Київський національний торговельно-економічний університет

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЕКТ

на тему:

«Розробка адаптивного landing page кафедри»

Студента 4 курсу, 9 групи,
спеціальності
122 «Комп'ютерні науки» _____ Колесникова Микити Миколайовича
підпис студента

Науковий керівник
Доктор технічних наук, професор _____ Краскевич Валерій Євгенович
підпис керівника

Гарант освітньої програми
кандидат технічних наук, доцент _____ Демідов Павло Георгійович
підпис керівника

Київ 2020

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«20» грудня 2019р.

Завдання
на випускний кваліфікаційний проект студенту

Колесникову Микиті Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема випускного кваліфікаційного проекту
«Розробка адаптивного landing page кафедри»
Затверджена наказом ректора від *«04» грудня 2019 р. № 4111*
2. Строк здачі студентом закінченої роботи *29 травня2020 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: розробка адаптивного landing page.
Об'єкт дослідження: адаптивний landing page.
Предмет дослідження: методи та технології сучасної web-розробки.
4. Перелік графічного матеріалу_____
_____
_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант	Підпис, дата

	(прізвище, ініціали)	Завдання видав	Завдання прийняв
1	Пурський О.І.	15.12.2019 р.	15.12.2019 р.
2	Пурський О.І..	15.12.2019 р.	15.12.2019 р.
3	Пурський О.І.	15.12.2019 р.	15.12.2019 р.

6. Зміст випускного кваліфікаційного проекту (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. Теоретичні аспекти моделювання сучасного landing page

1.1.Теоретичний аналіз поняття landing page

1.2.Поняття сучасності landing page у front-end розробці

1.3. Тенденції веб-дизайну сайтів

1.4. Технології написання програмного коду та верстки сайту

РОЗДІЛ 2. Огляд методик верстки landing page на HTML/CSS (SASS)/ JavaScript

2.1. Метод табличної верстки

2.2. Метод технології табличної верстки HTML/CSS

2.3. Метод технології CSS Float

2.4. Метод технології CSS FlexBox

2.5 Метод технології CSS Grid

2.6 Mobile-first та Desktop-first методи адаптивної верстки

РОЗДІЛ 3. Створення адаптивного landing page

3.1. Планування алгоритму створення сторінки

3.2. Моделювання процесу розробки

3.3. Розробка адаптивного Landing Page

3.4. Налагодження роботи сервера та БД

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

7. Календарний план виконання роботи

№ Пор.	Назва етапів випускного кваліфікаційного проекту	Строк виконання етапів роботи	
		За планом	фактично

1	2	3	4
1	*Вибір теми випускного кваліфікаційного проекту*	*01.10.2019*	*01.10.2019*
2	*Розробка та затвердження завдання на випускний кваліфікаційний проект*	*15.12.2019*	*15.12.2019*
3	*Вступ*	*03.02.2020*	
4	*РОЗДІЛ 1.* *Теоретичні аспекти моделювання сучасного landing page*	*28.02.2020*	
5	*РОЗДІЛ 2.* *Огляд методик розробки landing page*	*06.04.2020*	
6	*РОЗДІЛ 3.* *Створення адаптивного landing page*	*12.05.2020*	
7	*Висновки*	*15.05.2020*	
8	*Здача випускного кваліфікаційного проекту на кафедру науковому керівнику*	*20.05.2020*	
9	*Попередній захист випускного кваліфікаційного проекту*	*03.06.2020*	
11	*Виправлення зауважень, зовнішнє рецензування випускного кваліфікаційного проекту*	*09.06.2020*	
12	*Представлення готового зшитого випускного кваліфікаційного проекту на кафедру*	*12.06.2020*	
13	*Публічний захист випускного кваліфікаційного проекту*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «15» грудня 2019 р.

9. Керівник випускного кваліфікаційного проекту
Краскевич В.Є.
(прізвище, ініціали, підпис)

10. Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

11. Завдання прийняв до виконання студент-дипломник
Колесников М. М.
(прізвище, ініціали, підпис)

12. Відгук керівника випускного кваліфікаційного проекту
_____
_____
_____

____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____

Керівник випускного кваліфікаційного проекту

_________________ ____________.

(підпис, дата)

13. Висновок про випускний кваліфікаційний проект

Випускний кваліфікаційний проект студента ____________________

(прізвище, ініціали)

може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми________________________Демідов П.Г.____

(підпис, прізвище, ініціали)

Завідувач кафедри ______________________________Пурський О.І.____

(підпис, прізвище, ініціали)

«_____»_________________2020 р.

Анотація

У випускному кваліфікаційному проекті здійснено комплексну розробку Landing page кафедри комп'ютерних наук та інформаційних систем київського національного торговельно-економічного університету. Емпіричним шляхом досліджено інструментарій, технології та методології верстки сучасних веб-сайтів з використанням тенденцій веб-дизайну сайтів за останні 5 років. Визначено оптимальні рішення розробки як наслідок порівняльного аналізу характеристик методів та технологій кодування. Створено діаграми прецедентів використання та послідовності процесу розробки. Проаналізовано та розроблено базову серверну логіку для взаємодії із даними користувачів що надходять з клієнтської частини веб-сайту. Імплементовано роботу бази даних.

Ключові слова: landing page, адаптивність, клієнтська частина, серверна частина.

Annotation

In the final qualification project, a comprehensive development of the Landing page of the Department of Computer Science and Information Systems of the Kyiv National University of Trade and Economics was carried out. The layout tools, technologies and methodologies of modern websites with the use of web design trends for the last 5 years have been empirically studied. The optimal development solutions as a consequence of the comparative analysis of the coding methods characteristics and technologies are determined. Use case and sequences diagrams of development process are created. The basic server logic for interaction with user data coming from the client part of the website is analyzed and developed. The work of the database is implemented.

Keywords: landing page, adaptability, client-side, server-side.

ЗМІСТ

ВСТУП

Актуальність проекту полягає в тому, що робота з абітурієнтами та їх батьками має всі ознаки ринку – кафедрам, що приймають майбутніх студентів, для успіху на ринку необхідно мати відповідний рекламний матеріал. Враховуючи поширеність Інтернету, особливо серед молоді, створення Landing page буде мати позитивний вплив на зацікавленість абітурієнтів у певній кафедрі.

Розвиток глобальної інформаційної інфраструктури спричинив значне зростання користувачів мережі Інтернет, станом на 2019 рік близько 71% людей в Україні є активними користувачами мережі [1]. Інтернет-маркетинг на сьогоднішній день надзвичайно поширений у світі.

Мета роботи полягає у розробці адаптивного landing page.

Завдання, що забезпечать реалізацію мети:

1. Аналіз сучасних моделей landing page;
2. Огляд інструментів розробки web-сайтів;
3. Визначення тенденцій дизайну для верстки;
4. Планування структури HTML-сторінки та SASS (CSS) коду шаблонних елементів;
5. Розробка HTML-сторінки з паралельною розробкою CSS стилів сторінки;
6. Створення, налагодження та під’єднання хмарної бази даних Mongoose для зберігання заповненої користувачем контактної інформації.

Об’єкт дослідження – адаптивний landing page.

Предмет дослідження – методи та технології сучасної web-розробки.

Методи дослідження ґрунтуються на принципах інформаційного моделювання предметної області, на методах порівняльного аналізу а також на загальнонауковому аналітичному методі.

Практична значущість роботи полягає в отриманні навичок верстки, налагодженні серверної частини web-сайту а також взаємодії клієнтської частини з серверною частиною та базою даних

РОЗДІЛ 1.

ТЕОРЕТИЧНІ АСПЕКТИ МОДЕЛЮВАННЯ СУЧАСНОГО LANDING PAGE

1.1. Теоретичний аналіз поняття landing page

Landing page - це «легкий» (мається на увазі невеликий за обсягом), як правило односторінковий сайт, що використовується для залучення зацікавленого сегменту аудиторії до цільових товарів, послуг або акцій. Як правило, на цільову сторінку потрапляють завдяки переходу з контекстної реклами або запитах у пошукових системах.

На подібних SPA (англ. Single-page application – односторінковий додаток) розташована необхідна для відвідувача інформація в такій формі, щоб максимально фокусуватися на ній.

Хороша цільова сторінка зосереджена на певному потоці трафіку - скажімо, з електронної пошти, що, наприклад, рекламує електронну книгу. Оскільки цільова сторінка орієнтована лише на людей які (ймовірно) зацікавлені в цій книзі, і оскільки ця електронна книга має ексклюзивну інформацію, що детально розглядає хвилюючу аудиторію тему, є можливість конвертувати більший відсоток відвідувачів веб-сайту у потенційних клієнтів. Спостерігається майже ідентична ситуація із кафедрою вищого навчального закладу: дана цільова сторінка орієнтована на майбутніх студентів університету, її задача полягає у вступі (конвертації) більшого відсотку абітурієнтів до певної кафедри.

Правильний Landing page спрямований на стимулювання бажання гостя лендінгу зробити корисну дію: реєстрація на сайті, оформлення замовлення, дзвінок з офісу компанії, підписка на сервіси і тому подібне. Як правило веб-сторінки, в абстрактному понятті, схожі на швейцарські ножі – мають багато функціональних можливостей та цілей, проте на відміну від них, цільові сторінки (landing page) розроблені з єдиною ціллю, відомою як заклик до дії (call to action, скорочено CTA). Саме ця спрямованість робить цільові сторінки одним з

найкращих варіантів для збільшення коефіцієнтів конверсії маркетингових кампаній.

Завдяки правильній спрямованості, Landing page може забезпечити підвищення конверсії до 300% [2]. Як правило, цільові сторінки – це невеличкий, симпатичний і в міру лаконічний дизайн. Основний акцент такий: на сторінці не має бути факторів, що можуть відволікти від її змісту.

Головна ціль лендінгу – конвертувати якомога більше потенційних клієнтів, що зацікавлені в пропозиції та бажають більше дізнатися про товар, послугу або акції.

1.2.Поняття сучасності landing page у front-end розробці

Сучасність – розтяжне поняття, у веб-розробці одними з найголовніших факторів оцінки роботи як «зробленої» є здатність виконувати свої основні функції, а також наявність привабливого зовнішнього вигляду клієнтської частини. Це одна з основних причин, чому навіть через одинадцять років після винаходження технології CSS-Flexbox досі існують сайти що створені на застарілій та не зручній у використанні технології CSS-Float [3] – воно працює.

Сучасність веб-сайтів, до яких також відноситься й Landing Page, включає в себе такі основні пункти:

- Привабливий дизайн, за тенденціями на поточний момент часу;
- Оптимізація клієнтської частини сайту під більшість існуючих розширень екранів у світі;
- Використання технологій та методів, що підтримуються більшістю браузерів та значно полегшують як процес розробки сайту, так і процес його оптимізації;
- Наявність анімацій та плавних переходів при взаємодії з інтерактивними елементами сторінки.

Шаблонів красивого дизайну в Інтернеті є надзвичайно багато, ключовими характеристиками краси сайту є сумісні кольори, шрифт, привабливі для очей картинки, фото, графічні елементи що вписуються у загальну концепцію дизайну,

тощо.

У світі поширені найрізноманітніші розширення екранів пристроїв, з яких користуються мережею Інтернет. Найрозповсюдженішими сегментами користувачів Інтернету в Україні є користувачі персонального комп'ютера та мобільних пристроїв [4]. Отже, необхідно оптимізувати цільову сторінку так, щоб з нею можна було взаємодіяти і через ПК, і через планшети і особливо через смартфони, оскільки більша кількість абітурієнтів в основному користується саме цими пристроями.

На момент 2020 року створена велетенська кількість найрізноманітніших методологій верстки, фреймворків та бібліотек для взаємодії із HTML-сторінками. Для створення односторінкового Landing Page, доцільно буде використовувати методи верстки Flexbox, як зручний та швидкий спосіб розташування елементів на сторінці, та препроцесора SASS, що забезпечить дотримання ієрархічної структури у CSS-коді і, як наслідок, приведе до більш впорядкованої й читабельної побудови коду, що також є вагомим елементом зручності та ефективності при розробці.

