Кафедра  *програмного забезпечення*

ПОЯСНЮВАЛЬНА ЗАПИСКА

до бакалаврської кваліфікаційної роботи на тему:

Система пошуку геолокації та інформації про автоматичні дефібрилятори

*Студент групи ПЗ-42 Бойчук Т.В.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Керівник проекту** |  | ( | *доц. Коротєєва Т.О.* | ) |

(підпис)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Консультанти** |  | ( | *доц. Воскресенська Т. І.* | ) |

(підпис)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | ( |  | ) |

(підпис)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | ( |  | ) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | ( |  | ) |

|  |  |
| --- | --- |
| **Завідувач кафедри** | *проф. Яковина В. С.* |

(підпис)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| “ |  | ” |  | 20 |  | р. |

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут ІКНІ

Кафедра Програмне забезпечення

Спеціальність 121 «Інженерія програмного забезпечення»

“ЗАТВЕРДЖУЮ”

Завідувач

кафедри програмного забезпечення

*проф. Яковина В. С.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| « |  | » |  | 201 |  | р. |

**ЗАВДАННЯ**

|  |  |  |  |
| --- | --- | --- | --- |
| **на кваліфікаційну роботу (проект) студента** **групи** | ПЗ-42 | **ОКР** | бакалавр |

Бойчука Тараса Віталійовича

(прізвище, ім’я, по-батькові)

1. Тема роботи (проекту) Система пошуку геолокації та інформації про автоматичні дефібрилятори

(у разі виконання комплексної роботи в дужках вказується “комплексна робота(проект)” )

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| затверджена наказом по університету від | « | *24* | » | *квітня* | 20 | *20* | р. № | *1239-4-08* |

2. Термін подання студентом закінченої роботи (проекту) *1 червня 2020р.*

3. Вхідні дані для роботи (проекту) постановка задачі, первинні вимоги до проекту, матеріали з інтернету

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

Вступ

1. Небезпека проблеми раптової серцевої смерті, та важливість екстреної допомоги з використанням дефібрилятора.
2. Постановка задачі розробки системи пошуку геолокації та інформації про автоматичні

дефібрилятори.

1. Архітектура та проектування системи для пошуку геолокації та інформації про автоматичні

дефібрилятори.

1. Реалізація та тестування веб-системи.
2. Економічне обґрунтування доцільності роботи.

Висновки

5. Перелік графічного матеріалу діаграма прецедентів, діаграма розгортання, фізична модель бази даних,

діаграма послідовності.

6. Перелік програмних продуктів, які належить використати в процесі розроблення роботи (проекту) Node.js, React,

MongoDB, Formik, Mapbox, Passport.js.

7. Консультування роботи (проекту), із зазначенням розділів роботи

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Розділ | Консультант | Завдання видав | | Завдання прийняв | |
| підпис | дата | підпис | дата |
| *Економіка* | *доц. Воскресенська Т. І* |  | *04.04.2020* |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

8. Дата, коли видано завдання 4 квітня 2020 р.

Керівник *доц. Коротєєва Т.О.*

(підпис)

Завдання прийняв до виконання Бойчук Т.В.

(підпис)

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів роботи (проекту) | Термін виконання етапів роботи (проекту) | Примітка |
|  | Аналіз програмних засобів для пошуку геолокації на інформації про автоматичні дефібрилятори які наявні на ринку | 04.04.2020 - 11.04.2020 |  |
|  | Створення специфікації вимог | 11.04.2020 – 13.04.2020 |  |
|  | Вибір шляхів та засобів розробки | 13.04.2020 - 15.04.2020 |  |
|  | Проектування програмної системи | 15.04.2020 - 23.04.2020 |  |
|  | Розробка програмного забезпечення | 23.04.2020 - 17.05.2020 |  |
|  | Тестування програмної системи | 17.05.2020 - 20.05.2020 |  |
|  | Формування пояснювальної записки | 20.05.2020 - 1.06.2020 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Студент Бойчук Т.В.

(підпис)

Керівник *доц. Коротєєва Т.О.*

(підпис)

Видавництво Львівської політехніки. Наклад 5000. Зам. 180288. 2018

**АНОТАЦІЯ**

Метою бакалаврської кваліфікаційної роботи було створення програмної системи для пошуку геолокації та інформації про зовнішні дефібрилятори.

Програма розроблена за принципами клієнт-серверної архітектури. Серверна частина була розроблена на платформі Node.js з використанням таких бібліотек як Mongoose, Passport.js та Express.js. Клієнтська частина була розроблена за допомогою бібліотеки React, та великої кількості суміжних пакетів для роботи з формами та картами.

Розроблений засіб повністю адаптивний, та володіє зручним та мінімалістичним інтерфейсом. В ньому також наявна система ролей, в якій неавторизовані користувачі мають доступ тільки до функціоналу для пошуку пристроїв, інформації та шляху до них. Користувачі які авторизовані як модератор або адміністратор мають змогу додавати, редагувати чи видаляти інформацію про пристрої з системи.

До системи створено специфікацію вимог. Проектування та розробка програмного продукту здійснювалась згідно вимог описаних у специфікації та спираючись аналізу предметної області.

Робота містить реалізацію програмної системи та пояснювальну записку. Загальний розмір записки 92 сторінки.

**ABSTRACT**

The purpose of the bachelor qualification work was to create software system for finding geolocation and detailed information of automated external defibrillators

The program was developed on the principles of client-server architecture. The server part was developed on the Node.js platform using libraries such as Mongoose, Passport.js and Express.js. The client part was developed using the React library, and a large number of related packages for working with forms and MapboxMaps.

The developed software is completely adaptive, and has a convenient and minimalist user interface. It also has a role system in which unauthorized users have access only to the functionality for searching information and paths to automated external defibrillators. Users who are authorized as a moderator or administrator can add, edit, or delete information about defibrillators from the system.

A requirements specification has been created for the system. The design and development of the software product was carried out according to the requirements witch are described in the specification and based on the analysis of the subject area.

The work includes software implementation and system explanatory note. The total volume of pages of system explanatory note is 92.

**ЗМІСТ**

[**ВСТУП** 9](#_Toc41565887)

[**РОЗДІЛ 1 Небезпека проблеми раптової серцевої смерті, та важливість екстреної допомоги з використанням дефібрилятора** 12](#_Toc41565888)

[**1.1.** **Раптова серцева смерть** 12](#_Toc41565890)

[**1.2.** **Важливість першої невідкладної допомоги** 12](#_Toc41565891)

[**1.3.** **Автоматичний зовнішній дефібрилятор** 14](#_Toc41565892)

[**1.4.** **Принципи роботи та можливості існуючих систем для пошуку АЗД** 17](#_Toc41565893)

[**Висновки до розділу 1** 20](#_Toc41565894)

[**РОЗДІЛ 2 Постановка задачі розробки системи пошуку геолокації та інформації про автоматичні дефібрилятори** 21](#_Toc41565895)

[**2.1.** **Загальна постановка завдання** 21](#_Toc41565897)

[**2.2.** **Специфікація вимог до програмного продукту** 21](#_Toc41565898)

[**2.2.1.** **Призначення та мета** 21](#_Toc41565899)

[**2.2.2.** **Продукти-аналоги** 21](#_Toc41565900)

[**2.2.3.** **Загальний опис** 22](#_Toc41565901)

[**2.2.4.** **Характеристики системи** 23](#_Toc41565902)

[**2.2.5.** **Вимоги зовнішніх інтерфейсів** 28](#_Toc41565903)

[**2.2.6. Інші нефункціональні вимоги** 29](#_Toc41565904)

[**2.2.7.** **Атрибути якості програмного продукту** 30](#_Toc41565905)

[**2.3.** **Вибір засобів для виконання завдання** 30](#_Toc41565906)

[**Висновок до розділу 2** 31](#_Toc41565907)

[**РОЗДІЛ 3 Архітектура та проектування системи для пошуку геолокації та інформації про автоматичні дефібрилятори** 32](#_Toc41565908)

[**3.1** **Проектування поведінки розроблюваної системи** 32](#_Toc41565910)

[**3.2. Загальна архітектура системи** 34](#_Toc41565911)

[**3.3. Проектування бази даних** 36](#_Toc41565912)

[**3.4. Проектування серверної частини** 38](#_Toc41565913)

[**3.5. Проектування клієнтської частини** 39](#_Toc41565914)

[**Висновок до розділу 3** 42](#_Toc41565915)

[**РОЗДІЛ 4 Реалізація та тестування веб-системи** 43](#_Toc41565916)

[**4.1. Налаштування та реалізація серверної частини** 43](#_Toc41565918)

[**4.2. Опис роботи системи та програмних рішень на клієнтській частині** 43](#_Toc41565919)

[**4.3. Тестування програмного забезпечення** 52](#_Toc41565920)

[**4.3.1. Вступ.** 53](#_Toc41565921)

[**4.3.2. Створення тестових випадків** 53](#_Toc41565922)

[**4.3.3. Модульне тестування** 54](#_Toc41565923)

[**4.3.4. Функціональне тестування** 55](#_Toc41565924)

[**4.3.5. Тестування інтерфейсу** 55](#_Toc41565925)

