МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА НАНОЕЛЕКТРОНИКИ ТА МОДИФІКАЦІЇ ПОВЕРХНІ

БАКАЛАВРСЬКА РОБОТА

зі спеціальності 153 – «Мікро- та наносистемна техніка»

на тему:

«Розробка та проектування онлайн-сервісу для діагностичного центру»

Гніденко Анастасія Вадимівна

Завідувач кафедрою	**Науковий керівник**
_____ проф. О.Д.Погребняк	_____ доц. О.В.Ющенко
«___»_____2022 р.	«___»_____2022 р.

Суми – 2022

РЕФЕРАТ

Робота складається з вступу, огляду теоретичного матеріалу, де розібрані наукові статті із проектування та розробки веб-сайтів, оригінального програмного коду висновків та списку використаних джерел.

Звіт містить 41 сторінок, 27 малюноків та 20 літературних джерел.

Об’єктом дослідження є .

Мета роботи – розгляд основних понять і структури сайту, його побудова, створення форми.

КЛЮЧОВІ СЛОВА: front-end, JavaScript, лендінг, функція, компонент, код.

ЗМІСТ

ВСТУП

Інтернет – всесвітня інформаційно-комунікативна мережа, всесвітня система обробки та зберігання інформації. Інтернет з’явився унаслідок розробок США в 70-тих роках. На основі інтернету працює WWW – глобальна, гіпертекстова інформаційна система (всесвітнє павутиння) та багато інших систем передачі даних. До 2021 року кількість користувачів перевищила 5 млрд (близько 70% населення світу).Причину розвитку інтернету можна пояснити тим, що на початку XXI століття інтернет замінив більшість класичних джерел отримання інформації.

Веб-сайт – засіб передачі інформації, сукупність веб-сторінок, на якому побудований майже весь інтернет.

У сукупності всі веб-сайти являють собою всесвітню мережу. Із його допомогою з’являється купа можливостей для способів полегшення життя населення. Через них можна навчатися, продавати свої товари чи послуги, розміщувати різноманітну інформацію, також оновлювати класичні способи отримання інформації чи інших ресурсів. При створенні сайту йому завжди дається адреса (лінк), по якому його можна знайти – це домен сайту. Вона присвоюється за допомогою IP-адреси. Сам же сайт розміщений на хостингу (на безкоштовному чи платному, це залежить від мети та намірів автора). І також бувають статичні та динамічні. Статичний веб-сайт – створений із статичних веб-сторінок, які зберігаються і завантажуються на хостинг. При потребі зміни наповнення – веб-розробник переписує сторінки та заново завантажує. Є динамічні – сайт. який постійно готовий до оновлення інформації. Інформаційне наповнення вже готове, дані завантажені в бази даних. Динамічність пояснюється тим, що на етапі створення такого сайту пишеться достатньо важкий механізм (зазвичай десятками. іноді сотнями розробників),який оновлює, зберігає. фільтрує, валідує, обробляє інформацію, надану користувачами.

У даній роботі представлена інформація по створенню динамічного веб-сайту.

1. СТРУКТУРА САЙТУ

1.1. Front-end

WEB-додаток – це клієнт-серверний додаток, у ролі клієнта завжди виступає браузер.

Front-end – це публічна сторона веб-сайтів та веб-додатків, із якою користувач контактує напряму. У нього входить відображення динамічних користувальницьких інтерфейсів, функціональних задач, обробка запитів. Front-end – це все, що бачить користувач, відкривши сайт, та все, із чим він може взаємодіяти.

Front-end розробка – розробка публічної сторони сайту, лицьова. Розробник працює повністю над функціоналом, над усіма елементами, щоб вони не тільки статично займали своє місце, а й виконували свою функцію.

Відкривши сайт, усі кнопки, картинки, форми, анімації тощо, що ви бачите – фронтенд. Для створення цих елементів використовуються три різних компонента – HTML, CSS та JavaScript. [1]

1.2. HTML

HTML (HyperText Markup Language) надає інформацію браузеру, яким має бути зміст сторінки, наприклад, «заголовок», «параграф», «список», «елемент списку».

HTML є основним будівельним блоком Інтернету. Це так бо мовний скелет сайту, його наповнення. Інші технології, крім HTML, використовуються для опису зовнішнього вигляду (обгортки) веб-сторінки (CSS) або логіки(JavaScript).

Якщо без JavaScript чи CSS створити сайт можна, то без HTML – ніколи. Дана технологія хоч і не така об’ємна, як інші в розробці сайтів, але є фундаментальною.

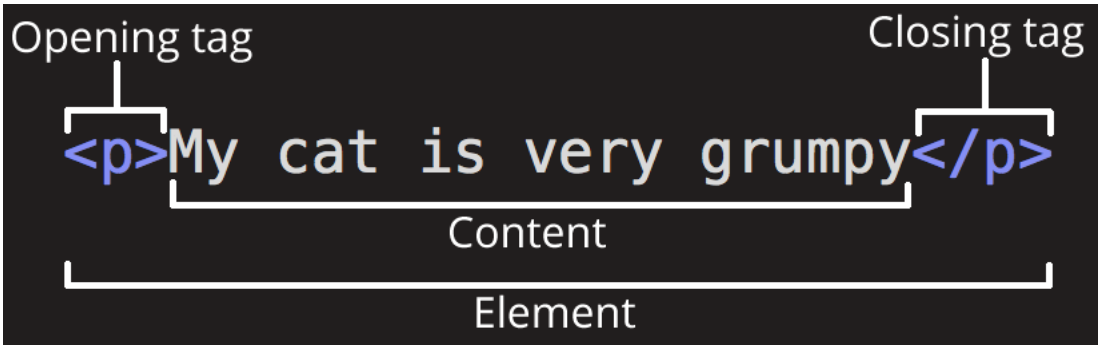

Рисунок 1.1 - Анатомія елемента HTML

Основні частини нашого елемента такі:

Відкриваючий тег: складається з назви елемента (у даному випадку p), загорнутого у відкриваючу та закриваючу **кутові дужки** . Це вказує, де починається або починає діяти елемент — у цьому випадку, де починається абзац.

1. **Закриваючий тег:** це те саме, що і початковий тег, за винятком того, що він містить *косу риску* перед назвою елемента. Це вказує, де закінчується елемент — у цьому випадку, де закінчується абзац. Відмова додати закриваючий тег є однією зі стандартних помилок початківців і може призвести до дивних результатів.
2. **Вміст:** це вміст елемента, який у даному випадку є просто текстом.
3. **Елемент:** початковий тег, закриваючий тег і вміст разом складають елемент.

Рисунок 1.2 – Атрибути

Атрибути містять додаткову інформацію про елемент, який ви не бажаєте ві

Атрибути містять додаткову інформацію про елемент, який ви не бажаєте відображати в фактичному

вмісті. Тут *ім'я*class атрибута та *значення* атрибута . Атрибут дозволяє надати елементу неунікальний ідентифікатор, який можна використовувати для націлювання на нього (та будь-які інші елементи з таким же значенням) з інформацією про стиль та іншими речами.editor-noteclassclass

Атрибут завжди повинен мати таке:

Пробіл між ним і назвою елемента (або попереднім атрибутом, якщо елемент уже має один або кілька атрибутів).

Ім'я атрибута із знаком рівності.

Значення атрибута, укладене в лапки, що відкривають і закривають. [

"Гіпертекст" відноситься до посилань, які з'єднують веб-сторінки одна з одною, або в межах одного веб-сайту, або між веб-сайтами. Посилання є основним аспектом Інтернету. Завантажуючи вміст в Інтернет і посилаючи його на сторінки, створені іншими людьми, ви стаєте активним учасником всесвітньої мережі. [2]

1.3. CSS

CSS (каскадні таблиці стилів) говорить браузеру, як відображати елементи, наприклад, «після першого параграфа відступає в 20 пікселів» або «весь текст у елементі тіла має бути темно-сірим і написаний шрифтом Verdana».

```
p {
  color: red;
}
```

Рисунок 1.3 – Приклад CSS

Наприклад, цей CSS надає текст абзацу, встановлюючи колір червоного кольору

Рисунок 1.4 – Анатомія набору правил CSS

Вся структура називається **набором правил** . (Термін *набір правил* часто називають просто *правилом* .) Зверніть увагу на назви окремих частин:

Селектор - це назва елемента HTML на початку набору правил. Він визначає елемент(и) для стилізації (у цьому прикладі <p>елементи). Щоб стилізувати інший елемент, змініть селектор.

Праворуч від властивості — після двокрапки — є **значення властивості** . Це вибирає один із багатьох можливих виглядів для даної властивості. (Наприклад, color на додаток до red.)

Інші важливі частини синтаксису:

- Окрім селектора, кожен набір правил має бути загорнутий у фігурні дужки. ({})
- У кожній декларації ви повинні використовувати двокрапку (:), щоб відокремити властивість від її значення або значень. [3]

1.4. JavaScript

JavaScript (JS) – це інтерпретована, об'єктно-орієнтована мова з функціями першого класу. Найширше застосування знаходить мову сценаріїв веб-сторінок, але також використовується і в інших програмних продуктах, наприклад, node.js або Apache. JavaScript – мова із динамічною типізацією, але на більш просунутих рівнях застосовують TypeScript, щоб статично типізувати.

