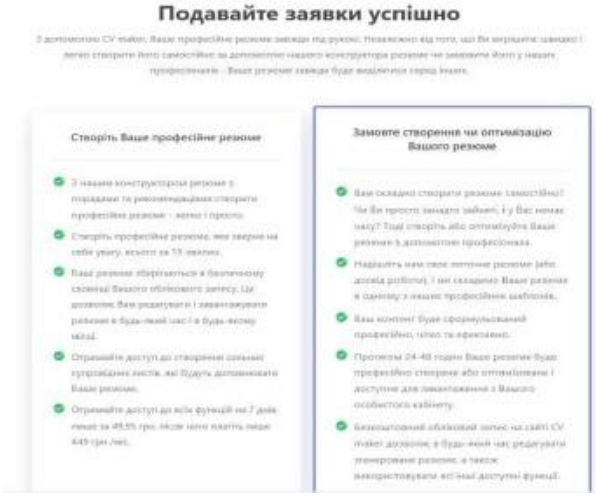


Рисунок 1.3 Поради до створення резюме

Як скласти ідеальне резюме? Скористатись **конструктором резюме,** де знайдете поради до кожного розділу, які допоможуть скласти резюме найкращим чином. Крім того, ось декілька загальних порад:

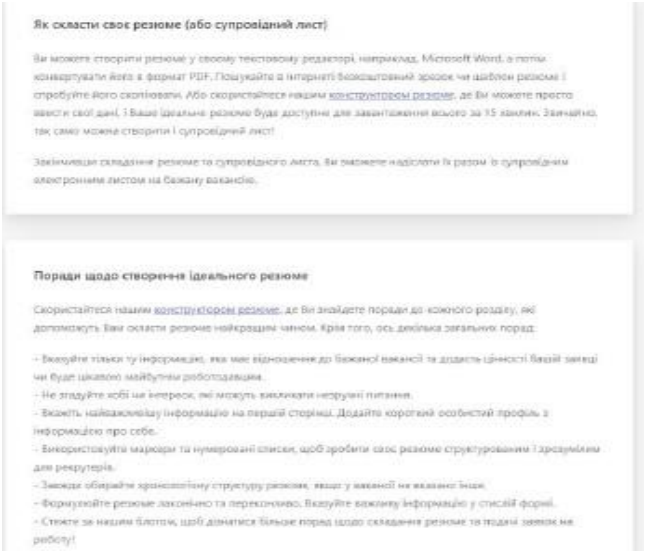


Рисунок 1.4 Як скласти ідеальне резюме

Отже, CVmaker є онлайн-інструментом, яка допомагає людям створювати професійні резюме . CVmaker надає корисні поради щодо того, як написати ефективне резюме та як використовувати ключові слова, щоб підвищити його шанси на успіх у пошуку роботи.

Canva – безкоштовний конструктор для створення резюме. Налаштувавши шаблони, користувач додатку заощадить час, адже все, що йому потрібно зробити, — це додати інформацію про свій професійний досвід і налаштувати наявний дизайн за необхідності. Шаблони Canva затверджені дизайнерами, тому користувачу не потрібно турбуватися про те, як вони виглядають. Адже вони вже виглядають чудово!

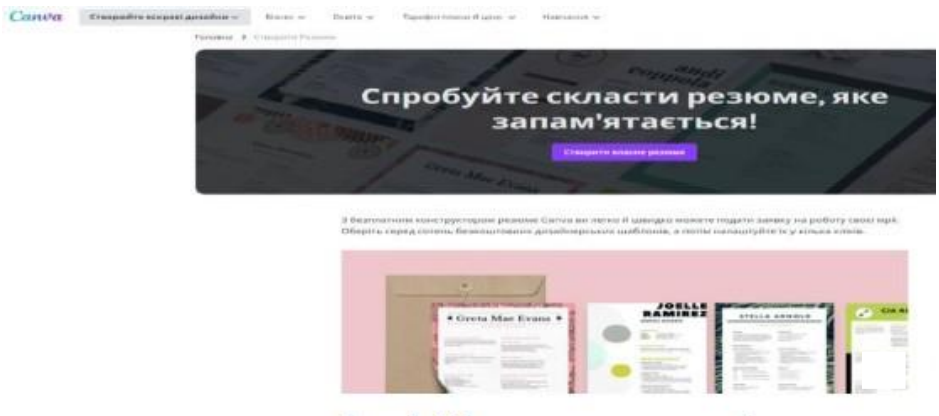


Рисунок 1.5 Резюме, яке запам'ятається

Резюме, яке запам’ятається. З безкоштовним конструктором резюме Canva користувач додатку легко й швидко може подати заявку на роботу своєї мрії. Користувач обирає серед сотень безкоштовних дизайнерських шаблонів, а потім налаштовує їх у кілька кліків.

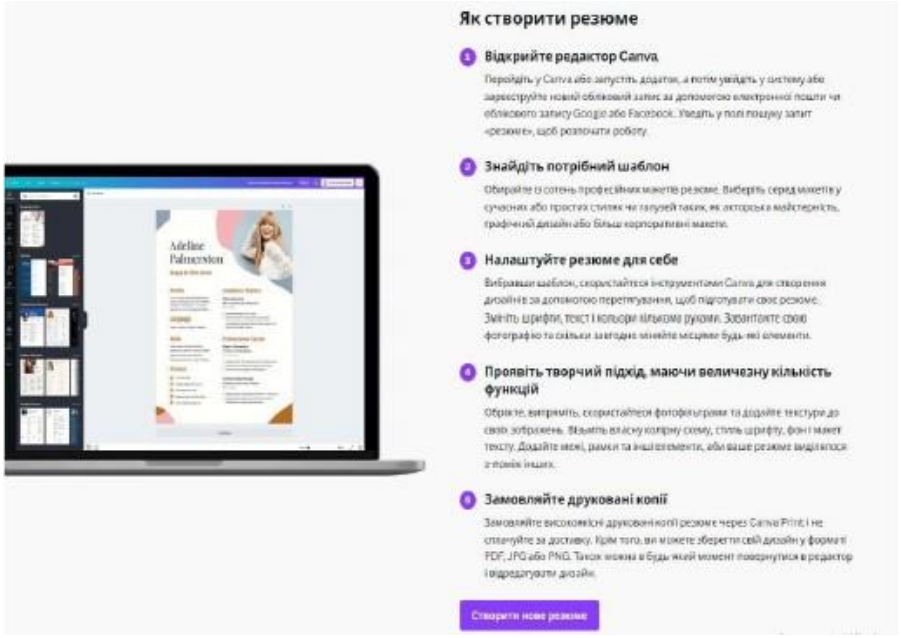


Рисунок 1.6 Як створити резюме на сайті

Далі описана покрокова інструкція для створення резюме, за допомогою якої, користувач зможе швидко створити резюме на свій смак.

Рисунок 1.7 Покрокова інструкція

З безкоштовним конструктором Canva, користувача позбавили важкої роботи з розробки дизайну та форматування резюме. Налаштовуючи шаблони додатку, користувач заощадить час, адже все, що йому потрібно зробити, — це додати інформацію про свій професійний досвід і налаштувати наявний дизайн за необхідності.

Якщо ви адвокат, який працює в корпоративному секторі, або стиліст інтер'єрів у творчій індустрії, графічні дизайнери Canva створили широкий спектр шаблонів, які підійдуть для будь-якої галузі, де б ви не працювали. Для творчих спеціальностей у додатку є барвисті шаблони з художніми, ілюстративними елементами. Якщо ж ви хочете щось більш формальне, можете переглянути колекцію мінімалістичних шаблонів, що обов'язково вразить найбільш консервативних менеджерів з підбору кадрів. У середньому рекрутери витрачають на перегляд одного резюме шість секунд. Тому для тих, хто шукає особливий спосіб презентувати свої навички, шаблони додатку Canva пропонують унікальні зразки дизайну, які допоможуть виділитися. Від інфографіки з хронологією до презентації на одну сторінку, і навіть секторних діаграм — за допомогою кількох простих натискань ви можете чітко відобразити всі свої кар'єрні досягнення. Canva надає можливість редагувати дизайн необмежену кількість разів. При поданні заявки на роботу часто найбільше часу займає коригування супровідного листа та резюме для кожної нової вакансії, на яку ви подаєте заявку. З безкоштовним конструктором Canva всі ваші дизайни супровідних листів і резюме автоматично зберігаються в редакторі. Створюйте велику кількість версій свого резюме і вносьте невеликі правки, де це необхідно.

Шаблони Canva затверджені дизайнерами, тому користувачу не потрібно турбуватися про те, як вони виглядають.

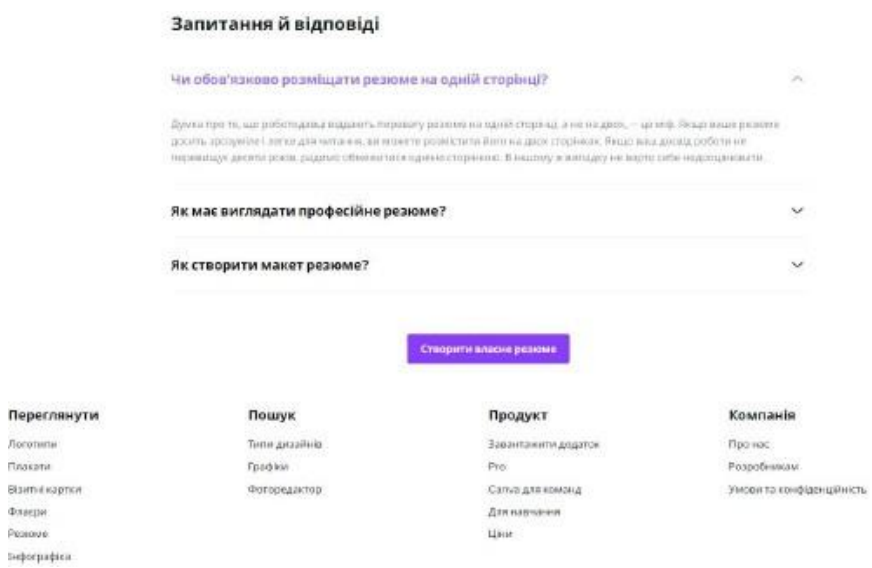


Рисунок 1.8 Найчастіші запитання

Як має виглядати професійне резюме в Canva?

Обов’язкові елементи такі :

- Контактні дані
- Загальна інформація про себе
- Основні факти з досвіду роботи
- Інформація про основні вміння, освіту та навчання
- Будь-які додаткові сертифікати та дипломи

Якщо залишилося вільне місце, додайте хобі та рекомендації з попередніх місць роботи. Як створити макет резюме? У найбільш ефективних макетах резюме використовують професійні шрифти, одинарну відстань між рядками, зрозумілі заголовки розділів та поля шириною 1,27 см. Найдоцільніше на початку зазначити найбільш актуальний та останній досвід роботи. Перша сторінка має містити загальну інформацію і контактну інформацію у верхній частині резюме.

CV2you - це інструмент для створення резюме з використанням готових шаблонів, який дозволяє самостійно скласти професійне резюме онлайн без необхідності залучення дизайнера. Навіть якщо ви володієте технічними навичками або не користуєтеся графічними редакторами, створення резюме для пошуку роботи стає легким завданням.

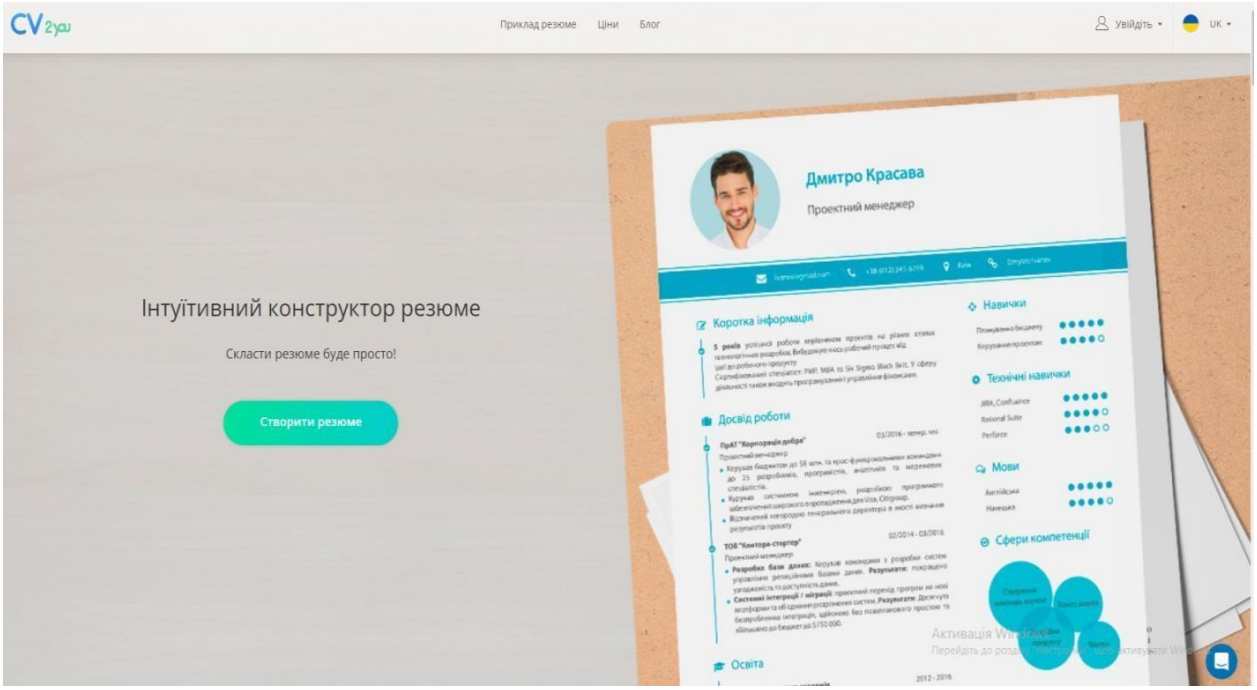


Рисунок 1.9 Інтуїтивний конструктор резюме

Конструктор резюме - це онлайн-сервіс, який надає всі необхідні інструменти: шаблони з різними дизайнами, підказки та приклади. Користувача крок за кроком ведуть, надаючи необхідну допомогу. Конструктор дозволяє змінювати розташування блоків та їх послідовність всередині шаблону, а також автоматично сортує ваші навички. Щодо дизайнерських можливостей, можете вибирати шрифти і колірні стилі. Все це просто і легко зрозуміло. Після заповнення всієї необхідної інформації отримуєте готове резюме онлайн (для поділу посиланням), а також можливість завантажити його у форматі PDF.

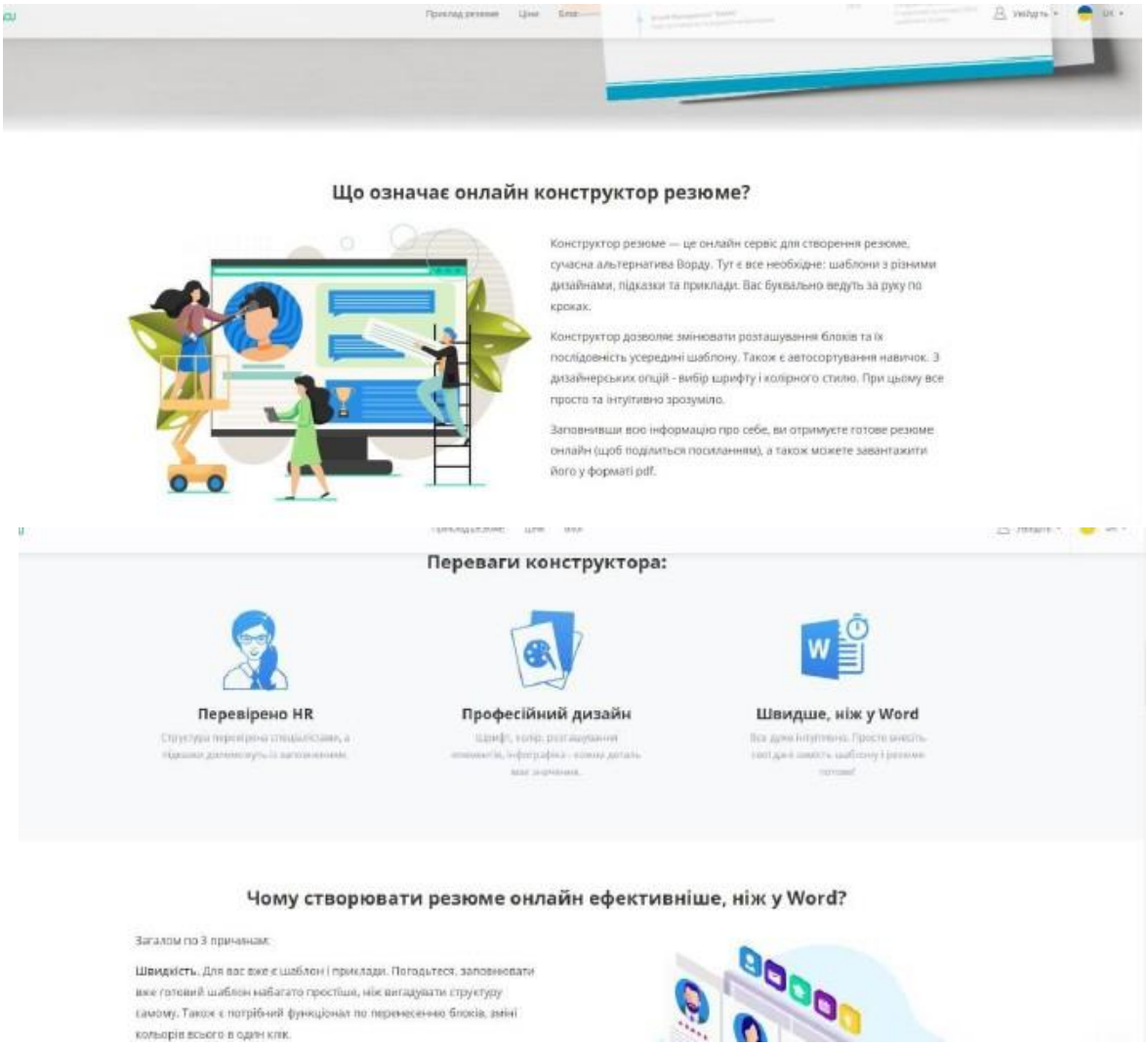


Рисунок. 1.10 Переваги конструктора резюме

Загалом, є три ключові причини, чому слід скористатися конструктором резюме:

1. Швидкість: за допомогою готового шаблону та прикладів зможете швидко заповнити своє резюме. Використання готового шаблону значно спрощує процес, порівняно зі створенням структури з нуля. Крім того, конструктор надає функціонал для перенесення блоків та зміни кольорів всього в один клік, що також збільшує швидкість редагування.
2. Функціональність: шаблони резюме розробляються спільно з дизайнерами, рекрутерами та HR-менеджерами. Це означає, що кожний елемент детально продуманий, починаючи від колірної схеми та шрифтів, закінчуючи автоматичним сортуванням блоків за датою. Усі шаблони не тільки візуально привабливі, але й функціональні, містять всі необхідні елементи.
3. Якість: дослідження The Ladders показує, що рекрутер витрачає лише 6 секунд, щоб оцінити резюме. В порівнянні зі стандартними шаблонами у Word, резюме, створені за допомогою конструктора, виглядають більш креативно. Таким чином, конструктор резюме виконує своє завдання - привертає увагу та утримує інтерес менеджера з підбору персоналу.