[**4.3.6. Критерії успішності виконання тестування** 55](#_Toc41565926)

[**Висновок до розділу 4** 55](#_Toc41565927)

[**РОЗДІЛ 5 Економічне обґрунтування доцільності роботи** 56](#_Toc41565928)

[**5.1. Економічна характеристика проектного рішення** 56](#_Toc41565930)

[**5.2. Інформаційне забезпечення та формування гіпотези щодо потреби розроблення проектного рішення (програмного продукту)** 57](#_Toc41565931)

[**5.3. Оцінювання та аналізування факторів зовнішнього та внутрішнього середовищ** 58](#_Toc41565932)

[**5.4. Формування стратегічних альтернатив** 60](#_Toc41565933)

[**5.5. Бюджетування** 63](#_Toc41565934)

[**5.6. Вибір стратегії** 69](#_Toc41565935)

[**Висновки до розділу 5** 70](#_Toc41565936)

[**ВИСНОВКИ** 71](#_Toc41565937)

[**СПИСОК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ** 72](#_Toc41565938)

[**ДОДАТОК А. Варіанти використання** 74](#_Toc41565939)

[**ДОДАТОК Б. Діаграма прецедентів** 75](#_Toc41565940)

[**ДОДАТОК В. Варіанти використання** 76](#_Toc41565942)

[**ДОДАТОК Г. Звіт про модульне тестування** 82](#_Toc41565943)

[**ДОДАТОК Д. Інструкція користувача** 83](#_Toc41565944)

[**ДОДАТОК Е. Компонента що містить карту, та її функціонал** 86](#_Toc41565945)

**ВСТУП**

Дослідження багатьох передових країн зафіксували різкий ріст смертності людей, викликаної порушенням серцевої діяльності, яке призводить до раптової зупинки серця. Найчастішою причиною раптової зупинки серця, кардіальної смерті, є дуже часті (близько 350 на одну хвилину) серцеві скорочення. Це явище називається фібриляція серцевих шлуночків. Під час фібриляції серцеві скорочення є неефективними, тому серце не в змозі транспортувати кров по судинах, і кровообіг тканин та органів зупиняється. В таких випадках потрібна негайна госпіталізація, проте можливо сильно збільшити шанси на виживання потерпілого, надавши йому першу невідкладну допомогу. Серед найбільш ефективних заходів рятування життя людині, в якої раптово зупинилось серце є серцево-легенева реанімація із застосуванням автоматичного дефібрилятора. Було доведено, що кожна хвилина після зупинки серця, без здійснення дефібриляції, зменшує шанс потерпілого вижити на 10%. З іншого боку, вчасно зроблена дефібриляція, виконана протягом однієї-двох хвилин після раптової зупинки серця, відновлює роботу серця у більшості з постраждалих.[1]

Саме тому у більшості розвинених країн автоматичні зовнішні дефібрилятори встановлені у всіх місцях скупчення людей та просто на вулицях на видному місці. Такі дефібрилятори повністю автоматичні, тому скористатися ними може будь-яка людина, навіть та, що не має медичної освіти. Достатньо лишень витягти пристрій з коробки та приєднати електроди на грудну клітку потерпілого. Автоматичний зовнішній дефібрилятор самостійно проаналізує стан пацієнта і тільки при необхідності виконає розряд, який повинен зняти фібриляцію серцевих шлуночків. Більшість з таких дефібриляторів мають голосові підказки та візуальний супровід. Одразу після здійснення розряду, пристрій буде голосовими командами підказувати подальший алгоритм здійснення серцево-легеневої реанімації.[2, 3, 4]

В Україні ситуація з дефібриляторами є невтішною, оскільки їх можна знайти далеко не в кожному місці скупчення людей. Проте українське суспільство розуміє необхідність забезпечити цими приладами вокзали, торгово-розважальні центри, підприємства чи аеропорти. В інтернеті зараз розміщено безліч петицій щодо закупівлі та розміщення АЗД в різних містах нашої країни, а Міністерство охорони здоров’я сприяє введенню законів про обов’язкову наявність цього медичного приладу у вищезгаданих місцях.

Немає сумнівів, що Україна наслідуватиме приклад передових держав, і АЗД будуть встановлені там, де вони потрібні. Проте можуть залишитися ще дві проблеми:

* Люди, які стануть свідками того, як інша людина потерпає від зупинки серця, не будуть здатні вчасно знайти місце, де зберігається пристрій для автоматичної дефібриляції.
* Міністерству охорони здоров’я знадобиться інструмент для зручного адміністрування розміщення та стану цих пристроїв, адже саме вдале розміщення АЗД буде гарантувати найшвидший доступ до нього.

Обидві ці проблеми зможе покрити програмна система, яка дозволить користувачам отримати швидкий доступ до детального розташування найближчого дефібрилятора, а органам охорони здоров’я дасть змогу адмініструвати систему, візуалізувати ситуацію з розміщенням дефібриляторів для визначення та впровадження найбільш гармонічного розташування пристроїв, а також децентралізувати відповідальність за певні райони шляхом створення користувачів, які будуть слідкувати за станом дефібриляторів тільки у своєму районі зможуть редагувати інформацію тільки про дефібрилятори, за належний стан і розташування яких вони несуть відповідальність.

*Актуальність теми.*

В Європі сотні чоловіків та жінок (50% з яких навіть не мали жодних симптомів хвороби серця) помирають від раптової зупинки серця, стається це майже кожні 45 секунд. Запобігти такій великій кількості смертей можливо — негайно провівши серцево-легеневу реанімацію. Отже, критично важливо, щоб будь-яка людина в будь-якому місті України змогла вчасно отримати доступ до автоматичного зовнішнього дефібрилятора, який в рази збільшує шанси врятувати потерпілого. Саме тому, система для пошуку розташування найближчого автоматичного дефібрилятора *–* без перебільшення *–* може допомогти спасти життя потерпілому.[5]

*Мета і задачі дослідження.*

Метою роботи є розроблення програмної системи пошуку геолокації та інформації про автоматичні дефібрилятори, головна ціль якої є можливість пошуку локації найближчого до користувача дефібрилятора та шляху до нього, а також можливість адмініструвати систему людьми, які відповідають за наявність та розміщення пристроїв.

*Об’єкт дослідження –* процес пошуку та адміністрування пристроїв для автоматичної дефібриляції.

*Предмет дослідження –* методи та реалізації для швидкого та інтуїтивно зрозумілого пошуку найближчого дефібрилятора та можливості адміністрування розташування та інформації про наявні в системі дефібрилятори.

*Практичне значення одержаних результатів.*

Розроблена система дозволить людям, які стали свідком того, як інша людина потерпає від зупинки серця, отримати швидкий доступ до детального розташування найближчого дефібрилятора, а органам охорони здоров’я дасть можливість адмініструвати систему, візуалізувати ситуацію з розміщенням дефібриляторів для визначення та впровадження найбільш гармонічного розташування пристроїв, а також децентралізувати відповідальність за певні райони шляхом створення користувачів, які будуть слідкувати за станом дефібриляторів тільки в своєму районі і зможуть редагувати інформацію тільки про дефібрилятори, за належний стан і розташування яких вони несуть відповідальність.

**РОЗДІЛ 1**

**Небезпека проблеми раптової серцевої смерті, та важливість екстреної допомоги з використанням дефібрилятора**

* 1. **Раптова серцева смерть**

Незважаючи на те, що сучасна медицина сильно розвинулась у лікуванні серцевих захворювань та в розробці новітніх технологій для лікування та діагностики, розвинені країни дотепер потерпають від проблеми раптової серцевої смерті. Звертаючись до статистики, можна сказати, що 88% з усіх випадків раптової смерті викликані саме серцевою смертю, а в розвинутих країнах кожного року раптово помирає 1 з 1 тисячі осіб. [6]

Раптова серцева смерть – це смерть, викликана кардіальною причиною, що розвинулась моментально, або наступила на протязі години від початку гострих симптомів. Причиною цього можуть бути більш як 30 захворювань. Найбільш поширеними причинами є: кардіальна патологія, серцево-судинні захворювання та спадкові фактори. Іншими факторами ризику, що можуть сприяти раптовій серцевій смерті, є :

* Вік більше 50 років.
* Високий рівень холестерину (в той час як про свій рівень холестерину знають лише 1,3% населення України).
* Паління (а цією шкідливою звичкою володіють 40% наших громадян).
* Підвищений артеріальний тиск (вищий за 140мм ртутного стовпчика). [8]

Проте, навіть якщо в людини не було жодних серцевих захворювань та шкідливих звичок, в неї також є ризик, що її серце може зупинитися, адже у 12% потерпілих від раптової зупинки серця після обстеження не було виявлено жодних факторів ризику, які могли б спричинити трагедію. [6, 8]

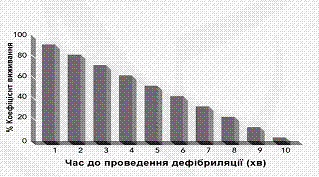
* 1. **Важливість першої невідкладної допомоги**

Без сумнівів, якщо ви стали свідком того, як людина потерпає від зупинки серця, необхідно негайно зателефонувати в службу для швидкої допомоги, оскільки потерпілому необхідна госпіталізація. Проте карета швидкої допомоги їхатиме на виклик якийсь час, а допомога у випадку з зупинкою серця потерпілому потрібна негайно, тому слід якнайшвидше почати серцево-легеневу реанімацію.