Стандартом мови JavaScript є ECMAScript. Станом на 2012 рік всі сучасні браузери повністю підтримують ECMAScript 5.1. Старі версії браузерів підтримують принаймні ECMAScript 3. 17 червня 2015 року відбувся випуск шостої версії ECMAScript. Ця версія офіційно називається ECMAScript 2015, яку найчастіше називають ECMAScript 2015 чи просто ES2015. З останнього часу стандарти ECMAScript випускаються щорічно. Ця документація відноситься до останньої версії чернетки, якою є ECMAScript 2018. [4]

JavaScript і Java плутати не слід, так як це 2 взагалі різні продукти різних торгових марок, з'явилися в різний час. Вони мають різний синтаксис, різне застосування і семантику. Java використовується лише для бекенду вже багато років.

1.5. Backend

Бекенд - це серверна сторона будь-якого сайту або програми, яка відповідає за все те, що насправді відбувається за кордонами екрану. "Back-end" тому так і називається, з англійської перекладається як " ззаду". Спеціалісти даного напрямку обробляють всі запити та дії, які надсилає користувач, взаємодіючи з програмними інтерфейсами. Тобто якщо на якомусь веб-сайті користувач хоче зареєструватися, то вводячи його дані в поле вводу, яке створене безпосередньо фронтендом – дані відправляються на сервер, де вони обробляються, зберігаються в базах даних, наприклад, SQL чи Mongo, та дають команду, щоб відкрити дані сайту, як зареєстрованому користувачу. Вся ця логіка – створена серверною частиною сайту.

Сервер — це програмний або апаратний пристрій, який приймає запити, зроблені через мережу, і відповідає на них . Пристрій, який робить запит і отримує відповідь від сервера, називається клієнтом . В Інтернеті термін «сервер» зазвичай відноситься до комп'ютерної системи, яка отримує запити на веб-файли та надсилає ці файли клієнту. [7]

Бекенд-технології мають свої особливості та риси, є купа різних способів і бібліотек, кожну технологію можна описати по-різному.

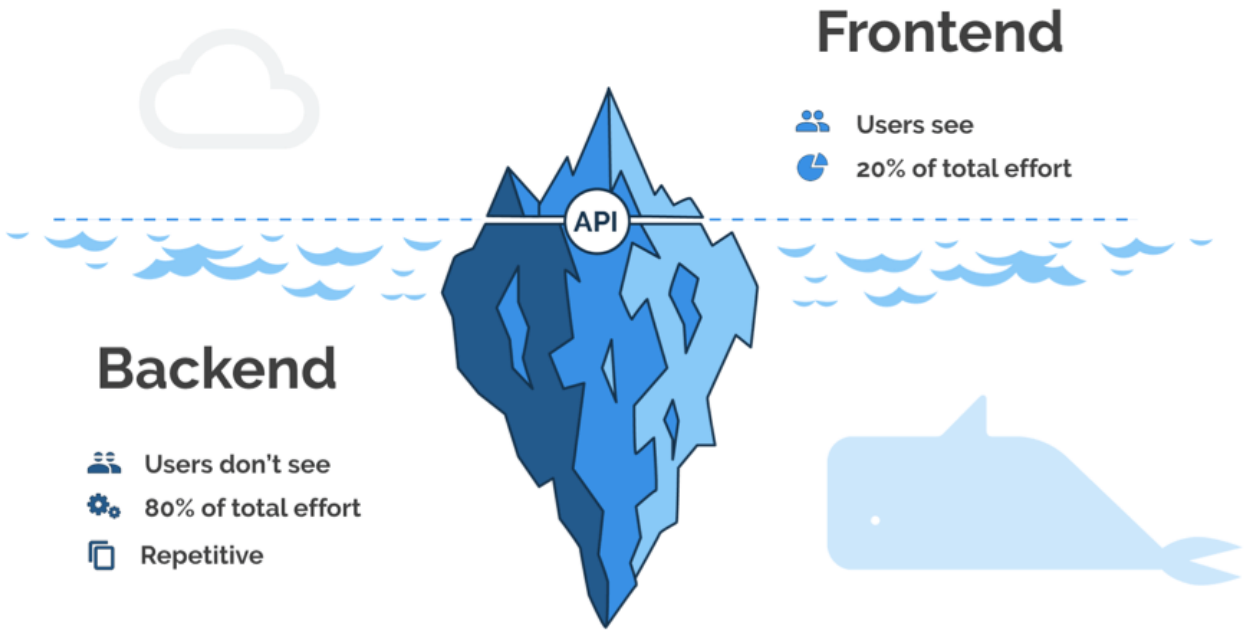

Рисунок 1.5 – Співвідношення бекенду з фронтендом

1.6. Популярні мови, які використовуються для створення програми на стороні сервера

- Java

Java – серверна мова програмування із динамічною типізацію, підтримує поліморфізм, наслідування. Відрізняється швидкістю та високим рівнем безпеки, простіше, ніж більшість інших мов, які використовуються для створення серверних програм завдяки послідовному використанню об'єктної моделі. Великий стандартний набір бібліотек класів надає Java розробникам потужні інструменти на всіх платформах.

Цю серверну мову використовували такі компанії, як LinkedIn або Uber. Він відомий переважно своєю універсальністю – його можна використовувати в різних проектах. Однак він не такий зручний для початківців, як інші популярні

серверні мови. Крім того, він має багато бібліотек з відкритим кодом, які можна використовувати для різних цілей, що може прискорити процес розробки.

Його використовують у розробці веб-сайтів, мобільних бізнес-додатків та програмного забезпечення для комп'ютерів.

- PHP

PHP — це серверна мова програмування, створена для розробки веб-сайтів. Його використовували Facebook і WordPress. Він має багато популярних фреймворків, таких як Laravel, Symfony або Codelgniter, які використовуються розробниками для створення як простих, так і складних бізнес-рішень.

PHP є технологією з відкритим кодом, тому програмісти можуть використовувати її (і її фреймворки) безкоштовно – це впливає на витрати на розробку бізнес-рішення. Він також користується великою підтримкою громади. Це, а також доступ до фреймворків, які надають розробникам корисні інструменти, дають змогу проектній команді розробити платформи за відносно короткий час.

Проте більшість веб-сайтів створені за допомогою PHP. Він використовується для розробки платформ електронної комерції, а також CMS, таких як WordPress. Він має кілька застосувань у бізнесі.

- Python

Ця мова програмування зазвичай використовується в наукових обчисленнях, науці про дані, математиці та інженерії. Він має спеціальні пакунки, такі як NumPy і SciPy, і бібліотеки, наприклад, PyTorch для цих цілей. Чи означає це, що його не можна використовувати для розробки бізнес-додатків? Зовсім ні. Він має кілька фреймворків для розробки серверів, наприклад Django, який використовувався багатьма популярними платформами, такими як Spotify або Instagram.

Python має чіткий синтаксис і інтуїтивно зрозумілий у використанні – це робить його популярним вибором для молодих розробників, які починають

кодувати. Це дозволяє створювати код, який буде легко читати тим, хто може приєднатися до проекту на пізнішому етапі розробки. Оскільки він популярний у багатьох сферах, він є однією з найбільш швидкозростаючих мов серверної системи.

Його можна використовувати як для настільних комп'ютерів, так і для розробки веб-додатків.

- Ruby

Ця письмова мова часто використовується для онлайн-додатків. Колись його використовували багато популярних соціальних мереж, таких як Airbnb і Twitter. Це хороший варіант для швидкої розробки додатків. Це оригінальна технологія, тому її можна вдосконалювати і вдосконалювати.

Велика річ у цій мові – це її люди. Вони діляться ідеями та допомагають один одному знайти рішення спільних проблем. Це дозволяє команді проекту швидко вирішувати проблеми та знаходити найкраще рішення. Ще однією потужною особливістю Ruby є його простота як бекенд-мови.

Можна використовувати для створення прототипів для веб-додатків і мобільних пристроїв.

Існує багато інших мов сервера JavaScript, відомих як мови інтерфейсу, але їх також можна використовувати для проектування сервера. Щодня розробляється багато нових мов, які будуть домінувати на ринку в майбутньому. Хорошими прикладами є Swift, мова програмування, яку Apple використовує для запуску iOS, або Golang (Go), мова низького рівня, яка швидко розвивається. Можна використовувати для розгортання частини мобільного веб-сервера.

2. ВЕБ-РОЗРОБКА

2.1. Що таке фреймворк

Коли JavaScript був запущений у 1996 році, він іноді додавав взаємодії та інтересу спілкування, які вже мали деякі сценарії. Інтернет став місцем читання. Популярність JavaScript продовжує зростати. Розробники JavaScript створили інструменти для вирішення своїх проблем і поклали їх у багаторазові ящики, які називаються бібліотеками, щоб вони могли поділитися своїми рішеннями з іншими.