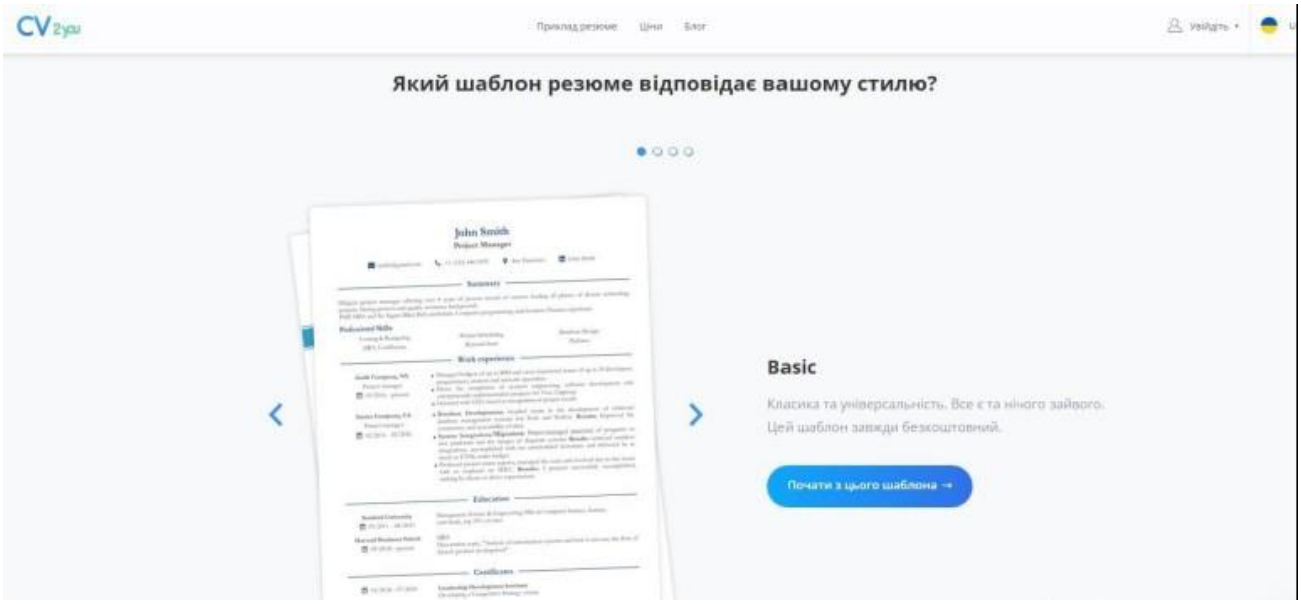


Рисунок. 1.11 Шаблони резюме

1.2 Аналіз принципів роботи веб – додатку

Веб - додаток можна розглянути як складну систему. Він реалізує наступні основну функцію: представлення послуг користувачу.

Функції даного веб – додатку :

1. надання онлайн допомоги користувачу;
2. реєстрація;
3. забезпечення безпеки особистої інформації покупця.

Проаналізувавши принципи роботи веб-додатків, можна зробити висновок, що будь-який веб-додаток повинен складатися з клієнтської та серверної частин. Основною метою створення веб-додатку є резюме. На сьогоднішній день існують різні підходи до створення веб-додатків, кожен з яких має свої переваги та недоліки. Додаток для створення резюме - це чудовий спосіб показати, що ви кращі за своїх конкурентів. Тому перед початком розробки проводиться ретельний аналіз, який допоможе виявити ваші унікальні сильні сторони і виділити ваші переваги. Цей веб-додаток вимагає реєстрації та авторизації користувача. Ви також можете додавати та редагувати інформацію про себе. Якщо у вас виникли запитання, зверніться до служби підтримки.

1.3 Постановка задачі

Веб – додаток для створення резюме. Сформулюємо основні вимоги до розроблюваного web-додатку:

- представлення компанії в мережі Інтернет;
- допомога користувач у написанні резюме;
- надання підтримки клієнтам.

Основні користувачі додатку, а значить і основна аудиторія розроблюваного додатка – це люди, віком 18-40 років, які прагнуть знайти роботу. Прийнявши до уваги ці дані, вони повинні бути використані для розробки простого, легкого, але привабливого та яскравого дизайну всіх сторінок додатку. Додаток повинен мати наступні функції:

1. Реєструвати та авторизувати користувачів, використовуючи відповідні механізми для перевірки унікальності клієнта.

2. Система повинна забезпечувати збереження та відновлення даних користувача.

3. Інформація в додатку повинна бути актуальною та відповідати дійсності.

Вимоги до надійності. Додаток повинен забезпечувати базовий захист від несанкціонованого доступу. Достовірність даних, введених користувачем, повинна бути обов'язковою. Додаток повинен використовувати протокол SSL для передачі даних. Вимоги до Usabillity. Веб-додаток повинен відповідати наступним характеристикам:

-Дизайн повинен бути привабливим.

-Інформація в каталозі повинна бути структурована.

-Користувач не повинен відчувати інформаційного перевантаження при навігації сторінками.

-Навігація по додатку повинна бути простою ізрозумілою.

-Швидкість обробки запитів і завантаження сторінок завжди повинна бути високою і мінімізувати використання ресурсів сервера.

-На ресурсі має бути тільки якісний матеріал, що відповідає призначенню сервіса.[1]

ВИСНОВОК ДО РОЗДІЛУ 1

У цьому розділі розглядаються та аналізуються основні причини та тенденції зростання популярності веб-додатків для створення резюме. Прогнози показують, що популярність і попит на ці додатки неухильно зростає. Також розглядаються аналогічні продукти та описуються їхні основні переваги та недоліки, щоб виділити їхні особливості та висвітлити основні помилки, яких слід уникати. Аналіз функціональних принципів веб-додатків показує, що будь-який веб-додаток повинен складатися з клієнтської та серверної частини; основним завданням веб-додатку є створення резюме, яке є частиною самого веб-додатку. Сьогодні існують різні підходи до побудови веб-додатків, кожен з яких має свої переваги та недоліки. Додатки для створення резюме - це ефективний спосіб показати, чим ви кращі за своїх конкурентів. Тому перед початком розробки проводиться ретельний аналіз. Загалом, веб-програми для створення резюме можуть спростити процес написання резюме, пропонуючи широкий вибір шаблонів, легке редагування та зручність доступу до вашого резюме з будь-якого місця. Цей додаток дозволить ефективно представити професійну інформацію та зберегти її у зручних форматах для подальшого використання.

РОЗДІЛ 2
АНАЛІЗ ІНСТРУМЕНТІВ ДЛЯ СТВОРЕННЯ ВЕБ - ДОДАТКІВ

2.1 Характеристика та аналіз обраного проекту

Враховуючи вимоги до основного та додаткового функціоналу, вказані в Розділі 1, створюваний додаток буде клієнт-серверною програмою з віддаленим сервером та клієнтом. Додаток повинен надати користувачу можливість реєстрації або авторизації, перегляду та редагування інформації користувача, створення власного резюме.

Пояснення до всіх вимог функціоналу наведено в таблиці нижче:

Таблиця 2.1 Вимоги до функціоналу додатку

Функція	Пояснення
Реєстрація	Надати можливість реєстрації на даному веб – ресурсі.
Автентифікація	Надати можливість користувачу автентифікації в системі, використовуючи вже існуючий профіль. Якщо користувач існує – надати можливість використовувати додаток через уже існуючий профіль. В іншому випадку перейти до функції реєстрації.
Редагування профілю	Надати можливість змінювати таку інформацію як контактний номер, електронну адресу та інше.

Продовження Таблиці 2.1 Вимоги до функціоналу додатку

Функція	Пояснення
Перегляд основної статистики користувача	Надати можливість переглядати основну статистику користувача, яка включає: загальну інформацію, контактний номер, електронну адресу, дані про освіту та інше.
Можливість публікувати та зберігати інформацію	Надати можливість публікувати чи редагувати інформацію про користувача.
Можливість отримати онлайн підтримку	Надати можливість користувачу звернутись за допомогою у вирішенні будь – якого питання.

2.2 Етапи створення веб – додатка

Щоб розробити веб-додаток, необхідно не тільки мати ідею, але й чітко розуміти послідовність кроків, забезпечити максимальну відповідність початковим вимогам, уникнути збоїв і не втратити потенціал додатку. Щоб уникнути подібних проблем, варто створити план розробки додатку:створення веб-додатку - це детальне завдання, в якому задіяний не один співробітник, а ціла професійна команда. Для відображення додатку на комп'ютері або смартфоні користувача використовується браузер.[2]

У разі якщо ви розробляєте веб – додаток для бізнесу, у нього мають бути головні цілі:

1) залучити клієнтів;

2) розповісти про послугу;

3) розповісти про вашу компанію;

4) вибудувати довгострокові відносини з покупцями;

5) збільшити охоплення і впізнаваність бренду

Етапи розробки додатка:

- Визначення предмету та основних цілей проекту
- Підготовка технічного завдання
- Прототипування додатку, макетування та дизайн
- Замовлення та програмування
- Наповнення змістом
- Тестування та виявлення багів
- Здача готового проєкту

З чого почати роботу над реалізацією? Чітке розуміння мети і кінцевого результату допоможе вибудувати структурний ланцюжок проекту і визначити, які етапи розробки необхідні для досягнення мети. Всі ці аспекти слід обговорити на ранній стадії розробки.

Технічне завдання є офіційним документом і є основою для подальшої роботи. У ньому вказуються всі деталі, включаючи структуру додатку(кількість сторінок, розділів, категорій, блоків), вимоги до дизайну, функціональне, візуальне і текстове наповнення та технічну компетентність. Технічне завдання - це вимоги, які постійно використовуються під час розробки додатку. Основними етапами розробки веб-продукту є прототипування, макетування та дизайн. На цьому етапі створюються макети для втілення ідей в реальність. Тестування - це завершальний етап, на якому проводяться різні види перевірок, такі як виявлення помилок, некоректної функціональності та загальної продуктивності ресурсу. Ще одним важливим аспектом є розгортання додатку в інтернеті. Щоб додаток міг бути доступним для користувачів, його переносять на хостинг. Крім того, необхідно вибрати доменне ім'я. Після розгортання в Інтернеті виконується фінальне тестування для перевірки його продуктивності.[3]

На ранніх стадіях процесу розробки додатку необхідно затвердити вимоги до проекту.

Для подальшої розробки необхідно відповісти на ключові питання при визначенні вимог і затвердженні ідей.

Мета додатку: Чому у розробника є бажання створити веб-додаток? Чи відповідає ідея потребам цільової аудиторії? Цільова аудиторія: Хто є потенційними користувачами розроблюваного веб-додатку? Які основні особливості додатку? Чим він виділяється і що робить його унікальним? Аналіз конкурентів: хто є конкурентами? Чи існують ідеї для кращого вирішення проблеми? Технічні питання є дуже важливим етапом розробки. Маркетингові дослідження виявляють реальні потреби та інтереси ринку. Вони гарантують, що ідеї будуть відповідно скориговані і засновані на знайденій інформації. Дослідження також дає вам чіткий перелік потреб, який може допомогти вам визначити пріоритети у вашому бізнесі.

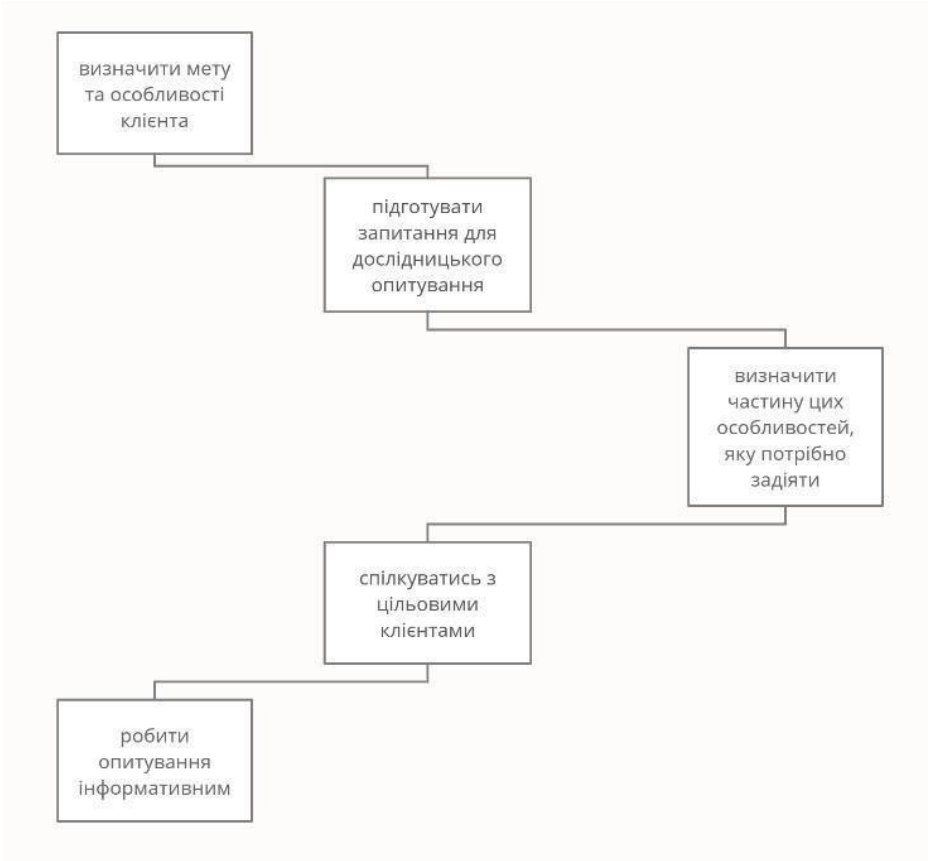


Рисунок 2.1 Діаграма дослідження ринку

Крім того, маркетингові дослідження мають такі переваги: вони дають краще розуміння споживача, чітке уявлення про конкуренцію, дозволяють всебічно оцінити продукт перед запуском, виявляють можливості для залучення аудиторії та враховують особливості та вимоги користувачів. Основна мета додатку - забезпечити гарний користувацький досвід з привабливим візуальним оформленням. Юзабіліті додатку важливе на етапі розробки. На цьому етапі створюються екземпляри головних сторінок додатку та визначаються основні функції, що активуються при натисканні на елементи інтерфейсу. Після того, як юзабіліті додатку перевірено, можна приступати до розробки. Процес тестування під час розробки гарантує, що не буде серйозних помилок, які потрібно виправити до завершення. Після того, як додаток буде випущено в продакшн, його потрібно ретельно проаналізувати, оскільки можна отримати зворотній зв'язок від користувачів і врахувати їхні зауваження для майбутніх удосконалень[3].

2.3. Інструменти розробки

2.3.1 Редактор VSCODE

VisualStudioCode - це інструмент, призначений для створення, редагування та налагодження сучасних додатків для хмарних систем. Це середовище розробки є першим інструментом в серії Visual Studio, який працює на декількох платформах. Visual Studio Code базується на проекті Atom, програмному забезпеченні з відкритим вихідним кодом, розробленому на GitHub. VisualStudioCode, надбудова над Atom Shell, що використовує двіжок браузера Chromium та Node.js, має більше можливостей, ніж Atom. Він пропонує не лише інтегроване середовище розробки, але й більше інструментів та можливостей.

VisualStudioCode - це флагманський редактор від Microsoft з низкою ключових функцій. Він використовується для створення та налагодження програм і має підтримку Markdown з інтеграцією Git. Редактор також має функцію попереднього перегляду, яка дозволяє користувачам переглянути файл README.md перед завантаженням його на GitHub. Конфігурація в VisualStudioCode проста за допомогою JSON-файлів, а нещодавно з'явився зручний графічний інтерфейс користувача. Редактор має вбудований дизасемблер, інструменти для роботи з Git, засоби рефакторингу, легку навігацію по коду, автозавершення та довідку, підтримку розробки на платформах ASP.NET та Node.js і повноцінне інтегроване середовище розробки. Підтримувані мови і технології включають C++, JavaScript C#, TypeScript, Jade, PHP, XML, Python, Batch, F#, DockerFile, Objective-C, Luna, PowerShell.

2.3.2. HTML, CSS

HTML - це мова, яка використовується для визначення структури сторінки документа. Вона може форматувати звичайний текст за допомогою таких структур, як абзаци, заголовки і списки, а також створювати різні посилання на інші сторінки. HTML - це текстова мова, і інструкції форматування, відомі як теги, вбудовуються в частини документа. Ці теги вказують браузеру, як форматувати і відображати інформацію на екрані. Наразі існує близько 140 різних тегів HTML, деякі з яких є застарілими і несумісними з сучасними браузерами. HTML вважається офіційним веб-стандартом, а його специфікації розробляються Консорціумом Всесвітньої павутини(W3C).

Найбільшим оновленням мови стало впровадження HTML5 у 2014 році, яке додало нові семантичні теги, такі як <header>, <article> і <footer>. HTML був представлений у 1989 році Тімом Бернерсом-Лі як частина Всесвітньої павутини. HTML був запропонований як частина технології для розробки розподіленої гіпертекстової системи. Основна ідея гіпертекстової інформаційної системи полягає в тому, щоб дозволити користувачам переглядати документи (сторінки тексту) в зручному для них вигляді та порядку, а не в порядку, як при читанні книги. HTML-документ - це файл з розширенням .html.

Файли з розширенням .html можна переглядати за допомогою веб-браузера, наприклад, Google Chrome або Safari. Браузери інтерпретують HTML-файли і відображають їхній вміст для користувачів. Веб-сайт зазвичай складається з декількох різних HTML-сторінок, таких як домашня сторінка, сторінка опису, а також сторінка контактів. Кожна HTML-сторінка складається з набору тегів, що формують ієрархію і структурують вміст документу за допомогою розділів, абзаців, заголовків та інших елементів. Більшість елементів HTML мають відкриваючі тазакриваючі теги, представлені в форматі <tag></tag>. HTML визначає структуру електронного документа з дизайном на рівні друку і може включати різні елементи, такі як зображення, аудіо та відео.