1.3.Тенденції веб-дизайну сайтів

Дизайнер малює кнопки, банери та інші графічні елементи – прототип сайту отримує естетичний зовнішній вигляд. Варто відзначити, що дизайнер працює не над кожною сторінкою, а над шаблонами кількох основних, застосовуючи тенденції веб-дизайну. Для роботи з макетами сайту буде використовуватися програмний засіб Photoshop, у якому є всі необхідні інструменти для створення дизайну.

При розробленні моделі сайту, доцільно керуватися базовими тенденціями веб-дизайну сайтів на поточний 2020 рік:

- Велика кількість білого простору;
- Чуйні (гумові) логотипи;
- Використання градієнту;
- Шрифти без зарубок;

- Напівпрозорі кнопки;
- Плавна анімація взаємодії з інтерактивними елементами сайту;
- Висока якість графічних елементів сайту.

Тенденція на велику кількість білого простору з'явилася нещодавно – близько трьох-чотирьох років тому, однак стала фундаментальною в веб-дизайні завдяки своїй здатності концентрувати увагу на головному реченні. Білий простір робить екран візуально більшим, не створює зайвих відволікаючих факторів для користувача. Білий колір комбінується з усіма іншими, через це в якості акцентів або дизайнерських ідей є свобода вибору абсолютно яких завгодно відтінків – майже у будь-якому випадку, вкупі це буде виглядати симпатично.

Адаптивні логотипи в залежності від розмірів екрану пристрою можуть автоматично адаптуватися й залишатися ключовою відмінністю сайту або бренду. Це є хорошою практикою для SEO-оптимізації (оптимізації сайту для поліпшення своїх позицій у запитах пошукових систем).

Використання градієнтного переходу кольорів – також є оптимальним варіантом фону сторінки з точки зору колористики та відчуття наповненості сайту. Нескладний інструмент, використання якого придає оригінальності та свіжості.

Анімації на веб-сайті не є необхідністю, проте вважається кращою практикою, оскільки такі елементи вносять значний вклад у сприйняття сучасності та технологічності сайту. Як правило, майже на будь-якому веб-ресурсі можна побачити плавні зміни стилів елементу при синергії з ним: це може бути елемент навігаційного меню із зміною кольору або розмірів при наведенні, гарна анімація при натисканні на чекбокс, поступова зміна позиціонування тексту-підказки поля вводу при введенні у нього інформації – будь-яка помітна зміна, що позначить інтерактивність елементу. Дане рішення має надзвичайно позитивний вплив на візуальне задоволення від відвідування веб-сайту.

Загалом веб-дизайн в останні роки стрімко крокує в мінімалізм, тому непоганим рішенням також стане використання напівпрозорих кнопок. Такі кнопки

виглядають стильно, вони вписуються у концепцію майже будь-якого сайту через свою мінімалістичність та універсальність [5].

Шрифти мають важливий вплив на сприйняття інформації, і тут ситуація дуже схожа на ту що зараз притаманно веб-дизайну в цілому – переважає стиль мінімалізму, літери та цифри без зарубок. Ідеальним прикладом є стандартний шрифт що використовує компанія Apple – Helvetica Neue, або поширений на просторах Інтернету Manrope.

На даний момент часу, у світі існують типи екранів із різною щільністю пікселів, зокрема відрізняється Retina компанії Apple, що наявний у всіх сучасних пристроях даної корпорації. Такі екрани мають підвищену щільність розташування пікселів, що має вплив на сферу веб-розробки. Для пояснення проблеми розмитості необхідно розібратися у термінології принципу.

Фізичні пікселі (Device pixel або Physical pixel) – звичні для широкого розуміння пікселі: найменші елементи будь-якого екрану, кожен з яких має свій колір і яскравість. Щільність екрану (Screen density) – це кількість фізичних пікселів дисплея. Зазвичай вимірюється в пікселях-на-дюйм (PPI: pixels per inch). Apple, розробивши Retina-екрани з подвійною щільністю пікселів, стверджує, що людське око не здатне розрізнити більшу щільність. CSS-пікселі (CSS pixels) - абстрактна величина, яка використовується браузерами для точного відображення контенту на сторінках, незалежно від екрану (DIPs: device-independent pixels) [11]. Наприклад:

 </ div>

Такий блок на звичайних екранах буде займати область 200x300 пікселів, а на Retina-екранах той же блок отримає 400x600 пікселей. Таким чином, на Retina-екранах щільність пікселів в 4 рази більше, ніж на звичайних (Рис. 1.1):

Рисунок 1.1. Щільність пікселів на різних екранах [11]

Растрові пікселі (Bitmap pixels) – найменші частини, складові растрового зображення (.PNG, .JPG, .GIF та інші) Кожен піксель містить інформацію, про колір і розташуванні в системі координат зображення. У деяких форматах піксель може містити додаткову інформацію, наприклад, прозорість.

Крім растрового розширення, зображення в інтернеті мають абстрактні розміри в CSS-пікселях. Браузер стискає або розтягує графічний елемент відповідно до його властивостей CSS-ширини та довжини. При відображенні на звичайному екрані, один растровий піксель відповідає одному CSS-пікселю [11].

На Retina-екранах кожен растровий піксель множиться в 4 рази, що і створює проблему розмитих зображень на сайті (Рис 1.2), що створює потребу високої якості для растрових зображень, таких як фото.

Рисунок 1.2. Відображення графічних елементів на веб-сторінці[11]

1.4. Технології написання програмного коду та верстки сайту

Верстка передбачає собою використання певного редактору коду для зручної, ефективної та швидкої роботи. На даний момент розвитку IT-індустрії

існує багато сотень редакторів коду, кожний з яких краще взаємодіє з певною мовою програмування і заточений під різні цілі. Для створення Landing page не потрібне якесь спеціальне середовище розробки, буде достатньо будь-якого інструменту, тож робочим середовищем буде універсальний Visual Studio Code із встановленими сніпетами (анлг. snippets), що дають можливість використовувати скорочення основних тегів HTML та властивостей CSS для примноження швидкості написання коду.

HTML (HyperText Markup Language), є найпоширенішою мовою розмітки веб-сторінок у світі, CSS (Cascading Style Sheet) – це мова стилів, що використовується для надання елементам HTML сторінки візуальних змін, таких як колір, позиціонування на сторінці, межі, відступи, форми та розмір об’єктів, анімації і так далі. Препроцесор SASS – це метамова на основі CSS, призначена для збільшення рівня абстракції CSS коду та спрощення файлів каскадних таблиць стилів. Ці технології і використовуються у верстці даного Landing page.

Існує 2 основних та принципово різних методи верстки сайтів:

- Desktop-first
- Mobile-first

Desktop-first метод передбачає собою в першу чергу розробку веб-сайту для екранів великого розміру таких як у персональних комп’ютерів, а потім для менших екранів мобільних пристроїв.

Mobile-first навпаки – спочатку йде праця над найменшими розширеннями екранів, і вже потім, поступово, для екранів ширше й ширше, аж поки не буде досягнуто максимального брейкпоінта (Breakpoint) – максимального розміру екрану що розповсюджений серед населення (як правило, це ширина екрану в 1920 пікселів).

Різниця версій сайту полягає кількості функціоналу та креативних елементів – у мобільній версії сайту їх завжди буде менше. За суб’єктивною точкою зору автора, з методу Desktop-first легше поступово переходити до розробки адаптивної та гумової версії сайту для мобільних пристроїв.

Кращою практикою є спочатку кодувати загальні стилі (кольори, шрифти і тому подібне), і тільки потім для кожного з діапазона розширень створювати свій власний стиль-коректор, котрий описує (коригує) позиціонування елементів, їх розміри і т.д. Цей метод є комбінацією двох методів, проте більше підходить під опис саме Desktop-first методу.

У ході роботи гарантовано будуть траплятися помилки, як незначні так і більш помітні, через емпіричний характер процесу веб-розробки. Це означає, що для швидкого та ефективного виявлення кореня проблем є необхідність у зручному інструменті для тестування та взаємодії із сторінкою вживу. Стандартним та найбільш доцільним рішенням є технологія, що присутня на даний момент у кожному браузері, проте найліпше реалізована у браузері Google Chrome – ChromeDevTools. Це набір інструментів веб-розробника, вбудованих безпосередньо в браузер. DevTools може допомогти швидко редагувати сторінку та діагностувати проблеми, що в кінцевому підсумку допомагає швидше створювати веб-додаток [22]. Даний інструмент розробки дозволяє прямо в браузері на сторінці проводити різного роду тестування, вимірювання та аналіз без впливу на локальні файли ресурсу, що є оптимальним для даної роботи.

РОЗДІЛ 2.

ОГЛЯД МЕТОДИК РОЗРОБКИ LANDING PAGE

2.1. Метод табличної верстки

За такого методу верстки, всі елементи сайту розташовуються у таблицях та їх комірках. Створюється файл-шаблон з розміткою і використовується як каркас для всіх інших сторінок. Фактично від файлу до файлу змінюється лише основний контент. Шапка сайту, його підвал і меню беруться з уже готового шаблону і зазвичай залишаються незмінними. Суть цього методу у тому, що весь сайт являє собою одну велику таблицю. Тобто весь вміст має бути укладений в один парний тег - table. В одну таблицю можна вкласти необмежену кількість інших елементів, в тому числі і інших таблиць.

Основні теги:

- Тег <table>, декларує таблицю;
- Тег <th>, заголовок таблиці;
- Тег <tr>, рядок таблиці;
- Тег <td>, одна комірка таблиці;
- Тег <col>, одна або декілька колонок;

Використання таблиць з невидимим кордоном - відомий спосіб верстки, що застосовувався на старих сайтах. Така таблиця фактично являє собою модульну сітку, в якій можна розміщувати окремі елементи веб-сторінки. Завдяки універсальності таблиць і великій кількості параметрів, які керують їх видом, таблиці надовго стали стандартом для верстки веб-сторінок. Даний метод є застарілим оскільки це один з перших стандартів верстки, і йому на зміну прийшли більш зручні та швидкісні методи [6].

Переваги табличної верстки**:**

- Основне призначення– створення таблиць даних;
- Підтримка абсолютно усіма браузерами;

Недоліки табличної верстки:

- Якщо використовувати як метод верстки усього сайту – він буде важким: сторінки багато важитимуть, повільно завантажуватися;
- Наявність надлишкового коду, структура виглядає дуже громіздкою;
- Як наслідок попереднього пункту, надзвичайно важко оптимізувати сторінку під усі розширення екранів.

Отже, табличну верстку доцільно використовувати за необхідністю зробити на сайті таблицю, проте як метод верстки він застарілий та менш зручний ніж інші існуючі на сьогоднішній день методи.

2.2. Метод технології CSS Float

Float – це властивість CSS-позиціонування, що робить елементи веб-сторінки «плаваючими». У своєму найпростішому використанні властивість Float представляє собою властивість обертання тексту навколо зображень, від назви технології, Float – плавати (Рис. 2.1).

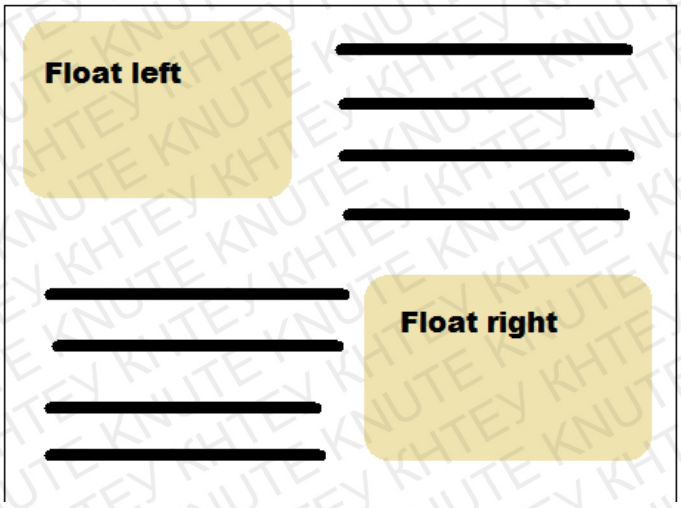

Рис. 2.1. Приклад використання CSS Float

Встановлення float на елемент за допомогою CSS відбувається так:

```
#sidebar {
  float: right;
}
```

Де #sidebar – це звернення у стилях CSS до ідентифікатору (id) елемента HTML

сторінки, а «float: right;» це встановлення властивості елемента із значенням «right».