Зараз JavaScript є невід'ємною частиною Інтернету, використовується на 95% усіх веб-сайтів , а Інтернет є важливою частиною сучасного життя. Користувачі пишуть статті, керують своїми бюджетами, транслюють музику, дивляться фільми та миттєво спілкуються з іншими на великій відстані за допомогою текстового, аудіо- або відеочату. Інтернет дозволяє нам робити те, що раніше було можливим лише в рідних програмах, встановлених на наших комп’ютерах. Ці сучасні, складні, інтерактивні веб-сайти часто називають веб-додатками **.**

Поява сучасних фреймворків JavaScript значно полегшила створення високодинамічних інтерактивних програм. Фреймворк — це бібліотека, яка надає уявлення про те, як створювати програмне забезпечення. Ці перспективи забезпечують передбачуваність і однорідність програми; передбачуваність дозволяє програмному забезпеченню масштабуватися до величезних розмірів, зберігаючи його; передбачуваність і ремонтопридатність важливі для продуктивності та довговічності програмного забезпечення. [8]

П'ять фреймворків JavaScript, які зараз домінують на ринку з точки зору популярності та використання:

- React
- Vue
- Angular

- Ember
- Backbone.js.

2.2. React

Безсумнівно лідер у світі JS. Ця структура JavaScript використовує реактивний підхід, а також вводить багато власних концепцій інтерфейсу веб-розробки. Щоб використовувати React, ви повинні навчитися використовувати багато інших інструментів, щоб досягти високого ступеня гнучкості при проектуванні інтерфейсів. Наприклад, ось менш вичерпний список бібліотек, які можна використовувати з React: Redux, MobX, Fluxy, Fluxible або RefluxJS. React також можна використовувати з jQuery AJAX, Get API, Superagent і Axios.

2.3. Angular

Сьогодні Angular став дуже просунутим і модульним для використання для розробки інтерфейсу.

Angular має кілька додаткових проблем. Розробники майже зобов’язані використовувати TypeScript для забезпечення безпеки типів у програмах Angular.

2.4. Vue

Концепція Vue взята з Angular і React, але Vue краще. Цього року Vue завантажили 40 мільйонів разів, і було зафіксовано лише 4 прямих експлойта. Вони всі закріплені. За допомогою Vue ви можете зберігати логіку компонентів, макет і таблиці стилів в одному файлі. React працює так само, без таблиць стилів. Vue використовує реквізити та об’єкти стану, щоб дозволити компонентам спілкуватися один з одним. Цей підхід також існував у React до того, як був прийнятий Vue. Як і Angular, Vue очікує, що ви змішуєте HTML-макети з JavaScript. Ви повинні використовувати директиви Vue, такі як v-bind або v-if, щоб вставити значення з логіки компонента в шаблон. Однією з причин розглянути можливість використання Vue замість React є бібліотека Redux, яка часто використовується у великих програмах React. Як описано в розділі React,

коли React + Redux збільшується, ви витратите багато часу, вносячи невеликі зміни в кілька програм.. [9]

3. REACT. СТВОРЕННЯ ФОРМИ ЗА ДОПОМОГОЮ REACT

3.1. Детально про React. Особливості

React.js був випущений інженером-програмістом, що працює у Facebook, Джорданом Уоці в 2011 році. React - це бібліотека JavaScript, орієнтована на створення декларативних інтерфейсів (UI) з використанням концепції, заснованої на компонентах. Він використовується для обробки шару представлення і може використовуватися для веб-додатків та мобільних додатків. Основна мета React - бути широким, швидким, декларативним, гнучким і простим. [10]

На відміну від звичайного базового JS, код в React.js маємо писати в одному файлі із розширенням JSX. Це значно полегшує роботу, не потрібно писати багато шифрів та комбінацій коду щоб зв’язати розмітку(HTML) і логіку(JS). Він називається JSX і є синтаксичним розширенням JavaScript. Рекомендуємо використовувати його з React, щоб описати, як повинен виглядати інтерфейс користувача. Після компіляції вирази JSX стають звичайними викликами функцій JavaScript та обчислюються як об'єкт JavaScript.

JSX виглядає як HTML всередині JS і образовує компонент.

3.2. Компоненти

Компоненти дозволяють розділити інтерфейс користувача на незалежні, багаторазові частини, кожна із них ізольована. Раніше компоненти можна було написати у вигляді класів і у вигляді функцій, але функціональні мали менше можливостей, таких як стан, життєвий цикл і не мали переваги, поки не вийшла новіша версія React 16.8 – тоді з’явилися так звані хуки. Із того моменту класові компоненти відійшли в сторону, так як компоненти, написані ними, були занадто важкими для читання.

Класовий компонент має такий вигляд:

```
class Clock extends React.Component {
  render() {
    return (
```

```
      <div>
        <h1>Привет, мир!</h1>
        <h2>Сейчас {this.props.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
}
```

Функціональний компонент виглядає так:

```
function Clock(props) {
  return (
    <div>
      <h1>Привет, мир!</h1>
      <h2>Сейчас {props.date.toLocaleTimeString()}.</h2>
    </div>
  );
}
```

[11]

Тут вже можна зрозуміти, чому розробники віддають перевагу функціональним компонентам. Класові ще використовують, але рідко, це так званий «легасі» код (старий).

Функціональний компонент можно об’явити як у вигляді звичайної, так і у вигляді стрілочної функції, вони майже не відрізняються.

```
const Modal = (props) => {
    return (
        <div>
            Hello World
        </div>
    )
}
```

Рисунок 3.1 – Створення компоненту за допомогою стрілочної функції

```
function MainButton (props) {
    return (

            <button
                 type={"submit"} className={'orangeButton'}  onClick={props.onPress} >
                {props.name}
            </button>
    )
}
```

Рисунок 3.2 – Створення компоненту за допомогою звичайної функції

3.3. Props

Props з англійської перекладається як «параметри».

На рисунку 3.2, де представлені прості функціональні компоненти, в якості аргументів передаються props. Оскільки кожен компонент лежить в окремому файлі сам по собі, його можна використовувати безліч разів, не потрібно переписувати код компонента, просто імпортувавши в батьківський і тегнути його. На даному уривку коду застосований компонент із назвою <Input/>, та собой представляє елемент HTML input - це звичайне поле вводу для невеликого тексту, який застосовується при реєстраціях чи коротких анкетах.

```
import React from 'react';
import Input from "../input/input";

const Registration = () => {
    return (
        <div>
            <Input/>
            <Input/>
            <Input/>
            <Input/>
            <Input/>
            <Input/>
            <Input/>
        </div>
    );
};

export default Registration;
```

Рисунок 3.3 – Використання компоненту «Input» без props

Даний код видає такий результат на екрані:

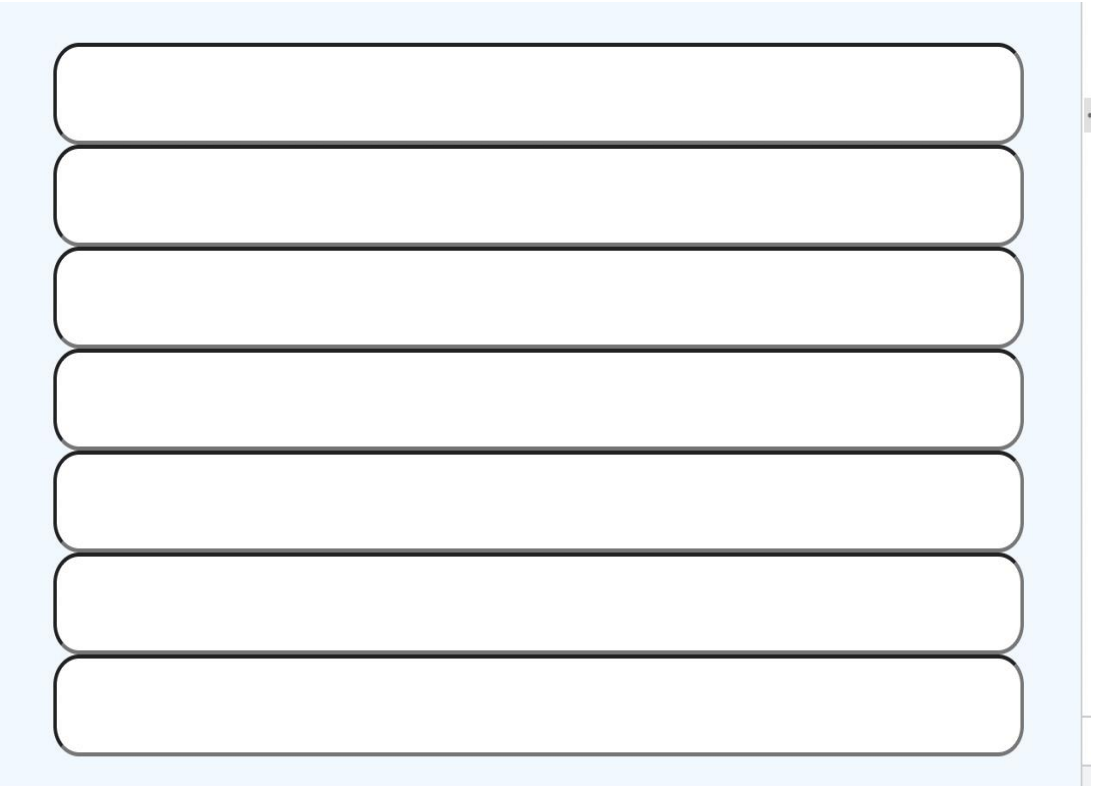

Рисунок 3.4 – Результат коду

Але тут не зрозуміло, що потрібно вводити, скільки символів, взагалі ніяких умов та допоміжної інформації. Якщо кожне поле підписувати вручну, створювати новий компонент для кожного підпису чи іншого елемента – це значно перевантажує код проекту, займає досить багато пам'яті та значно перевантажує пристрій, унаслідок чого працює повільніше, код читається дуже погано. Для вирішення цієї проблеми при створенні бібліотеки додали цю властивість, що полегшує життя і зменшує коду рази, а іноді і у десятки разів.