HTML має розвинені інструменти для роботи з різними аспектами веб-розробки. Вона дозволяє визначати формати заголовків, виділяти текст за допомогою шрифтів, створювати списки і таблиці, а також створювати форми для взаємодії з користувачем. Інструменти HTML також можна використовувати для створення посилань на інші сторінки та ресурси, встановлення мета-тегів для пошукової оптимізації, вбудовування скриптів для динамічної зміни контенту та багато іншого. HTML-це потужний інструмент для створення структурованого та багатофункціонального веб-контенту.

Плюси HTML:

•HTML є широко використовуваною мовою з великою кількістюресурсів і величезною спільнотою.

•Вона працює в будь-якому веб-браузері і має відкритий код, що робить її абсолютно безкоштовною.

•HTML надає чисту і послідовну розмітку, яка сприяє легкому розумінню і редагуванню коду.

•Легка інтеграція з серверними мовами, такими як PHP і Node.js, дозволяє створювати повноцінні веб-додатки.

•Поєднання HTML з CSS і JavaScript дозволяє досягти багатогокористувацького досвіду та реалізувати розширені ідеї.

HTML є основою веб-розробки і працює у всіх браузерах, що підтримують W3C. Ця мова використовується для створення структури сайтів да додатків і може бути доповнена CSS для стилізації та JavaScript для додавання динамічної функціональності; CSS використовується для визначення зовнішнього вигляду документа, в той час як JavaScript використовується для інтерактивних елементів, анімації та інших розширених можливостей. Поєднання цих технологій надають інструменти для створення гарних та інтерактивних веб-додатків. HTML є основною мовою розмітки веб сторінок, і її стандарти підтримуються всіма сучасними браузерами. Це робить HTML універсальною та зручною мовою, яку можна використовувати для створення веб-сайтів та додатків, доступних для всіх користувачів. HTML має велику кількість ресурсів, документації та шалену підтримку спільноти. Це означає, що велика кількість ресурсів та експертів завжди доступні, щоб допомогти вам в розробці та вирішенні проблем. HTML має простий синтаксис і є легкою для розуміння.[4]

Зазвичай CSS використовується для візуального оформлення веб-сторінок, написаних у форматі HTML або XHTML, але формат CSS можна застосовувати і до інших XML-документів. Специфікація CSS була розроблена Консорціумом Всесвітньої павутини і все ще перебуває в стадії розробки. Автори та відвідувачі веб-сторінок використовують CSS для визначення кольорів, шрифтів, макета та інших аспектів зовнішнього вигляду сторінки. Одна з головних переваг CSS полягає в тому, що вміст сторінки (зазвичай у HTML, XML або подібній мові розмітки) і представлення документа (анотація в CSS) можуть бути розділені. Таке розділення має кілька переваг, наприклад, покращує сприйняття і доступність контенту, забезпечує більшу гнучкість і контроль над представленням контенту в різних середовищах, робить контент більш структурованим і простим, а також допомагає уникнути повторень. CSS також дозволяє адаптувати контент до різних умов перегляду, таких як комп'ютерні монітори, кишенькові пристрої (КПК), друковані матеріали, телебачення, а також пристрої, що підтримують шрифт Брайля і голосові браузери. Один і той самий документ HTML або XML може відображатися по-різному залежно від використовуваного CSS.

1. Стилі автора:

а) зовнішні таблиці стилів (зазвичай в окремому файлі)

б) внутрішні таблиці стилів, які включені як частина документу

в) стилі для кожного окремого елементу

2. Стилі користувача:

а) локальний .css- файл, який вказаний користувачем для використання на веб-сторінках і в налаштуваннях браузера. такому як Opera, Google Chrome.

3. Стилі переглядача

а) стандартизований стиль переглядача, наприклад стандартний стиль для елемента, визначеного браузером, використовується коли відсутня інформація про стиль елемента або вона є неповною.

CSS також можна використовувати для визначення того, як виглядатиме веб-сторінка при перегляді в середовищі, відмінному від веб-браузера. Це пов'язано з тим, що такі елементи веб-сторінки, як кнопки навігації або веб-форми, не мають сенсу на друкованій сторінці. Хоча така практика не дуже поширена на багатьох веб-сайтах, можливість створювати готові до друку таблиці стилів є потужною і привабливою (з досвіду, більшість веб-майстрів не роблять цього, тому що бюджет сайту не виправдовує цієї додаткової роботи).

CSS - це один з найпотужніших інструментів, який може вивчити веб-дизайнер, оскільки він може впливати на весь зовнішній вигляд сайту. Добре розроблену таблицю стилів можна швидко оновлювати і змінювати те, що є візуально важливим на екрані, щоб зосередити увагу відвідувачів не змінюючи розмітку HTML. Основна цінність CSS полягає в тому, що його дуже мало потрібно вивчати: це дуже проста і легка у використанні мова, яку можна використовувати в найрізноманітніших форматах. Крім того, браузери змінюються щодня, тому те, що добре працює сьогодні, може не мати сенсу завтра, оскільки нові стилі підтримуються, а інші стають застарілими або менш популярними з будь-якої причини. CSS дозволяє веб-дизайнерам використовувати один файл, щоб суттєво змінювати стиль усіх сторінок на веб-сайті. Це допомагає створювати легкі, креативні веб-сайти, які швидко реагують на запити користувачів і вражають їх при перегляді. Тому це важливий елемент сучасних веб-сайтів, який не можна ігнорувати. Окрім вищезазначених фактів, CSS корисна у багатьох випадках. Пробіли, вирівнювання і позиціонування: Це означає, що CSS має неперевершений потенціал для візуального позиціонування, наприклад, полів, маркірування і відступів тексту. CSS також дозволяє збільшити кількість зображень, що корисно для користувачів зі слабким зором".

Ось деякі круті фішки CSS:

1.Flexbox. Flexbox - це потужна техніка верстки, яка дозволяє легко організовувати, вирівнювати та розподіляти елементи всередині контейнера. display: flex, flex-direction, justify-content, align-items та інші функції можна використовувати для створення гнучких макетів для різних розмірів екранів.

2.Сітки. Сітки CSS - це ще одна потужна техніка верстки, яка дозволяє створювати багатоколонкові та багаторядкові макети за допомогою таких властивостей, як display:grid, grid-template-columns і grid-template-rows, що дозволяє легко керувати розміщенням елементів у сітці.

3.Анімація. CSS дозволяє створювати приголомшливі анімації без використання JavaScript: такі функції, як ключові кадри, анімація і перехід дозволяють створювати рухомі ефекти, змінювати кольори і розміри. Це робить їх більш динамічними та привабливими. Це дозволяє створювати більш динамічні та привабливі веб-сторінки.

4.Псевдокласи та псевдоелементи. CSS надає різноманітні псевдокласи та псевдоелементи, які дозволяють формувати елементи відповідно до їхнього стану або положення в документі. Наприклад, псевдоклас :hover дозволяє вам змінювати стиль елемента при наведенні на нього курсору, а псевдоклас :hover дозволяє вам змінювати стиль елемента при наведенні на нього курсору.

Ми ознайомилися з основними перевагами використання CSS при розробці веб-сайтів. Він має широкий спектр можливостей використання не тільки в ситуаціях розробки та дизайну, але і в тому, як він впливає на інші елементи бізнесу, такі як SEO, цифровий маркетинг і пошукові системи. Крім того, одним з найважливіших моментів, що розглядаються на глобальному рівні, є доступність, яку CSS пропонує користувачам. Виживає найсильніший - це не лише закон природи, він також повністю перейнятий технологіями.

2.3.3. JAVASCRIPT

JavaScript (JS) - це динамічна об'єктно-орієнтована мова програмування, яка реалізує стандарт ECMAScript і надає можливість взаємодіяти з користувачем на стороні клієнта, здійснювати контроль над браузером, асинхронно обмінюватися даними з сервером, а також змінювати структуру і зовнішній вигляд веб-сторінок в Інтернеті. Мову JavaScript класифікують як прототипну мову програмування з динамічною типізацією, підмножина об'єктно-орієнтованого програмування. Вона також частково підтримує й інші парадигми програмування, такі як імперативні та часткові типи функцій, і має такі архітектурні особливості, як динамічна і слабка типізація, автоматичне керування пам'яттю, прототипування та функції як першокласні об'єкти. JavaScript використовується для наступних цілей: [5].

- Написання сценаріїв на веб-сторінках щоб надати їм інтерактивності.
- Створення односторінкових та складних веб-застосунків, таких як React, AngularJS, Vue.js.
- Програмування на стороні back end з використанням Node.js, Express.js.
- Розробка складних застосунків, таких як Electron, NW.js.
- Створення мобільних застосунків, наприклад таких як React Native, Cordova.
- Використання в середині PDF-документів та інші застосування.

Java та JavaScript - це дві мови програмування зі схожими назвами, але різними значеннями. Незважаючи на схожість, наприклад, стандартні бібліотеки та домовленості щодо імен, вони відрізняються: На початку свого розвитку JavaScript була орієнтована на аматорів, і тому багато професійних програмістів ставилися до неї скептично. Однак це змінилося з появою AJAX, який також привернув увагу професійних програмістів. Пізніше були внесені зміни відповідно до стандарту ES6+, і JavaScript отримав багато корисних функцій, яких не вистачало для ефективного програмування. В результаті було розроблено та вдосконалено багато JavaScript-додатків, включаючи тестування та налагодження, створення бібліотек, а також фреймворків. В результаті використання JavaScript вийшло за межі браузера: Хоча JavaScript має низку рис об'єктно-орієнтованої мови, його підтримка об'єктів чимало різниться від традиційних мов ООП через концепцію прототипів. JavaScript також має риси функціональних мов, де функції є першокласними об'єктами, а об'єктами є списки, носії, анонімні функції, що надає мові додаткової гнучкості. JavaScript має синтаксис, подібний до C, але основні ключові відмінності від C полягають у наступному:

- Об'єкти можуть динамічно змінювати тип через механізм прототипів
- Функції виступають об'єктами першого класу.
- Обробка винятків.
- Автоматизоване приведення типів.
- Автоматичний garbage collection.
- Анонімні та стрілочні функції.

JavaScript використовується у веб додатках на стороні клієнта, тобто у браузері. Це значить, що він може працювати на всіх операційних системах, і веб-інтерфейсах, побудованих на JavaScript.

JavaScript може знаходити широке застосування в таких областях як:

- AJAX скрипти використовуються у відомому підході, який передбачає асинхронний обмін даними між браузером користувача та ресурсами сервера.
- Браузерні операційні системи: частка коду деяких браузерних операційних систем складається зі скриптів, написаних на JavaScript
- Закладки: JavaScript широко застосовується у програмах, які розміщуються в закладках браузера.
- Серверні додатки: JavaScript використовують і для написання фрагментів коду, які можуть виконуватися на стороні сервера, зокрема у Java 6.
- Мобільні додатки: JS також може бути корисним і у цьому популярному IT напрямку.
- Віджети: JavaScript використовується для написання різноманітних міні-програм, що використовуються у робочому просторі і є дуже зручними.
- Прикладне програмне забезпечення: Об'єктно-орієнтована мова JavaScript використовується також для створення окремих програм, включаючи невеликі ігри.

Не зважаючи на те, що для браузера у нас є дуже багато мов клієнта для маніпуляції HTML, як VBScript, чому в основному використовується JavaScript? Чому це лідер? Чому б не замінити JavaScript на інші мови виконання сценаріїв? Причини є:

. Цю мову підтримує кожен браузер.

. Більшість клієнтських мов, які знаходяться у використанні, використовують JavaScript як проміжний.

. JavaScript легко вивчити порівняно з іншими мовами сценаріїв. Крім того, JavaScript є мовою, яку починаються вивчати однією з перших

Оскільки JavaScript є строго нетипізованою мовою програмування і може працювати в різних середовищах зі своїми особливостями сумісності, програмістам потрібно ретельно перевіряти, чи правильно працює їхній код у різних можливих конфігураціях. Недотипізація визнана однією з найбільших проблем JavaScript. Тому восени 2012 року Microsoft анонсувала TypeScript - мову програмування, яка компілюється в JavaScript і містить чимало важливих доповнень, що спрощують розробку для програмістів.

2.3.4. MYSQL і PHP

MySQL - це безкоштовна реляційна система управління базами даних. Сьогодні MySQL є однією з найпоширеніших систем управління базами даних. Здебільшого її використовують для створення динамічних веб-сторінок, оскільки вона пропонує хорошу підтримку різних мов програмування. Чому MySQL стала такою популярною?

. Спочатку система була створена для обмеженого використання. Сьогодні вона вже є сумісна з такими платформами, як Microsoft Windows, Linux та macOS.

. Простота використання. Ви можете змінювати вихідний код. Зробіть це самостійно.

. Висока продуктивність. Сумісність з багатьма моделями кластерних серверів.

. Безпека. Система доступу до бази даних за допомогою облікових записів забезпечує високий рівень захисту бази даних. Можливе шифрування паролів та автентифікація на боці хосту. Важко створити веб-сайт без бази даних. Найзручніше зберігати мінливий контент сайту, продукти електронної комерції та новини в базі даних.

Як працює MySQL? Базова структура системи «клієнт-сервер» є дуже простою. Клієнт підключається до сервера через мережу. Він робить запит через інтерфейс користувача (GUI). Далі у середовищі MySQL відбуваються такі операції:

. спочатку створюється база даних MYSQL, визначається відношення таблиці;

. потім клієнт робить запит, використовуючи при цьому команди SQL;

. далі сервер відповідає на запитану інформацію, яка надсилається клієнту.

Бази даних MYSQL - це реляційні бази даних. Це назва бази даних, організованої у вигляді зв'язаних таблиць. Відкритий вихідний код можна змінювати. Однією з переваг PHP є те, що вона дозволяє веб-розробникам швидко створювати динамічні веб-сторінки:

1. Скрипти для запуску на стороні сервера: PHP - найпоширеніша мова.
2. Скрипти для запуску в командному рядку: Можна створювати PHP-скрипти, які працюють незалежно від веб-сервера або браузера.
3. Створення додатків з графічним інтерфейсом користувача(GUI) для запуску на стороні клієнта.

Вибираючи PHP, ви можете отримати велику свободу вибору операційної системи та сервера. Розглянемо кілька причин популярності мови PHP:

- Вона має простий та дуже інтуїтивно зрозумілий синтаксис, що дозволяє навіть новачкам швидко та без проблем освоювати PHP
- Кросплатформність та гнучкість: PHP сумісний майже з усіма сучасними популярними платформами, наприклад такими як Linux та Windows.

Висока масштабованість: PHP може максимізувати продуктивність програми навіть при збільшенні апаратних ресурсів. Веб-додатки, розгорнуті на декількох серверах, можуть ефективно справлятися з високими навантаженнями і великим трафіком.

Активний розвиток і вдосконалення. Розробники постійно працюють над вдосконаленням та додатковими функціями, щоб збільшити функціональність мови, спростити її синтаксис, використання та покращити захист від можливих атак: MySQL і PHP - дві дуже популярні технології, які доволі часто використовуються разом при розробці веб-додатків. Розглянемо кожну технологію окремо. MySQL: MySQL - це система управління базами даних, що забезпечує структуроване зберігання інформації та організацію даних. Заснована на реляційній моделі бази даних, вона обробляє дані за допомогою мови запитів SQL; переваги використання MySQL полягають у наступному [8].

1. Надійність: MySQL є достатньо стабільною та надійною базою даних з дуже високим рівнем безпеки.
2. Простота використання: Вона має зрозумілий інтерфейс, який дозволяє легко створювати, модифікувати та робити запит для отримання даних
3. Масштабованість: MySQL також підтримує розподілені системи та може обробляти великий потік даних.
4. Широка співпраця: Розробники активно підтримують MySQL, надаючи документацію, плагіни та інші корисні ресурси. PHP: PHP - це мова програмування, яку широко використовують для розробки веб-додатків. Вона слугує мовою сценаріїв на стороні сервера, що в свою чергу означає, що виконання коду відбувається саме на сервері, а не на стороні клієнта. Далі неведені деякі переваги використання PHP:[6]

1. Простота використання PHP легко вивчати та розробляти завдяки простому синтаксису.

2. Широке використання: PHP є однією з найбільш популярних мов програмування для веб-розробки і також має чималу спільноту розробників.

3. Вбудована підтримка MySQL: PHP сама по собі має вбудовану підтримку для підключення до бази даних MySQL та надсилання запитів до неї.

4. Розширюваність: PHP підтримує ряд бібліоек, які можуть розширити функціональність мови.

ВИСНОВОК ДО РОЗДІЛУ 2

Отже, зробивши аналіз етапів створення веб – додатків, було складено чіткий план та дано відповіді на всі запитання, які фігурували у вказаних етапах:

- Метою програми є веб – додаток для створення резюме.
- Цільова аудиторія: користувачі, що бажають просто та зручно розповісти про свої навички для подальшого працевлаштування.
- Аналіз конкурентів: існуючі аналоги додатку, який планується розробляти, було проаналізовано в Розділі 1 дипломної роботи
- Проаналізовані переваги та недоліки редактоу VSCODE .

Розробка веб-додатку вимагає більше, ніж просто мати ідею, вона також вимагає чіткого розуміння послідовності подій, щоб забезпечити максимальну відповідність початковим вимогам, уникнути збоїв і не втратити потенціал додатку. Щоб уникнути таких проблем, варто створити план розробки додатку. Ми виділили основні етапи розробки веб-додатків:

- вибрати тематику та основну мету проекту;
- зробити розробку технічного завдання;
- прототипування, макетування, аналіз та дизайн;
- верстка та програмування;
- наповнення додатку контентом;
- тестування додатку;
- здача готового проекту.

В цьому розділі був описаний редактор VSCODE, а також мови програмування, які були використани при створенні веб-ресурсу.

РОЗДІЛ 3

РОЗРОБКА ВЕБ – ДОДАТКА ДЛЯ СТВОРЕННЯ РЕЗЮМЕ

3.1 Проектування бази даних (БД) системи веб - додатку

Структура сутності веб-додатка визначає організацію та взаємозв'язок між його основними компонентами. Основні сутності, які можуть бути присутніми у структурі веб-додатка, включають наступні:

1. Користувачі: Це сутність, яка представляє користувачів системи. Вона може включати дані про користувача, такі як ім'я, електронну пошту, пароль, роль, права доступу та інше.
2. Моделі даних: Моделі даних відображають структуру та взаємозв'язок між даними, що використовуються в додатку. Вони можуть бути представлені у вигляді класів або схем бази даних і містять поля, методи та валідацію даних.