Серцево-легенева реанімація – це невідкладна процедура, що спрямована на відновлення життєдіяльності організму шляхом виведення потерпілого зі стану клінічної смерті. Ця процедура передбачає непрямий масаж серця (компресія грудної клітки) і штучну вентиляцію легень. Завдяки своїм натискам можливо механічно відновлювати кровообіг, щоб забезпечити транспортування крові до мозку, тим самим не допустити його біологічної смерті. Проте ця реанімація не буде ефективною якщо не зняти фібриляцію серцевих шлуночків за допомогою дефібрилятора. [5]

Фібриляція серцевих шлуночків – це дуже часті (до 350 на хвилину), хаотичні та різночасові скорочення волокон фібрилу (серцевого м’яза), під час яких серцеві скорочення неефективні, і серце не здатне гнати кров по судинах. [1, 2]

В більшості випадків клінічна смерть починається саме з фібриляції шлуночків, тобто серце не зупиняється, як це показують у фільмах, воно змінює свій ритм на такий, що унеможливлює кровообіг. Дефібриляцію потрібно здійснити як найшвидше, адже кожна хвилина затримки знижує шанси на виживання потерпілого на 10-15%. У випадку, якщо пройшло 7-10 хвилин з моменту появи фібриляції, шанси на виживання потерпілого стають близькі до нуля. [7]



*Рис. 1.1. Залежність коефіцієнта виживання від часу, що пройшов до початку проведення дефібриляції (зниження виживаності на 10% кожну хвилину затримки) [7]*

Звісно, можна робити серцево-легеневу реанімацію і виграти деякий час до приїзду медиків, які в свою чергу мають дефібрилятори і зможуть провести процедуру дефібриляції, проте середній час прибуття карети швидкої від 8 до 12 хвилин (мова йде про країни з більш досконалою і налагодженою системою у сфері екстреної медичної допомоги, в Україні цей час буде явно більший). Виходячи з цього стає очевидним, що тільки наявність та максимально швидкий доступ до автоматичного зовнішнього дефібрилятора зможе гарантувати вчасну дефібриляцію та зможе врятувати потерпілого від раптової серцевої смерті. [5, 8]

* 1. **Автоматичний зовнішній дефібрилятор**

Автоматичний зовнішній дефібрилятор (АЗД) – це портативний автоматичний прилад, який самостійно визначає стан пацієнта (діагностує небезпечні для життя, хаотичні та різночасові скорочення серця – фібриляцію шлуночків), та у випадку виявлення аномалій здійснює розряд (дефібриляцію), який здатний відновити звичайний ритм серця.

Даний пристрій є дуже простий у використанні, тому ним може скористатися будь-яка людина, яка пройшла короткий тренінг (протягом 1 години). Моделі АЗД розміщені в протиударному кейсі, який без зусиль відкривається, і пристрій автоматично вмикається. В комплекті також будуть знаходитись самоклеючі електроди, які слід приклеїти на грудну клітку в місцях, які зображені на відповідних рисунках на самих електродах, а клеми від електродів потрібно увімкнути у відповідне гніздо на блоці управління. Особливостями цього пристрою, які роблять його максимально простим для використання, є:

* Мінімалізм інтерфейсу керування (зазвичай пристрій містить тільки одну кнопку для здійснення розряду, і тільки у випадку, якщо після проведення аналізу він визначив що це необхідно, в іншому випадку «розряд» здійснити він не дозволить)
* Голосовий супровід. Пристрій буде повідомляти вас про ті дії які він зараз виконує, попередить коли необхідно буде натиснути кнопку, щоб здійснити «розряд», і дасть застереження, щоб ніхто не торкався пацієнта, коли буде виконуватись дефібриляція. Після виконання дефібриляції пристрій даватиме підказки про те, як здійснювати серцево-легеневу реанімацію, і навіть задасть ритм, в якому потрібно буде здійснювати компресію грудної клітки, звуковими сигналами [6]

Практика передових країн показала, що застосування автоматичного дефібрилятора до приїзду екстрених служб – збільшує шанси на успішно проведені дії щодо рятування постраждалих.

Статистика говорить, що у публічних місцях та закладах, не оснащених АЗД, у випадку раптової серцевої смерті виживають не більше 5% людей.

Щодо закладів, оснащених АЗД, у випадках раптової серцевої смерті зафіксовані такі дані: [7]

* В аеропортах та на борту літаків – 40-60%
* На робочих місцях (підприємствах, офісах та ін.) – 52%
* У місцях відпочинку (ресторани, концертні зали на ін.) – 74%

Таблиця 1.1

**Час реакції та коефіцієнт виживання [7]**

|  |  |  |
| --- | --- | --- |
| **Вид першої допомоги** | **Час до першого розряду,**  **хв** | **Частка виживаності**  **%** |
| Традиційна швидка допомога | 10 | <10 |
| Пожежні служби | 6 | 40 |
| Поліція | 5 | 50 |
| Дефібриляція на місці | 3 | >70 |

Згідно з таблицею, яка сформована за даними США, середній час прибуття швидкої допомог, навіть у дуже розвиненій економічно країні, становить близько 12 хвилин, тож очевидним стає те, що в більшості випадків раптової сердечної смерті врятувати потерпілого зможе тільки швидкий доступ до АЗД. [6, 7]

За підрахунками, при достатньому оснащені АЗД громадських місць, установ та закладів в Україні прогноз врятованих життів при виникненні раптової сердечної смерті становитиме близько 30-50 тисяч осіб на рік.

Місця в яких рекомендується встановлювати АЗД: [7]

* Навчальні заклади.
* Офіси, банки та міністерства
* Аеропорти, автовокзали, залізничні вокзали та станції метро
* Стоматологічні кабінети та поліклініки
* Літаки, потяги, автобуси
* Кораблі та пароплави
* Різноманітні заклади відпочинку (готелі, казино)
* Пожежно-рятувальна служба
* Торгівельні центри, державні та громадські заклади та установи
* Медичні заклади та медичний транспорт

На цей момент часу Україна недостатньо забезпечена пристроями АЗД, проте суспільство враховує позитивний досвід активного використання АЗД іншими країнами, тому можливо спостерігати, що в інтернеті зараз розміщено багато петицій про закупівлю та розміщення АЗД в різних містах нашої країни, а Міністерство охорони здоров’я сприяє врегулюванню законодавства щодо обов’язкової наявності цього медичного приладу у громадських місцях. Враховуючи ці фактори, можна прогнозувати, що невдовзі в Україні у більшості багатолюдних місць на помітному місці будуть розташовані шафи з автоматичними зовнішніми дефібриляторами.

* 1. **Принципи роботи та можливості існуючих систем для пошуку АЗД**

Для того, щоб ці пристрої максимально ефективно допомагали рятувати життя людей, потрібно забезпечити максимально швидкий доступ до них, а також на етапі активного впровадження даних пристроїв надзвичайно корисним буде інструмент для того, щоб зручно адмініструвати їхнє розміщення (гармонічно розподіляти АЗД по місцевості, щоб унеможливити ситуації, коли одне місце, наприклад, перенасичене дефібриляторами, а в іншому вони відсутні) та стан.

Для того, щоб забезпечити дані функціональні вимоги, добре підходить веб-застосунок або мобільний додаток, який буде активно працювати з картами та геолокацією користувача. Дана система повинна ідентифікувати найближчий АЗД на карті відповідно до місця локації користувача і за потребою візуалізувати найкоротший шлях до пристрою, а також відобразити відстань до нього і час, який займе подолання цієї відстані.

Такий сервіс як Google MapboxMaps не підходить для наших цілей, тому що в даному сервісі будь-хто зможе додати розташування пристрою (навіть якщо насправді його там немає), а також, якщо доданий пристрій буде вже не доступний на доданому місці, або за якихось причин відсутній, він дальше буде зображений на карті й людина, що є відповідальна за дефібрилятор, не зможе моментально оновити інформацію про його поточний стан. Тому нам потрібна закрита система з доступною авторизацією, яка буде доступна тільки користувачам, відповідальним за розміщення та стан дефібриляторів в певному районі, і які зможуть адмініструвати інформацію про пристрої, за які несуть відповідальність. Прості користувачі зможуть, без будь-якої авторизації знайти бажаний пристрій, інформацію та шлях до нього.

**AED Locations**

Один з додатків, який покриває більшість вимог, є «AED Locations» - мобільний додаток, який доступний у магазинах додатків App Store та Google Market, а також в нього є веб-версія, яка доступна за таким посиланням - <https://aedlocations.co.nz/>.

Я буду аналізувати мобільний додаток з App Store, який доступний для стягування на всі моделі iPhone. Цей додаток на сьогодні ще не вийшов на глобальний ринок, і обслуговується тільки для одної країни, а саме Нової Зеландії, тому в даному додатку локації АЗД ви можете знайти тільки там.