На даному рисунки вказані стандартні атрибути елементу input, а їх значення в фігурних дужках, що піднімається таким чином до батьківського компонента і вже звідти можна вказати їх значення.

```
const Input = (props) => {
    return (
        <div>
            <label className={'labelName'}>
                {props.labelName}
            </label>
            <div>
                <input
                    onChange={e=>props.updateData(e.target.value)}
                    type={props.type}
                    placeholder={props.placeholder}/>
            </div>
        </div>
    );
};
```

Рисунок 3.5 – Props для компоненту Input

```
import React from 'react';
import Input from "../input/input";

const Registration = () => {
    return (
        <div>
            <Input labelName="Name" type="text" placeholder="Name"/>
            <Input labelName="Last name" type="text" placeholder="Last name"/>
            <Input labelName="Patronymic" type="text" placeholder="Patronymic"/>
            <Input labelName="E-mail" type="email" placeholder="E-mail"/>
            <Input labelName="Tel" type="tel" placeholder="Tel"/>
        </div>
    );
};

export default Registration;
```

Рисунок 3.6 – Використання компоненту «Input» з props

Та з цього коду маємо ось такий результат:

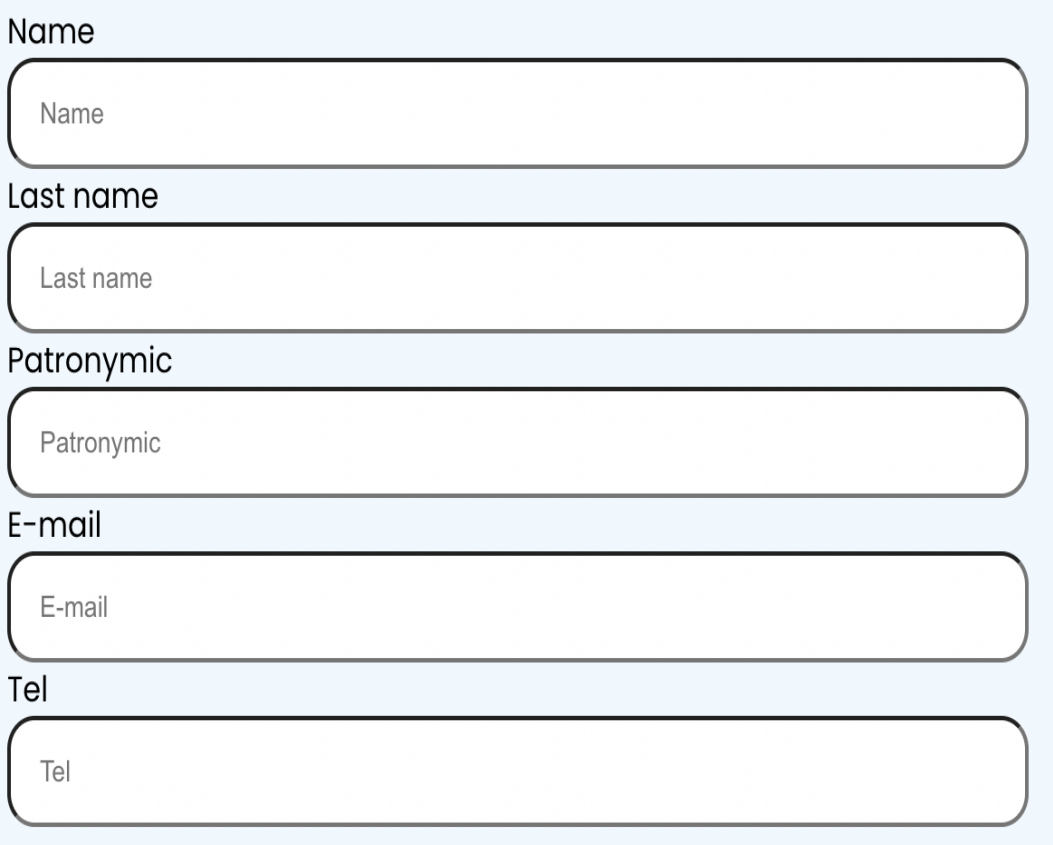

Рисунок 3.7 – Результат коду

У React HTML-елементи форми поводяться дещо на відміну від інших DOM-елементів, оскільки в елементів форми спочатку є внутрішній стан. Наприклад, у цю HTML-форму можна ввести ім'я:

```
<form>
  <label>
    Ім'я:
    <input type="text" name="name" />
  </label>
  <input type="submit" value="Надіслати" />
</form>
```

За замовчуванням браузер переходить на іншу сторінку при надсиланні HTML-форм, у тому числі й цій. Якщо вас це влаштовує, то не треба нічого змінювати, у React форми працюють як завжди. Проте, найчастіше форму зручніше обробляти за допомогою JavaScript-функції, яка має доступ до введених даних. Стандартний спосіб реалізації такої поведінки називається "керовані компоненти".

- Керовані компоненти

У HTML елементи форми, такі як <input>, <textarea> і <select>, зазвичай самі управляють своїм станом і оновлюють його, коли користувач вводить дані. У React мутабельний стан зазвичай міститься у властивості компонентів state і оновлюється лише через виклик setState().

Ми можемо скомбінувати обидва підходи та зробити стан React-компонента "єдиним джерелом правди". Тоді React-компонент буде рендерувати форму і контролювати її поведінку у відповідь на введення користувача. Значення елемента форми input у цьому випадку контролюватиме React, а сам елемент називатиметься керований компонент».

Використання керованих компонентів іноді може бути стомлюючим. Доводиться писати обробник події для кожного варіанта зміни ваших даних та проводити весь стан форми через компонент React. Це може особливо дратувати, якщо ви переносите існуючу кодову базу в React, або коли працюєте над інтеграцією програми React з іншою бібліотекою. У такій ситуації можуть стати в нагоді некеровані компоненти — альтернативна техніка реалізації введення даних у форму. [11]

4. ПРОЕКТУВАННЯ ОНЛАЙН-СЕРВІСУ ДЛЯ ДІАГНОСТИЧНОГО ЦЕНТРУ

4.1. Landing page для веб-сайту

Landing page (лендінг) – це односторінковий сайт або ж вхідна веб-сторінка для великого по структурі веб-сайту. Він використовується для маркетингових та рекламних цілей. Користувачі попадають на неї зі сторінки результатів пошуку, через рекламу, через перехід по соціальним мережам тощо.

Лендінг – це обличчя самого сайту, його мета – конвертувати користувачів в ліди; Лід (англ. Lead) – це користувач, який проявляє свій інтерес до продукту чи послуги. Якщо простими словами, то це потенціальний клієнт. Завдяки досвідченим та вмілим дизайнерам, маркетологам та СММ-спеціалістам, ці ж ліди і перетворюються на активних клієнтів.

Лендінг має бути водночас простий, але зі своєю родзинкою, має мати те, що клієнта точно зацікавить, зручний інтерфейс, та містити потрібну інформацію без перевантаження, бути ненав’язливим. За сам дизайн відповідають вище перераховані спеціалісти, а ось саме втілення цих бажань в реальність – фронтенд-розробники.

4.2. Landing page для діагностичного центру

Працюючи над даним проектом було досить багато ідей щодо дизайну сайту, зокрема лендінгу, був обраний даний дизайн, що зображений на фото.