Рекомендується починати проектування бази даних зі створення концептуальної та фізичної моделі даних. Спочатку треба створити концептуальну модель, що представляє цільову область, для якої буде розроблена база даних. Основою цієї моделі є модель "сутність-зв'язок" (ER-модель). Ця модель описує взаємозв'язки між інформаційними об'єктами в розглянутій предметній області. Визначимо структуру кожної сутності: Сутність "Адміністратор" зберігає інформацію про користувачів додатку. Сутність "Клієнт" використовується для відображення інформації про клієнта, тобто імені, прізвища, адреси електронної пошти, номера телефону і т.д. Фізична модель даних визначає спосіб зберігання даних на комп'ютері і надає інформацію про структуру, порядок записів і доступні шляхи доступу (табл. 3.1-3.2).

На основі визначених сутнісної та концептуальної моделей була створена реляційна модель даних. Це логічна модель, в якій таблиці виступають як відношення, а рядки - як атрибути. При створенні таблиць відразу виконується нормалізація.

ъОвdzfчw 3.1 ï ţ j dzwч'?dzОdzъsнj dz fйzlsdzsfйs' "ф нdz'dz"

ş dz ʁ ч	udzôw Оstœfвlzlsz	ъcfţf	N/NN	ł tswfифjдdzdzw
PK	IdMang	Text	Not Null	ł tœsfйscf? tfj tœəcfdzdzf? ş dz ч
	Name	Text	Not Null	ч ş dzОнj dzf? Оstœfвlzls "ł tœ'Ləcfĥ j, 'dzôw"
	Login	Text	Not Null	ł tœsfйscf? Оstœfвlzls "dzsG'dz"
	Password	Text	Not Null	ł tœsfйscf? Оstœfвlzls "tfОœtsdz"

ъОвdzfчw 3.2 ï ţ j dzwч'?dzОdzъsнj dz fйzlsdzsfйs' "ş dz' dzs"

ş dz ʁ ч	udzôw Оstœfвlzlsz	ъcfţf	N/NN	ł tswfифjдdzdzw
K	IdClient	Text	Not Null	ł tœsfйscf? tfj tœəcfdzdzf? ş dz ч
	Name	Text	Not Null	ч ş dzОнj dzf? Оstœfвlzls "ł tœ'Ləcfĥ j, 'dzôw ş dz' dzsO"
	Phone	Text	Not Null	ł tœsfйscf? Оstœfвlzls "lsj dzдw tsdz"

ł tœf tfœtsj ş lszəОdzdz вОLf нОdzf¬ нdzw нtsнОksлz 'L fйsətstœj dzdzw tœj Lʁ dzj, tftslstc'вdzs əcfLdzОцflscf, wş' нОdz dzдtsв¬'нdzs lвj tc'GОlscf lsО wş ətsdzf вlzнlzlsн əLО dzъsн'wlscf tsнcfdzL tsнdzfdz ∫ fиj ş'dz ş О tsfифzsədzf¬ ş œtsş'ə, wş' dzъsчdzО əcfş tsdzОlscf нdzw tfœtsj ş lszəОdzdzw вОLf нОdzf¬.

1. ЈlcfLdzОцj dzdzw fйzlsdzsfйsj ?: ЈlcfLdzОцlsj tsfифzsədz fйzlsdzsfйs', wş' вlzнlzlsн tfœcffйzlsdz əнtsнОksлz. υ əОм tsdzz əcftfОнş lz цj dzъsчlzlsн вlzlscf ş tstœcffйslzəОц', tœj Lʁ dzj, нtsfиф'н tœsвtslscf, tsfиф'lsО dzОəcfцş cf lstsĥ ts. Јlfйsфdzsə'lsн lə'wLş cf dz'чl цcfdzъcf fйzlsdzsfйswdzъcf, dzОtfœcfş dzОн, lə'wLtsş "tsнcfdznts вОGОlsн ts¬" dz'чl ş tstœcffйslzəОцfdzъcf lsО'¬ tœj Lʁ dzj.

2. ЈlcfLdzОцj dzdzw Оstœfвlzls'ə: ſ dzw ş tsчdzs' fйzlsdzsfйs' əcfLdzОцlsj dzОв'tcОstœfвlzls'ə Овts tftsdz'ə, wş' əcf¬tsцjlsj lвj tc'GОlscf. l Оtfœcfş dzОн, нdzw fйzlsdzsfйs' "tœj Lʁ dzj" цj dzъsчlzlsн вlzlscf tftsdzw, lsGş' wş lGdtsdzsətsş, ş tsdzsGş lsdzО 'dzwtstœdzОцfw, нОlsО fйsətstœj dzdzw, нОlsО tsdzsədzj dzdzw lstsĥ ts.

3. Визначення первинних ключів: Для кожної сутності визначте унікальний ідентифікатор або первинний ключ, який буде використовуватися для ідентифікації записів в цій сутності. Наприклад, у сутності "резюме" може бути первинний ключ "resume_id".
4. Визначення зв'язків: Встановіть зв'язки між сутностями, які ви визначили раніше. Наприклад, зв'язок "один до багатьох" між користувачами та їх резюме означає, що один користувач може мати багато резюме, але кожне резюме належить тільки одному користувачеві. У цьому випадку, ви можете використати ідентифікатор користувача (наприклад, "user_id") як зовнішній ключ у таблиці "резюме", щоб зв'язати їх.
5. Нормалізація: Розгляньте можливість застосування нормалізації для вашої бази даних, щоб забезпечити ефективну організацію та уникнути дублювання даних. Розділіть дані на логічні таблиці та зв'язуйте їх за допомогою зовнішніх ключів.
6. Створення таблиць: Створіть таблиці у базі даних відповідно до вашого проектування. Кожна таблиця повинна відповідати одній сутності, і має бути відповідність між полями таблиці та атрибутами сутності.
7. Запити та операції: Визначте, які типи запитів та операцій ви будете виконувати з базою даних. Це можуть бути запити для отримання резюме користувача, додавання нового досвіду роботи, оновлення резюме тощо. Врахуйте ці запити при проектуванні структури бази даних та створенні необхідних операцій.[7]

Ці кроки дозволять вам спроектувати структуру бази даних вашого додатку для резюме. Також важливо враховувати ефективність і масштабованість бази даних, оскільки в майбутньому вона буде робити обробку значної кількості даних і транзакцій. Вибір правильної бази даних для веб-додатку має кілька переваг:

1. Ефективне зберігання даних та управління їми: Правильно підібрана база даних може ефективно зберігати та керувати даними веб-додатку.

2. Надійність і безпека: бази даних повинні забезпечувати надійне зберігання та захист інформації. Дуже важливо обрати правильну базу даних із відповідними механізмами резервного копіювання, щоб відновлення даних і безпеки допомогла захистити дані від втрати, пошкодження або несанкціонованого доступу.

3. Сумісність зі стеком технологій: Вибір бази даних, інтегрованої зі стеком технологій, спрощує розробку й обслуговування застосунків; якщо веб-додаток використовує певні технології або фреймворки, то він повинен підтримувати ці технології й надавати відповідні драйвери.

4. Швидкість і продуктивність: Правильно обрана база даних збільшить швидкість роботи вашого веб-додатка та забезпечить вищу продуктивність. Розширені можливості: вибір правильної бази даних дає змогу використовувати низку розширених можливостей, важливих для веб-додатків, таких як кешування, транзакції, повнотекстовий пошук, реплікація даних тощо. Беручи до уваги потреби та характеристики веб-додатка, вибір правильної бази даних може призвести до підвищення продуктивності. Модель "Клієнт-Сервер" є однією з основних архітектурних моделей для розробки веб-додатків. В цій моделі додаток розділяється на дві частини: клієнтську (frontend) та серверну (backend). Клієнтська частина:

- Клієнтська частина повинна відповідати за взаємодію з користувачем і представлення інтерфейсу програми, який може бути реалізований за допомогою HTML, CSS і JavaScript.
- Клієнтська частина працює на стороні користувача веб-браузера. Вона надсилає запит на сервер, отримує відповідь і відображає її у вигляді веб-сторінки або інтерактивного інтерфейсу.

Серверна частина:

- Серверна частина відповідає за обробку запитів від клієнта, збереження та обробку даних, а також взаємодію з базою даних.
- Серверна частина може бути реалізована з використанням різних технологій, таких як PHP, Node.js, Ruby, Python тощо.
- Вона надсилає відповіді на клієнтські запити, які містять необхідну інформацію чи результати обробки запиту. Взаємодія:
- Клієнтська частина взаємодіє з серверною частиною за допомогою мережевих запитів, таких як HTTP запити. Клієнтська частина надсилає запити до сервера, включаючи дані, параметри чи команди.
- Серверна частина обробляє ці запити, виконує необхідні операції, отримує доступ до бази даних, якщо потрібно, та генерує відповіді, які надсилаються назад клієнту.

Переваги моделі "Клієнт-Сервер" включають:

- Розділення обов'язків: Клієнтська та серверна частини можуть бути розроблені незалежно одна від одної, що дозволяє розподілити роботу між фронтенд та бекенд розробниками.
- Масштабованість: Серверна частина може бути масштабована окремо від клієнтської частини, що дозволяє забезпечити швидку та надійну роботу додатку при великому навантаженні.
- Універсальність: Клієнтська частина може працювати на різних пристроях та платформах, що дозволяє забезпечити доступ до додатку з різних пристроїв.
- Безпека: За допомогою серверної частини можна забезпечити контроль доступу до даних та захист від несанкціонованого доступу. Загалом, модель "Клієнт-Сервер" є популярним вибором для розробки веб-додатків, оскільки вона дозволяє ефективно розділити обов'язки між клієнтом та сервером, забезпечити масштабованість та забезпечити швидку та надійну роботу додатку.[8]

3.2 Розробка інтерфейсу веб – додатку для створення резюме

Щоб розробити інтерфейс для додатку з створення резюме, вам знадобиться використати декілька технологій і компонентів. Ось кілька важливих елементів, які ви могли б врахувати при проектуванні інтерфейсу.

1. Форма введення даних: Резюме зазвичай містить різні розділи, такі як освіта, досвід роботи, навички тощо. Вам потрібно створити форму, де користувачі зможуть ввести ці дані. Використовуйте різні типи полів введення, такі як текстові поля, випадаючі списки та флажки, щоб забезпечити зручний спосіб введення даних.
2. Шаблони резюме: Додайте можливість вибору з готових шаблонів резюме. Це дозволить користувачам швидко обрати вигляд свого резюме з популярних варіантів.
3. Перегляд та редагування: Забезпечте можливість перегляду та редагування введених даних. Ви можете використовувати таблиці або картки, щоб користувачі змогли легко оглядати та змінювати свої дані.
4. Попередній перегляд: Додайте функцію попереднього перегляду, де користувачі зможуть побачити, як виглядатиме їх резюме в обраному шаблоні перед збереженням або друкуванням.
5. Збереження: Надайте можливість зберегти резюме та експортувати його у різних форматах.
6. Менеджер аккаунтів: Розгляньте можливість створення системи керування обліковими записами, де користувачі зможуть створювати, зберігати та керувати своїми резюме.
7. Інтуїтивний дизайн: розмістіть елементи візуально зрозуміло і використовуйте зрозумілу мову для підказок та інструкцій.
8. Ці рекомендації можуть слугувати основою для розробки інтерфейсу додатку з створення резюме. Не забувайте також досліджувати ринок, аналізувати потреби користувачів та здійснювати постійне вдосконалення свого додатку на основі отриманих повідомлень та фідбеку

Інтерфейс веб-додатку містить в собі графічні елементи та взаємодію з користувачем. Для створення ефективних і привабливих інтерфейсів веб-додатків можна використовувати різні методи і підходи. Нижче наведені елементи інтерфейсу, які найчастіше використовуються при розробці веб-додатків.[9]

1. HTML/CSS: HTML використовується для створення структури веб-сторінки, а CSS - для задання її зовнішнього вигляду і стилю; HTML і CSS можна використовувати для створення різних елементів, таких як кнопки, форми, таблиці, списки та зображення.
2. JavaScript: JavaScript - це мова програмування, що використовується для реалізації інтерактивних функцій на веб-сторінках. Її можна використовувати для додавання анімації, опрацювання подій, динамічної зміни вмісту сторінки, перевірки форм і багато чого іншого.
3. Бібліотеки для графічного дизайну: такі бібліотеки, як Material-UI, Bootstrap і Foundation, надають розробнику набір готових стилів, компонентів і шаблонів для швидкого розроблення зручних і привабливих інтерфейсів.
4. Гарний дизайн: за різноманітності пристроїв і розмірів екранів важливо забезпечити гарний дизайн, який може швидко пристосуватися до різних пристроїв. Використання медіазапитів, гнучких макетів елементів і масштабованих зображень може забезпечити зручний інтерфейс незалежно від пристрою, на якому відкрито застосунок.
5. Анімація і переходи: Додавання анімації і плавних переходів між сторінками та елементами може поліпшити користувацький досвід і зробити застосунок більш вражаючим.
6. Тестування призначеного для користувача інтерфейсу: Важливо протестувати призначений для користувача інтерфейс, щоб переконатися в його функціональності, зручності використання та відповідності вимогам. Беручи до уваги потреби проєкту і досвід команди, можна вибрати найбільш підходящу технологію і підхід для створення користувацького інтерфейсу веб-додатка.

При виборі правильного інтерфейсу може бути кілька переваг:

1. Користувацький досвід: добре продуманий і збалансований інтерфейс дозволяє користувачам взаємодіяти з додатком у комфортний та ефективний спосіб. Добре продуманий інтерфейс полегшує пошук інформації, виконання завдань та навігацію в додатку.

2. Задоволення потреб користувачів: Вибір правильного інтерфейсу вимагає врахування потреб та очікувань цільової групи користувачів. Задоволення цих потреб підвищить задоволеність від використання додатку, забезпечить позитивний досвід та лояльність користувачів.

3 Ефективність та продуктивність: Вибір правильного інтерфейсу може значно підвищити ефективність роботи користувачів. Зрозумілість і простота використання можуть зменшити кількість помилок, пришвидшити виконання завдань і підвищити продуктивність.

4. Конкурентна перевага: Функціональний і привабливий інтерфейс надає веб-додатку перевагу над конкурентами. Користувачі з більшою ймовірністю оберуть додаток зі зручним інтерфейсом, що сприяє популярності та успіху проекту.

5. Легкість підтримки та розширення: Вибір правильного користувацького інтерфейсу робить веб-додаток простішим у підтримці та розширенні. Якщо інтерфейс добре структурований і модульний, легше вносити зміни, додавати нову функціональність і забезпечувати сумісність з майбутніми вимогами.

Для користувачів веб-ресурсів мають бути реалізовані прості, інтуїтивно зрозумілі та зручні інтерфейси. Також варто звернути увагу на використання гібридного дизайну. Такий дизайн повинен однозначно компенсувати положення та розмір вікон, шрифтів та елементів дизайну відповідно до роздільної здатності монітора користувача.

Коли вибираєте інтерфейс для свого додатка, треба звернути увагу на декілька важливих аспектів:

1. Користувацький досвід (UX): переконайтеся, що додаток є зручним для користувача. Створіть інтерфейс, який дозволить користувачам швидко та ефективно досягати своїх цілей, враховуючи їхні потреби та розуміючи, як вони взаємодіють з додатком.
2. Візуальний дизайн: Створіть привабливий та цілісний зовнішній вигляд додатку. Використовуйте естетичні та логічні рішення щодо шрифтів, кольорової палітри, компонентів та графічних елементів. Використовуйте сучасні тенденції дизайну, щоб зробити додаток привабливим, гарним і сучасним.
3. Навігація та інтуїтивно зрозумілий інтерфейс: Забезпечте легку навігацію та логічну структуру додатку. Треба організувати елементи інтерфейсу так, щоб користувачі могли швидко орієнтуватися і легко виконувати необхідні дії, дотримуючись чіткої моделі взаємодії.
4. Швидкість та продуктивність: Треба звернути увагу на швидкість з якою завантажуються сторінки, відгуки користувачів та загальну продуктивність вашого додатку. Оптимізуйте використання ресурсів, мінімізуйте час відгуку сервера, забезпечте швидку роботу додатку та задоволеність користувачів.
5. Доступність: Переконайтеся, що додаток доступний для людей з різними потребами та обмеженими можливостями.
6. Тестування: Протестуйте інтерфейс, щоб переконатися, що він працює, простий у використанні та коректний. Тестуйте з реальними користувачами, отримуйте зворотній зв'язок і враховуйте його при подальшій розробці інтерфейсу.[10]

На всіх сторінках додатку (окрім реєстрації та авторизації) використовуються блоки "меню" та "футер". Вони є постійними і містяться в окремих файлах menu.js та slider.js у скрипті. Меню містить логотип додатку.

Головна сторінка містить:

1. Загальну інформацію про додаток.
2. Онлайн чат.
3. Створення резюме.

Вміст головної сторінки формується за допомогою окремих блоків. Схема створення контенту головної сторінки показана на рисунку 3.2.