Для властивості Float є чотири можливих значення:

- Left, позиціонування елемента вліво;
- Right, що розташовує блок вправо;
- None (за замовчуванням), елемент не буде мати властивості Float;
- Inherit, наслідує значення float для даного елемента від батьківського [7].

Крім простого прикладу обгортання тексту навколо зображень, Float можна використовувати для створення цілих макетів веб-сторінок (Рис. 2.2).

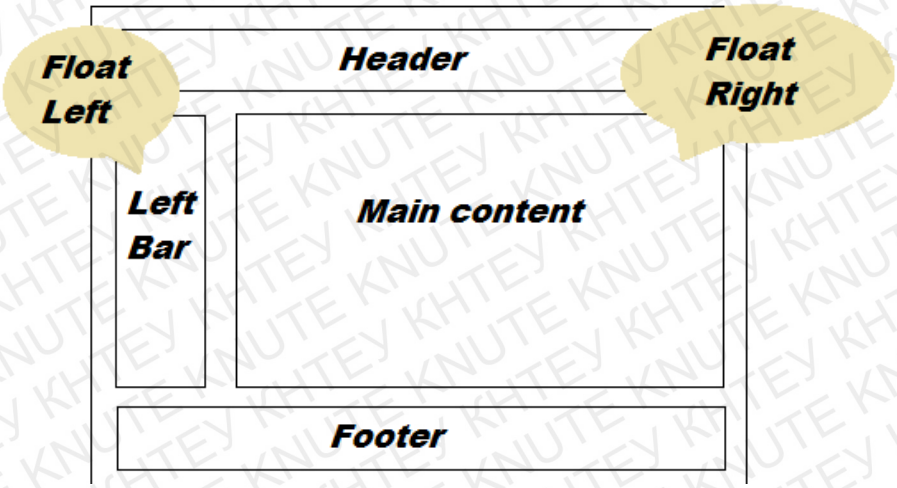

Рис. 2.2. Приклад верстки цілої сторінки за методом CSS-Float

Переваги методу верстки CSS Float:

- Підтримка усіма браузерами;
- Основна функція – загорнути текст навколо якогось блоку;
- Можна позиціонувати блоки на сторінці;
- Не громіздка архітектура коду;

Недоліки методу верстки CSS Float:

- При використанні цього методу верстки сторінки можуть виникати проблеми з позиціонуванням блоків: елементи будуть «засмоктуватися» у певну частину сторінки та накладатися один на одного;
- Потребує наявність у коді так званого ClearFix-у, що буде вирішувати основну

проблему, проте додасть громіздкості коду;

- Метод не надто зручний через вірогідність проблем з позиціонуванням, проте доволі швидкий у використанні.

Підсумовуючи, CSS Float все ще можна використовувати для верстки, проте на момент 2020 року інснує набагато ефективніші та зручніші технології для верстки веб-сторінок. У роботі буде використовуватися за необхідності зробити «плаваючий» текст.

2.3. Метод технології CSS FlexBox

CSS Flexible Box Layout, відомий як FlexBox, це властивість стилів елементів, що дозволяє майже як завгодно маніпулювати розташуванням груп елементів відносно один одного, що й виходить з назви (з англ. Flexible – гнучкий). На момент написання даної роботи, це все ще є найпопулярнішою та однією з найзручніших методологій не тільки позиціонування певних елементів сторінки, але й веб-сайту в цілому. Зустрічається практично на будь-якому сайті.

Модуль гнучких блоків, який зазвичай називають флексбоком (FlexBox), був розроблений як модель одновимірного компонування і як метод, який міг запропонувати розподіл простору між елементами в інтерфейсі та потужні можливості вирівнювання. Flexbox одновимірний, мається на увазі той факт, що flexbox має справу з компонуванням в одному вимірі за часом – або як рядок, або як стовпець. Це може бути протиставлено двовимірній моделі CSS Grid Layout, яка одночасно керує стовпцями та рядками разом.

Основна ідея Flexbox полягає наданні контейнеру можливості змінювати ширину, висоту і порядок елементів, щоб найкраще заповнити доступний простір (переважно для їх розміщення на всіх типах і розмірах екранів). Технологія розширює елементи, щоб заповнити вільний простір або зменшує їх, щоб уникнути переповнення.

Flexbox працює з двома осями – з головною віссю та поперечною. Головна

вісь визначається властивістю flex-direction (з англ. напрям), а поперечна вісь проходить перпендикулярно до неї. Всі маніпуляції з флексбоком відносяться до цих осей.

Row та row-reverse використовується для позиціонування елементів по горизонталі, тобто робить головною вісь абсцис. Column та column-reverse за основну вісь вважає ординату, тобто розташування у колонку, по вертикалі. Додаткове слово у цих властивостях “reverse” змінює напрям розташування елементів за віссю. Існує також нюанс із арабськими країнами, в яких у технології Flexbox за замовчуванням увесь текст та контент розташовується навпаки – справа наліво (right to left, скорочено rtl), на відміну від стандартного розташування зліва направо (left to right, ltr) [8].

Отже, головна вісь flex-direction може приймати одне з чотирьох значень (Рис. 2.3):

- row (за замовчуванням): зліва направо в ltr; справа наліво в rtl;
- row-reverse: справа наліво в ltr; зліва направо в rtl;
- column: те саме, що row, але зверху вниз;
- column-reverse: те саме, що і зворотний рядок, але знизу вгору.

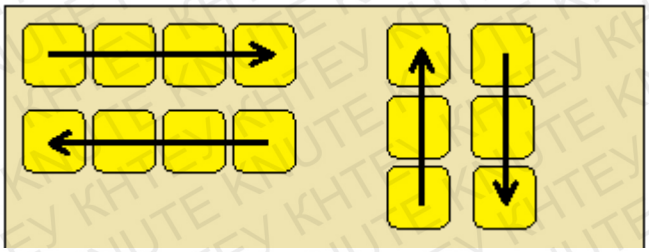

Рис 2.3. Напрямки Flexbox

Властивість justify-content та align-items призначені для розподілу залишку вільного простору між елементами. Дані властивості також здійснюють певний контроль над вирівнюванням елементів, коли вони переповнюють рядок або стовпець (Рис. 2.4).

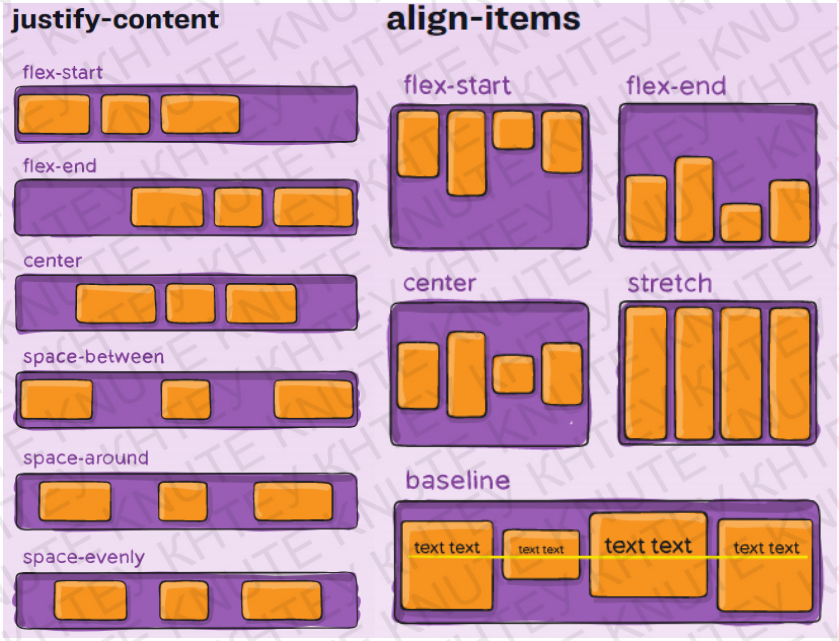

Рис 2.4. Основні властивості позиціонування Flexbox [9]

Методологія нараховує ще 6 потужних функціональних можливостей, проте перераховані даному у підрозділі є фундаментальними для зручної розробки веб-сайту.

Переваги Flexbox:

- Гнучкість, дозволяє майже як завгодно позиціонувати групи елементів;
- Легко адаптувати під різні розміри екранів через вбудовану гумовість (чутливість) та широкий спектр властивостей;
- Візуально зрозуміла взаємодія з HTML-сторінкою;
- Швидкий, ефективний та зручний інструмент;
- Підтримка усіма сучасними версіями браузерів (окрім старих версій Internet Explorer та Edge);
- Flexbox полегшує проектування адаптивної та чутливої структури компонування без використання Float або Positions.

Недоліки Flexbox:

- Потреба у вивченні великої кількості властивостей;
- У великих проектах (понад 2 тисяч рядків коду) код стилів може набувати громіздкості (один тільки Flexbox займає до 10% усього коду);

Через свою зручність, сучасність а також адаптованість під широкий спектр

браузерів, даний метод верстки веб-сайтів, у комбінації з іншими розглянутими технологіями, є оптимальним рішенням.

2.4 Метод технології CSS Grid

CSS Grid – найпотужніша система компонування на момент 2020 року, доступна в CSS. Це двовимірна система, вона може обробляти як стовпці, так і рядки, на відміну від технології Flexbox який є значною мірою одновимірною системою.

CSS Grid layout відмінно підходить для того, щоб розділити сторінку на основні області або визначити взаємозв'язок розміру, позиціонування і рівня між частинами контенту, що складається з HTML елементів.

Як і таблиці (<table>), Grid layout дозволяє вирівнювати елементи в стовпці і рядки. Проте, за допомогою модульної сітки CSS Grid працювати з елементами набагато простіше, ніж з таблицями. Наприклад, дочірні елементи grid-контейнера можуть нашаровуватися один на одного як і інші елементи за допомогою CSS [10].

CSS Grid налічує надзвичайно багато вбудованих властивостей, проте фундаментальними, на яких базується уся робота, є такі:

- display: grid (використовується для декларування відображення модульної сітки);
- grid-template-columns та grid-template-rows (визначає стовпці та рядки сітки із розділеним простором списком значень. Значення представляють собою розмір, а простір між ними представляє лінію сітки.);
- grid-template-areas (визначає шаблон сітки шляхом посилання на імена областей цієї сітки, які вказані властивістю grid-area. Повторювання певного імені області сітки змушує вміст охоплювати ці комірки. Сам синтаксис забезпечує візуалізацію структури сітки);

Grid Template Layout дозволяє адаптивно та чутливо верстати без допомоги спеціальних властивостей CSS media queries (Рис 2.5), використовуючи усього три рядки:

```
.grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
  grid-gap: 1rem;
}
```

Рисунок 2.5. Приклади використання CSS Grid при розмірах ширини екрану 360 (смартфон) та 720 (планшет)

Переваги CSS Grid:

- Один з найефективніших інструментів розробки при використанні Best Practices у верстці;
- Зручний та семантично зрозумілий у використанні;
- Відсутність необхідності змінювати та адаптувати код під кожний можливий розмір екрану, через вбудовану в функціонал технології чутливість;

Недоліки CSS Grid:

- Головним недоліком є відсутність підтримки деякими браузерами, проте з часом цей недолік зникне;
- За об’ємом інформації для вивчення не поступається великій бібліотеці у програмуванні (налічує понад 28 унікальних CSS властивостей для батьківських та дочірніх елементів).