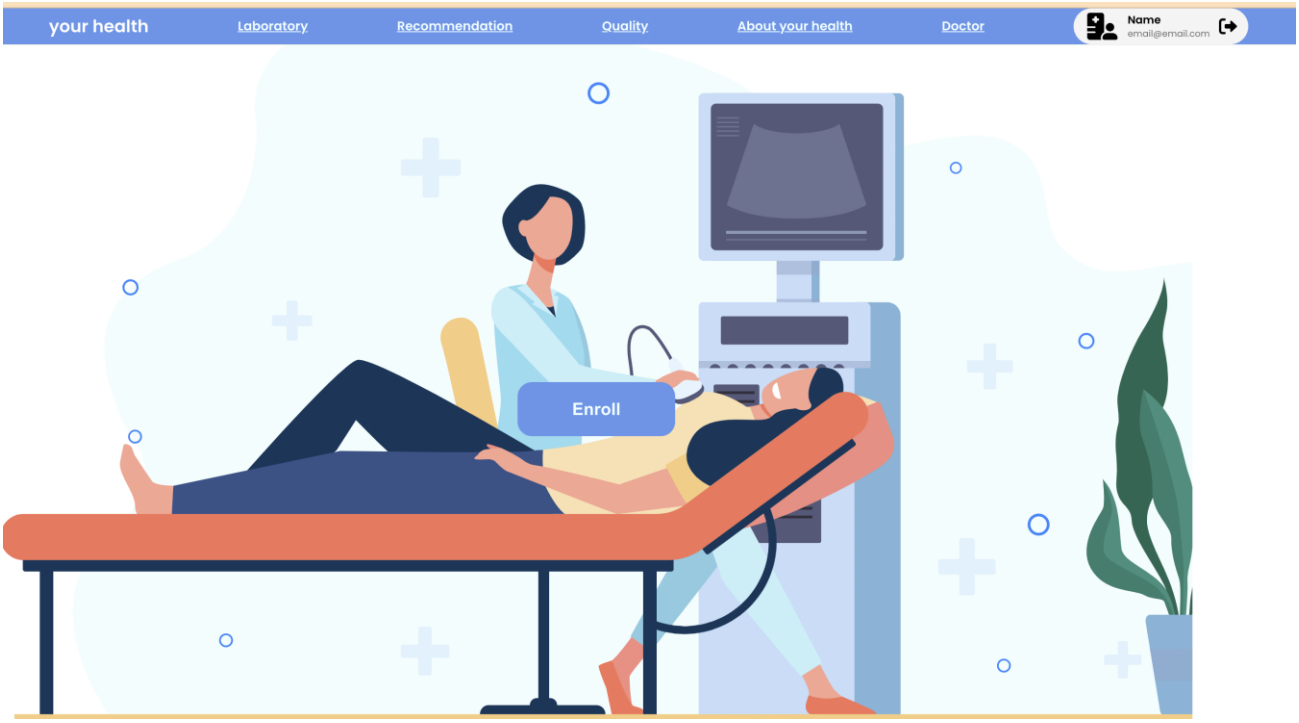

Рисунок 4.1 Landing page

Процес створення та проектування власного проекту на ReactJS занадто тонкий та має масу нюансів, так як він достатньо відрізняється від інших своїх попередників.

App.js – кореневий файл кожного проекту, створеного за допомогою цієї бібліотеки.

На даному рисунку зображена структура даного проекту.

На рядках 1-5 зображені імпорт файлів із зовнішніх модулів:

- App.css – стилі, які належать даному компоненту.
- Modal – компонент, який представляє собою модальне вікно, яке відкривається, при нажатті на кнопку.
- useState – це хук стану (додаткова можливість функціонального компоненту, яка дає змогу використовувати, як класовий компонент, якими користувалися до виходу React 16), використовується для того, щоб компонент перетворити на керований, динамічний. У даному

випадку використовується для дії появи модального вікна, а сама ж анімація прописана у його власному файлі зі стилями.

- Content – власне, контент самого веб-сайту, у якому містяться всі компоненти кожної із сторінок та компоненти, які їх створюють.
- Header – це та фіксована частина лендінгу, на якій розташовані розділи сайту, його заголовок, відображення авторизованого облікового запису:

Рисунок 4.2 - Header

Клас “background” – відображає фон лендінгу.

Тег “button” – відображає кнопку. Текст між даним парним тегом – контент самої кнопки, тобто її назва. У фронтенді є безліч інтерфейсів, побудованих на подіях “event” – представляють собою будь-яку подію, яка відбувається в DOM (об'єктна модель документа) є програмним інтерфейсом для HTML, XML і SVG документів. DOM – структурована модель документа у вигляді дерева.

У кнопці може відбуватися лише одна подія – onClick, що і відображена на даному скриншоті. Вона відбувається, як уже і зрозуміло в момент натискання на кнопку. onClick={() => setModalActive(true)} – описує стрілочною функцією, що після натискання на кнопку відбувається подія і модальне вікно, що за умовчуванням закрите - описується хуком useState.

const [modalActive, setModalActive] = useState(false) – описується початковий стан модального вікна.

```
import './App.css';
import Modal from "./components_ui/modal/modal";
import {useState} from "react";
import Content from "./layout/content/content";
import Header from "./layout/header/header";

function App() {
    const [modalActive, setModalActive] = useState( initialState: false)
  return (
    <div className="background">
        <Header/>
      <button className={'open-btn'} onClick={() => setModalActive( value: true)}> Enroll</button>
      <Modal active={modalActive} setActive={setModalActive}>
        <Content closeModal={() => setModalActive( value: false)}/>
      </Modal>
    </div>
  );
}

export default App;
```

Рисунок 4.3 App Component

4.3. Структура побудови модального вінка

Структура побудови коду модального вікна має такий вигляд:

Перші 2 рядка коду відповідають за імпорт необхідних модулів для правильної побудови компонента, а імпортувати можна лише тоді, якщо всередині компонента, який потрібно експортувати, є модуль експорту. В інакшому випадку IDE не бачить цей файл, немає в полі видимості.

На 4 рядку об’явлений компонент за допомогою стрілочної функції, у якості аргументів із батьківського компонента приймає active – стан модального вікна, setActive – функцію, що його змінює стан та children, що передані як props.

Усередині батьківського розташований ще один, який керує стилями, за допомогою якій і з’являється контент у модальному вікні – усе це відбувається за умовою виконання події onclick.Умова створена за допомогою тернарного оператора: «якщо active ? тоді відкрито клас modal__content active : інакше modal__content»

```
import React from "react";
import './modal.css'

const Modal = ({active, setActive, children}) => {
    return (
        <div className={active? 'modal active': 'modal'} onClick={()=> setActive(true)}>
            <div className={active? 'modal__content active' : 'modal__content' } onClick={e=> e.stopPropagation()}>
                {children}
            </div>
        </div>
    )
}

export default Modal
```

Рисунок 4.4 Модальне вікно

Уже відомо, що кожен елемент може мати класи, що описують стилі, але в нашому головному контейнері модального вікна трішки гнучкіша ситуація:

className={active? 'modal active': 'modal'}

Оскільки, div – це елемент HTML, то записати сам JavaScript потрібно у фігурних дужках, як і представлено. Стан «active» - має логічний тип даних Boolean, тобто або “так”, або “ні”, тому дані умови означають “якщо модальне вікно – активне, то присвоюємо стилі класу «modal active», якщо ні – «modal» ”. Тепер модальне вікно має змогу і відкриватися, і закриватися.

Children – це props за умовчуванням, якщо використовуємо слоти, як у даному випадку. Майже повністю у коді цього проекту використовується наслідування – один компонент в одному, як в матрьошці. Реакт має занадто потужну модель композиції, це є одним із принципів ООП (об’єктно-орієнтоване програмування). Так як деякі компоненти не можуть знати заздалегідь свої дочірні компоненти, ми використовуємо {props.children}. У даному випадку як дочірній компонент можна помістити будь який, але в батьківському компоненті потрібно використати замість одиночного тега – парний, та всередину помістити те, що потрібно відобразити:

```
<Modal active={modalActive} setActive={setModalActive}>
  <Content closeModal={() => setModalActive( value: false)}/>
</Modal>
```

Рисунок 4.5 Дочірній компонент

В даній частині коду як батьківський компонент – використали <Modal/>, як дитину - <Content/>. Компонент <Content/> - як і говорить назва сама за себе, представляє собою контент, у нашому випадку це форма реєстрації та вибір послуги і запис на неї, докладніше буде описано у наступному підрозділі.