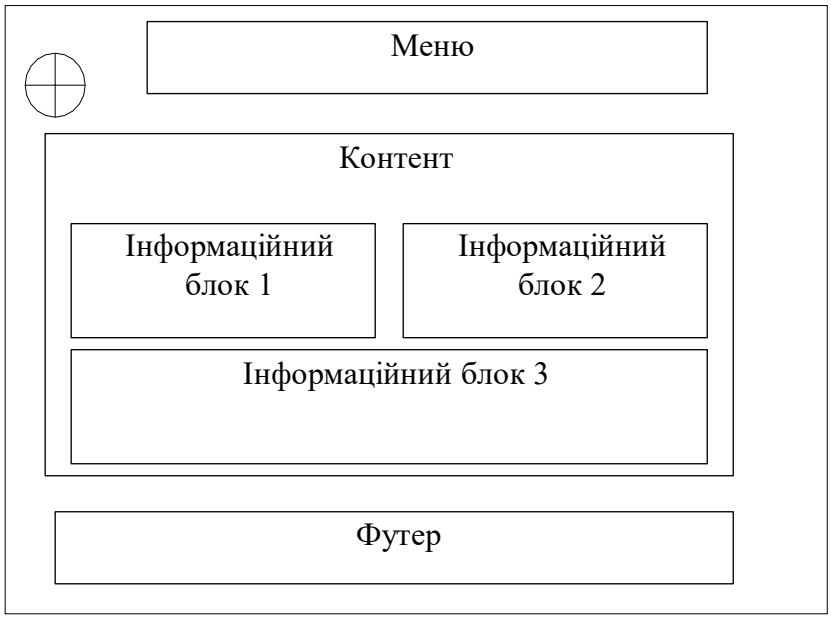


Рисунок 3.2 – Схема головної сторінки веб – додатку

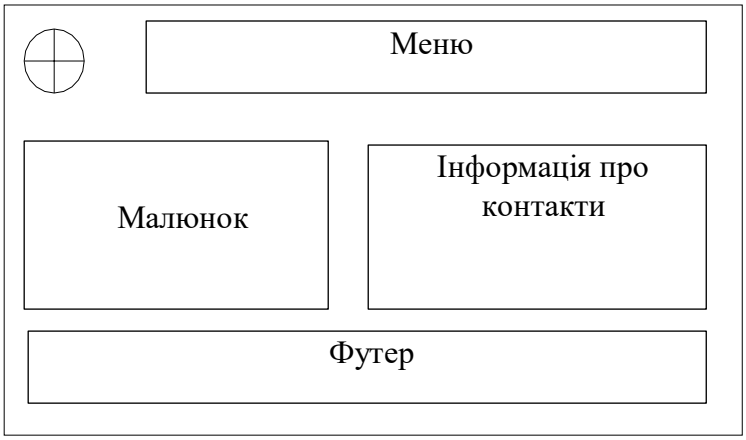


Рисунок 3.3 – Схема сторінки Інформація про контакти додатку

3.3. Проектування архітектури системи для веб – додатка

Архітектура веб-додатків зазвичай будується на основі архітектури клієнт-сервер, в якій один або кілька клієнтських пристроїв запитують інформацію з сервера. Веб-додатки можуть бути побудовані на основі технологій тонкого і товстого клієнта. У товстому клієнті обробка даних відбувається на робочій станції клієнта, а браузер запускає програму, яка безпосередньо взаємодіє з базою даних. У тонких клієнтах веб-сервер отримує динамічний запит на веб-сторінку, виконує всі необхідні операції для її відображення і відправляє її клієнту для відображення в браузері. Веб-додатки не мають високоінтерактивного інтерфейсу з великою кількістю користувацьких інструментів. У той же час, користувачі веб-додатків часто використовують мобільні пристрої для перегляду інформації, тому мобільний трафік необхідно контролювати. Розробники постійно працюють над додатковими функціями, які збільшують функціональність мови, спрощують її синтаксис і покращують захист від можливих атак.

На рисунку 3.4 схематично показано цю модель розробки. Пунктирна лінія розділяє традиційні клієнтське та серверне середовища виконання, а стрілки вказують напрямок руху даних між клієнтською та серверною сторонами. Серверна частина бере на себе всі функції рендерингу запиту, тоді як клієнтська частина займається обробкою згенерованого статичного контенту. Слід зазначити, що в даному випадку представлена реалізація "тонкого клієнта".

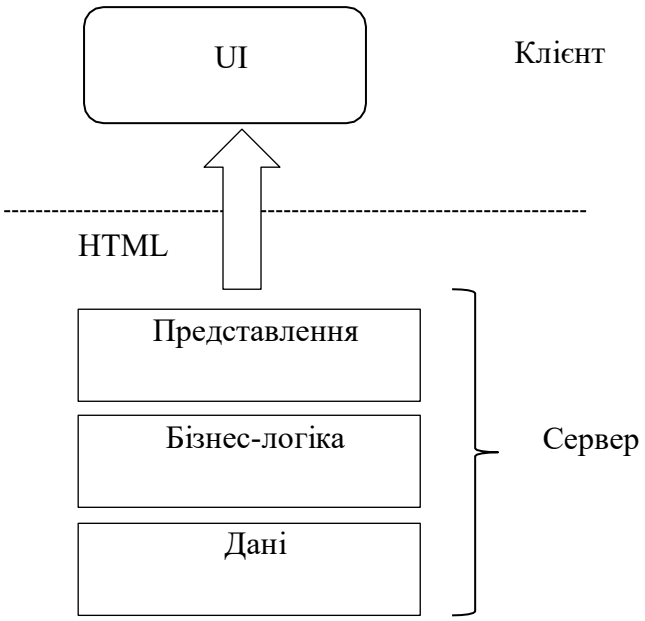


Рисунок 3.4 – Схема патерну "тонкий" клієнт

Клієнти повинні мати можливість отримати інформацію про додаток, задати запитання, створити та додати власну інформацію. На статичній веб-сторінці вміст визначається в момент створення сторінки. Коли користувач отримує доступ до статичної сторінки, на ній завжди відображається однакова інформація. Прикладом статичної веб-сторінки є сторінка, яка відображає інформацію про компанію. На динамічних веб-сторінках вміст змінюється на основі взаємодії користувача або даних із зовнішніх джерел.

Узагальнена архітектура веб – додатка для створення резюме наведена на рис. 3.5.

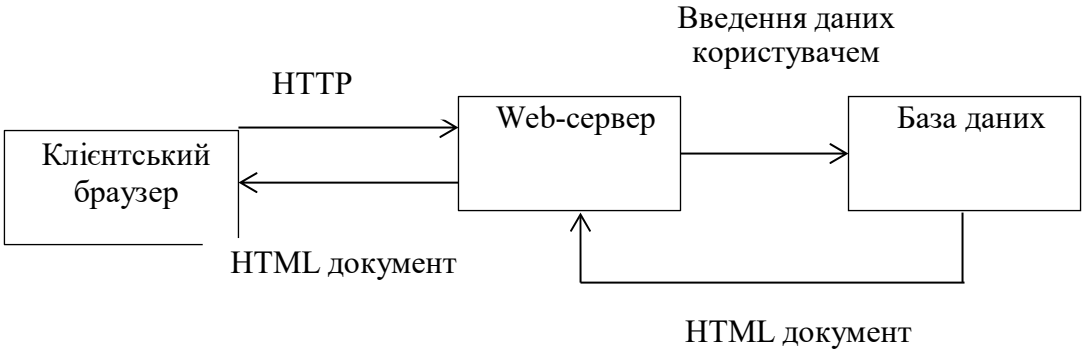


Рисунок 3.5 – Узагальнена архітектура веб - додатка

Фізичне представлення системи може бути показано у вигляді діаграми компонентів. Ця діаграма може показувати архітектуру системи та залежності між програмними компонентами. Основними графічними елементами цієї діаграми є компоненти та інтерфейси, а також залежності між компонентами.

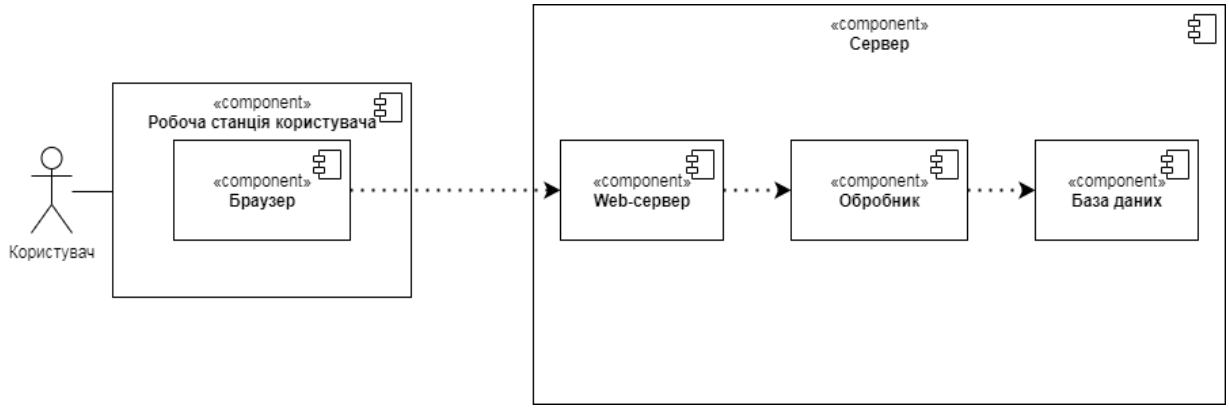


Рисунок 3.6 – Діаграма компонентів

На основі розробленого веб-додатку можна виділити інструменти для його розробки. Важливим етапом у розробці будь-якого веб-проекту є проектування архітектури. Правильно спроектована архітектура забезпечує безпеку, ефективність, та надійність додатку. Загальні кроки, яких слід дотримуватися при проектуванні архітектури веб-додатку, такі:

1. Визначення вимог: ретельно продумайте вимоги до веб-додатка. Розмір, функціональність, продуктивність, масштабованість і безпека - ось лише деякі з міркувань під час визначення вимог.

2. Створення концептуальної моделі: створіть концептуальну модель, яка показує взаємозв'язки між різними компонентами програми. Це може також бути щось накшталт графічної діаграми або концептуального опису архітектури.

3. Вибрати технологію: вибрати набір технологій, які будуть використовуватися у веб-додатку. Це можуть бути або мови програмування, або фреймворки, сервери, бази даних, API тощо. Враховуйте вимоги проєкту та досвід команди.

4. Визначити структуру додатка: визначити структуру веб-додатка, включно з поділом функціональності на компоненти і модулі. Розгляньте такі питання, як поділ логіки клієнта і сервера, маршрутизація, автентифікація та авторизація, кешування та обробка помилок.

5. Проектування бази даних: якщо веб-додаток використовує базу даних, розробіть структуру бази даних, включно з таблицями, зв'язками, індексами та правилами цілісності даних. Враховуйте ефективність і масштабованість бази даних.

6. Розгортання і масштабованість: розгляньте розгортання веб-додатка, включно з хостингом, конфігурацією сервера, налаштуваннями безпеки, резервним копіюванням і моніторингом. Плануйте масштабування таким чином, щоб потужність додатка зростала разом із навантаженням.

7. Тестування: визначте стратегію тестування веб-додатка. включіть модульне, інтеграційне та функціональне тестування, щоб переконатися, що додаток працює правильно і відповідає вимогам.

8. Створення документації: завершіть архітектурний дизайн вашого веб-додатка, створивши документацію, що буде описувати інтеграцію, структуру, компоненти, та інші важливі аспекти вашого додатка. Це допоможе іншим розробникам зрозуміти ваш проект і полегшить майбутню підтримку.

Загальна мета проектування архітектури веб-додатків - створити гнучкий, ефективний і надійний додаток, який буде відповідати вимогам проекту і задовольнятиме потребам користувачів. Існують різні архітектурні підходи до проектування веб-додатків, залежно від вимог проекту та специфікацій розробки. Нижче описані деякі загальні архітектурні моделі для веб-додатків:

1. Модель-вид-контролер (MVC): є одним із найпоширеніших патернів проектування веб-додатків. Він ділить додаток на такі основні компоненти: модель (дані), представлення (відображення) та контролер (логіка обробки); MVC дає змогу керувати логікою бізнес-процесу та його представленням користувачеві окремо.
2. Модель-вид-модель (MVVM): цей патерн дуже схожий на MVC, але фокусується на зв'язуванні даних і відокремлює логіку представлення від бізнес-логіки; модель ViewModel використовується для керування даними та взаємодії з уявленнями; модель ViewModel використовується для керування даними та взаємодії з уявленнями.
3. Мікросервіси: даний підхід використовує розподілену архітектуру, де додаток, в свою чергу, складається з набору незалежних сервісів, що взаємодіють через API. Кожен сервіс виконує певну функцію і може бути розгорнутий, масштабований і оновлений незалежно один від одного.
4. Односторінкові застосунки (SPA): у SPA весь код застосунку завантажується разом із першою сторінкою, а взаємодія із сервером здійснюється через AJAX-запити. Це дає змогу створювати багатокомпонентні додатки, які працюють динамічно без необхідності перезавантажувати сторінки..

Для веб-додатка, який допомагає створювати резюме, можна розглянути наступну архітектуру та інтерфейс.

1. Архітектура:

- Фронтенд: Використовуйте фреймворк або бібліотеку для створення інтерфейсу користувача. Наприклад, можна використовувати React або Angular. Фронтенд-додаток відповідає за відображення інтерфейсу користувача, обробку взаємодії з користувачем та передачу даних на сервер.
- Бекенд: Розробіть серверний бекенд, який обробляє запити від фронтенду та зберігає дані. Можна використовувати мови програмування, такі як Python або Node.js, та фреймворки, такі як Django або Express.js. Бекенд відповідає за обробку логіки додатка, валідацію даних, зберігання та отримання даних з бази даних.
- База даних: Використовувати підходящу базу даних для зберігання резюме та пов'язаних даних. Наприклад, можна використовувати реляційну базу даних, таку як MySQL або PostgreSQL. Створити схему бази даних, яка відображає структуру резюме та інших сутностей, які можуть бути присутніми (наприклад, освіта, досвід роботи тощо).

2. Інтерфейс:

- Реєстрація та вхід в систему: Надавати можливість користувачам створити обліковий запис або увійти до системи за допомогою електронної пошти або соціальних мереж.
- Створення резюме: Розробити інтерфейс, який дозволить користувачам створювати резюме. Надати форми та поля для введення інформації про освіту, досвід роботи, навички, контактні дані тощо. Додавати можливість додавання розділів та редагування існуючих даних.
- Перегляд та редагування резюме: Забезпечувати можливість переглядати та редагувати резюме, які були створені. Дозволити користувачам змінювати інформацію, додавати нові розділи, видаляти або переміщувати існуючі розділи.

Вигляд веб-додатку для створення резюме може бути різним в залежності від дизайну та функціональних вимог. Однак, ось кілька загальних принципів та компонентів, які можуть бути використані для створення привабливого та зручного вигляду додатку:

1. Заголовок та логотип: Додавати заголовок вашого додатку та можливо логотип, які будуть видні на кожній сторінці. Це створить єдиний ідентифікатор додатку для користувачів.

2. Меню навігації: Використовувати верхнє меню або бічне меню для навігації між основними розділами додатку. Меню може бути статичним або розкривається під час наведення курсора.

3. Створення резюме: Розробити інтуїтивно зрозумілий та привабливий інтерфейс для створення резюме. Використовувати форми, поля введення, списки випадаючих меню та кнопки для додавання та редагування інформації про особисті дані, освіту, досвід роботи, навички та контактні дані.

4. Макет резюме: Показувати макет резюме у режимі реального часу, коли користувач вносить зміни. Це дозволить користувачам бачити, як їх резюме виглядає на вигляд під час редагування.

5. Перегляд та редагування: Забезпечити можливість переглядати та редагувати розділи резюме окремо. Користувачі можуть переходити від одного розділу до іншого, вносити зміни та оновлювати інформацію.

6. Дизайн шаблону резюме: Надавати користувачам вибір з кількох дизайнерських шаблонів резюме. Вони можуть обирати стиль, кольорову палітру та розташування елементів.

7. Перегляд та завантаження: Забезпечте можливість користувачам переглядати своє резюме в переглядачі або завантажувати його у форматі PDF або іншому форматі.

8. Профіль користувача: Розробити сторінку профілю, де користувачі зможуть переглядати та оновлювати свої особисті дані, пароль, налаштування приватності та інші деталі свого облікового запису.

Під час вибору архітектури застосунку важливо враховувати такі аспекти:

1. Масштабованість: враховуйте потенціал зростання обсягу даних, трафіку і користувачів. Виберіть архітектуру, яка дає змогу додатку масштабуватися без збоїв і дозволяє легко додавати нові компоненти.

2. Продуктивність: враховуйте час відгуку та вимоги до обробки запитів. Виберіть архітектуру, яка може ефективно виконувати обробку даних, здійснювати запити до бази даних та виконувати інші операції для забезпечення швидкодії програми.

3. Розширюваність: забезпечити можливість додавання нових функцій і змін до застосунку без необхідності переписувати наявний код. Також ви маєте вибрати архітектуру , яка підтримує простоту розширення і модульність.

4. Надійність: треба забезпечити стабільність і надійність застосунку. Ви маєте вибрати архітектуру, яка забезпечує управління помилками та аварійне відновлення, включно з резервуванням і механізмами відновлення.

5. Безпека: розгляньте вимоги безпеки додатка. Виберіть архітектуру, яка забезпечує механізми безпеки, що відповідають вимогам додатка, такі як захист даних, контроль доступу тощо.

6. Простота розроблення та підтримки :подивіться, чи доступні необхідні інструменти та технології для розроблення та підтримки обраної архітектури. Виберіть архітектуру, яка дає змогу розробникам ефективно працювати та підтримувати додаток.

7. Витрати: зробіть аналіз витрат на підтримку, розробку, а також інфраструктуру, пов'язану з обраною архітектурою. Враховуйте баланс між потенційними витратами та перевагами, які пропонує архітектура.

8. Сумісність: переконайтеся, чи обрана архітектура сумісна з використовуваною технологією.

ВИСНОВОК ДО РОЗДІЛУ 3

У цьому розділі було описано реалізацію програми проєкту, а також була створена база даних проекту, основні функції веб-додатка та алгоритми роботи. Веб-додаток було представлено й описано з погляду на його архітектурну структуру . Розроблення інтерфейсу веб-додатка ґрунтується на аналізі функцій, виконуваних ресурсом. Після аналізу інформації, представленої в розділі 3.1, було розроблено діаграму варіантів використання. У результаті аналізу інформації, представленої в діаграмі варіантів використання, було визначено основні права користувачів групи 'клієнти'.

Проектування архітектури веб-додатка - важливий етап у розробці будь-якого веб-проекту. Добре спроектована архітектура забезпечує ефективність, масштабованість, безпеку і надійність програми. Крім того, вона дає змогу додавати нові функції та вносити зміни в застосунок без необхідності переписувати наявний код. Ми обрали архітектуру, яка підтримує модульність і простоту розширення, забезпечили стабільність і надійність застосунку. Ми обрали архітектуру, яка дає нам змогу справлятися з помилками і відновлюватися після збоїв, наприклад, за допомогою механізмів резервного копіювання та відновлення. Ми також розглянули вимоги до безпеки застосунку, враховували доступність інструментів і технологій, необхідних для розроблення та підтримки обраної архітектури.

РОЗДІЛ 4

ПРЕДСТАВЛЕННЯ РОБОТИ ТА ГРАФІЧНОГО ВИГЛЯДУ ПРОЄКТУ

4.1. Основні сторінки додатка

Загальні поради та опис, як може виглядати графічний інтерфейс додатку для створення резюме.

1. Головний екран: Зазвичай головний екран додатку для створення резюме містить меню або панель навігації, яка дозволяє користувачу переходити до різних секцій або функцій. Наприклад, такі секції можуть включати створення нового резюме або перегляд вже створених резюме.

2. Форма створення резюме: Графічний вигляд форми створення резюме може включати поля для введення основної інформації, такої як ім'я, контактна інформація, освіта, досвід роботи тощо. Користувач може заповнювати ці поля, натискаючи на них або використовуючи випадаючі списки.