Незважаючи на те, що цей додаток не набув поки великої популярності й використовується тільки в одній країні, було здійснено аналіз і визначено вдалі та невдалі рішення, а також чого не вистачає даному додатку.

На рисунку 1.2 ми бачимо мінімалістичний та зручний інтерфейс даного додатку, який зустрічає нас при відкритті програми. Як ви можете побачити на рисунку, в додатку автоматично визначається локація користувача, на відповідному місці на карті встановлюється червона кругла іконка, і карта централізується на ній. В нижній панелі розміщено список АЗД, який посортований по відстані до пристрою (від найближчого до найдальшого), а також під кожним елементом списку виведена відстань до нього в метрах. Це дуже зручно, проте більш доцільно було б виводити також час, за який можна добратися до пристрою, адже саме він є ключовий у випадках раптової серцевої смерті.

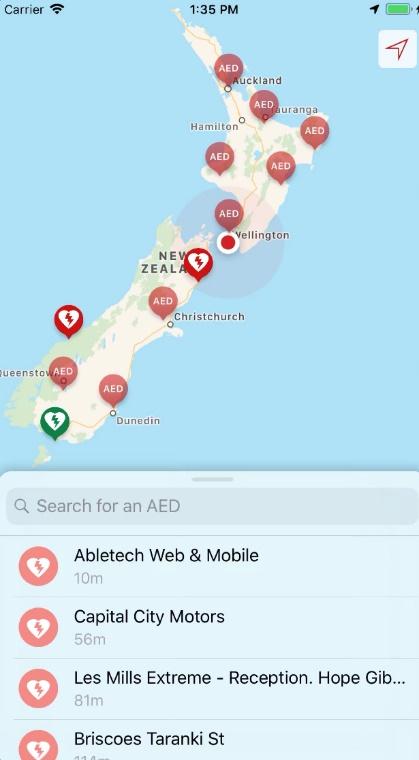


Рис. 1.2. Скріншот з додатку AED Locations

А також в нижній панелі розміщене поле для введення адреси, в яке користувач зможе ввести необхідне значення і отримає список пристроїв адреса який відповідає введеному значенню, саме це можна бачити на рисунку 1.3.(скрін зліва).

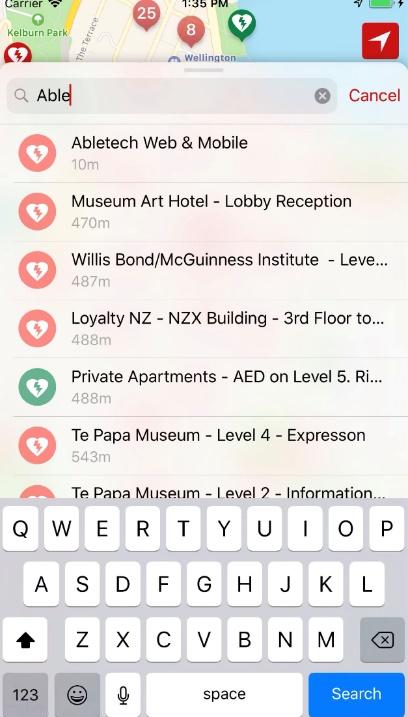
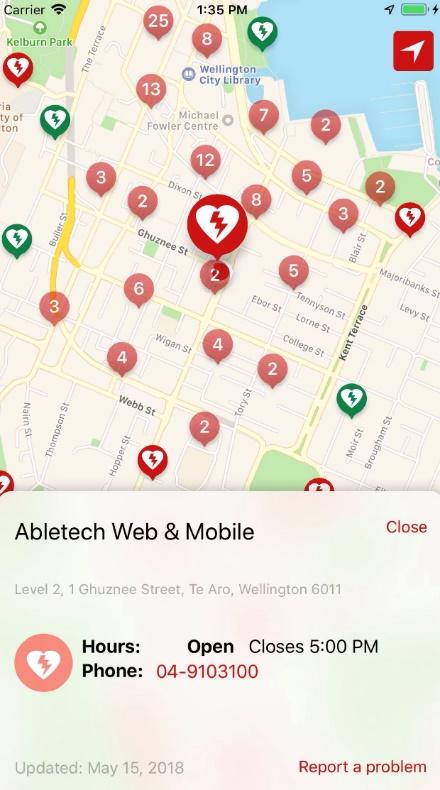
 

Рис. 1.3. Скріншоти з додатку AED Locations

Після кліку на один зі значків на карті, що символізують про розміщення в тому місці АЗД, даний значок збільшиться, а в нижній панелі буде виведена необхідну інформацію про даний пристрій, зокрема адресу, за якою знаходиться пристрій, в які годині він доступний, коли останній раз оновлялася інформація про нього, і номер відповідальної за пристрій людини. Саме зображено на рисунку 1.3.(скрін справа). Також можна відзначити, що доступні дефібрилятори позначені зеленою іконкою, а недоступні — червоною. На мою думку, слід показувати користувачу тільки доступні дефібрилятори, щоб не нагромаджувати інтерфейс. В додатку також реалізована кластеризація, тобто якщо в певній локації(в залежності від масштабу) багато дефібриляторів, то значки які символізують про їхню наявність групуються в один на якому зазначена кількість дефібриляторів в певній зоні.

Ще одним з найбільш явних недоліків є те, що в користувача немає можливості зобразити найкоротший шлях до вибраного пристрою .

Для того, щоб додати інформацію про новий дефібрилятор або повідомити інформацію про наявний, потрібно написати на вказану в додатку електронну пошту. Набагато доцільніше було б створити систему авторизації яка б дозволила відповідальним людям за певний район адмініструвати інформацію про наявні в цьому районі АЗД, це б дозволило також децентралізувати нашу систему.

**Висновки до розділу 1**

В цьому розділі описано поширену сьогодні проблему раптової серцевої смерті, а також важливість методів екстреної медичної допомоги потерпілому у випадках її настання. Опрацьовуючи матеріали по цій темі, було визначено, що одним з ключових аспектів, який збільшує шанси потерпілого на виживання, є наявність та максимально швидка доступність до автоматичних зовнішніх дефібриляторів (АЗД), адже тільки з ними можна за максимально короткий час, отже й найбільш ефективно, провести реанімацію пацієнта.

На ринку наявна дуже невелика кількість систем, які б оптимізували процес швидкого пошуку та адміністрації АЗД, а ті, що існують, не обслуговують інформацію про АЗД в Україні та не відповідають всім вимогам, тому я вважаю, що є необхідність створити таку систему яка б дозволила АЗД врятувати найбільшу кількість життів.

**РОЗДІЛ 2**

**Постановка задачі розробки системи пошуку геолокації та інформації про автоматичні дефібрилятори**

* 1. **Загальна постановка завдання**

Завдання полягає в створенні системи, яка допоможе будь-яким користувачам швидко знаходити найближчий АЗД та шлях до нього, а також дозволить певним медичним структурам адмініструвати стан, інформацію та розташування цих пристроїв. Для цього було вирішено розробити веб-застосунок зі зручним та інтуїтивно зрозумілим інтерфейсом та можливістю взаємодії з картою та геолокацією користувача, системою авторизації, та алгоритмами для знаходження найближчого АЗД з можливістю побудувати шлях до нього.

* 1. **Специфікація вимог до програмного продукту**

**Вступ**

* + 1. **Призначення та мета**

Основними призначеннями даної системи є пошук геолокації та інформації про автоматичні дефібрилятори, а також можливість адміністрування цих даних. За допомогою такої системи неавторизовані користувачі, які стали свідком раптової серцевої смерті, зможуть максимально швидко виявити найближчий до них АЗД, отримати найкоротший шлях до нього, а також інформацію про сам пристрій і його детальне розташування. В даній системі буде реалізована авторизація, яка дозволить створювати користувачів, які зможуть слідкувати за станом дефібриляторів у своєму районі та зможуть редагувати інформацію про пристрої, які містяться там.

* + 1. **Продукти-аналоги**

Враховуючи те, що АЗД поширені по всьому світу, систем, які оптимізують процес пошуку та адміністрування пристроїв, на мій погляд, несправедливо мало, а ті, що існують не вийшли на глобальний ринок і не володіють всім необхідним функціоналом. Одна з таких систем була проаналізована та описана у першому розділі цієї записки до дипломної роботи. Мова йде про додаток AED Locations (<https://aedlocations.co.nz/>), який обслуговує розташування та інформацію про автоматичні дефібрилятори виключно для однієї країни – Нової Зеландії.

* + 1. **Загальний опис**

**2.2.3.1. Характеристики продукту**

Основними можливостями системи є:

* Відображення геолокації користувача на карті
* Відображення розташування АЗД на карті
* Відображення детальної інформації про вибраний дефібрилятор та про його розташування
* Генерація та відображення найкоротшого шляху від місця локації користувача до вибраного пристрою
* «Екстрена кнопка», при натисканні якої система визначить найближчій дефібрилятор і побудує шлях до нього
* Відображення списку всіх наявних дефібриляторів з сортуванням по відстані від користувача (від найближчого)
* Застосування до списку дефібриляторів вибраних фільтрів
* Можливість авторизації як адміністратор або модератор
* Для адміністратора наявна можливість створення нових модераторів
* Модератор може скористатися функцією відновлення паролю
* Адміністратор може «блокувати» дефібрилятори – модератори або неавторизовані користувачі не будуть їх бачити ні в списку ні на карті.
* Додавання нових дефібриляторів та редагування існуючих (модератори можуть редагувати тільки ті дефібрилятори які самі створили)

**2.2.3.2. Класи користувачів**

Система матиме кілька класів користувачів, що будуть володіти різними привілеями і для яких буде доступний різний функціонал.