Резюмуючи властивості даної технології, її доцільно використовувати на великих, насичених деталями сайтах у якості каркасу сторінки, на якій буде

змінюватися тільки основний контент, в залежності від поточної сторінки сайту, що є дуже схожим на принципи роботи сучасних веб-фреймворків таких як ReactJs, Angular, JQuery та інших. Для менших веб-ресурсів, таких як Landing page, більш релевантним буде використання Flexbox, однак CSS Grid Layout надзвичайно зручний у використанні інструмент для розробки карточок.

2.5 Mobile-first та Desktop-first методи адаптивної верстки

Mobile-first – це, як вже було сказано вище у пункті 1.4 даного проекту, метод адаптивної верстки що передбачає собою розробку сайту, у першу чергу, для коректного відображення на екранах мобільних пристроїв таких як, наприклад, сучасні смартфони та планшети. За такого методу увесь контент на сторінках має дещо інші UX (User Experience – досвід користувача, зручність у використанні) та UI (User Interface – інтерфейс користувача, зовнішній вигляд та візуальна зрозумілість функціональних можливостей певних елементів сайту), зокрема позиціонування об’єктів, їх розміри, а також окремі частини дизайну, що будуть перероблені у більш компактні габарити.

Головною ідеєю є те, що спочатку йде розробка під мобільні гаджети, і тільки після цього для розмірів екранів все ширше й ширше, аж до розширення моніторів персональних комп’ютерів. Загалом, цей метод розробки є зручним у ситуації, коли є в наявності макет дизайну як для смартфонів, так і для ПК, дана методика не підійде для імпровізованої верстки через свою невизначеність у точній розмітці.

Основними принципами та засобами для роботи з адаптуванням веб-сайту до мобільних версій є:

- Відносні одиниці вимірювання – проценти від розміру екрану (%), процентна ширина та висота вікна (vh, vw), комбінація та відносного до поточного розміру шрифтів (px, em);
- Відсутність горизонтальної прокрутки сторінки та розташування контенту у межах вікна екрану, що забезпечить комфортне споживання інформації;
- Налагодженість відображення графічних елементів, або «ретинізація»;

- Наявність компактного випадаючого меню, що у закритому стані майже не займає простір сторінки.

Відносні одиниці вимірювання надають сайтові можливість самостійно пристосовувати розміри об’єктів до розширення екрану (блоки контенту, графічні елементи, текст та інші), що надзвичайно позитивно впливає на адаптованість сайту у цілому. У випадках невеликих, за обсягами контенту, сайтів, дотримання даного принципу є ключовим, що може відбивати необхідність у використанні більш різноманітних варіантів позиціонування елементів на різних розмірах екранів пристроїв.

Відсутність горизонтальної прокрутки, або скролбара (англ. Scroll Bar), також є правилом грамотної верстки. Дизайн потребує такої реалізації, щоб не відчувалася незручність у його сприйманні. Горизонтальна прокрутка сайту, особливо на крихітних екранах смартфонів, є вкрай незадовільним рішенням з точки зору UX, оскільки дана ситуація змушує користувача прикладати зусилля для сприйняття інформації яка, у таких випадках, погано структурована та потребує гортання сторінки від одного боку до іншого. Мобільна версія, як правило, передбачає собою позиціонування контенту у стовпчик, на відміну від ПК-версії. У комбінації з використанням відносних величин, із даним принципом майже не буде виникати проблем із зайвим гортанням сторінки через свою вбудовану чутливість.

Існує кілька способів оптимізації графіки для відображення на Retina-екранах. Кожен має і плюси, і мінуси, необхідно в кожному конкретному випадку обрати, що має більший пріоритет: продуктивність, простота реалізації, підтримка браузерами або якісь інші параметри:

- HTML і CSS-масштабування;
- Визначення щільності пікселів екрану;
- Масштабна векторна графіка;

Найпростіший спосіб підготувати графіку до Retina-дисплея – розділення навпіл фізичних розмірів зображення. Наприклад, щоб показати фотографію

200x300 пікселів на екрані зі збільшеною щільністю пікселів, необхідно завантажити фото розміром 400x600 пікселів і зменшити його, використовуючи CSS-селектори або HTML-атрибути. Таким буде відображення на звичайному екрані (Рис 2.6):

Рисунок 2.6. Схематичне відображення фото на звичайному екрані[11]

Й таким на Retina (Рис 2.7):

Рисунок 2.7. Схематичне відображення фото на Retina-екрані[11]

Найпростішим способом оптимізації є надання параметрів width та height тегу img або відповідному селектору у CSS. Такий спосіб зручно використовувати на невеликих, односторінкових сайтах.

Такого ж результату можна досягти, використовуючи Javascript для того, щоб ділити навпіл розміри зображень для Retina-екранів.

Плюси HTML і CSS-масштабування:

- Простота реалізації;
- Кроссбраузерність.

Мінуси HTML і CSS-масштабування:

- Пристрої зі звичайними екранами будуть завантажувати зайві мегабайти;
- На звичайних екранах чіткість зображень може постраждати через алгоритми стискання.

Визначення щільності пікселів екрану передбачає собою використання @media-запитів у CSS або javascript-реалізацію

Плюси визначення щільності пікселів:

- Простота реалізації;
- Кроссбраузерність.

Мінуси визначення щільності пікселів:

- Пристрої зі звичайними екранами будуть завантажувати зайві мегабайти;
- На звичайних екранах чіткість зображень може постраждати через алгоритми стискання.

Незалежно від використовуваного методу, растрові зображення за своєю природою залишаються обмеженими в масштабуванні. Векторна графіка не має такого недоліку.

SVG (Scalable Vector Graphics) – формат на основі XML, що застосовує векторну графіку (використовується для відображення тексту), такий елемент не може розмитим при масштабуванні, оскільки подібні зображення автоматично «перемальовуються» під будь-який розмір екранів. Підтримується більшістю браузерів. Найпростіший спосіб використання SVG-зображень – імплементувати його в HTML тег img або CSS-параметрами background-image і content: url (). Просте SVG-зображення може бути як завгодно масштабоване.

Переваги SVG-зображень:

- Незмінюване зображення для всіх пристроїв;
- Простота реалізації;
- Нескінченне масштабування.

Недоліки SVG-зображень:

- Не підходить для складної графіки через великі розміри файлу;
- Не найпопулярніший формат зображень, через що є необхідність конвертувати потрібний графічний елемент у даний формат.

Desktop-first передбачає собою розробку гумового сайту у першу чергу для моніторів персональних комп’ютерів і згодом, поступово, для більш менших дисплеїв аж найменших екранів смартфонів.

Загалом, керуючись даним методом також доцільно буде застосовувати принципи адаптивної розробки: відносні одиниці вимірювання, запобігання горизонтального скролу, ретинізація. Головною відмінністю від Mobile-first методу є творча свобода, оскільки за відсутності макету сайту, створювати його імпровізуючи значно легше, через велику кількість сайтів-прикладів, і, як наслідок, творчі процеси адаптації та видозмінення певних елементів сторінки під різні дисплеї також полегшуються – зменшувати у розмірах та позиціонувати габаритний контент зрозуміліше для сприйняття ніж навпаки, йти від малого до більшого.

Також за такого метода зручніше розробляти та перевіряти роботу анімацій інтерактивних елементів веб-ресурсу, оскільки у мобільній версії даний функціонал загалом менш помітний, як через розміри екранів, так і через особливості взаємодії із подібними елементами, наприклад як наслідок того що людський палець застосовується безпосередньо на екрані мобільного пристрою, частина візуальних ефектів, що по стандартам триває від 0.2 до 0.5 секунд, не помічається під пальцем що натискає на елемент синергії.

Можна зробити висновок, що при безальтернативному виборі серед двох протилежних методів розробки сайту, у Desktop-first спостерігається більша кількість переваг, оскільки основні принципи сучасних Best Practices застосовуються в обох методах, і через відсутність професіонально сконструйованого макету сайту більш доцільним та зручним є саме розробка в першу чергу для персональних комп’ютерів. Таке рішення значно збереже час на планування та процес роботи.

2.6. Методи взаємодії серверної та клієнтської частин веб-додатку

Серверна частина сайту – це всі технології, необхідні для обробки вхідного запиту, генерації та відправлення відповіді клієнту. У випадку даної роботи, завданням серверної частини landing page є валідація даних форми та їх подальший запис до бази даних. Зазвичай включає три основні частини:

- Сервер – комп'ютер, який приймає запити;
- Додаток – програма, запущена на сервері, яка слухає запити, отримує інформацію з бази даних та надсилає відповідь;
- База даних – використовуються для організації та збереження даних [23].

У веб-розробці «клієнтська сторона» позначає все у веб-додатку, що має візуальне відображається на клієнті – на пристрої кінцевого користувача. Сюди входить те, що бачить користувач, наприклад текст, зображення а також будь-які дії, які програма виконує в браузері користувача [24].

Форма HTML, що знаходиться на клієнтській частині, використовується для збору даних користувача. Потім введена інформація користувача може бути відправлена на сервер для обробки. Тег <form> має свої особливості. По суті це звичайний блок , проте так само як і функції в програмуванні, він приймає в себе аргументи – а саме дані з полів введення таких як <input> різних типів (checkbox, text, radio, тощо), <textarea>, <button> та інших.

Атрибутами тегу форми що дозволяють взаємодіяти з сервером є action (з англ. «дія») та method (з англ. «метод»). Action має містити в собі ім’я події – так званого роута (англ. «route», шлях) – що буде міткою для оброблення даних форми на сервері сайту. Method – це метод HTTP-запиту (HTTP, HyperText Transfer Protocol). HTTP означає протокол передачі HyperText. Даний протокол є базовим, й використовується всесвітньою павутиною, цей протокол визначає, як повідомлення форматується та передається, а також які дії веб-сервери та браузери повинні вживати у відповідь на різні команди [25]. В основному використовується 4 типи запитів:

- GET – для «віддавання» на веб-сторінку відповіді з серверу, як правило саму

сторінку;

- POST – для фіксування змін або запису інформації з клієнтської частини на серверну;
- PUT – замінює всі поточні дані цільового ресурсу завантаженим вмістом;
- DELETE – для видалення даних.

Форма, як правило, використовує саме метод POST [26].

Валідація – це автоматична перевірка, щоб переконатися, що введені дані не є неправильними. Однак слід розуміти що така перевірка не може забезпечити точність даних. Для визначення правил заповнення форми, необхідним є використання спеціальних HTML атрибутів. А саме:

- Minlength та maxlength – відповідно мінімальна та максимальна довжина рядку тексту;
- Required – визначає необхідність заповнення поля;
- Type – в деяких випадках, як наприклад з електронною поштою, автоматично визначає правила коректної форми запису в полі введення, використовуючи регулярні вирази;
- Template – визначає шаблон тектсу, по якому має бути заповнене поле. Визначається використанням регулярних виразів.

Регулярні вирази (англ. «regular expressions») – формальна мова пошуку і здійснення маніпуляцій з підрядками в тексті, заснована на використанні метасимволів, наприклад «/», «*», «.» та багато інших. Для пошуку використовується рядок-зразок, що складається з символів і метасимволів та задає правило пошуку [27].

Node.js — це JavaScript–оточення побудоване на JavaScript–движку Chrome V8. Як асинхронне подієве JavaScript–оточення, Node.js спроектований для побудови масштабованих мережевих додатків [28], і через родову схожість із браузерною мовою програмування має перевагу у вигляді відсутності потреби у вивченні іншої серверної мови а також можливість оперування, по суті, однією мовою одразу обидві частини сайту.

Express – це мінімалістичний і гнучкий веб-фреймворк для додатків Node.js, що надає великий набір функцій для мобільних і веб-додатків. Маючи в своєму розпорядженні безліч службових методів HTTP і проміжних оброблювачів, створити надійний додаток можна швидко і легко [20].