3. Розділи резюме: Додаток може надавати розділи для структурування резюме, наприклад, "Освіта", "Досвід роботи", "Навички", "Проекти" тощо. Кожен розділ може мати свою власну форму або поля для введення відповідної інформації. Користувач може додавати нові розділи або видаляти існуючі, в залежності від своїх потреб.

5. Перегляд: Після створення резюме користувач може переглянути його увигляді попереднього перегляду перед збереженням або експортом.

Спочатку з'явиться екран входу в обліковий запис (або реєстрації), залежно від того, увійшов користувач уже в систему чи ні. На малюнку 4.1 показано форму входу в додаток. Ця форма має два поля введення, одне з яких містить адресу електронної пошти, а інше пароль користувача.

Кнопка 'SIGN UP' дозволяє користувачеві перейти на екран реєстрації. Кнопка 'LOG IN' дає змогу перейти на сторінку з авторизацією користувача, де починається процес авторизації після того, як у відповідні поля вводяться правильні значення.

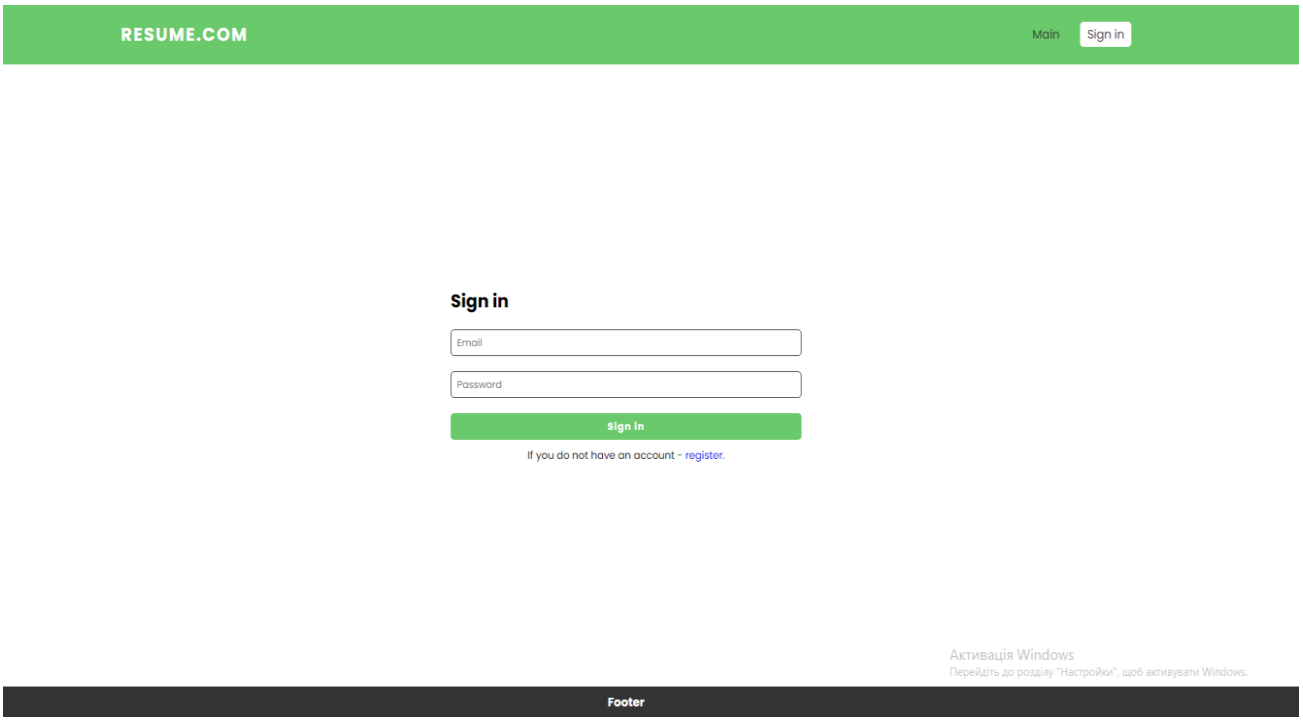


Рисунок. 4.1 – Екран входу LOG IN

На малюнку 4.2 показана форма реєстрації нового користувача. Дана форма містить 5 полів введення, відповідно, ім’я, призвіще, адреса електронної пошти, пароль та повторне введення паролю. Натиснувши на кнопку «SIGN UP» буде розпочато процес реєстрації після введення відповідних значень. Кнопка «LOG IN» дозволяє користувачу повернутися до екрану входу в додаток, якщо користувач вже зареєстрований.

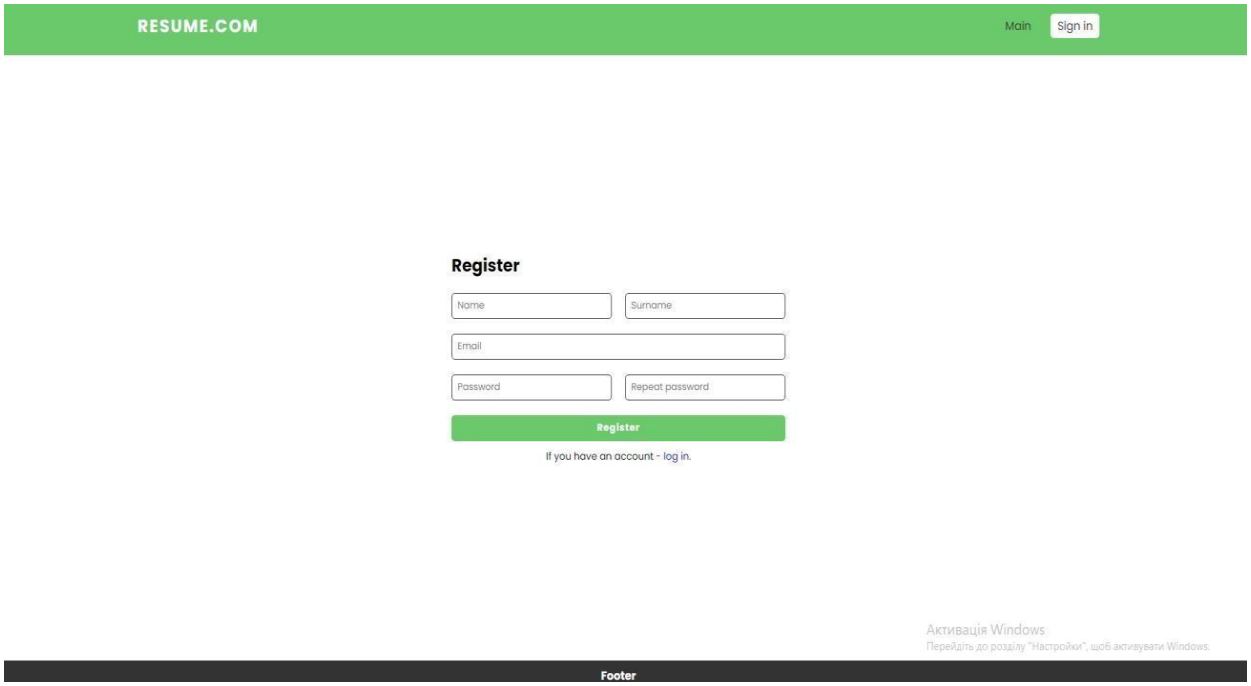


Рисунок. 4.2 – Екран реєстрації SIGN UP

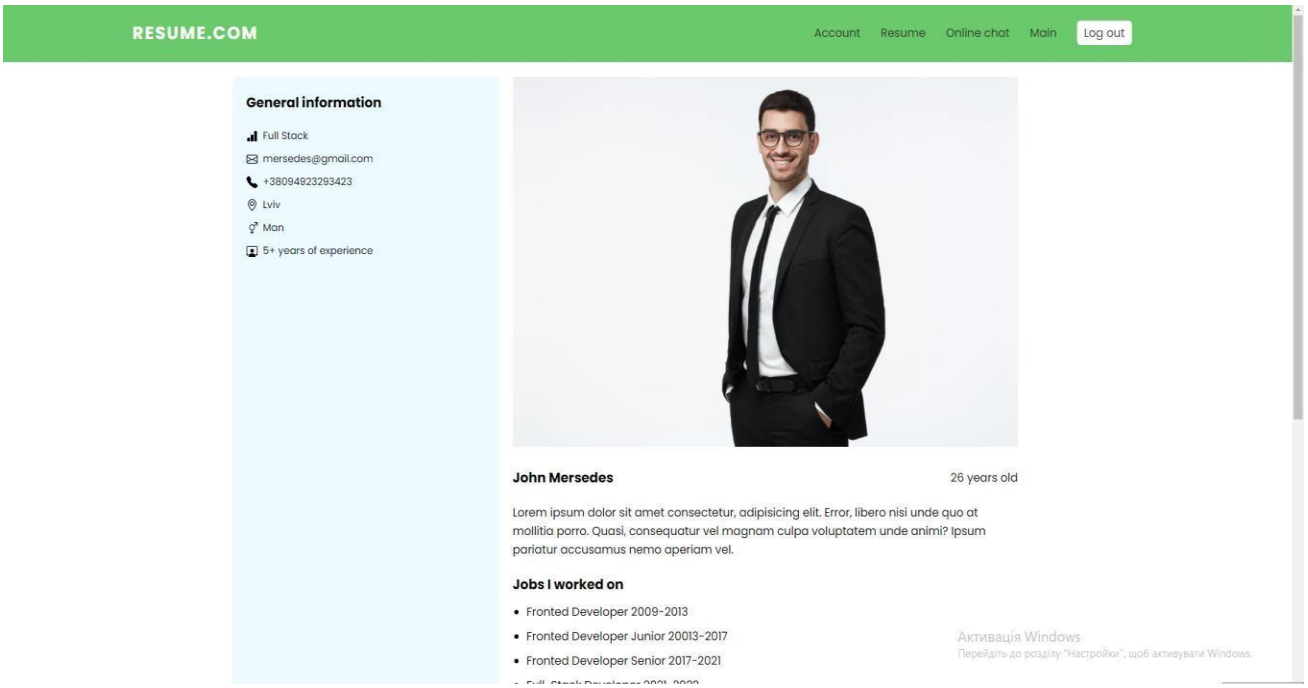


Рисунок 4.3. Екран готового резюме

На рисунку 4.3 представлено вигляд готового резюме. На ній користувач має можливість ознайомитись із готовим резюме та перейти до сторінки з редагуванням резюме,якщо його щось не влаштовує.

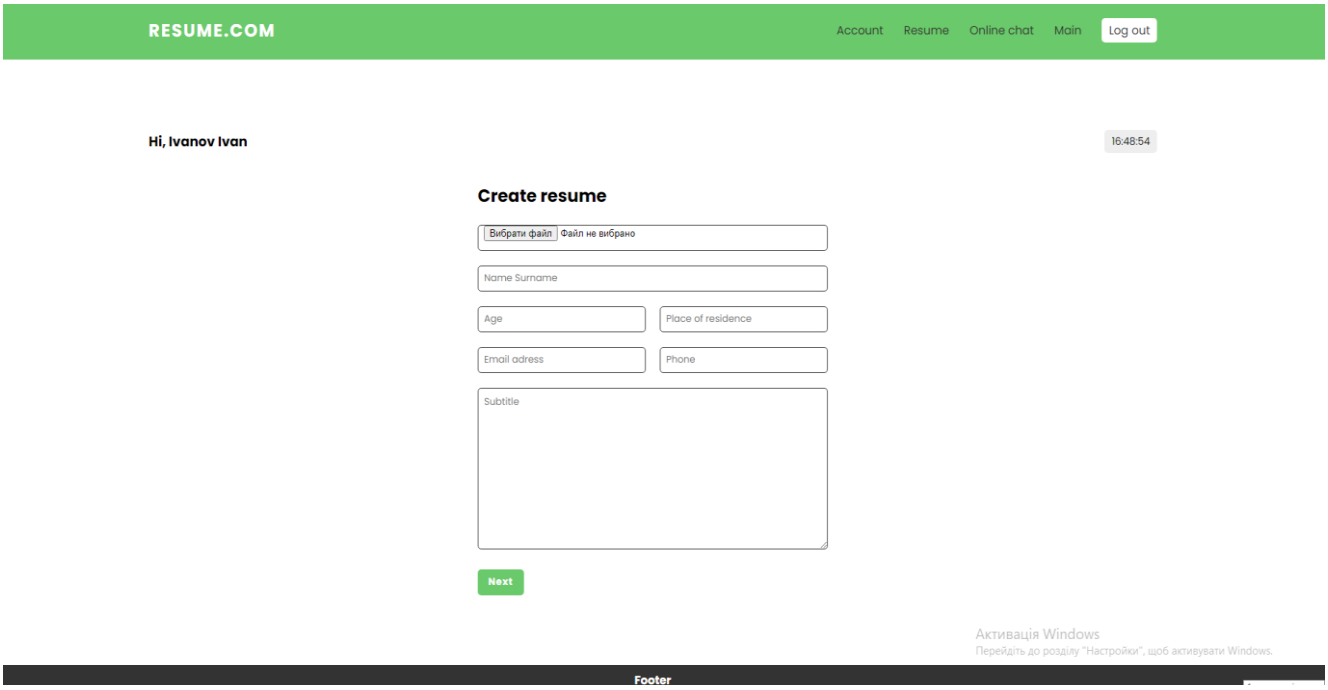


Рисунок 4.4. Екран для створення резюме

На рисунку 4.4 представлена сторінка для створення резюме.. На ній користувач має можливість додати власну інформацію для створення резюме, а саме: додати фото, додати здобуту освіту, вік, додати місця минулої роботи, , електронну адресу, телефон та інше.

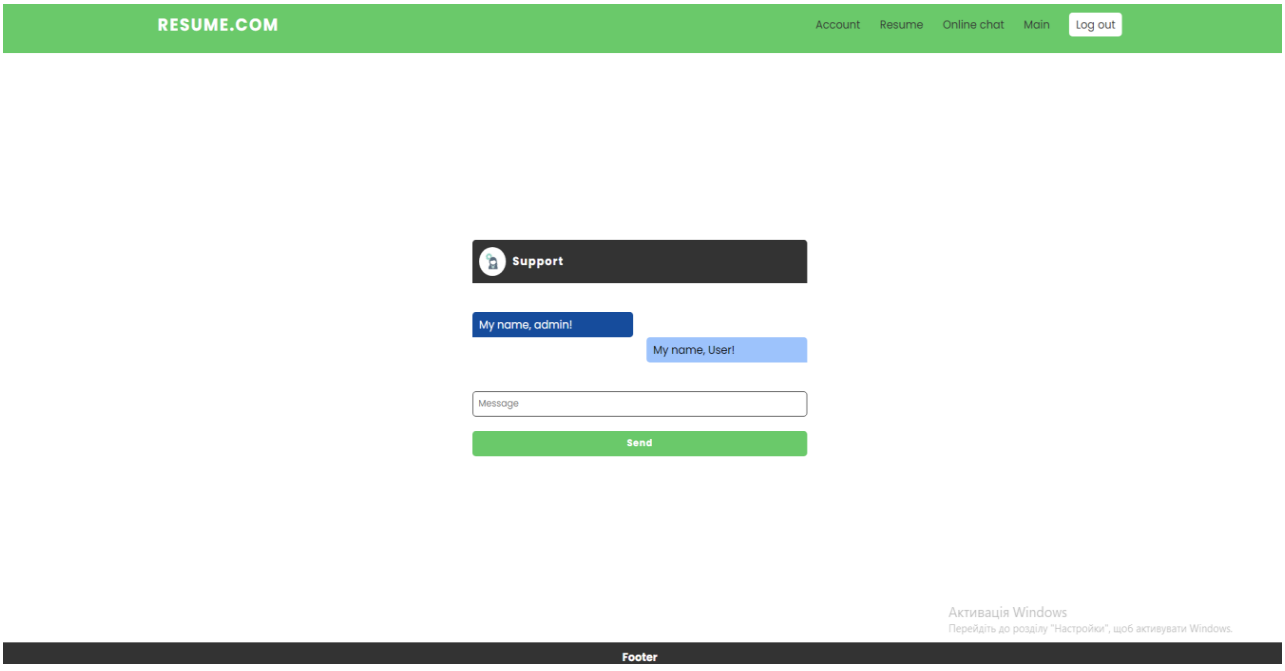


Рисунок 4.5. Екран підтримки

На рисунку 4.5 представлена сторінка для підтримки користувачів. На ній користувач має можливість звернутись з будь – якого питання до адміна, і швидко отримати відповідь.

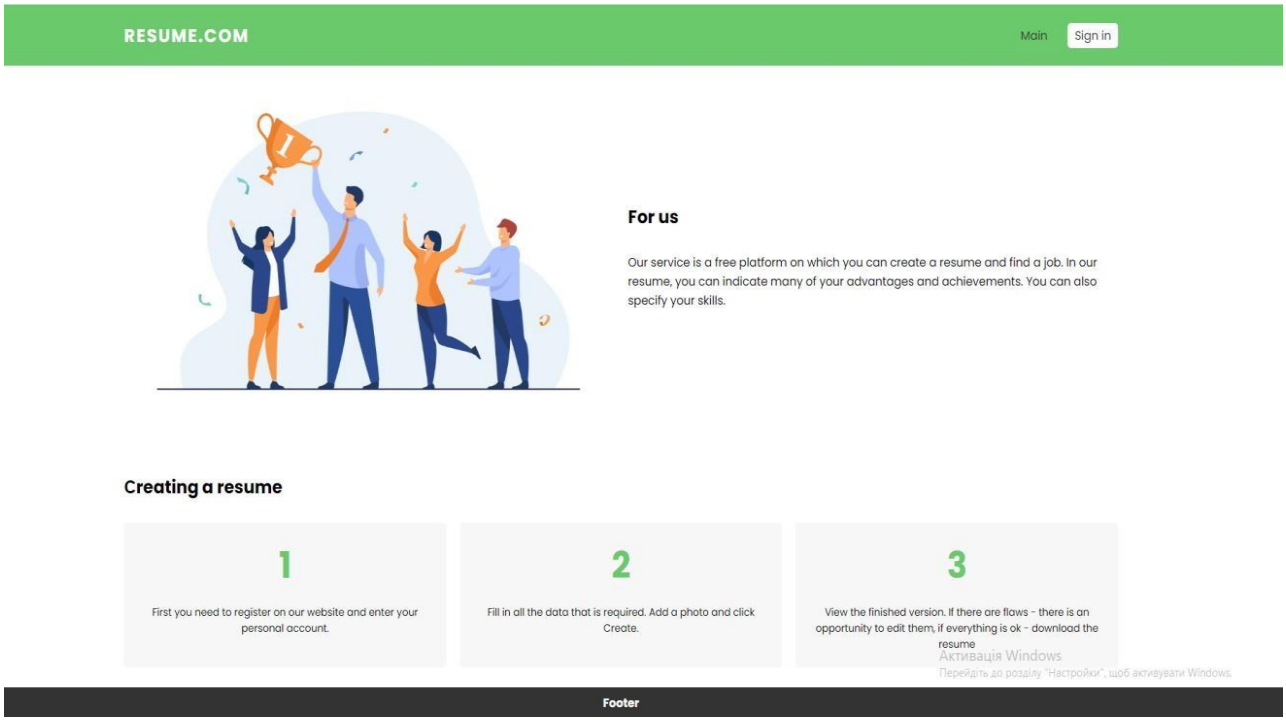


Рисунок. 4.6. Екран інформації про додаток

На рисунку 4.6 представлена сторінка про додаток та 3 етапи створення ідеального резюме. На ній користувач має можливість ознайомитись з етапами створення резюме та інформацією про самий додаток. Дана сторінка знайомить клієнта з додатком, його концепцією. З цієї сторінки користувач може взяти корисну інформацію та поближче познайомитись з веб – додатком. На даній сторінці було представлено актуальну інформацію про створення резюме, його основні пункти й інформацію, яку потрібно вности в своє резюме.