* Звичайний користувач – йому не потрібно проходити процедуру авторизації. З цими правами з системою будуть працювати представники цільового сегмента користувачів – люди, які стали свідками того, що хтось потерпає від зупинки серця, тому для них доступний тільки найнеобхідніший функціонал: пошук найближчого дефібрилятора та шляху до нього, а також генерація та візуалізація шляху до вибраного дефібрилятора та виведення детальної інформації про нього
* Модератор – для входу в систему повинен пройти авторизацію. Має можливість додавання в систему нових АЗД, а також може редагувати інформацію про пристрої, які він додав
* Адміністратор – має можливість реєструвати нових модераторів, додавати нові дефібрилятори, редагувати інформацію про будь-який пристрій та блокувати його

**2.2.3.3. Середовище функціонування**

Для функціонування серверної частини системи потрібні:

* Платформа з встановленим Node.js та всіма необхідними пакетами, які описані в файлі package.json
* Сервер бази даних – MongoDB, на якому створені та заповнені необхідні колекції даних

Для того, щоб коректно працювала клієнтська частина, потрібен будь-який із сучасних веб-браузерів.

* + 1. **Характеристики системи**

**Req. 1. Відображення локації користувача**

*Опис:* При відкритті веб-аплікації користувач повинен одразу бачити своє місце розташування.

*Пріоритет:* Середній.

*Функціональні вимоги:*

1. Система запрошує доступ до геолокації користувача при першому відкритті веб-аплікації.
2. Відображенні відповідної іконки у місці розташуванні користувача, якщо він дозволив доступ до своєї геолокації.
3. При першому відкритті системи карта центрується на місці розташування користувача.

**Req. 2. Відображення місць в яких розташовано АЗД**

*Опис:* Для будь-якого користувача на карті будуть відображені певні іконки, які символізують про наявність на відповідній локації АЗД.

*Пріоритет:* Високий.

*Функціональні вимоги:*

1. Якщо в одному місці декілька дефібриляторів, або на карті встановлено масштаб, при якому іконки про наявність дефібриляторів «налізають» одна на одну, іконки збиваються в кластер, який показується замість іконок на певній локації, і на якому зображена кількість пристроїв.

**Req. 3. Візуалізація інформації про вибраний дефібрилятор**

*Опис:* Будь-який користувач, клікнувши на карті на значок про на наявність дефібрилятора, зможе побачити детальну інформацію про даний пристрій у стилізованому вікні.

*Пріоритет:* Високий.

*Функціональні вимоги:*

1. Анімація відкриття та загрузки вікна з потрібною інформацією.
2. Закриття вікна при кліку на відповідну кнопку призначену для закриття або при збільшенню або зменшенню масштабі карти.
3. Відображення модального вікна з фотографіями пристрою при кліку на кнопку «Фотографії».

**Req. 4. Відображення списку дефібриляторів в бічному меню**

*Опис:* В бічному меню користувач буде бачити список дефібриляторів, де кожен елемент буде містити назву та адресу. Адміністратор і модератор буде також бачити для кожного елемента списку кнопки для редагування, видалення або блокування (тільки у випадку з адміністратором).

*Пріоритет:* Високий.

*Функціональні вимоги:*

1. Підсвічування елемента списку, який відповідає «активному» дефібрилятору.
2. Сортування дефібриляторів в залежності від відстані користувача до них (від найближчого до найдальшого).

**Req. 4. Фільтрація списку з дефібриляторами**

*Опис:* Користувач має змогу задати певні фільтри (адресу, назву пристрою, мову інтерфейсу та наявність інформаційних табличок), відповідно до яких буде формуватися список дефібриляторів

*Пріоритет:* Середній.

*Функціональні вимоги:*

1. Елементи після фільтрації також сортуються по відстані до користувача.

**Req. 5. Пошук найближчого дефібрилятора**

*Опис:* В системі всім користувачам доступна «екстрена кнопка», натиснувши на яку, буде знайдено найближчий АЗД і шлях до нього.

*Пріоритет:* Високий.

*Функціональні вимоги:*

1. Визначення найближчого АЗД до користувача.
2. Центрування карти на локації знайденого пристрою.
3. Визначення і відображення найкоротшого маршруту.
4. Визначення та відображення відстані до пристрою та часу, щоб дістатися до нього.

**Req. 6. Відображення шляху до вибраного АЗД**

*Опис:* При відкритті вікна для перегляду детальної інформації про пристрій, доступна кнопка, клікнувши на яку, система проведе найкоротший шлях до вибраного пристрою

*Пріоритет:* Високий.

*Функціональні вимоги:*

1. Визначення і відображення найкоротшого маршруту.
2. Визначення та відображення відстані до пристрою та часу, щоб дістатися до нього.

**Req. 7. Реєстрація нових модераторів**

*Опис:* Користувач з правами адміністратора має можливість зареєструвати нового користувача з правами модератора. Для цього йому потрібно перейти в розділ «Особистий кабінет» та натиснути на кнопку «Зареєструвати користувача». Після цього система зобразить модальне вікно з полем вводу електронної пошти модератора, якого слід зареєструвати в систему. На введену електронну скриньку прийде повідомлення з посиланням, за яким користувач повинен перейти для створення паролю, який він буде використовувати для авторизацію в систему з правами модератора.

*Пріоритет:* Середній.

*Функціональні вимоги:*

1. Формування повідомлення, яке містить посилання, за яким користувач якому надійшов лист зможе перейти і створити пароль.
2. Надсилання повідомлення на вказану адміністратором пошту для реєстрації.
3. Після того, як користувач перейшов за посиланням і створив пароль в базі даних користувачів, створюється відповідний запис, в якому міститься зашифрований пароль і пошта користувача.

**Req. 8. Додавання нового дефібрилятора в систему**

*Опис:* Користувач з правами адміністратора або модератора може додати новий АЗД в систему, клікнувши на кнопку «Додати новий дефібрилятор», яка знаходиться в нижній частині бічного меню. Після цього в бічному меню відкриється форма додавання з відповідними полями, які описують розташування та інформацію про пристрій.

*Пріоритет:* Високий.

*Функціональні вимоги:*

1. Перевірка на коректність при введенні даних в поля форми для додавання дефібриляторів.
2. У випадку, якщо користувач здійснить подвійний клік на певне місце на карті, здійснюється встановлення на цьому місці специфічної іконки, та визначення адреси цього місця і запис його в поле «адреса» форми додавання нового дефібрилятора.
3. При введенні адреси в поле вводу «адреса», система здійснює запити на Google API та повертає користувачу адреси, які співпадають із введеним в поле.
4. Відображення повідомлення про успішне додавання пристрою або про те що в поля форми заповнені некоректно.

**Req. 9. Адміністрування інформації про існуючі в системі АЗД**

*Опис:* Користувач з правами модератора може редагувати інформацію про пристрій, який він додав в систему, клікнувши на кнопку «Редагувати», яка знаходиться в елементі зі списку дефібриляторів, який відповідає певному АЗД. Також модератори можуть видаляти АЗД, на які вони мають права, клікнувши на кнопку «Видалити», яка знаходиться справа від кнопки для редагування. Користувач з правами адміністратора може редагувати і видаляти будь-який дефібрилятор, який наявний в системі, також може «блокувати» дефібрилятор, натиснувши на кнопку блокування. Після «блокування» пристрій не буде доступний для неавторизованих користувачів та користувачів з правами адміністратора.

*Пріоритет:* Високий.

*Функціональні вимоги:*

1. Перевірка на коректність при введенні даних в поля форми для редагування дефібриляторів.
2. У випадку, якщо користувач здійснить подвійний клік на певне місце на карті, здійснюється встановлення на цьому місці специфічної іконки, та визначення адреси цього місця і запис його в поле «адреса» форми редагування нового дефібрилятора.
3. При введенні адреси в поле вводу «адреса», система здійснює запити на Google API та повертає користувачу адреси які співпадають із введеним в поле.
4. Відображення повідомлення про успішне редагування пристрою або про те, що в поля форми заповнені некоректно.
5. Блокований дефібрилятор буде приховуватись на карті та в списку дефібриляторів для неавторизованих користувачів та для модераторів.
   * 1. **Вимоги зовнішніх інтерфейсів**
        1. **Користувацькій інтерфейси**

Інтерфейс системи (десктоп версія) складається з:

* Карти, на якій розміщені іконки про розташування дефібриляторів, та місце розташування користувача
* Бічне меню в якому міститься список пристроїв та «хедер» в якому містяться кнопки входу в обліковий запис (у випадку якщо користувач не авторизований) або кнопка для управління особистим кабінету, якщо користувач авторизований в систему
* На правому-верхньому куті карти розташовані три кнопки (рисунки 2.1, 2.2 і 2.3):



Рис. 2.1. Кнопка для пошуку найближчого дефібрилятора та відображення найкоротшого шляху до нього



Рис. 2.2. Кнопка для пошуку наступного дефібрилятора та відображення найкоротшого шляху до нього



Рис. 2.3. Кнопка яка видаляє згенеровані шляхи та центрує карту на розташуванні користувача

**2.2.6. Інші нефункціональні вимоги**

* + - 1. **Вимоги безпеки**

В проекті наявна система авторизації, яка заборонить неавторизованим користувачам додавати або змінювати інформацію про наявні в системі дефібрилятори. Також є система ролей, серед яких модератори та адміністратори. Користувачі обидвох ролей зможуть додавати нові пристрої в систему, проте модератор зможе редагувати тільки ті, що сам і додав, а адміністратор інформацію про будь-який дефібрилятор, що наявний в системі.