MongoDB - це нереляційна база даних документів. Бази даних NoSQL (вони ж "not only SQL") не є табличними та зберігають дані інакше, ніж реляційні таблиці. Бази даних NoSQL бувають різних типів, засновані на їх моделі даних. Основними типами є документ, ключ-значення, широкий стовпець та графік. Вони легко забезпечують гнучкі схеми та масштабування з великою кількістю даних та великим завантаженням користувачів [29]. Mongoose – це бібліотека об'єктних даних моделювання для MongoDB та Node.js [30], що зберігає дані використовуючи хмарні технології. Через зручність імплементування коду взаємодії із базою даних на сервері Node.js, використання mongoose є релевантним рішенням проблеми зберігання інформації користувачів веб-ресурсу даної проектної роботи.

Модель документа MongoDB проста для розробників для вивчення та використання, але все ж надає всі можливості, необхідні для задоволення найскладніших вимог у будь-якому масштабі [29].

РОЗДІЛ 3.

СТВОРЕННЯ АДАПТИВНОГО LANDING PAGE

3.1. Планування алгоритму створення сторінки

Для створення алгоритму розробки сторінки, доцільним буде визначити усі необхідні складові роботи:

- Визначити завдання;
- Знайти контент (текст, фото, картинки та інше);
- Визначити шаблони верстки, що будуть використовуватись на сторінці;
- Створити сервер та базу даних, шо буде приймати заяви зі сторінки.

Завданням даної цільової сторінки буде прорекламувати кафедру комп'ютерних наук та інформаційних технологій факультету інформаційних технологій (ФІТ) як для абітурієнтів, так і для студентів київського національного торговельно-економічного університету.

Необхідно зробити декілька секцій контенту, де буде вказана поточна інформація та особливості кафедри, розміщення на території університету та на карті, контакти ВНЗ а також форма зворотнього зв'язку з кафедрою. За правилами Best Practices у сфері веб-розробки, необхідно буде розмістити інформацію якомога стисліше та привабливіше – небагато тексту, більше фото і посилань на основне джерело інформації.

Текстове наповнення сторінки буде взято з офіційного сайту КНТЕУ, на сторінці про кафедру ФІТ а також буде взято відео- та фото-матеріали про KNUTE Hub та Smart Library.

Під шаблонами на сторінці мається на увазі шрифти, кольори сайту, однотипні блоки інформації, тощо.

Так як флаг факультету інформаційних технологій має блакитні кольори, то й Landing Page буде мати переважно блакитний колір, у поєднанні із сучасними тенденціями у веб-дизайні (розділ 1.3). Також увесь контент на сайті повинен бути загорнутий у так званий контейнер – дочірній блок секції інформації, що

позиціонує матеріали по центру вікна а також має певну довжину, яка зазвичай дорівнює 1100-1200 пікселів або 70-80% ширини вікна.

Сервер та база даних буде необхідна для прийняття та опрацювання заявок від форми зворотнього зв'язку. Це буде зовсім крихітний сервер, як і БД, проте їхня роль є ключовою – саме заради отримання контактної інформації відвідувача й створюються «сайти-візитки».

3.2. Моделювання процесу розробки

З метою визначення чіткого вектору дій та кращого розуміння задач, було розроблено моделі процесу розробки, а саме діаграму прецедентів (Use Case) та діаграму послідовності (Sequence Diagram). Моделювання здійснено за допомогою програмного засобу StarUML.

UML, скорочено від Unified Modeling Language – це стандартизована мова моделювання, що складається з інтегрованого набору діаграм, розроблених для надання розробникам програмного забезпечення конкретизації, візуалізації, побудови та документування артефактів програмних систем, а також для бізнес-моделювання та інших задач. UML являє собою сукупність найкращих інженерних практик, які виявилися успішними в моделюванні великих і складних систем; ця мова використовує переважно графічні позначення для вираження дизайну програмних проектів [12].

Use Case – це письмовий або візуалізований опис того, що користувачі будуть робити на веб-сайті. З точки зору користувача, діаграма варіантів використання визначає поведінку системи під час відповіді на запит або дію основного актору (як правило, людини). Кожен випадок використання представлений у вигляді послідовності простих кроків, починаючи з мети користувача та закінчуючимоментом, коли ця мета виконується.

Діаграма прецедентів використання допомагає пояснити, як повинна поводити себе система а також допомагає виявити проблему, якщо щось не так. Дана модель надає перелік цілей, котрий у комерційних випадках розробки може

бути використаний для встановлення вартості та складності розроблюваної системи. [13].

На діаграмі зображені наступні об’єкти: основний актор – абітурієнт (як правило), та два вторинних актори – адміністратор, сервер (Рис. 3.1).

Абітурієнт має такі можливості на сайті:

- Зайти на сайт;
- Ознайомлюватися з контентом;
- Заповнювати форму зворотнього зв’язку;
- Відправляти валідовану форму.

Прямо при заповненні своїх контактних даних, на клієнтській частині сайту (на сторінці) буде виконуватися процес валідації даних – дані будуть перевірятися на правильність заповнення, згідно з запрограмованими правилами заповнення полів введення. При неправильному заповненні поля буде вискакувати підказка, наприклад, «у полі номера телефону має бути 10 цифр», в іншому випадку форму неможливо буде відправити, тощо.

В момент відправлення форми, інформація відправляється вторинному акторові – серверу, що також повинен буде провалідувати дані на правильність заповнення, і при незадовільному результаті повідомляти про помилки користувачу. У випадку успішної валідації на сервері, дані записуються у підключену базу даних. Дані 2 опції будуть єдиним існуючим функціоналом серверу, окрім підгрузки сторінки сайту.

Другий вторинний актор – адміністратор – має 2 варіанти взаємодії з системою – це надіслати електронного листа абітурієнтові, або дзвінок абітурієнтові за даними, що записані у базі.

Успішним завершенням Use Case є заповнення та надсилання контактної форми на сервер. Невдалим завершенням можуть бути наступні причини:

- Сайт не було підгружено через погане інтернет-з’єднання користувача;
- Сайт не було підгружено через наявність проблем з хостингом;

- Сайт не було підгружено через несправності з роботою пристрою, під’єднаного до мережі Інтернет у користувача;
- Домен сайту заблоковано на території користувача;
- Відсутність електричного струму у загальній мережі електричного постачання.

Отже, генеральний сценарій успіху виглядає таким чином:

1. Людина заходить на цільову сторінку.
2. Ознайомлюється із лаконічною інформацією.
3. Якщо це студент, абітурієнт або хтось із його родичів, людина заповнює контактну форму та натискає кнопку відправлення.

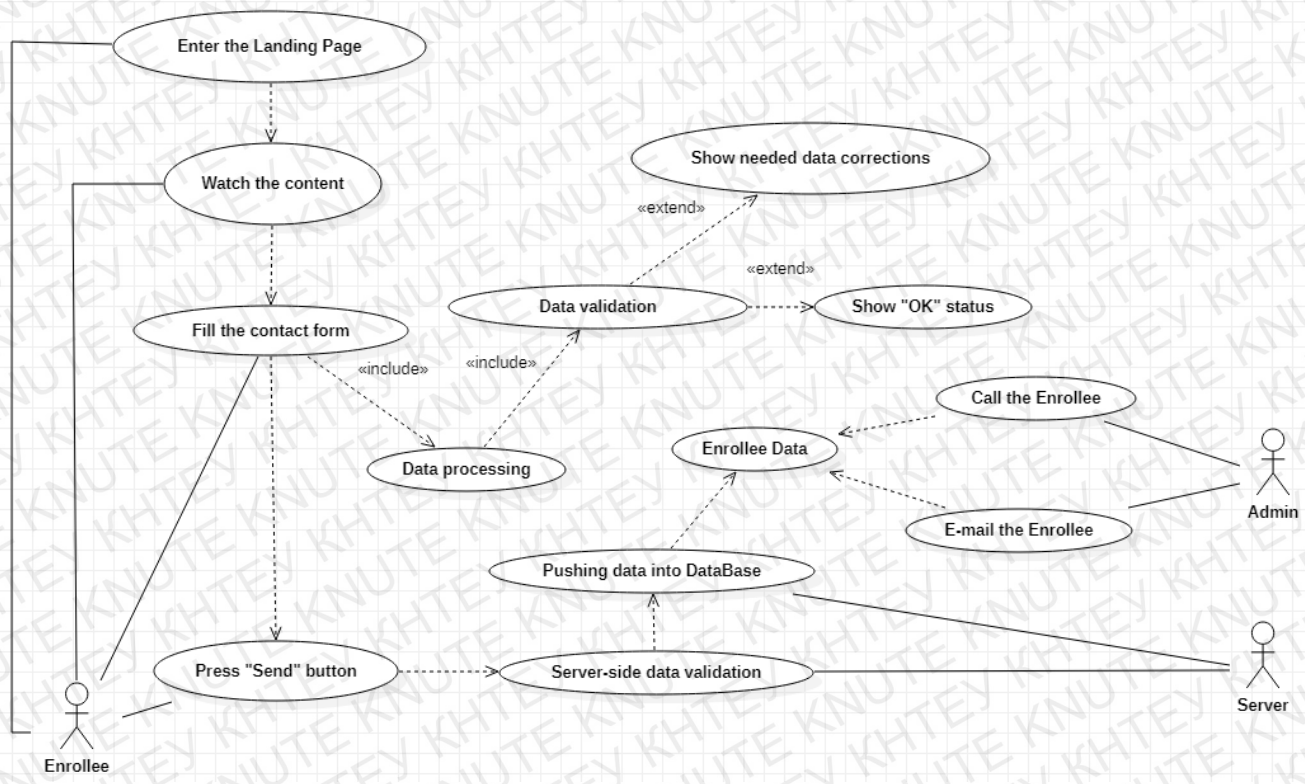

Рис. 3.1. Діаграма прецедентів використання

Діаграми послідовності UML – це діаграми взаємодії, які детально описують спосіб виконання операцій. Вони фіксують взаємодію між об'єктами в контексті співпраці. Діаграми послідовності орієнтовані на час, і вони візуально показують порядок синергії, використовуючи вертикальну вісь діаграми, щоб представити час, які повідомлення надсилаються та коли [14].

На моделі послідовності зображені ті ж самі об’єкти що й у моделі Use Case, за вийнятком відсутнього вторинного актору «сервер». Послідовність операцій на

Landing Page виглядає наступним чином:

1. Основний актор «абітурієнт» взаємодіє з об’єктом «цільова сторінка» - заходить на сайт;
2. Об’єкт «цільова сторінка», за сценарію успіху Use case, взаємодіє з об’єктом «форма» - основний актор заповнює її;
3. Об’єкт «форма» синергує із об’єктом «валідація на клієнтській частині» - йде перевірка введених даних;
4. Валідація повертає булево значення – true або false, в залежності від правильності заповнених абітурієнтом даних;
5. За пройденого етапу валідації, розблоковується можливість натискання на кнопку відправлення форми – при кліку дані відправляються на об’єкт «серверна валідація даних», де йде перевірка інформації;
6. За успішної валідації, сервер записує контактні дані в об’єкт «база даних»;
7. Сервер, в залежності від поверненного після валідації булевого значення, відправляє подяку або помилку на Landing Page;
8. Вторинний актор «адміністратор» спостерігає за роботою об’єкта «цільова сторінка»;
9. Вторинний актор «адміністратор» взаємодіє з об’єктом «база даних» та контактує із абітурієнтом.