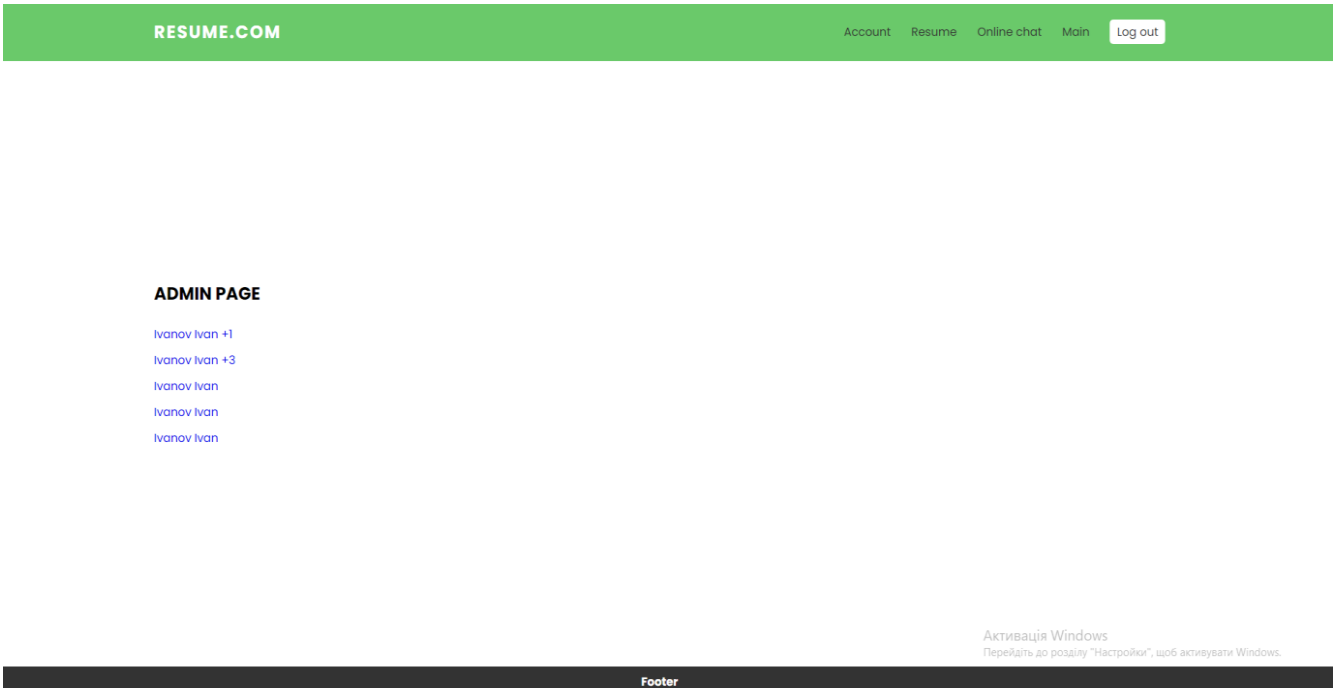


Рисунок 4.7. Екран адміна

На рисунку 4.7. представлена сторінка адміна. На ній адмін переглядає інформацію про зареєстрованих користувачів в додатку та може перейти до чату підтримки, якщо користувач до неї звернувся.

Навігація є важливою складовою веб-додатків, оскільки вона дозволяє користувачам переходити між різними сторінками та функціональними частинами додатку. Ось декілька можливих способів організації навігації у вашому додатку для створення резюме:

1. Верхнє меню: Розмістіть горизонтальне верхнє меню, де можна розмістити основні розділи вашого додатку, наприклад "Домашня сторінка", "Створити резюме", "Мої резюме", "Налаштування" та "Вихід". Користувачі зможуть вибирати потрібний розділ, натискаючи на відповідний пункт меню.

2. Плаваюча панель із навігацією: Використовуйте плаваючий панель або меню, яке розташовується на постійній позиції на екрані, незалежно від прокрутки сторінки. Це забезпечить постійний доступ до навігаційних посилань, навіть при прокручуванні вмісту сторінки.

3. Кнопки дії: Додайте кнопки дій, які відображаються на кожній сторінці, наприклад "Зберегти", "Редагувати", "Додати новий розділ" тощо.

4. Пошук: Додайте можливість пошуку в додатку, яка дозволить користувачам швидко знайти потрібну інформацію або резюме за певними критеріями.

5. Віджети або панелі: Використовуйте віджети або панелі, які можна розмістити на кожній сторінці і містять посилання на інші важливі розділи або функції додатку.

Створюючи ідеальну навігацію, потрібно включити декілька аспектів:

1. Чітка структура меню: Розробити логічну структуру меню, де основні секції та функціональні можливості додатка будуть легко доступні. Меню повинно бути видимим і легкодоступним на всіх сторінках.

2. Консистентність: Зберігати однаковий стиль та розташування меню на всіх сторінках додатка, щоб користувачі легко могли орієнтуватися та переключатися між розділами.

3. Інтуїтивний дизайн: Використовувати зрозумілі та зрозумілі піктограми або написи, що відображають функціональні можливості. Користувач повинен зрозуміти, куди він буде перенаправлений або яку дію виконає, клікнувши на певний елемент навігації.

4. Виокремлення активної сторінки: Позначати активну сторінку або розділ в меню, щоб користувачі знали, на якій сторінці вони знаходяться.

5. Пошук та фільтрація: Забезпечити можливість швидкого пошуку і фільтрації контенту в додатку, якщо це необхідно. Це допоможе користувачам знаходити необхідну інформацію або функції швидше та ефективніше.

6. Контекстні посилання: Забезпечити контекстні посилання або швидкі посилання, які допомагають користувачам переходити до пов'язаних розділів або функцій без необхідності повного навігування.

7. Тестування та зворотний зв'язок: Провести тестування навігації з реальними користувачами та отримайте їхні відгуки. Прослухати їхні пропозиції щодо поліпшення навігації та вносьте необхідні зміни.

4.2. Функіонал веб - додатка

Наступні кроки:

1. Реєстрація та вхід в систему:

- Користувач може створити обліковий запис, вказавши необхідну інформацію, таку як електронна пошта та пароль.
- Після реєстрації користувач може увійти в систему, використовуючи свої облікові дані.

2. Додавання основної інформації:

- Користувач може ввести основну інформацію про себе, таку як ім'я, прізвище, контактні дані та заголовок резюме.
- Користувач може також додати свою профільну фотографію та інші особисті дані, які він вважає за потрібне.

3. Додавання освіти та досвіду роботи:

- Користувач може ввести свою освіту, включаючи назву закладу, спеціальність та роки навчання.
- Користувач може додати свій досвід роботи, вказавши назву компанії, посаду, період зайнятості та опис своїх обов'язків та досягнень.

4. Додавання навичок та вмінь:

- Користувач може вказати свої навички та вміння, які він володіє, наприклад, програмування, мови програмування, м'які навички тощо.
- Він також може оцінити свій рівень володіння кожною навичкою (наприклад, по шкалі від початківця до експерта).

5. Перегляд та редагування резюме:

- Користувач може переглянути своє резюме у вигляді, який буде використовуватися для подання.

6. Збереження резюме:

- Користувач може зберегти своє резюме у системі для майбутнього використання та редагування.

4.3. Графічний вигляд

Графічне оформлення резюме має бути професійним, привабливим і простим у використанні. Нижче наведені рекомендації щодо графічного оформлення:

1. Мінімалістичний дизайн: використовуйте простий, чистий дизайн, щоб забезпечити легку навігацію і читабельність інформації. Уникайте перевантаження сторінки зайвими деталями.

2. Професійні шрифти: обирайте професійні, легкі для читання шрифти для тексту, які підходять для читання на екрані. Уникайте використання шрифтів, які важко читати, складних або незвичних.

3. Кольорова палітра: використовуйте обмежену кольорову палітру, яка створює гармонійний вигляд, що відповідає бренду. Зверніть увагу на кольорові контрасти, щоб текст був читабельним і можна було розрізнити елементи.

4. Ілюстрації та фотографії: використання візуальних елементів, таких як ілюстрації та фотографії, додає вашому додатку привабливості. Переконайтеся, що вони відповідають стилю додатку та підкреслюють його професіоналізм.

5. Інтерактивність: розгляньте можливість використання інтерактивних елементів, які покращують користувацький досвід. Наприклад, використовуйте кнопки, переходи, анімацію, поступове завантаження даних тощо.

6. Адаптивний дизайн: переконайтеся, що додаток реагує на різні пристрої та розміри екранів. Переконайтеся, що він добре виглядає як на десктопних, так і на мобільних пристроях.

7. Простота використання: забезпечте просту та інтуїтивно зрозумілу навігацію, щоб ті хто користуються додатком могли швидко знаходити потрібну їм інформацію та легко орієнтуватися в додатку.

ВИСНОВОК ДО РОЗДІЛУ 4

Цей розділ підсумовує детальний посібник користувача для розроблюваного веб-ресурсу. Описано структуру та зміст сторінок додатку. Додаток містить різні сторінки і користувач може переміщатися між ними. Показано візуальний інтерфейс всіх основних сторінок додатку та вигляд деяких дій користувача. Навігація є важливою складовою веб-додатку, оскільки вона має дозволяти користувачам легко переміщатися між різними сторінками додатку. Проте, загалом, сторінки додатка для створення резюме мають мати такі характеристики:

1. Головна сторінка: На головній сторінці розміщений вступний екран або панель управління, почати створювати нове резюме. Також в додатку доступні додаткові опції, наприклад, перегляд збережених резюме.

2. Сторінка створення резюме: Ця сторінка містить форму або редактор, де користувач зможе ввести свою професійну інформацію. Розроблений веб-додаток використовує адаптивний дизайн, який працює на різних пристроях і розмірах екрана. Для тексту було обрано легкочитабельні та професійні шрифти, які підходять для читання на екрані. Розглянули такі можливості як впровадження елементів бренду, колір, стиль і логотип для того щоб створити відчуття єдності та впізнаваності програми.

ВИСНОВКИ

Під час виконання дипломної роботи було створено багатокористувацький веб – додаток для створення резюме із покращеним функціоналом. Даний сервіс надає можливість користувачам легко і зручно створювати ідеальне резюме

У першому розділі ми проаналізували та розглянули основні причини та тенденцію росту популярності веб – додатків для створення резюме. Було виявлено, що за прогнозами зростання попиту на робочі місця їх популярність та потреба у них буде тільки зростати. Також було розглянуто найпопулярніші аналоги зі схожою тематикою, охарактеризовано їх переваги та недоліки.

У другому розділі було описано причини вибору редактору VSCODE та програм, за допомогою яких буде створено додаток. Також, проаналізувавши етапи створення веб – додатків, був складений короткий план розробки: мета програми: веб – додаток для створення резюме; цільова аудиторія: користувачі, що бажають просто та зручно розповісти про свої навички для подальшого працевлаштування; аналіз конкурентів: існуючі аналоги веб – додатків.

У третьому розділі було поетапно розписано програмну реалізацію проекту. Описано створену базу даних проекту та обгрунтовано її вибір, а також описано основні функції та алгоритми веб – додатку. Було розглянуто веб – додаток з точки зору архітектурної його будови.

У четвертому розділі було продемонстровано роботу програми, користувацький інтерфейс та дії, які може виконувати користувач. У розділі представлений готовий веб – додаток. Було надано детальні інструкції щодо того, як користуватися додатком. Слід також зазначити, що після завершення розробки веб – додатка був зроблений аналіз та висновки, що були враховані майже всі зазначені на початку вимоги, яким він мав відповідати.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Архітектури та технології WEB-служб. Конспект лекцій для студентів спеціальності 121 Інженерія програмного забезпечення усіх форм навчання. [Електронний ресурс]. Режим доступу до ресурсу: http://eir.zntu.edu.ua/bitstream/123456789/2852/1/Stepanenko_Summary_of_lectures.pdf.
2. Використання веб-ресурсів. [Електронний ресурс]. Режим доступу до ресурсу: http://galanet.at.ua/publ/1-1-0-23 .
3. Етапи створення веб-ресурсів. [Електронний ресурс]. Режим доступу до ресурсу: http://edufuture.biz/index.php?title =Етапи створення веб- ресурсів.
4. HTML [Електронний ресурс]. Режим доступу до ресурсу: https://ua.wikipedia.org/wiki/HTML5.
5. JavaScript. [Електронний ресурс]. Режим доступу до ресурсу: https://learn.javascript.ua/
6. MySQL. [Електронний ресурс]. Режим доступу до русерсу: https://www.mysql.com/ .
7. SitePoint. [Електронний ресурс]. Режим доступу до ресурсу: https://www.sitepoint.com/.
8. "Web-програмування". [Електронний ресурс]. Режим доступу до ресурсу:https://web-programming.com.ua/.
9. "ProCode". [Електронний ресурс]. Режим доступу до ресурсу: https://procode.com.ua/.
10. "UAWEB". [Електронний ресурс]. Режим доступу до ресурсу: https://uaweb.info/.

ДОДАТОК 1

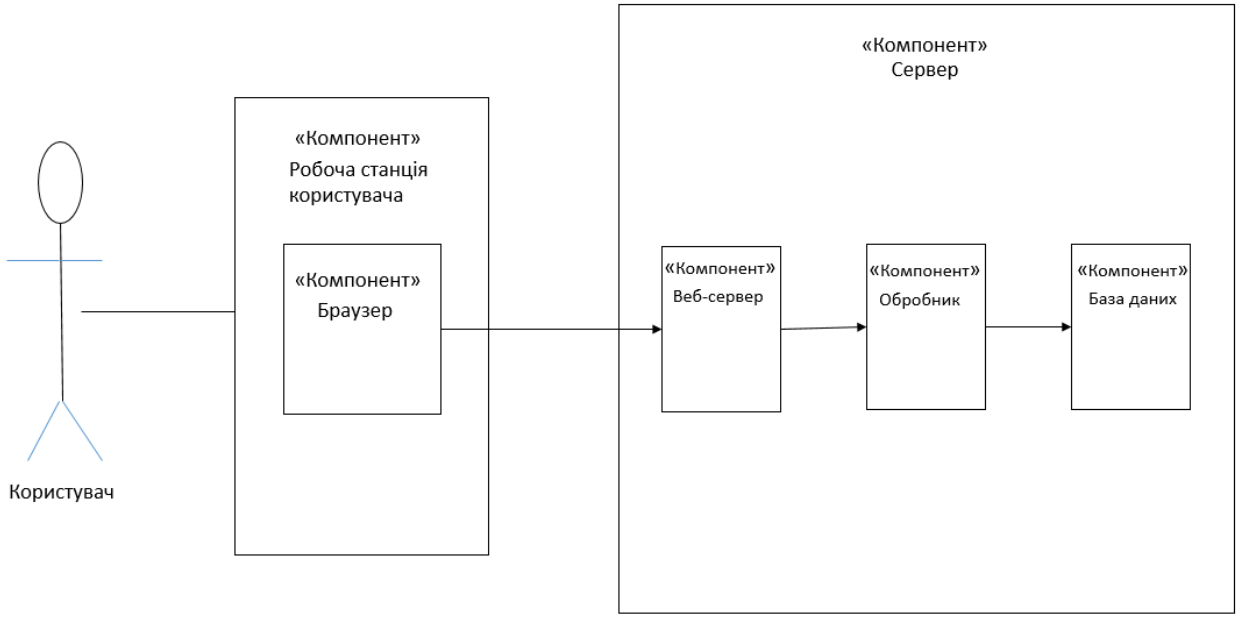
Додаток для створення резюме

Даіграма компонентів

ІАЛЦ.467200.004 Д1

Аркушів 1

Київ 2023 р



					ІАЛЦ.467200.004 Д1		
		№ докум.	Підпис	Дата			
Розробив		Бондік К.Г.			*Додаток для створення резюме* Діаграма компонентів (Структурна схема)	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Валько В.В.					
Н. Контр.		Волокита А.М.				НТУУ КПІ ім. Ігоря Сікорського,ФІОТ,ІП-94	
Затвердив							

ДОДАТОК 2

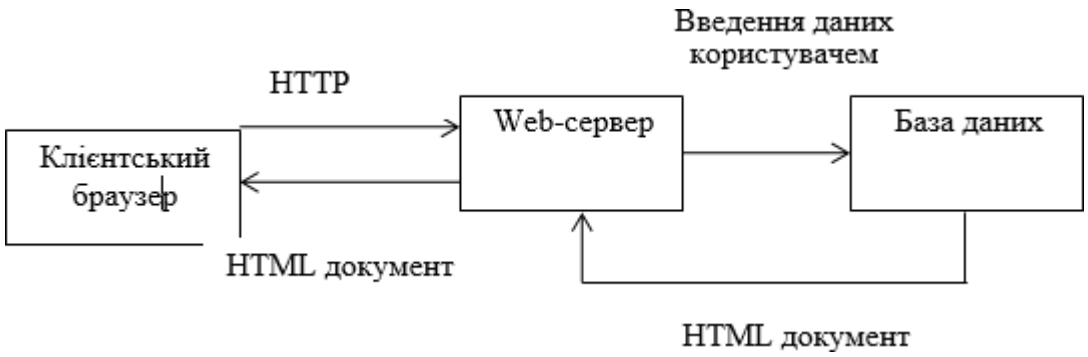
Додаток для створення резюме

Алгоритм дії програмного забезпечення

ІАЛЦ.467200.005 Д2

Аркушів 1

Київ 2023 р



					ІАЛЦ.467200.005 Д2		
		№ докум.	Підпис	Дата			
Розробив		Бондік К.Г.			*Додаток для створення резюме* *Діаграма використання* (Функціональна схема)	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Валько В.В.					
						НТУУ КПІ ім. Ігоря Сікорського,ФІОТ,ІП-94	
Н. Контр.		Волокита А.М.					
Затвердив							