4.4. Реєстрація

Після нажаття на кнопку відкривається модальне вікно із усіма необхідними полями для реєстрації:

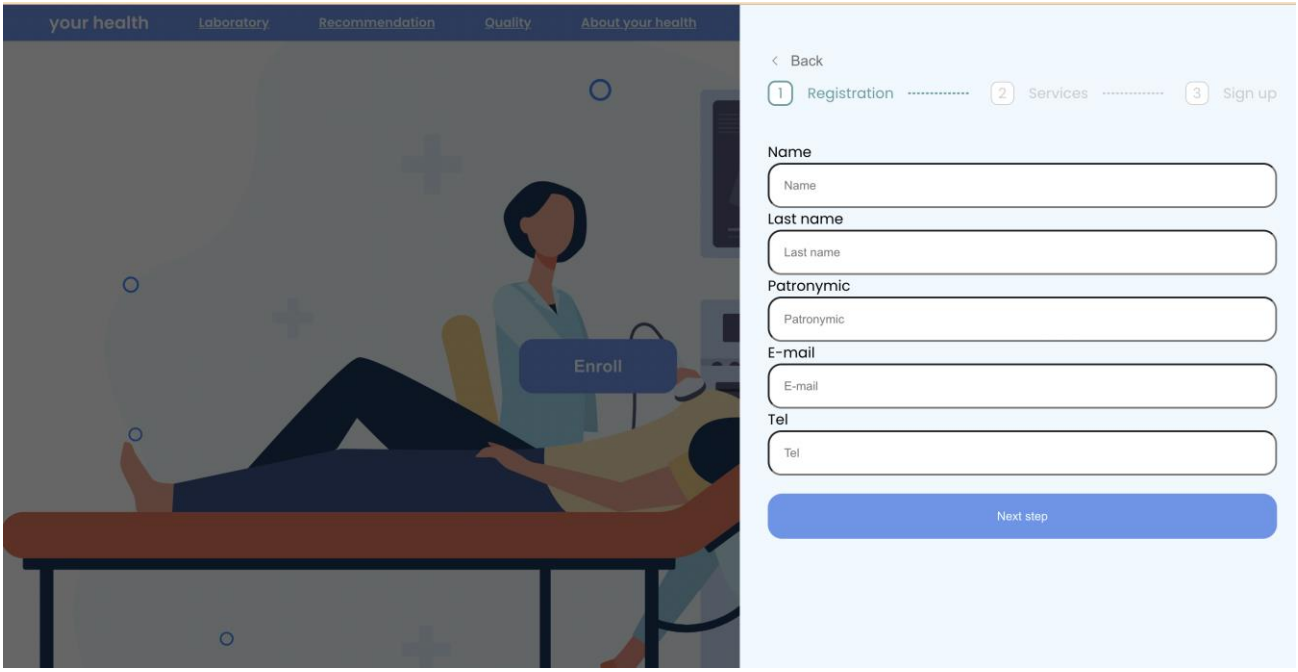

Рисунок 4.6 – Форма реєстрації

На даному кроці потрібно заповнити усі поля для реєстрації, щоб ідентифікувати користувача та створити власну історію записів та консультацій.

Для побудови реєстрації був використаний лише один компонент <Input/> із різними заголовками – це було створено за допомогою одних із особливостей даної бібліотеки – props. Може звучати незвичайно чи бути тяжко для розуміння, але це значно полегшує і скорочує код, адже не потрібно усе раз за разом дублювати.

Код цього компоненту має саме такий вигляд:

```
import React from 'react';
import './input.css';

const Input = (props) => {
    return (
        <div>
            <label className={'labelName'}>
                {props.labelName}
            </label>
            <div>
                <input
                    onChange={e=>props.updateData(e.target.value)}
                    type={props.type}
                    placeholder={props.placeholder}/>
            </div>
        </div>
    );
};
```

Рисунок 4.7 – Input component

Заголовок даного поля записано із використанням елементу <label/> HTML, даний тег застосовується попарно, заголовок передаємо із батьківського компонента, достатньо лише помістити у контент елемента (між його тегами), обернути у фігурні дужки, бо це є вираженням JavaScript, а його ми завжди заключаємо у дужки в компонентах React.

onChange – це стандартна подія в полях вводу input. У даному випадку це подія вводу тексту у це поле. {e=>props.updateData(e.target.value)} – функція, яка

виповнюється, якщо дана подія відбувається. Аргументом приймається e(event) – він стандартний для подій DOM. (e.target.value) потрібен, щоб «витягнути» дані із цього поля і в подальшому їх обробляти, відправляти на сервер тощо. Якщо це не застосувати, то даними буде неможливо заповнити це поле.

Батьківський компонент, із якого передаються дані, має такий вигляд:

```
import React from 'react';
import Input from "../input/input";

const Registration = () => {
    return (
        <div>
            <Input labelName="Name" type="text" placeholder="Name"/>
            <Input labelName="Last name" type="text" placeholder="Last name"/>
            <Input labelName="Patronymic" type="text" placeholder="Patronymic"/>
            <Input labelName="E-mail" type="email" placeholder="E-mail"/>
            <Input labelName="Tel" type="tel" placeholder="Tel"/>
        </div>
    );
};

export default Registration;
```

Рисунок 4.8 – Registration component

Для <input type=”E-mail”/> та <input type=”Tel”/> є додаткові особливості та атрибути. Для страхування того, що користувач введе там коректну інформацію,

Існує валідація. Поле можно провалідувати, застосувавши регулярні вираження.

```
const validateEmail = (email) => {
  return String(email)
    .toLowerCase()
    .match(
      /^(([^<>()[\]\\.,;:\s@"]+(\.[^<>()[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/
    );
```

}; - так виглядає код для валідації паролю.

А такий вигляд має валідація для номеру телефону:

```
const validatetel= (tel=> {
  return String(tel)
    .toLowerCase()
    .match(
  ^(\+?)([0-9 ]*)([0-9 ])*$|(^ *$)     );
```

Вони завжди мають один і той же вигляд для цих типів полей, змінюються. Якщо набраний текст не відповідає умовам регулярного вираження, то далі текст не відправляється, не проходить реєстрація. Це створено для того, щоб зменшити потік некоректної інформації.

4.5. Послуги

Після першого кроку – реєстрації переходить на другий крок, на другу сторінку. Це сторінка із переліком послуг, які може надати даний діагностичний центр.

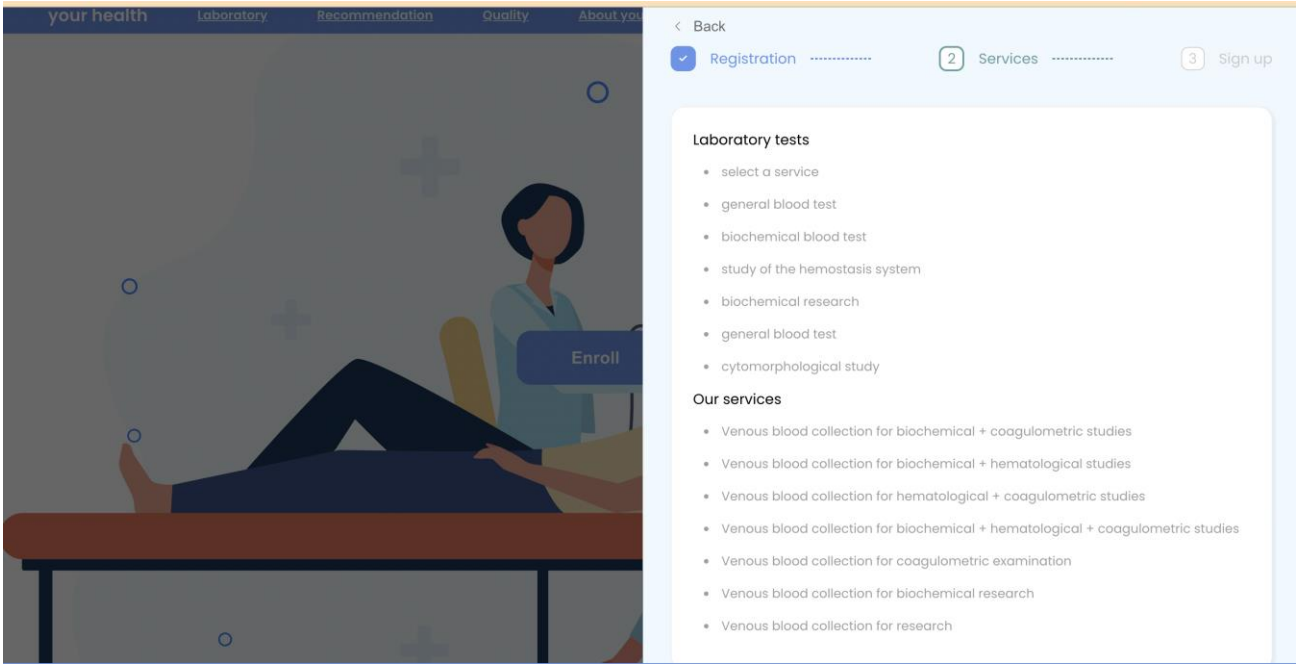

Рисунок 4.9 – Список послуг

Кожна людина зобов’язана піклуватися за своє здоров’я, одним із способів є лабораторні дослідження. За допомогою них можна виявити патологічні зміни чи хвороби на клітинному рівні. Це те, що не помітить жоден терапевт чи кардіолог. Завжди пацієнтів направляють на дослідження аналізів, щоб виявити точний діагноз та призначити лікування.

Головна мета цього центру – забезпечити людей сучасною, точною та якісною лабораторною діагностикою. В центрі працюють висококваліфіковані лікарі та лаборанти, які працюють на найсучаснішому німецькому обладнані. Саме це забезпечує грамотний підхід та точні результати.

Від пацієнта потребується відповідальність та дотримання усіх правил та рекомендацій для підготовки до дослідження.