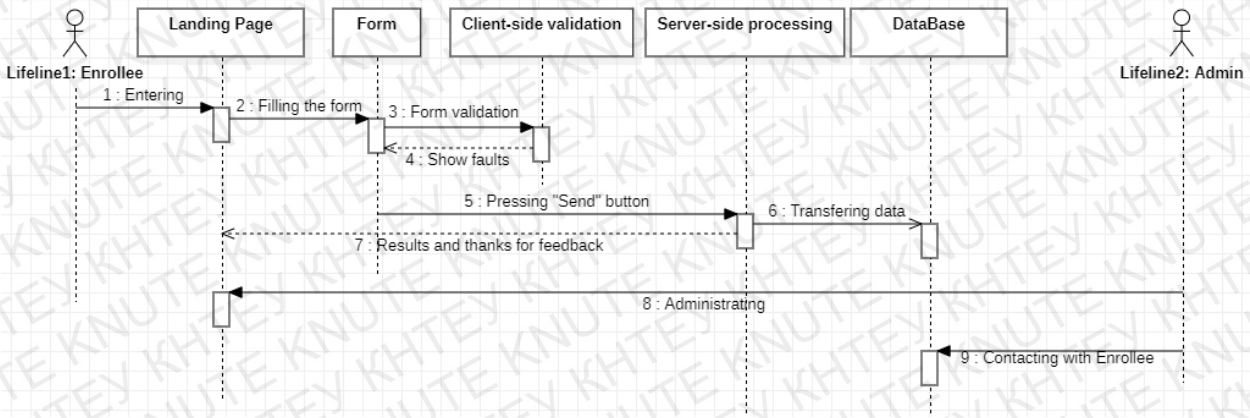

Рис. 3.2. Діаграма послідовності

3.3. Розробка адаптивного Landing Page

Найпершою дією безпосередньо при розробці веб-сайту є планування щодо використання та створення однакових блоків-обгорток контенту – шаблонів. Перш за все маються на увазі шрифти на сторінці: їх розміри, сімейство, дистанція між рядками, колір тексту та інше. Наступними частинами що будуть зустрічатися на сторінці часто – це кнопки, загальні стилі для заголовків та звичайного тексту, однакові властивості полів зворотнього зв'язку, спільна стилістика усіх елементів таких як закруглення меж елементів, тощо. Дана дія значно зменшить час розробки.

Для зручності користування, доцільним буде створення окремого файлу стилів, звідки будуть імпортуватися загальні стилі щоб не «забруднювати» код. З метою зручного звернення, у роботі будуть використані змінні технології SCSS – @mixin. Mixin дозволяє визначати стилі, які можна повторно використовувати у верстці, допомагаючи уникнути використання не семантичних назв класів, таких як «.float-left» або «.without-padding», та розповсюдження великої кількості повторюваних рядків коду [15] (Рис. 3.3).

```scss
$grey: #6c727f;
$black: #161616;
$blue: #005bea;
$darkBlue: #0f1c76;

@mixin default_flex {
  display: flex;
  justify-content: space-between;
  align-items: center;
}
```

```scss
@mixin simple_transition {
  opacity: 1;
  transition: 0.2s;
  &:hover {
    opacity: 0.9;
  }
}
@mixin button_template {
  font-family: 'Manrope-Bold';
  font-size: 18px;
  background-color: #fff;
  padding: 16px 30px 17px 30px;
  border-radius: 8px;
  cursor: pointer;
  @include simple_transition;
}
```

Рис. 3.3. Приклад використання змінних @mixin

Шапка сторінки включає в себе навігацію на сайті. У випадку даної проектної роботи, header (семантичний тег HTML, що визначає верхнє меню сайту – шапку) буде містити посилання на офіційний сайт університету, а також якорі – тобто посилання на певні блоки контенту на сторінці (Рис 3.4).

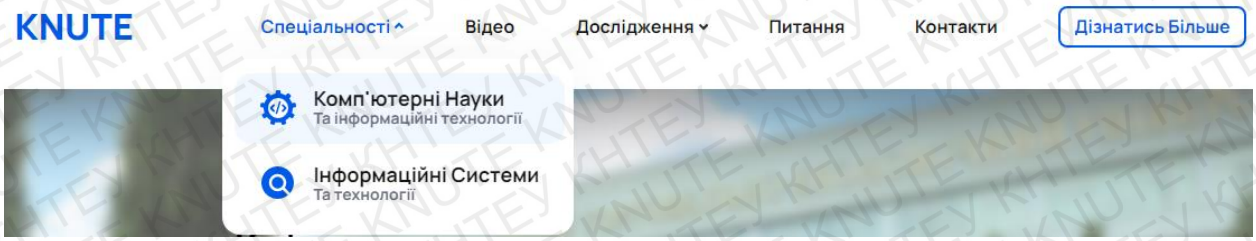

Рис. 3.4. Шапка сайту під розширення екранів ПК

Зазвичай, даний блок знаходиться вгорі сторінки, а на мобільних пристроях меню навігації статично прикуте до верху сторінки, й не пропадає при гортанні вниз, а також приймає метаморфозу у вигляді невеличкої кнопки з випадаючим меню, яку у професійній мові верстки називають «бургером» (з англ. Burger Menu, така назва прижилася через схожість). Це стандарт UX, що має вагомий вплив на зручність в експлуатації на невеликих екранах.

Як і будь-якому інтерактивному елементу, випадаючому меню, згідно із сучасними тенденціями веб-дизайну, необхідно запрограмувати плавну анімацію взаємодії.

При розробці анімацій, є наступні можливі рішення:

- Маніпулювання CSS-властивостями, такими як transform, transition та іншими;
- Використання @keyframes-animations у комбінації із базовими методами метаморфоз CSS-властивостей;
- Застосування браузерної мови програмування javascript.

За будь-якого методу буде проходити процес маніпулювання CSS-властивостями, проте найпоширенішими є випадки, у яких більш доцільним буде нескладна та швидка у розробці зміна характеристик елементу без залучення спеціальних команд. Даний спосіб є найуніверсальнішим, оскільки він є базою для усіх інших.

Правило @keyframes визначає код анімації, яка створюється шляхом поступового переходу від одного набору стилів CSS до іншого. Під час анімації можна багаторазово змінювати властивості елементів. За даного методу вказується, коли зміна характеристик елементу відбудеться за ключовими словами "від" і "до"

(from, to), або більш точний аналог – за процентним відношенням до часу процесу анімації, де зміни відбудуться при певному проценті виконання, де 0% – це початок анімації, а 100% – її завершення [16].

За програмної реалізації майже немає рамок використання, оскільки даний спосіб передбачає вільний доступ до будь-якого елементу сторінки незалежно від ієрархії розміщення елементів. Основними методами взаємодії з об'єктами на сторінці є document.querySelector() – що витягує елемент за назвою класа, іменем тегу або за унікальним ідентифікатором переданим в аргумент функції, та .addEventListener() що надає обраному за допомогою .querySelector() елементу функцію-слухача, що буде виконувати запрограмовані операції тільки за спричинення певних подій, таких як натискання на елемент, наведення та відведення курсору миші, відпускання клавіши миші та інших.

У якості каркасу анімації доречним буде імпортування готового анімаційного рішення із спеціалізованих на даній проблематиці сайтів. У випадку даної роботи, на сайті працюють у синергії одночасно декілька адаптованих під сучасні мотиви типів анімації згортання та розгортання випадаючих меню (Рис. 3.5).

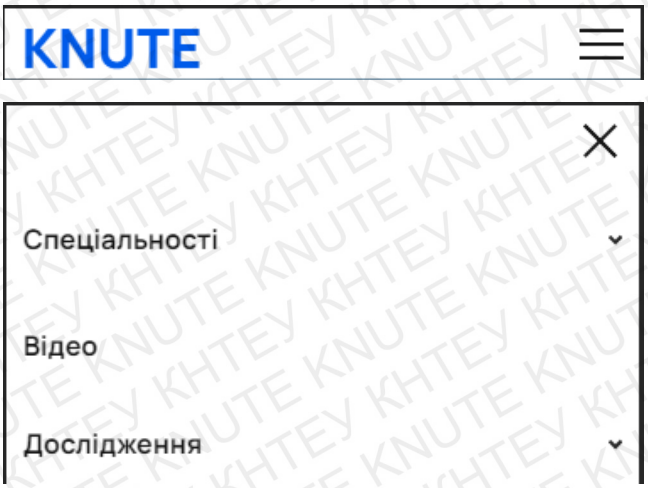

Рис. 3.5. Шапка сайту на розширеннях екранів смартфонів

Як наслідок ситуації із випадаючим меню, робота складається не тільки з верстки версії сайту під екрани персональних комп'ютерів – величезним сегментом коду буде саме адаптація кожної із секцій контенту під більшість розмірів екранів. Дана ситуація є показовою, оскільки один єдиний варіант меню навігації на смартфонах

не буде виглядати так само добре як на desktop-версії. Випадків подібних цьому буде надзвичайно багато, всі вони потребують адаптації. Кращою практикою в адаптуванні веб-сайтів є кодування нових стилів та оновленого, спеціалізованого веб-дизайну під різну ширину екранів, керуючись так званими брейкпоінтами (англ. Breakpoint, точка перелому) що визначаються як розширення екрану у пікселях, при якому зверстана сторінка «ламається», тобто виходить за рамки екрану та формує меню горизонтальної прокрутки. Дана проблема є основною, і за відсутності динамічних одиниць розмірів зустрічається буквально постійно. Проте, навіть за гумової верстки виникають ситуації, що потребують імплементування нових характеристик елементів на сайті. CSS-селектор @media screen дозволяє впровадити налаштування верстки певних елементів в залежності від розмірів екрану, як правило саме від ширини.

Використання media-запитів виглядає наступним чином (Рис. 3.6):

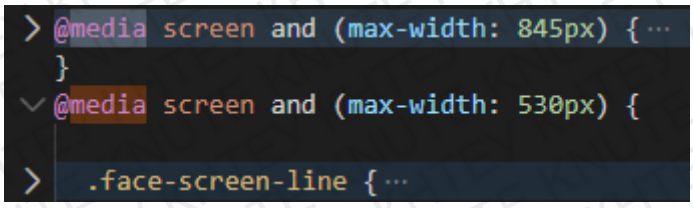

Рис. 3.6. Приклад використання media-запиту

Лицьова секція слугує обкладинкою сторінки, тому потребує деякої візуальної обробки. На сьогоднішній день популярним рішенням є легке розмиття фотографії високого розширення а також накладення ефекту тіні всередину блока, шо буде виступати як приємне оку фонове зображення. Такого ефекту можна досягти без редагування фотографії: використовуючи ієрархію блоків, поверх блоку з фоновою фотографією накладається напівпрозорий кольоровий елемент що динамічно розтягується на ширину батьківського блоку, з використанням спеціалізованої CSS-властивості backdrop-filter. Даний CSS-фільтр заднього фону надає можливість застосовувати графічні ефекти, такі як розмивання чи зміщення кольорів до області за елементом. Оскільки це стосується всього, що стоїть за елементом, щоб побачити ефект, є необхідність зробити елемент або його фон хоча б частково прозорим [17] (Рис 3.7). Даний ефект можна порівняти із видимістю, яка

з’являється при погляді через скло.

```
.face-screen {
  overflow: hidden;
  position: relative;
  z-index: 2;
  height: 500px;
  background: url(../img/knute/knute.jpg) center;
  box-shadow: inset 0px 0px 90px 0px rgba(0,0,0,1);
  background-size: cover;
  .blur {
    @include default_flex;
    width: 100%;
    height: 100%;
    background-color: rgba(255,255,255, 0.3);
    backdrop-filter: blur(5px);
```

Рис 3.7. Використання backdrop-filter

Оскільки на кафедрі Комп'ютерних Наук та Інформаційних Систем є 2 спеціальності а також ведеться робота над двома дослідженнями, сприятливими елементами сторінки будуть картки з інформацією. Картка – це відносно невеликий блок, що містить у собі коротку інформацію про продукт. Подібні рішення можна спостерігати на будь-якому сайті або журналі, що пропонує товари або послуги (Рис. 3.8). Оптимальна адаптація під маленькі екрани передбачає собою розташування карток не у ряд, а у стовпчик.

Рис. 3.8. Використання карток у landing page

Landing page або індексна сторінка багатосторінкового сайту може мати секцію з відповідями на поширені запитання, що має на меті економити час відвідувачам. Сучасною реалізацією такого рішення є використання випадаючого

списку або тексту при взаємодії із вкладкою-питанням, що також потребує анімації для прийнятного візуального сприйняття.