В колекції інформації про користувачів системи паролі зберігаються в зашифрованому вигляді, тому, якщо базу даних скомпрометують, зловмисники не зможуть ввійти в систему, адже паролі до облікових засобів, які вони отримають, будуть зашифровані.

* + - 1. **Вимоги продуктивності**

Для списку дефібриляторів в бічному меню повинна бути реалізована пагінація, тобто на клієнтській частині зберігатимуться тільки 10 найближчих дефібриляторів, і при скролінгу списку вниз, здійснюватимуться запити до бази даних, і тільки тоді на клієнт в список завантажуватимуться нові дефібрилятори. Завдяки такій реалізації виключається сценарій, при якому в список дефібриляторів на клієнті завантажувались б усі наявні в базі дефібрилятори, адже це б сприяло довгому очікуванню користувача і навантаженню на його браузер.

* + 1. **Атрибути якості програмного продукту**

Веб-аплікація повинна бути повністю адаптивною, тобто користувачі як з мобільних пристроїв повинні також мати змогу зручно і швидко користуватися системою. Повинна бути забезпечена валідація введених користувачем даних в будь-які наявні в системі поля, а також передбачені всі виняткові ситуації. Для того, щоб забезпечити супровід даного продукту, та перенесення ресурсів системи на іншу платформу, повинні бути розроблені супутні інструкції.

* 1. **Вибір засобів для виконання завдання**

Дана система, що розробляється, складається з клієнтської та серверної частини. Для розробки серверної частини буде використано платформу з відкритим кодом - Node.js.

Ключовим фреймворком на сервері є Express – це мінімалістичний і зручний веб-фреймворк для проектів Node.js, який надає широкий набір функцій для веб-аплікацій [11]. Його можливості будуть використовуватись, щоб специфікувати відповідним роутам певний функціонал і запускати сервер на певному порту.

PassportJS – це бібліотека для node.js, яку зручно використовувати для авторизації, та яку я буду використовувати для цих же цілей у своєму проекті [14].

Ще однією з ключових бібліотек для роботи сервера є Mongoose – це бібліотека, яка дозволить нашому серверу працювати з базою даних MongoDB.

Для збереження даних буде використовуватись не реляційну базу даних - MongoDB, а також MongoDB Compass як графічний інтерфейс для роботи з колекціями даних.

Для розробки клієнтської частини буде використана бібліотека React, яка ідеально підходить для управління станом всіх компонент інтерфейсу [9].

Оскільки в системі буде велика кількість компонент, слід також використати Redux – це менеджер станів, який дає дозволяє тримати стани всіх наших компонентів в одному місці, і дозволяє отримувати або змінювати будь-який стан з будь-якого компонента.

Для стилізації компонент інтерфейсу буде використовуватись фреймворк - Material-UI, який є найкращим інструментом у випадку якщо клієнт використовує бібліотеку React.

Для того, щоб зобразити карту на клієнті, розмістити на ній власні компоненти й дати можливість користувачу взаємодіяти з нею, буде використана платформу з відкритим кодом Mapbox.

Для реалізації такого функціоналу як повернення можливих адрес, які збігаються із введеними словами або знаходження адреси, яка відповідає заданим координатам я виконуватиму запити на Google API.

Для того, щоб зручно валідувати та керувати наявними в системі формами зручно буде використовувати бібліотеку Formik яка забезпечить нас великою кількістю методів для поставлених цілей.

**Висновок до розділу 2**

Спираючись на результати аналізу було створено специфікацію вимог до програмного забезпечення, в якій було описано основні сервіси системи та функціональні вимоги до них. Було визначено призначення та мету програмного продукту. Проаналізовано класи користувачів даної системи. Також були обрані програмні засоби для розробки даної веб-аплікації.

**РОЗДІЛ 3**

**Архітектура та проектування системи для пошуку геолокації та інформації про автоматичні дефібрилятори**

* 1. **. Проектування поведінки розроблюваної системи**

В попередньому розділі було визначено класи користувачів, які будуть наявні в системі, та функціональні вимоги які повинні бути реалізовані в проекті для задоволення потреб всіх цих класів користувачів. В даному розділі описано шляхи їхньої взаємодії з системою та доступний їм функціонал.

*Неавторизований користувач* увійшовши в систему буде бачити свою локацію на карті а також значки на ній, які символізують про наявність в тих місцях АЗД. В даного типу користувача, якого в подальшому будемо називати «Гість», може без всіляких обмежень працювати з картою, а саме:

* Може переміщатись по карті.
* Збільшувати або зменшувати масштаб карти.

Гість може клікнути на іконку АЗД на карті, щоб переглянути детальну інформацію про нього. Йому буде продемонстроване вікно в якому міститься детальний опис місця розташування АЗД та номера телефонів людей, які відповідальні за даний пристрій. В цьому вікні також міститься кнопка «Фотографії», клікнувши на яку буде відкрито нове вікно в якому міститься галерея фотографій даного пристрою. Також у верхній частині вікна детальної інформації про АЗД міститься кнопка, на яку гість може натиснувши, щоб система провела найкоротший шлях до вибраного АЗД. Будь якому користувачу також доступна «Екстрена кнопка», натиснувши на яку система автоматично визначить найближчий до користувача пристрій і побудує шлях до нього. Якщо за якихось причин найближчий дефібрилятор не задовольняє потреби користувача, він може натиснути на кнопку яка розміщена справа від «Екстреної кнопки», щоб знайти наступний за на відстанню від користувача пристрій. Також користувач без авторизації може переглядати список дефібриляторів який розміщений у бічному меню. Гість може клікнути на один з елементів списку, і карта централізується на дефібриляторі який відповідає цьому елементу списку. Гість також має можливість введення в поле яке знаходиться над списком дефібриляторів, тим самим шукаючи серед пристроїв ті, адреса яких відповідає тому, що було введено. Також користувач без авторизації може скористатися всіма доступними опціями фільтрування списку, вибравши мову інтерфейсу, наявність інформаційних табличок і годин коли доступний пристрій.

*Модератору* після авторизації в систему доступний весь функціонал, що й неавторизованому користувачу, а також:

* Клікнувши на кнопку в «Додати дефібрилятор», нижній частині бічного меню, буде відображена форма додавання, в якій користувач повинен вказати дані про пристрій який він додає, після чого натиснути на кнопку «Зберегти», щоб новий пристрій був доданий в систему.
* Користувач з правами модератора може вибрати з списку дефібрилятор який він раніше додав, і натиснути на кнопку «Редагувати». Після чого в бічному меню буде відображена форма редагування інформації про вибраний дефібрилятор, де всі поля будуть заповнені наявною інформацією про пристрій. Модератор може змінити вміст наданих полів та натиснути кнопку «Зберегти», після чого, у випадку коректності введених даних, інформація про даний пристрій буде оновлена.
* Може вибрати з списку раніше доданий ним дефібрилятор, і клікнути на кнопку «Видалити». Після цього буде відображене модальне вікно, де користувач буде мати змогу натиснути на кнопку «Так», щоб підтвердити видалення пристрою з системи, або відмінити свою дію, клікнувши на кнопку «Ні».

*Адміністратору* потрібно також пройти авторизацію, після чого йому стають доступні права всіх інших наявних в системі ролей, а також

* На відміну від модераторів, він може видаляти і редагувати будь-який дефібрилятор, а не тільки той, що сам і додав.
* Може вибрати елемент із списку дефібриляторів і клікнути на кнопку «Блокувати», після чого йому буде відображене модальне вікно з двома кнопками «Так» і «Ні». Клікнувши на «Так», іконка блокування вибраного дефібрилятор поміняє колір на червоний, а також даний дефібрилятор не будуть бачити в системі неавторизовані користувачі і користувачі з правами модератора. Якщо користувач здійснить клік на кнопку «Ні» в модальному вікні, операція по блокуванню буде відмінена.
* Може реєструвати нових користувачів з правами адміністратора. Перейшовши в особистий кабінет, йому буде доступна кнопка «Зареєструвати користувача», клікнувши на яку для нього буде відображене модальне вікно в якому міститься поле для введення електронної пошти користувача якого він хоче зареєструвати як модератора системи. Заповнивши поле, адміністратор може клікнути на кнопку «Реєстрація», для того щоб відправити на задану пошту лист для продовження реєстрації.