Перед аналізом крові потрібно дотриматися цих рекомендацій:

- важливою умовою для лабораторних досліджень є здача крові натщесерце — 6 -12 годинний період голодування.
- В день дослідження допустимо вживання невеликої кількості води.
- за 6 — 12 годин до дослідження слід виключити прийом алкоголю, куріння, прийом їжі, обмежити фізичну активність.
- виключити прийом ліків, якщо відмінити прийом ліків неможливо, необхідно проінформувати про це лабораторію.
- дітей до 5 років, перед здачею крові, бажано поїти кип’яченою водою (порціями до 150–200 мл., протягом 30 хвилин).
- для грудних дітей — перед здачею крові витримати максимально можливу паузу між годуваннями.[12]

А ось так цей перелік послуг бачить браузер, перед тим, як відобразити на веб-сторінці:

```
const analysisMock = [
    {id: "11", displayName: "select a service"},
    {id: "1", displayName: "general blood test"},
    {id: "2", displayName: "biochemical blood test"},
    {id: "3", displayName: "study of the hemostasis system"},
    {id: "4", displayName: "biochemical research"},
    {id: "5", displayName: "general blood test"},
    {id: "6", displayName: "cytomorphological study"},
]

const serviceMock = [
    {id: "11", displayName: "Venous blood collection for biochemical + coagulometric studies"},
    {id: "1", displayName: "Venous blood collection for biochemical + hematological studies"},
    {id: "2", displayName: "Venous blood collection for hematological + coagulometric studies"},
    {id: "3", displayName: "Venous blood collection for biochemical + hematological + coagulometric studies"},
    {id: "4", displayName: "Venous blood collection for coagulometric examination"},
    {id: "5", displayName: "Venous blood collection for biochemical research"},
    {id: "6", displayName: "Venous blood collection for research"}
]
```

Рисунок 4.10 – Об’єкти послуг

4.6. Запис на послуги

Вікно для запису включає до себе випадаючий список:

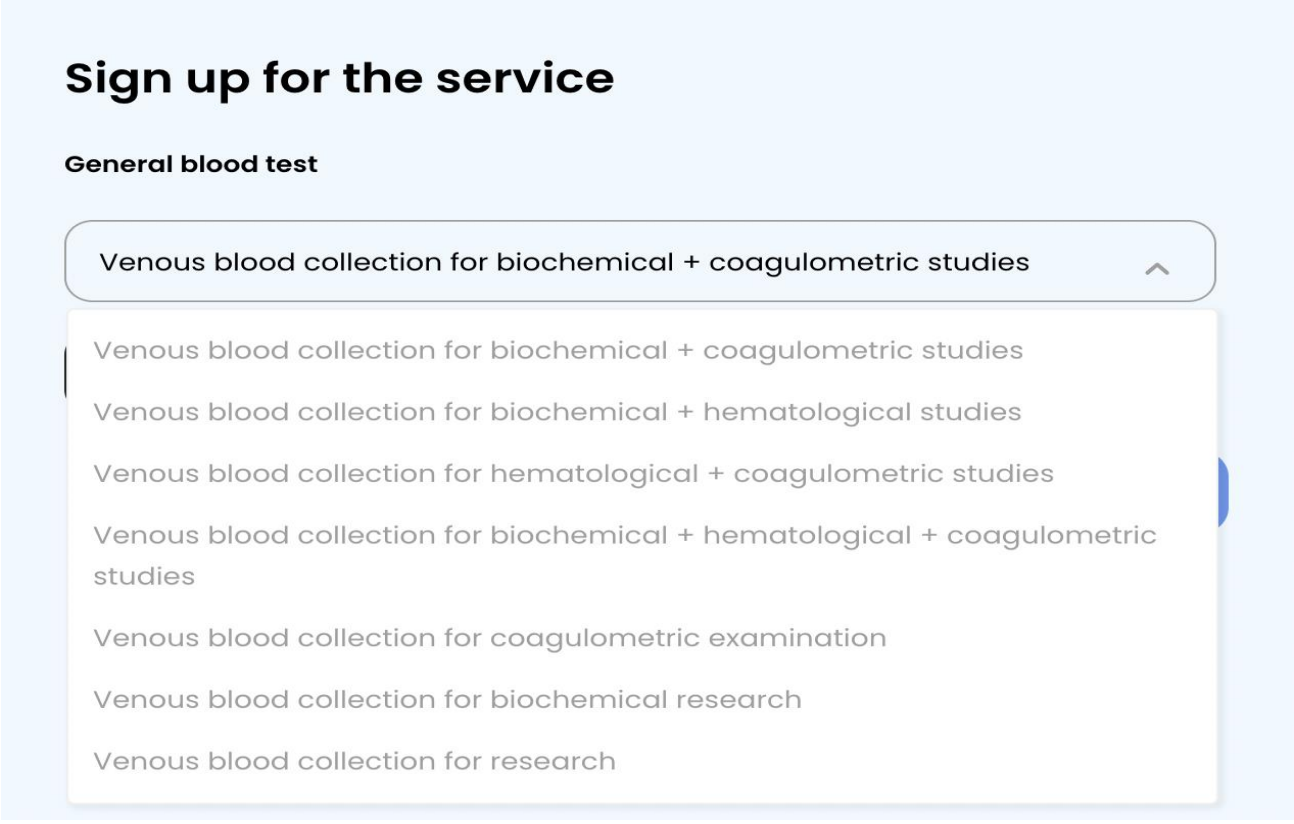

Рисунок 4.11 – Випадаючий список

За допомогою нього можна обрати послугу, на котру потрібно записатися.

За кулісами він виглядає ось так:

```
const dataMock = [
    {id: "11", displayName: "Venous blood collection for biochemical + coagulometric studies"},
    {id: "1", displayName: "Venous blood collection for biochemical + hematological studies"},
    {id: "2", displayName: "Venous blood collection for hematological + coagulometric studies"},
    {id: "3", displayName: "Venous blood collection for biochemical + hematological + coagulometric studi
    {id: "4", displayName: "Venous blood collection for coagulometric examination"},
    {id: "5", displayName: "Venous blood collection for biochemical research"},
    {id: "6", displayName: "Venous blood collection for research"}
]

const SignUp = (props) => {
    const [selected, setSelected] = useState(dataMock[0])
    const [date, setDate] = useState(new Date())
    return (
        <div>
            <h2>Sign up for the service</h2>
            <h5>General blood test</h5>
            <Dropdown selected={selected} setSelected={setSelected} mockk={dataMock}/>
```

Рисунок 4.11 – Об’єкт випадаючого списку

```
function DropDown ({selected, setSelected, updateMainSkill}, props)  {
     const [isActive, setIsActive] = useState( initialState: false);

    return (
        <div className={'dropdown'} >
            <div className={'dropdown-btn'} onClick={e => setIsActive(!isActive) }>
                <div className={'selectedItem'} >{selected.displayName}</div>
                { isActive ? <UpIcon/> : <DownIcon/> }

            </div>
            { isActive && (
                <div className={'dropdown-content'} >
                    {props.service.map(selected => (
                        <div onClick={e => {
                            e.preventDefault()
                            setSelected(selected);
                            setIsActive( value: false);
                            updateMainSkill(selected.uid)
                        }}
                             className={'dropdown-item'} >
                            {selected.displayName}
                        </div>
                    ))}
                </div>
                )}

        </div>
    );
}
```

Рисунок 4.12 – DropDown component

У випадаючий список передається масив об’єктів із двома ключами: id I displayName. Id – обов’язковий елемент, так як кожен має мати свій унікальний ідентифікатор, який потім буде використаний як ключ. Його не бачить користувач, використовується для обробки інформації на сервері. Це задля того, що кожен елемент був унікальним. Він перебирається методом перебору масивів map та заповнюється динамічно – кількість залежить від кількості полей у вихідному масиві.

{selected.displayName} – це назва опції із випадаючого меню.

Таким чином, ми не пишемо статично Analys усередині компоненту <DropDown/>

Момент відкривання і закривання списку опцій написаний лише за допомогою логічних операторів та стилів CSS. Тобто, при кліку список відкривається і закривається.

```
.dropdown-content {
    min-height: 48px;
    width: 100%;
    border-radius: 16px;
    border: 1px solid #f0f0f0;
    background: white;
    position: absolute;
    top: 110%;
    box-shadow: 2px 2px 3px 1px rgb(0 0 0 / 5%);
}

.dropdown-item {
    cursor: pointer;
    transition: all 0.2s;
    margin: 12px;
```

```
    font-size: 14px;
    font-weight: 400;
    line-height: 24px;
    letter-spacing: 0em;
    text-align: left;
    color: #9F9F9F;
}
```

Заповнивши усі поля, пройшовши 3 кроки – ви записані до центру на обрану послугу.

За день до візиту у лабораторії надходить смс-нагадування із адресою, номером кабінету та ціною.