За основу процесу згортання та розгортання текстової інформації будуть залучені стилі, що належать шапці сайту. Оскільки дані характеристики вже зустрічалися у коді, хорошим з точки зору оптимізації буде рішення винести базові стилі вкладки до загальних, щоб запобігти багаторазовому повторюванню рядків коду. Це яскравий приклад, як подібні превентивні міри зменшують габаритність коду: дана реалізація запобігла дублюванню 67 рядків коду (Рис. 3.9).

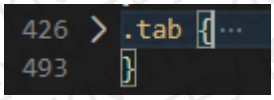

Рис.3.9. Стилі вкладки у згорнутому вигляді в редакторі коду

За таких дій секція питань є гумовою та має наступний вигляд на сторінці (Рис. 3.10):

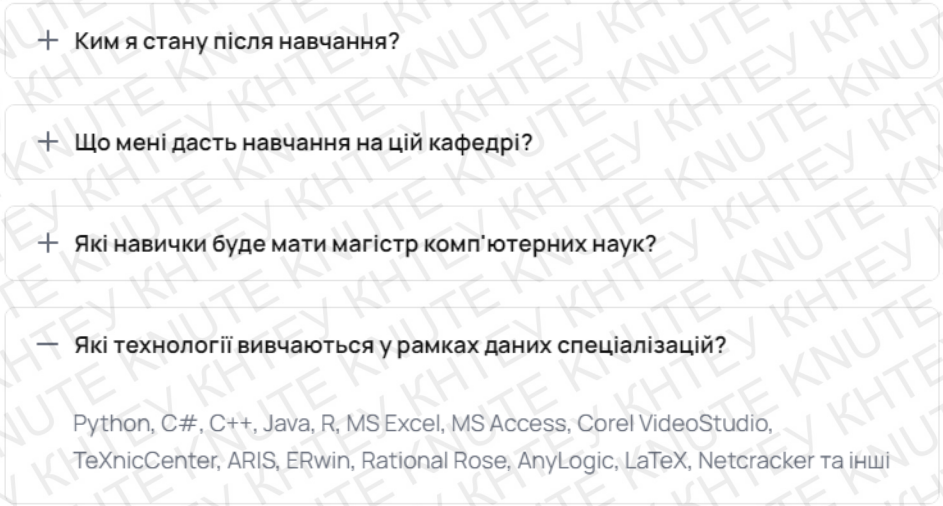

Рис. 3.10. Вкладки на сторінці

Розміщення відеоматеріалів із екскурсією по Б-корпусу на цільовій сторінці є непоганим рішенням оскільки в корпусі кафедри комп’ютерних наук та інформаційних систем є досить багато цікавих місць для відвідування, особливо KNUTE Hub та Smart Library як наслідок комп’ютеризації молоді.

IFrame (англ. Inline Frame, вбудований кадр) – це документ HTML, вбудований в інший HTML-документ на веб-сайті. Елемент HTML IFrame часто використовується для вставки вмісту з іншого джерела, наприклад реклами,

симуляції редактору коду, відео-матеріалів, тощо. Хоча IFrame поводить себе як вбудоване зображення, його можна налаштувати за допомогою власної панелі прокрутки незалежно від смуги прокрутки батьківської сторінки [19].

Станом на 2020 рік, імпортування відео-матеріалів на веб-сайти з інших платформ є завданням не складним. На поширених сайтах відео-хостингу є функція копіювання коду елементу відео, що значно полегшує імпортування такого блоку. Не дивлячись на свою очевидну відмінність від звичайних HTML тегів, стилізація відбувається так само як і з будь-якими іншими елементами. Тому необхідною умовою імплементації відео є надання динамічних значень висоти й ширини такого блоку (Рис. 3.11).

Дивіться відео про корпус нашої кафедри

Рис. 3.11. Імплементація відео

Основною метою Landing page є корисна дія користувача, у даному випадку заповнити та відправити форму. В секції зворотнього зв’язку необхідно мати наступні елементи:

- Контактна форма;
- Адреса корпусу;
- Місцезнаходження на карті.

Щоб дані були записані у JSON-об’єкт запиту контактної форми, необхідно надати полям введення атрибут name, тобто ім’я – для зчитування інформації з них

(Рис 3.12).

```
<form action="message" method="POST">
  <p>Зв'язатись</p>
  <div class="form-data">
    <label class='form-input'>
      <input name="name" placeholder=" " minLength="2" maxLength="18" required />
```

Рис. 3.12. Використання спеціальних атрибутів форми

Для коректного заповнення форми, використовується валідація форми на клієнтській частині.

Під час процесу кодування, запрограмувати перевірку для введення даних є важливим рішенням, оскільки це запобігає появі несподіваних або ненормальних даних, що можуть спричинити збій програми. Інакше кажучи, валідація не дозволяє отримувати неможливі для обробки дані [19].

Для створення механізму коригування даних під час їх введення, буде використовуватися тільки спеціалізовані властивості CSS, без імплементування javascript-коду. Псевдо-класи :valid та :invalid визначають стилі поля введення при, відповідно, валідному та невалідному заповненні.

Випадок даної роботи не передбачає собою комплексне використання усіх типів клієнтської валідації, оскільки необхідно перевірити усього 2 поля – електронну пошту та ім’я, що запише людина у формі (Рис. 3.13).

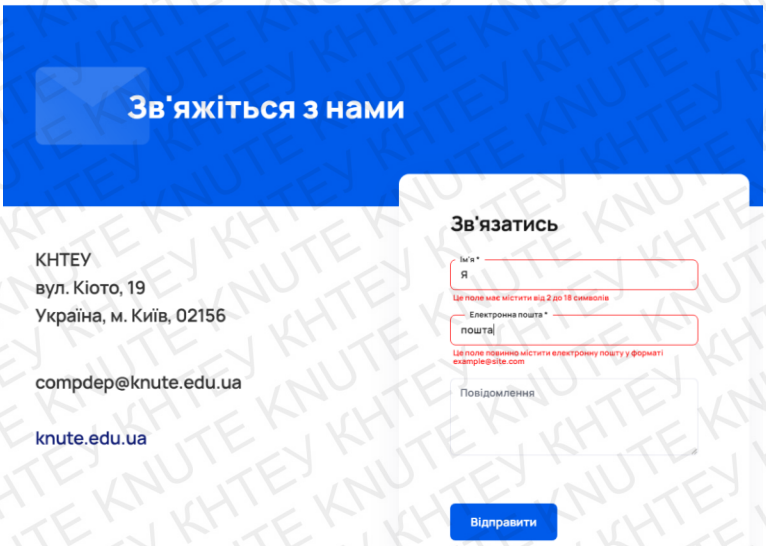

Рис 3.13. Секція зворотнього зв’язку та валідація форми

Також сучасним рішенням буде впровадження місцезнаходження на карті.

Використання сервісу Google Maps є доцільним, оскільки він є найпоширенішим. Імплементація передбачає собою використання вже знайомого тегу iframe (Рис. 3.14)

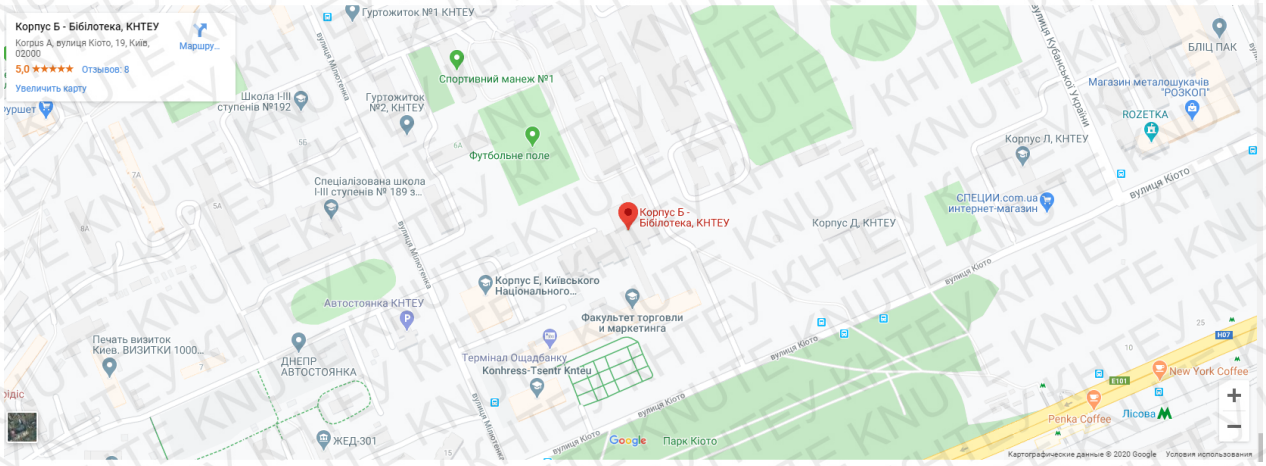

Рис. 3.14. Впровадження місцезнаходження на картах Google Maps до веб-сторінки кафедри

Рішенням питання адаптованості є зміна позиціонування елементів: спочатку розміщується заголовок, потім форма зворотнього зв'язку, далі карта, і тільки пісня неї йде поточна адреса та контакти. Таке розміщення є доцільним як з точки зору дизайнерських рішень, так і з точки зору зручності взаємодії із сторінкою.

На будь-якому сайті повинен бути його «підвал» – це блок даних внизу сторінки, що містить посилання на різні частини сайту, інші ресурси та іноді додаткову контактну інформацію. Такий елемент повинен бути загорнутий у семантичний HTML-тег <footer>. Як і увесь контент, дані підвалу сайту взято з офіційного сайту Київського Національного Торговельно-Економічного Університету (Рис. 3.15).

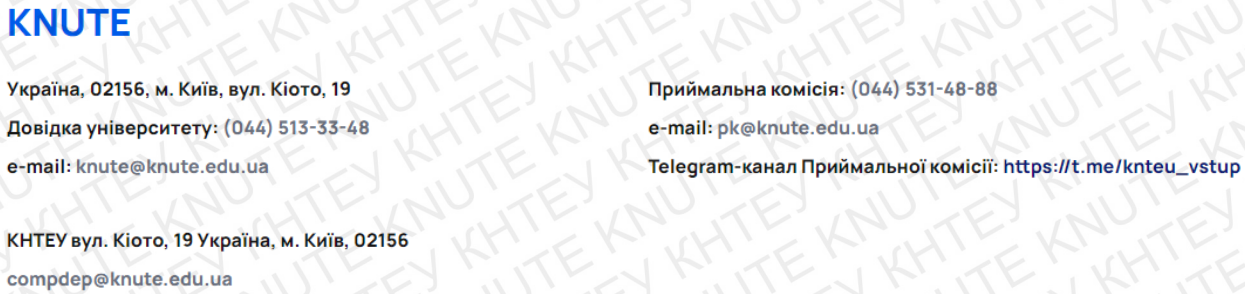

Рис. 3.15. Footer

3.4. Налагодження роботи сервера та БД

За умовою технічного завдання, дана проектна робота не потребує великої кількості налаштувань на сервері, тому використання фреймворку Express.js для Node.js є влучним рішенням, через свою простоту імплементування шаблонів серверу та відсутність великих габаритів програмного коду. Інсталювавши Node.js та пакетний менеджер npm, можна приступати до програмування. Для початку необхідно задекларувати використання модуля Express та створити сервер з базовими, необхідними налаштуваннями його роботи, такими як bodyParser (надає можливість серверу проглядати HTML сторінку на елемент відправлення форми), налаштування відповіді у форматі JSON (JavaScript Object Notation, формат обміну даними що легко читається як людиною, так і комп'ютером) та його запуск (Рис. 3.16).

```
const express = require('express');
const app = express();

app.use(express.json());
app.use(express.urlencoded({ extended: false }));

app.use(express.static('public'));

app.get('/', function (req, res) {
  res.sendFile('index.html')
})
app.listen(3001, function(){
  console.log("Listening on port 3001!");
});
```