В Додатку Б зображена діаграма прецедентів, на якій розміщені актори, які є класами користувачів, а також функціонал для кожного з наявних в системі користувачів.

**3.2. Загальна архітектура системи**

В процесі розроблення будь-якої програмної системи необхідно не тільки чітко визначити її вимоги і функціональність, а й також надзвичайно важливим є розроблення архітектури проекту. Адже якщо опустити цей етап, або здійснити значущу помилку в проектуванні, на етапі розроблення можуть виникнути проблеми які будуть нести за собою фінансові та часові втрати.

Найкращим варіантом для розроблення даної системи, буде побудувати архітектуру, базовану на клієнт-серверній взаємодії. Де на стороні клієнта працюватиме веб-застосунок з використанням бібліотеки React, за допомогою якого ми будемо представляти дані клієнту і опрацьовуватимемо його команди. Для авторизації, роботи з даними про дефібриляторами, реєстрації нових користувачів та інших операцій, які потребують виконання запитів до бази даних буде створена REST API, за допомогою програмного забезпечення на платформі Node.js – це буде нашою серверною частиною. На ній буде використаний фреймворк Express.js, щоб специфікувати «роути», на які клієнт здійснюватиме запити за допомогою протоколу HTTP, а також описано як наш сервер буде опрацьовувати ці запити, і що він буде повертати на клієнт. Для того, щоб дати можливість серверу працювати з базою даних буде використано бібліотеку mongoose, яка дозволить проводити операції з базою даних за допомогою протоколу SSL. Базою даних для системи буде служити MongoDB з колекціями для дефібриляторів, користувачів та фотографій.

Логічне представлення вище описаної архітектури зручно зобразити за допомогою діаграми розгортання, яка зображена на рисунку 3.1.



Рис. 3.1. Діаграма розгортання

**3.3. Проектування бази даних**

Для формування нашої бази даних було обрано MongoDB – саму популярну неструктуровану документ-орієнтовану систему управління базами даних. Дана СУБД завдяки своїй простоті працює набагато швидше ніж SQL бази даних, та дуже проста у використанні. Проте, не зважаючи на переваги, MongoDB не підтримує транзакції та не дозволяє створити структуру колекцій даних, яка б дозволяла б нам записувати в певні поля тільки дані певного типу, або зробити якісь поля обов’язковими для заповнення. MongoDB дозволяє нам додавати в запис в колекцію навіть таке поле, якого не має в попередніх записах.

Враховуючи це, велика відповідальність лежить на розробниках серверного застосунку який здійснюватиме запити до даної бази. Тож будуть сформовані певні схеми для кожної колекції бази даних саме на нашому серверному застосунку, за допомогою бібліотеки Mongoose.js. В кожній схемі які відповідатимуть певній колекції будуть визначені поля які наявні в колекції та тип даних цього поля. Була розроблена фізичні модель нашої бази даних (дивись рисунок 3.2.).

В колекції зі схемою «Defibrillator» містяться записи про наявні в системі пристрої АЗД. В записах наявні поля, які описують назву, адресу, локацію (координати розташування пристрою в форматі [географічна довжина, географічна широта]), детальне місце розташування пристрою, наявність інформаційних табличок, номери телефонів відповідальних за дефібрилятор людей, масив посилань на зображення, додаткову інформацію та ідентифікатор користувача який додав даний пристрій у систему.

В колекції зі схемою «User» містяться записи про наявні в системі користувачі. В кожному записі даної колекції міститься електронна адреса користувача, його роль (адміністратор або модератор) та пароль (зашифрований за допомогою бібліотеки bcryptjs).

Між колекціями «User» та «Defibrillator» існує залежність один до багатьох, тобто один користувач може створити декілька дефібриляторів, того може бути декілька записів в колекції дефібриляторів, де полю owner відповідатиме ідентифікатор одного користувача.

Між колекціями «Defibrillator» та «Images.Files» існує залежність один до багатьох, тобто один запис про дефібрилятор може містити декілька фотографій

В колекції зі схемою «Images.Files» містяться записи про наявні в системі зображення. Записи цієї колекції містять інформацію про розмір зображення (в кілобайтах), дату завантаження, назву файлу та тип файлу.

В колекції зі схемою «Images.Chunks» містяться записи які містять поле data, в якому зберігається частина закодованого в бінарному форматі зображення. Записи цієї колекції містять посилання на запис з колекції «Images.Files», який відповідає зображенню частинка якого зберігається в заданому записі, тобто між колекціями «Images.Files» та «Images.Chunks» існує залежність один до багатьох, адже одне зображення може складатися з багатьох частин.



Рис. 3.2. Фізична модель бази даних

**3.4. Проектування серверної частини**

Як вже було визначено в загальному описі архітектури системи, нашому програмному застосунку знадобиться серверна частина, яка надаватиме нам REST API для запитів зі сторони клієнта.

Платформа Node.js дуже добре підходить для реалізації даної задачі, адже вона володіє неблокуючим вводом-виводом, що є дуже великим плюсом для реалізації API до якого будуть звертатися багато клієнтів. Для даної платформи ми маємо змогу використовувати дуже зручний фреймворк - Express.js, який дозволить швидко описати всі шляхи (API endpoints), на які будуть поступати запити, які будуть оброблятися. Також до даної платформи ми матимемо змогу додати ще декілька дуже корисних модулів, зокрема бібліотеку PassportJS. За допомогою даної бібліотеки здійснюватиметься авторизація користувачів, і перевірятиметься наявність прав в користувача на виконання тої ти іншої операції.

Для формування нашої бази даних було обрано MongoDB. Тому для зручності роботи серверної частини з MongoDB в проекті слід також використати бібліотеку Mongoose.js – це «обгортка», яка дозволяє сервері створювати зручні та функціональні схеми документів, да дозволяє здійснювати запити до бази даних, та отримувати від сервера бази даних відповідь.

Отже на серверній частині слід описати декілька шляхів (API endpoints), на які будуть поступати запити з клієнтської частини, і які ми будемо обробляти. В даній системі будуть організовані чотири основні групи кінцевих шляхів:

* /api/auth – містить кінцеві точки, для запитів, які напрямлені на роботу з базою даних користувачів, зокрема на реєстрацію нового користувача і перевірки коректності введеного при авторизації пароля.
* /api/defibrillators – група кінцевих точок, на які клієнт здійснюватиме запити для маніпуляції з колекцією дефібриляторів, зокрема для читання записів з даної колекції а також для додавання нових записів, видалення та редагування наявних. Варто зазначити, що для здійснень операцій редагування, видалення, та додавання, клієнт повинен також передати в заголовки для запиту токен, який засвідчує про наявність в нього прав на виконання цієї операції.
* /api/gMapboxMap – група кінцевих точок для здійснення запитів, які опрацьовуються сервером, і взаємодіють через HTTP з Google API для маніпуляції з геолокаціями.
* /api/images– містить кінцеві точки, для запитів, які напрямлені на роботу з базою даних зображень. Мова йде про додавання в систему нових зображень та зчитування існуючих.

Принцип роботи серверної частини системи, та візуалізація запитів, та подальше їхнє опрацювання вдало можна зобразити за допомогою діаграми послідовності (дивись Додаток А).

**3.5. Проектування клієнтської частини**

Для реалізації клієнтської частини буду використовувати фреймворк React, основною парадигмою якого є модульність. Для нас це означає, що застосунок буде складатися з незалежних блоків, які називатимуться компонентами, і які володітимуть власною структурою та поведінкою.

Весь код компоненти буде знаходитись в окремій папці, і для того, щоб видалити його з програми достатньо буде просто видалити відповідну папку. Видалення компонент не буде впливати на роботу інших компонент, проте система втратить частину функціоналу.

Кожна компонента може містити всередині інші підкомпоненти, а ті в свою чергу ще інші. Таким чином архітектура нашої аплікації буде виглядати як ієрархія компонент, кожна з яких буде володіти своїм станом та поведінкою. Для забезпечення глобального стану, який зможуть читати та змінювати всі наявні в системі компоненти, нам знадобиться менеджер станів – Redux. Основними парадигмами цього інструменту є: сховище, дії та редуктори. Саме в сховищі міститиметься глобальний стан нашої аплікації, а за допомогою дій та редукторів є можливість описати спосіб, та сценарій маніпулювання глобального стану (сховища). Тобто в будь-якій компоненті можливо буде підключити сховище, щоб читати актуальний стан, та дії, які ми описали, щоб його змінювати.

На найвищому рівні нашої клієнтської частини розміщені три модулі (дивись рисунок 3.3.):

* Sidebar – компонента, яка містить структуру для управління, та відображення бічного меню та її внутрішніх компонентів..
* MapHolder – компонента для роботи та відображення карти, містить шари для відображення іконок які символізують про наявність дефібриляторів, місця розташування користувача, вікна детальної інформації про АЗД та інші.
* Auth – компонента яка містить весь функціонал для авторизації та реєстрації користувача, а також модальні вікна які використовуються для даних операцій.