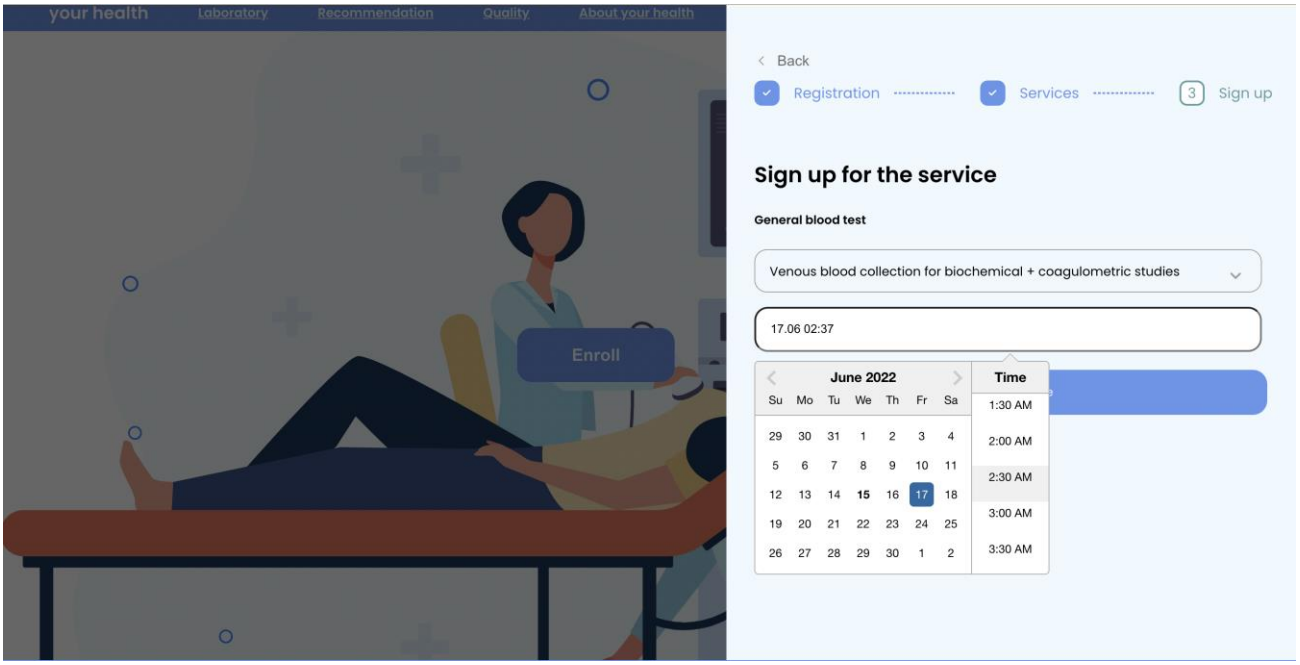

Рисунок 4.13 – Дата та час запису

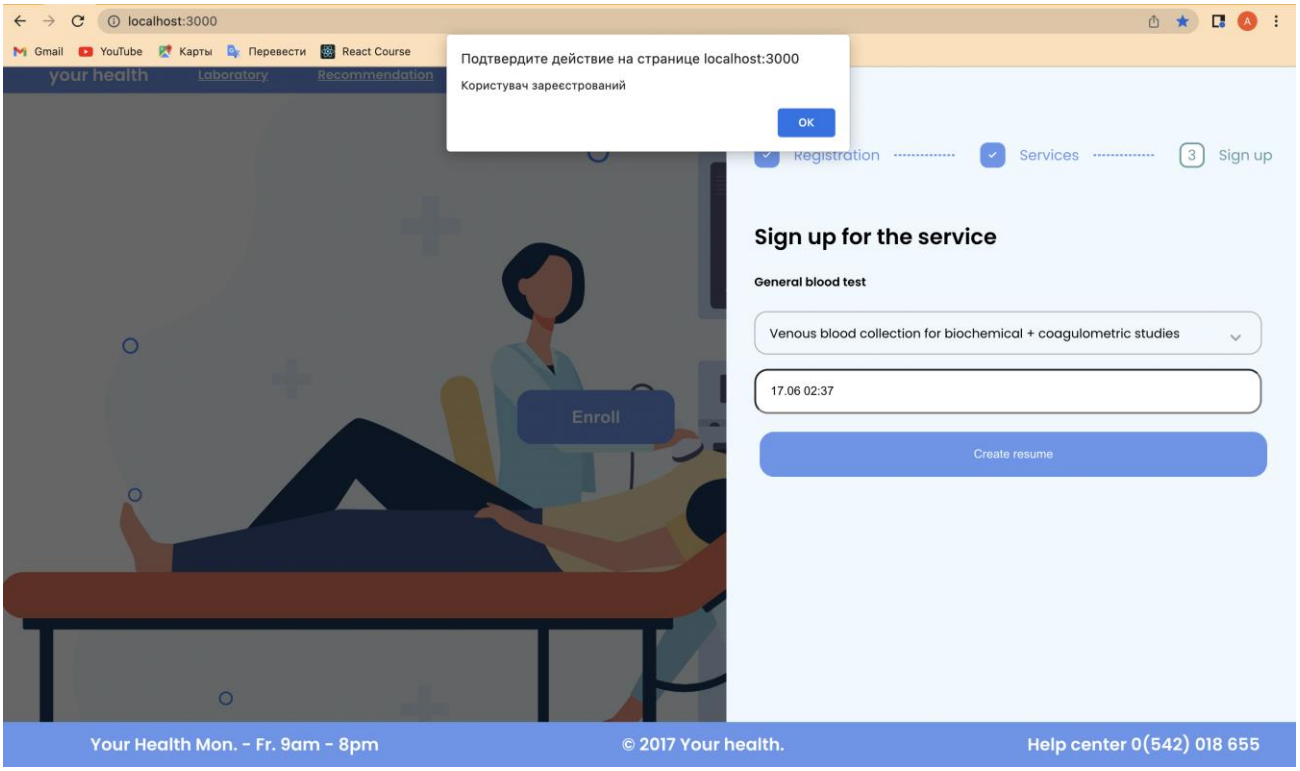

Рисунок 4.14 – Підтвердження запису

ВИСНОВКИ

Дана робота присвячується одній із найпопулярніших технологій сьогодення – веб розробці. За останні 20 років вона захопила повністю весь світ, усі сфери: від кулінарії до будівництва, від навчання до медицини. За останні 2 роки попит на клінічні дослідження виріс у кілька разів через пандемію коронавірусу – тому розробка такого сайту не викликає сумнівів.

На практичному дослідженні і роботі даного завдання було використано достатньо технологій, більшість із них обов’язкові для нормального функціонування та роботи сайту. React – найзручніша та найперспективніша технологія для створення користувальницьких інтерфейсів, це зумовлено конпонентно-функціональним підходом та props. Слід відмітити тонкий архітектурний підхід за допомогою слотів, що значно зменшило об’єм зайвого коду, при цьому логіка та вид змін не зазнали.

Велику роль зіграв CSS – надав приємний зовнішній вид усій структцрі даного сайту. Також у ході проектування розібрали важливу роль лендінгу – цілі, доступність, ненав’язливість.

Дане дослідження дозволило більш стисло пояснити та зобразити побудову сайтів, розказати про переваги даної бібліотеки у створенні клієнтської частини сторінки. Але для більш детального пояснення потребується задіення досить великого стеку технологій та введення додаткових параметрів і систем, щоб зрозуміти повністю роботу даної системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Всесвітнє павутиння [Електронний ресурс]
2. MDN HTML [Електронний ресурс] https://developer.mozilla.org/en-US/docs/Web/HTML
3. MDN CSS [Електронний ресурс] https://developer.mozilla.org/en-US/docs/Web/CSS
4. MDN JS [Електронний ресурс] https://developer.mozilla.org/en-US/docs/Web/JavaScript
5. Java Oracle Guide [Електронний ресурс] https://docs.oracle.com/en/database/oracle/oracle-database/12.2/jjdev/Java-overview.html#GUID-7E0D6882-64EA-43C2-B5ED-E89CD62B8178
6. WEZOM [Електронний ресурс] https://wezom.com.ua/blog/chto-takoe-back-end-razrabotka
7. Computer Hope [Електронний ресурс] https://www.computerhope.com/jargon/s/server.htm
8. MDN Framework [Електронний ресурс] https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Introduction
9. FreeCodeCamp [Електронний ресурс] https://www.freecodecamp.org/news/complete-guide-for-front-end-developers-javascript-frameworks-2019/
10. Flatlogic [Електронний ресурс] https://flatlogic.com/blog/what-is-react/
11. ReactJS [Електронний ресурс] reactjs.org/docs/state-and-lifecycle.html
12. Державний експертний центр МОЗ України
13. Responsive Web Design with HTML5 and CSS: Develop future-proof responsive websites using the latest HTML5 and CSS techniques, 3rd Edition
14. Deep Learning with JavaScript. Neural networks in TensorFlow.js

15. React 17. Разработка веб-приложений на JavaScript - Владимир Дронов https://akonit.net/353394-react-17-razrabotka-veb-prilozhenij-na-javascript

16. Full Front-End & Back-End Development (Bundle)

17. Murach's HTML and CSS

18. Best Book to Learn Java: Java: The Complete Reference

19. Learn Python 3 the Hard Way

20. The Internet Book: Everything You Need to Know about Computer Networking and How the Internet Works