Рис. 3.16. Базові налаштування express-серверу

Щоб ловити форму, сайтові необхідно зробити route за методом POST, який буде приймати дані із форми зворотнього зв'язку (Рис. 3.17). Роут приймає у себе аргументи шляху, по якому буде здійснено певну дію, та функцію, що й буде виконувати певні дії при переході на шлях. На великих сайтах використовується метод модуля express під назвою router, що може викликати HTTP методи, проте на невеликому сайті як landing page буде достатньо й виклику функції проміжної обробки, або middleware, за потрібним HTTP-методом. Різниця цих двох способів у їх прямому призначенні – мідлвейр використовується для налаштувань сервера

та обробки усіх запитів до того як вони дійдуть до роутів, router призначений лише для обробки шляхів. Функції шляхів приймають 2 обов'язкових аргумента: запит та відповідь (request, response). За участі двох об'єктів й відбувається увесь процес обробки даних: перевірка, операції, збереження до бази даних, відправлення, прийняття рішень, тощо.

```
app.post('/message', async function (req, res) {
    console.log(req.body);
})
```

Рис. 3.17. Функція проміжної обробки атрибута action форми

На даному POST-шляху необхідно обробити дані з форми. Серверна валідація захищає програму від несанкціонованих помилок та запобігає злому сайту. У роботі буде використовуватися модуль Ajv, що призначений для швидкого та якісного валідування за заздалегідь створеною схемою. Механізм роботи модуля наступний: схема пишеться як JSON-об'єкт у відокремленому файлі, а потім компілюється при використанні у програмному коді в швидкісну функцію, що повертає булево значення після перевірки [21] (Рис. 3.18).

```
// validation
const {
  name, mail, message
} = await req.body;
const validate = await ajv.compile(messageSchema);
const valid = await validate(req.body);
console.log(`VALIDATION: ${valid}`);
if (!valid) {
  const { errors } = validate;
  const result = {
    status: 'invalid data input',
  };
  console.log(errors);
  res.json(result);
}
```

```
const message = {
  "$schema": "http://json-schema.org/draft-07/schema#",
  "properties": {
      "mail": {
          "format": "email",
          "minLength": 2,
          "maxLength": 18,
          "require": true,
      },
      "name": {
          "type": "string",
          "minLength": 3,
          "maxLength": 18,
          "require": true,
      },
      "message": {
          "type": "string",
      }
  },
};
module.exports = message;
```

Рис.3.18. Програмний код валідації та json-схема

Перевірені дані необхідно записувати в базу даних. Оскільки MongoDB – нереляційна база даних, це означає, що є можливість зберігати в базі документи JSON, і структура цих документів може бути змінною, оскільки сувора типізація за схемою є відсутня, навідміну від реляційних, SQL-баз даних. Це одна з переваг використання NoSQL, оскільки такий метод прискорює розробку додатків. Для подальшого використання, Mongoose необхідно інсталювати та під'єднати до

серверу. Встановлення виконується шляхом написання у термінал команди “npm install mongoose”, а також “npm install” у директорії сайту, щоб встановити залежності і програмний код міг використовувати даний модуль.

Необхідні під’єднання та налаштування виконуються у файлі серверу (Рис. 3.19), де використовуються 2 основні методи даного модуля:

- Встановлення зв’язку з існуючою (або створення нової) базою даних mongoose.connect(), що приймає у себе API-ключ із паролем та об’єкт налаштувань;
- Процес під’єднання до хмарної бази даних mongoose.connection.once(), що також повинен обробляти помилку у разі її виникнення.

```
const mongoose = require('mongoose');
// DB connection
mongoose.connect('mongodb+srv://admin:1111@cluster0-7cnbh.mongodb.net/test?retryWrites=true&w=majority', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});
const db = mongoose.connection;
db.once('open', (err) => {
  if (err) throw err;
  console.log('Connected to db!');
});
```

Рис. 3.19. Налаштування mongoose

Не дивлячись на свою відносну свободу у структуруванні даних при їх записі, будь-якій базі даних незалежно від типу потрібна модель, за якою будуть зберігатися дані в колекцію бази даних. Такі моделі потребують створення окремих файлів, з імпортуванням у них модуля mongoose (Рис. 3.20).

```
const mongoose = require('mongoose');

const MessageSchema = new mongoose.Schema({
  name: String,
  email: String,
  message: String,
});

const Model = mongoose.model('message', MessageSchema);
module.exports = Model;
```

Рис. 3.20. Модель бази даних

Тестування форми зворотнього зв’язку дало позитивний результат (Рис. 3.21).

Рис. 3.21. Результат валідації форми у терміналі серверу та запис у колекції бази даних відповідно

Після розробки усіх необхідних структур сайту, таких як клієнтська та серверна частина, landing page є повністю готовим до використання.

ВИСНОВКИ

У випускному кваліфікаційному проекті представлено результати теоретичних та емпіричних досліджень, що полягають у розробці адаптивного Landing page кафедри комп'ютерних наук та інформаційних систем. Резюмуючи виконану роботу, були отримані наступні **висновки**:

1. Було проаналізовано ряд технологій Front-end та Back-end розробки та їх особливості, що забезпечило виконання мети даної випускної кваліфікаційної роботи – дослідження сучасних методів розробки адаптивного landing page. Технології, що були вивчені: HTML, CSS, SASS, CSS-таблиці, CSS-float, CSS-flexbox, CSS-grid, npm, JavaScript, Express.js, Ajv, mongoose.
2. Отримано навички використання кращих практик при верстці сторінок веб-сайту, таких як методи гнучкої, гумової верстки, використання особливостей таблиць каскадних стилей CSS для розробки адаптованих версій певних секцій сайту, способи оптимізації графічних елементів на клієнтській частині або ретинізація в яку входять наступні методи:
    - HTML і CSS-масштабування;
    - Визначення щільності пікселів екрану та використання JavaScript для підгрузки графічних елементів із різним рівнем якості зображень;
    - Масштабна векторна графіка SVG;
3. Використано сервіси Google Maps для відображення місцезнаходження корпусу університету на картах за допомогою тегу <iframe>, а також за такої ж схеми, на цільову сторінку було імплементовано використання відео-хостингу Youtube в якості способу реалізації інтеграції відео-матеріалу.
4. Було вивчено особливості та тенденції сучасного веб-дизайну, що мало вирішальний вплив на зовнішній вигляд клієнтської частини сайту та на його інтерактивні елементи, такі як кнопки, форма зворотнього зв'язку, посилання на інші ресурси та навігація по сайту, тощо.
5. Емпіричним шляхом досліджено чисельну кількість методологій верстки: адаптивні та чутливі (гумові) способи веб-розробки, mobile-first та desktop-first

методи, розділення повторюваних елементів веб-сайту на шаблони, ретинізація графічних об’єктів таких як растрові зображення, оптимізація навігаційного меню на екранах смартфонів та планшетів, імплементація сучасних тенденцій веб-дизайну на інтерактивні елементи, тощо.

6. Проаналізовано методи серверної шаблонізації, за допомогою використання фреймворків, модулів та бібліотек, базових методів HyperText Transfer Protocol (HTTP), функцій проміжної обробки шляхів, синтаксису мов програмування JavaScript та Node.js;
7. Шляхом дослідів було виявлено основні переваги й недоліки методів та інструментарію розробки веб-сайтів а також зроблено їх порівняльний аналіз;
8. Немає сенсу винаходити власну систему взаємодії елементів (не має значення – у верстці, чи у програмуванні) якщо більшість ситуацій у світі вже було кимось вирішено. Кращі практики – це досвід не одного десятку тисяч розробників, що емпіричним шляхом дійшли до найбільш зручного та\або найефективнішого варіанту розробки. Переймаючи цей досвід, мета випускного кваліфікаційного проекту була досягнута високим рівнем якості веб-розробки;
9. Імплементовано хмарну нереляційну базу даних mongoose та визначено фундаментальні принципи взаємодії додатку з базою, що дає змогу введеним користувачем даним можливість надійного зберігання;
10. Організовано цілісну систему, яку було створено безпосередньо через отримання базових навичок роботи програміста повного циклу, що включає у себе налагодження роботи серверу, створення веб-додатку на базі HTTP, під’єднання бази даних, валідації, створення чисельної кількості згрупованих HTML-елементів із дотриманням шаблонізації та ієрархії у їх стилізації, розроблення гумової та адаптивної версій сайту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Скільки людей в Україні користуються інтернетом: статистика [Електронний ресурс]. – Режим доступу: https://tech.informator.ua/2019/10/12/skolko-lyudej-v-ukraine-polzuyutsya-internetom-i-kakoj-portret-ukrainskogo-internet-polzovatelya-statistika/
2. Landing Page Statistics 2020 у. [Електронний ресурс]. – Режим доступу: https://truelist.co/blog/landing-page-statistics/
3. Flexbox history [Електронний ресурс]. – Режим доступу: https://annairish.github.io/historicizing/history
4. Screen resolution statistics [Електронний ресурс]. – Режим доступу: https://gs.statcounter.com/screen-resolution-stats/all/ukraine
5. Тенденции дизайна сайтов 2020 - примеры и описание ключевых элементов [Електронний ресурс]. – Режим доступу: https://impulse-design.com.ua/osnovnye-trendy-veb-dizajna-2018.html
6. Табличная верстка [Електронний ресурс]. – Режим доступу: http://htmlbook.ru/content/tablichnaya-verstka
7. All About Floats [Електронний ресурс]. – Режим доступу: https://css-tricks.com/all-about-floats/
8. Basic concepts of flexbox [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox
9. A Complete Guide to Flexbox [Електронний ресурс]. – Режим доступу: https://css-tricks.com/snippets/css/a-guide-to-flexbox/
10. A Complete Guide to Grid [Електронний ресурс]. – Режим доступу: https://css-tricks.com/snippets/css/complete-guide-grid/
11. Оптимизация графики для Retina-экранов [Електронний ресурс]. – Режим доступу: https://habr.com/en/post/150071/

12. What is Unified Modeling Language (UML)? [Електронний ресурс]. – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/
13. Use Cases [Електронний ресурс]. – Режим доступу: https://www.usability.gov/how-to-and-tools/methods/use-cases.html
14. What is Sequence Diagram? [Електронний ресурс]. – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/
15. @mixin and @include [Електронний ресурс]. – Режим доступу: https://sass-lang.com/documentation/at-rules/mixin
16. @keyframes animations [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/cssref/css3_pr_animation-keyframes.asp
17. backdrop-filter [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/CSS/backdrop-filter
18. What is IFrame? [Електронний ресурс]. – Режим доступу: https://whatis.techtarget.com/definition/IFrame-Inline-Frame
19. Validation [Електронний ресурс]. – Режим доступу: https://www.computerscience.gcse.guru/theory/validation
20. Express. Fast, unopinionated, minimalist web framework for Node.js [Електронний ресурс]. – Режим доступу: http://expressjs.com/
21. Ajv [Електронний ресурс]. – Режим доступу: https://www.npmjs.com/package/ajv
22. Chrome DevTools [Електронний ресурс]. – Режим доступу: https://developers.google.com/web/tools/chrome-devtools
23. Back-end Architecture [Електронний ресурс]. – Режим доступу: https://www.codecademy.com/articles/back-end-architecture
24. What Do Client-Side and Server-Side Mean? | Client Side vs. Server Side [Електронний ресурс]. – Режим доступу:

https://www.cloudflare.com/learning/serverless/glossary/client-side-vs-server-side/

25. HTTP - HyperText Transfer Protocol [Електронний ресурс]. – Режим доступу: https://www.webopedia.com/TERM/H/HTTP.html
26. HTTP – Methods [Електронний ресурс]. – Режим доступу: https://www.tutorialspoint.com/http/http_methods.htm
27. Regex [Електронний ресурс]. – Режим доступу: https://www.computerhope.com/jargon/r/regex.htm
28. Node.js [Електронний ресурс]. – Режим доступу: https://nodejs.org/uk/
29. MongoDB [Електронний ресурс]. – Режим доступу: https://www.mongodb.com/nosql-explained
30. Introduction to Mongoose for MongoDB [Електронний ресурс]. – Режим доступу: https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/