ДОДАТОК 3

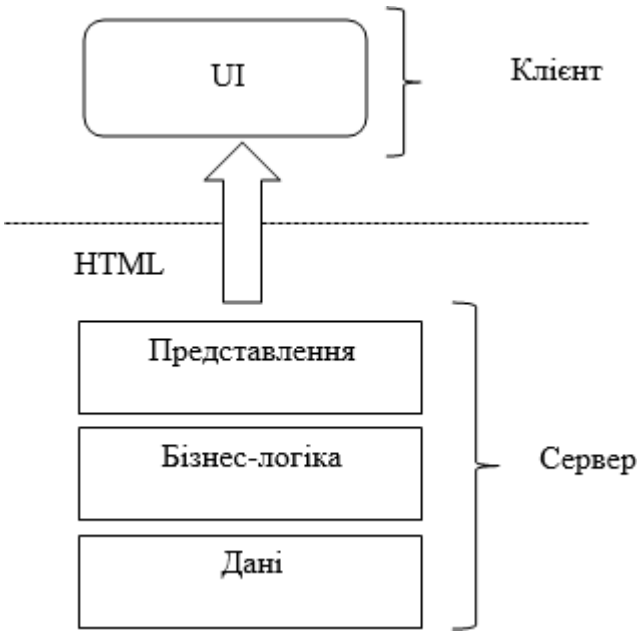
Додаток для створення резюме

Алгоритм дій програмного забезпечення

ІАЛЦ.467200.006 Д3

Аркушів 1

Київ 2023 р



					ІАЛЦ.467200.006 Д3		
		№ докум.	Підпис	Дата			
Розробив		Бондік К.Г.			*Додаток для створення резюме* Алгоритм дій програмного забезпечення (Принципова схема)	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Валько В.В.					
Н. Контр.		Волокита А.М.				НТУУ КПІ ім. Ігоря Сікорського,ФІОТ,ІП-94	
Затвердив							

ДОДАТОК 4

Додаток для створення резюме

Лістинг коду

ІАЛЦ.467200.007 Д4

Аркушів 13

Київ 2023 р

index.php

```
<?php
session_start();
require_once "../db/db.php";

if (isset($_POST)) {
    $data = $_POST;

    // login
    if (isset($data['login'])) {
        if ($data['email'] == 'admin@admin.com' && $data['password'] == 'admin') {
            $_SESSION['name'] = 'admin';

            header('Location: /admin.php');
            exit();
        }

        $sql = "SELECT * FROM users WHERE email = :email";
        $stmt = $db->prepare($sql);
        $stmt->bindValue(":email", $data['email']);
        $stmt->execute();

        $user = $stmt->fetch(\PDO::FETCH_ASSOC);

        if (!$stmt->rowCount()) {
            header( "refresh:4;url=/login.php");
            echo 'Your login or password is incorrect. Try again! You will be
redirect to login page after 4s';
            exit();
        }
        $userpassword = $data['password'];
        $hashpassword = $user['password'];

        if (password_verify($userpassword, $hashpassword)) {
            $_SESSION['name'] = $user['name'].' '.$user['surname'];
            $_SESSION['id'] = $user['id'];

            header('Location: /account.php');
        } else {
            header( "refresh:4;url=/login.php");
            echo 'Your password is incorrect. Try again! You will be redirect to
login page after 4s';
            exit();
        }

    }
```

```
    // register
    if (isset($data['register'])) {
        $sql = "INSERT INTO users(name, surname, password, email)
                VALUES (:name, :surname, :password, :email)";

        $password = $data['password'];
        $repeatPassword = $data['repeat_password'];

        if ($password == $repeatPassword) {
            $stmt = $db->prepare($sql);
            $stmt->bindValue(":email", $data['email']);
            $stmt->bindValue(":name", $data['name']);
            $stmt->bindValue(":surname", $data['surname']);
            $stmt->bindValue(":password", password_hash($data['password'],
PASSWORD_DEFAULT));
            $stmt->execute();

            header('Location: /login.php');
        } else {
            header('Location: /register.php');
            echo 'Password and repeat password must be equal';
        }
    }

   // create resume
    if (isset($data['create'])) {
        $userId = $_SESSION['id'];

        $img = '';
        $filename = $_FILES['file']["name"];
        $tempname = $_FILES['file']["tmp_name"];
        $uploadfile = '../img/' . basename($_FILES['file']['name']);

        if (move_uploaded_file($tempname, $uploadfile))  {
            $img = $filename;
        }

        $sql = "INSERT INTO resume(
                    user_id,
            full_name,
            age,
            place,
            email ,
            phone,
            subtitle,
            vaccancy,
```

```
            last_place,
            years_experience,
            sex,
            education,
            languages,
            skills,
            hobby,
            about_me,
            img
)
                VALUES (
                        :user_id,
            :full_name,
            :age,
            :place,
            :email ,
            :phone,
            :subtitle,
            :vaccancy,
            :last_place,
            :years_experience,
            :sex,
            :education,
            :languages,
            :skills,
            :hobby,
            :about_me,
            :img
                )";

        $jobs = $data['job'];
        $yearsStart = $data['start_year'];
        $yearsEnd = $data['end_year'];

        $countJobs = count($jobs);
        $education = [];
        for ($i = 0 ; $i < $countJobs; $i++) {
            $education[] = ['job' => $jobs[$i], 'start_year' => $yearsStart[$i],
'end_year' => $yearsEnd[$i]];
        }

        $education = json_encode($education);

        $languages = json_encode($data['language']);
        $skills = json_encode($data['skill']);
        $hobbys = json_encode($data['hobby']);
```

```
    $stmt = $db->prepare($sql);
    $stmt->bindValue(":user_id", $userId);
    $stmt->bindValue(":full_name", $data['full_name']);
    $stmt->bindValue(":age", $data['age']);
    $stmt->bindValue(":place", $data['place']);
    $stmt->bindValue(":email", $data['email']);
    $stmt->bindValue(":phone", $data['phone']);
    $stmt->bindValue(":subtitle", $data['sub_title']);
    $stmt->bindValue(":vaccancy", $data['vaccancy']);
    $stmt->bindValue(":last_place", $data['last_place']);
    $stmt->bindValue(":years_experience", $data['years_experience']);
    $stmt->bindValue(":sex", $data['sex']);
    $stmt->bindValue(":education", $education);

    $stmt->bindValue(":languages", $languages);
    $stmt->bindValue(":skills", $skills);
    $stmt->bindValue(":hobby", $hobbys);

    $stmt->bindValue(":about_me", $data['about_me']);
    $stmt->bindValue(":img", $img);
    $stmt->execute();

    header('Location: /resume.php');
}

// edit
if (isset($data['edit'])) {
    $userId = $_SESSION['id'];

    $img = '';
    if (!empty($_FILES['file']['name'])) {
        $filename = $_FILES['file']["name"];
        $tempname = $_FILES['file']["tmp_name"];
        $uploadfile = '../img/' . basename($_FILES['file']['name']);

        if (move_uploaded_file($tempname, $uploadfile))  {
            $img = $filename;
        }
    }
    $jobs = $data['job'];
    $yearsStart = $data['start_year'];
    $yearsEnd = $data['end_year'];

    $countJobs = count($jobs);
    $education = [];
    for ($i = 0 ; $i < $countJobs; $i++) {
```

```
            $education[] = ['job' => $jobs[$i], 'start_year' => $yearsStart[$i],
'end_year' => $yearsEnd[$i]];
        }

        $education = json_encode($education);

        $languages = json_encode($data['language']);
        $skills = json_encode($data['skill']);
        $hobbys = json_encode($data['hobby']);

        $prepareSql = '
                full_name = :full_name,
                age = :age,
                place = :place,
                email = :email,
                phone = :phone,
                subtitle = :subtitle,
                vaccancy = :vaccancy,
                last_place = :last_place,
                years_experience = :years_experience,
                sex = :sex,
                education = :education,
                about_me = :about_me,
                languages = :languages,
                skills = :skills,
                hobby = :hobby
                ';

        if (!empty($_FILES['file']['name'])) {
            $prepareSql.= ',img = :img';
        }

        $sql = "UPDATE resume SET $prepareSql WHERE user_id = $userId";

        $stmt = $db->prepare($sql);
        $stmt->bindValue(":full_name", $data['full_name']);
        $stmt->bindValue(":age", $data['age']);
        $stmt->bindValue(":place", $data['place']);
        $stmt->bindValue(":email", $data['email']);
        $stmt->bindValue(":phone", $data['phone']);
        $stmt->bindValue(":subtitle", $data['subtitle']);
        $stmt->bindValue(":vaccancy", $data['vaccancy']);
        $stmt->bindValue(":last_place", $data['last_place']);
        $stmt->bindValue(":years_experience", $data['years_experience']);
        $stmt->bindValue(":sex", $data['sex']);
        $stmt->bindValue(":education", $education);
        $stmt->bindValue(":about_me", $data['about_me']);
```

```
    $stmt->bindValue(":languages", $languages);
    $stmt->bindValue(":skills", $skills);
    $stmt->bindValue(":hobby", $hobbys);

    if (!empty($_FILES['file']['name'])) {
        $stmt->bindValue(":img", $img);
    }
    $stmt->execute();

    header('Location: /resume.php');
}

// send message to admin
if (isset($data['send_message'])) {
    $userFrom = $_SESSION['id'];
    $sql = "INSERT INTO messages(user_from, user_to, message)
            VALUES (:user_from, :user_to, :message)";

    $stmt = $db->prepare($sql);
    $stmt->bindValue(":user_from", $userFrom);
    $stmt->bindValue(":user_to", 0);
    $stmt->bindValue(":message", $data['message']);
    $stmt->execute();

    header('Location: /chat.php');
}

if (isset($data['send_message_from_admin'])) {
    $userFrom = 0;
    $userTo = $data['user_to'];
    $sql = "INSERT INTO messages(user_from, user_to, message)
            VALUES (:user_from, :user_to, :message)";

    $stmt = $db->prepare($sql);
    $stmt->bindValue(":user_from", $userFrom);
    $stmt->bindValue(":user_to", $userTo);
    $stmt->bindValue(":message", $data['message']);
    $stmt->execute();

    header('Location: /chat-admin.php?id='.$userTo);
}

// load pdf
if (isset($data['load_pdf'])) {
    $userId = $_SESSION['id'];

    $sql = "SELECT * FROM resume WHERE user_id = $userId";
```

```
        $stmt = $db->prepare($sql);
        $stmt->execute();
        $resume = $stmt->fetch(\PDO::FETCH_ASSOC);

        require '../vendor/autoload.php';

        $educationPlaces = '<ul class="w-ex">';
            foreach (json_decode($resume['education'], true) as $education) {
                $educationPlaces.= '<li>'.$education['job'].'
'.$education['start_year'].'-'.$education['end_year'].'</li>';
            }
        $educationPlaces.= '</ul>';

        $languages = '<ul class="w-ex">';
        foreach (json_decode($resume['languages'], true) as $lang) {
            $languages.= '<li>'.$lang.'</li>';
        }
        $languages.= '</ul>';

        $skills = '<ul class="w-ex">';
        foreach (json_decode($resume['skills'], true) as $skill) {
            $skills.= '<li>'.$skill.'</li>';
        }
        $skills.= '</ul>';

        $hobbys = '<ul class="w-ex">';
        foreach (json_decode($resume['hobby'], true) as $hobby) {
            $hobbys.= '<li>'.$hobby.'</li>';
        }
        $hobbys.= '</ul>';

        $imageFile = file_get_contents('../img/'.$resume['img']);
        $imageToBase64 = base64_encode($imageFile);

        $html = '<html><head><style>body { font-family: DejaVu Sans
}</style>'.'</head>';
        $resumeRight = '<div class="resume__right">
                    <div class="resume__right-img">
                        <img style="max-width: 400px" src="
'.$imageToBase64.'" alt="">
                    </div>
                    <div class="resume__right-desc">
                        <h3>'.$resume['full_name'].'</h3>
                        <p>'.$resume['age'].' years old</p>
                    </div>
                    <div class="resume__right-text">
                        <p>
```

```
                    '.$resume['subtitle'].'
                </p>
            </div>
            <div class="resume__right-desc">
                <h3>Jobs I worked on</h3>
            </div>
            '.$educationPlaces.'
            <div class="resume__right-desc">
                    <h3>Languages I know</h3>
            </div>
            '.$languages.'
            <div class="resume__right-desc">
                    <h3>Skills</h3>
            </div>
            '.$skills.'
            <div class="resume__right-desc">
                    <h3>Hobbys</h3>
            </div>
            '.$hobbys.'
            <div class="resume__right-desc">
                <h3>Briefly about me</h3>
            </div>
            <div class="resume__right-text">
                <p>
                    '.$resume['about_me'].'
                </p>
            </div>
        </div>';
$resumeLeft = '<div class="resume__left">
                <h3>General information</h3>
                <div class="contact_resume">
                     <p>'.$resume['vaccancy'].'</p>
                </div>
                <div class="contact_resume">
                     <p>'.$resume['email'].'</p>
                </div>
                <div class="contact_resume">
                    <p>'.$resume['phone'].'</p>
                </div>
                <div class="contact_resume">
                    <p>'.$resume['place'].'</p>
                </div>
                <div class="contact_resume">
                    <p>'.$resume['sex'].'</p>
                </div>
                <div class="contact_resume">
```

```
                            <p>'.$resume['years_experience'].' years of
experience</p>
                        </div>
                    </div>';

        $html.= '<div style="clear:both; position:relative;">
    <div style="position:absolute; left:0pt; width:192pt;">
       '.$resumeLeft.'
    </div>
    <div style="margin-left:200pt;">
        '.$resumeRight.'
</div>';
        $html.= '</html>';

        $dompdf = new \Dompdf\Dompdf();
        $dompdf->loadHtml($html);
        $dompdf->render();
        $dompdf->stream("/resume.pdf");

        header('Location: /resume.php');
    }
}
```

index.js

```
if ($('.clock').length) {
    function Timer() {
        let date = new Date;
        $('.clock').html(`${timeNice(date.getHours())}:${timeNice(date.getMinutes()
)}:${timeNice(date.getSeconds())}`);
    }

    setInterval(() => {
        Timer();
    }, 1000);

    function timeNice(num) {
        if (num < 10) {
            return `0${num}`;
        } else {
            return num;
        }
    }
}
```

```
let wrapScreen = $('.wrap-screen');
let i = 0;

function hidenScreen() {
    wrapScreen.each(function () {
        $(this).hide();
    });
}

function showScreen(i) {
    wrapScreen[i].style.display = 'block';
}

$('.btn-next').on('click', function () {
    i++;

    if (i != 0) {
        $('.btn-prev').show();
    }
    if (i == wrapScreen.length - 1) {
        $('.btn-next').hide();
    }
    hidenScreen();
    showScreen(i);
});

$('.btn-prev').on('click', function () {
    i--;

    if (i == 0) {
        $('.btn-prev').hide();
    }
    if (i != wrapScreen.length - 1) {
        $('.btn-next').show();
    }

    hidenScreen();
    showScreen(i);

    $(".send-btn").html('Create !').removeAttr("style");
});

if (wrapScreen.length) {
    hidenScreen();
    showScreen(i);
}
```

```
let usId = 0;
$('.add-education').on('click', function () {
    let newBlock =
        '<div class="us" id="'+usId+'"><' +
        'input type="text" name="job[]" placeholder="Job" required>' +
        '<input type="text" name="start_year[]" placeholder="Start year" required>'
+
        '<input type="text" name="end_year[]" placeholder="End year" required>' +
        '</div>';
    newBlock += '<span id="'+usId+'" class="delete_education">delete
education</span>';
    $('.education-block').append(newBlock);

    usId++;
});

let usIdLanguages = 0;
$('.add-languages').on('click', function () {
    let newBlock =
        '<div class="us-language" id="'+usIdLanguages+'"><' +
        'input type="text" name="language[]" placeholder="Language..." required>' +
        '</div>';
    newBlock += '<span id="'+usIdLanguages+'" class="delete_language">delete
language</span>';
    $('.languages-block').append(newBlock);

    usIdLanguages++;
});

let usIdSkills = 0;
$('.add-skills').on('click', function () {
    let newBlock =
        '<div class="us-skill" id="'+usIdSkills+'"><' +
        'input type="text" name="skill[]" placeholder="Skill..." required>' +
        '</div>';
    newBlock += '<span id="'+usIdSkills+'" class="delete_skill">delete
skill</span>';
    $('.skills-block').append(newBlock);

    usIdSkills++;
});

let usIdHobbys = 0;
$('.add-hobbys').on('click', function () {
    let newBlock =
        '<div class="us-hobby" id="'+usIdHobbys+'"><' +
```

```
        'input type="text" name="hobby[]" placeholder="Hobby..." required>' +
        '</div>';
    newBlock += '<span id="'+usIdHobbys+'" class="delete_hobby">delete
hobby</span>';
    $('.hobbys-block').append(newBlock);

    usIdHobbys++;
});

$(".send-btn").on('click', function (e) {
    e.preventDefault();

    let allRequiredFiledsNotEmpty = true;
    let countEmptyRequiredFields = $('[required]').filter(function() { return
this.value === '' }).length;

    if (countEmptyRequiredFields) {
        allRequiredFiledsNotEmpty = false;
    }

    if (allRequiredFiledsNotEmpty) {
        $('.create-resume').submit();
    } else {
        let text = 'You must fill ' + countEmptyRequiredFields + ' required
fields!';
        $(this).html(text).css({'background' : 'red', 'height' : '50px'});
    }
});
$(document).on('click', '.delete_education', function () {
    let id = $(this).attr('id');
    $(this).remove();
    $('#'+id+'.us').remove();
});

$(document).on('click', '.delete_language', function () {
    let id = $(this).attr('id');
    $(this).remove();
    $('#'+id+'.us-language').remove();
});

$(document).on('click', '.delete_skill', function () {
    let id = $(this).attr('id');
    $(this).remove();
    $('#'+id+'.us-skill').remove();
});

$(document).on('click', '.delete_hobby', function () {
```

```
    let id = $(this).attr('id');
    $(this).remove();
    $('#'+id+'.us-hobby').remove();
});
```