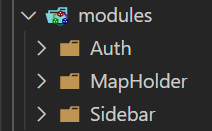


Рис. 3.3. Структура верхнього рівня клієнтського застосування

В клієнтському застосуванні також наявні модулі які часто використовуватимуться в інших компонентах, зокрема це:

* Компонента «Alert», яка містить вікно повідомлення, яке показується у випадку виконання користувачем деяких операцій.
* Компонента «ButtonBack», яка містить кнопку, при кліку на яку користувача відправлятиме на головну сторінку.
* Компонента «ConfirmationModalWrapper», в якій міститься модельне вікно для підтвердження дії, яку бажає виконати користувач.
* Компонента «Fields», в якій міститься поля введення, обгорнуті в сутність бібліотеки Formik, для зручного управління ними та перевірки коректності заповнення.
* Компонента «Form», в якій міститься форма, яка використовується для додавання нових дефібриляторів в систему, та редагування інформації про існуючі.
* В папці «http» містяться обгортки для http запитів за допомогою бібліотеки axios. Набагато зручніше буде використовувати дані обгортки для запитів, ніж знову їх описувати.
* Компонента «Loader», містить візуалізацію загрузки, яка використовується коли клієнт очікує відповіді від сервера.

Для зручності всі ці модулі були поміщені в окрему папку «shared» (дивись рисунок 3.4.), тому. якщо ми хочемо використати один з них у своїй компоненті, достатньо просто імпортувати цей модуль на початку файлу, за допомогою утиліти import.

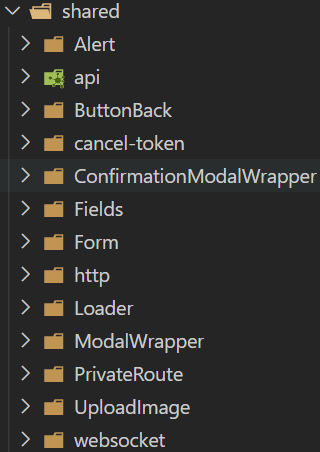


Рис. 3.4.Структура папки для збереження модулів які часто використовуються в системі

**Висновок до розділу 3**

В даному розділі було визначено загальну клієнт-серверну архітектуру системи, для відображення якої було сформовано діаграму розгортання. Також було описано доступні сценарії взаємодії з системою для всіх наявних класів користувачів, та сформовано діаграму прецедентів для наглядного відображення. Також було визначено структуру колекцій для баз даних, та зв’язки між ними. Для відображення наявних полів, та їхніх типів даних було сформовано фізичну діаграму бази даних. Оскільки серверна частина буде служити для нашої системи як REST API, було описано наявні кінцеві точки, на які клієнт буде відправляти запити, і сценарії для опрацювання кожного з цих запитів. Для демонстрації цього було сформовано діаграму послідовності.

**РОЗДІЛ 4**

**Реалізація та тестування веб-системи**

**4.1. Налаштування та реалізація серверної частини**

Ключовим інструментом для розробки серверної частини моєї системи є фреймворк express.js. За допомогою нього було визначено кінцеві точки, на які клієнт буде роботи запити.

app.use('/api/gMapboxMap', gMapboxMapRoute);

app.use('/api/auth', authRoute);

app.use('/api/defibrillators', defRoute);

app.use('/api/images', imageRoute);

app.use(passport.initialize());

require('./middleware/passport')(passport);

Функціонал обробки запитів для зручності був винесений в окремі файли. За допомогою цього ж фреймворку здійснюється запуск сервер на вибраному порту і відображаю на ньому клієнтську частину.

app.get('\*', (req, res) => {

  res.sendFile(

    path.resolve(\_\_dirname, 'client', 'build', 'index.html')

  );

});

Я використовував бібліотеку mongoose для взаємодії з базою даних, зокрема для визначення схеми колекцій та підключення бази даних до мого проекту:

mongoose.connect('mongodb+srv://Taras:123456789@boichuk-gt3rn.mongodb.net/defibrillatorDB', {

    useUnifiedTopology: true,

    useNewUrlParser: true,

    useFindAndModify: false,

    useCreateIndex: true

  });

**4.2. Опис роботи системи та програмних рішень на клієнтській частині**

Програмний продукт для пошуку геолокації та інформації про автоматичні дефібрилятори, повністю адаптивний, та доступний для використання через будь-який сучасний браузер, за умови наявності на пристрої користувача інтернету. Користувацький інтерфейс даної системи володіє всіма вимогами встановленими Якобом Нільсеном, тому він зручний та простий у використанні.

На рисунку 4.1. зображена десктопна версія користувацького інтерфейсу системи, яку бачить користувач при першому відкритті веб-застосунку . Як видно на рисунку, основними модулями клієнтської частини є бічне меню, яке розташоване зліва, та карта, яка розташована справа і займає більшу частину екрану.

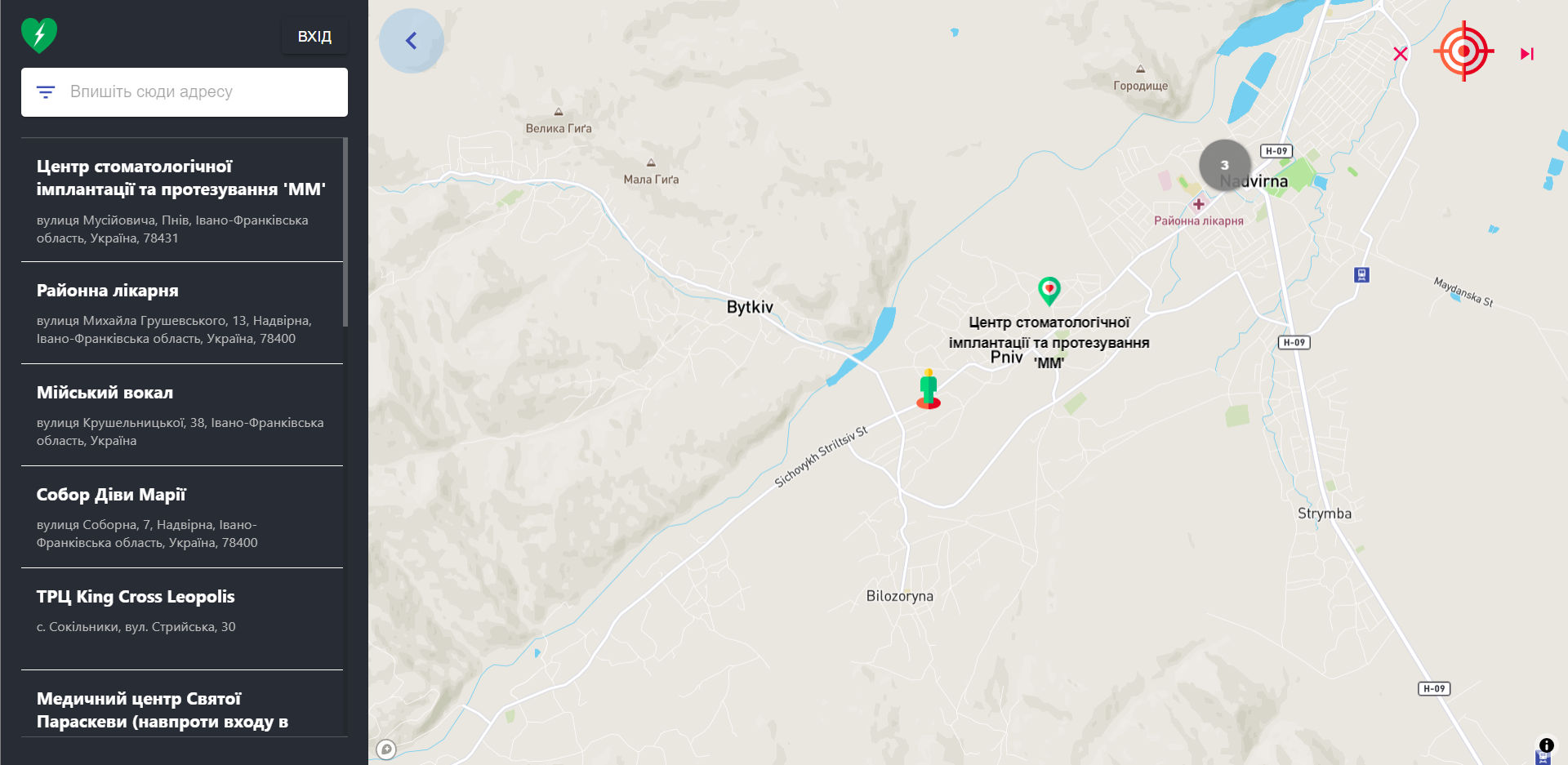


Рис. 4.1. Користувацький інтерфейс неавторизованого користувача для десктопу

Якщо поглянути на карту, то можна замітити, що в самому центрі розміщена специфічна іконка схожа на людину. Ця іконка відповідає локації користувача, і при першому відкритті веб-застосунку карта центрується саме на цьому місці. На карті також розміщені зелені іконки із зображенням серця всередині, які символізують про наявність у відповідному місці пристрою АЗД. Одразу під іконкою курсивним шрифтом відображена назва пристрою. На карті користувач також може бачити іконки у вигляді сірих кругів з числом в середині. Дане позначення - кластери символізує, що в цій локації розміщена кількість пристроїв яка відповідає числу всередині іконки, таким чином . Це реалізовано для того, щоб іконки АЗД не нагромаджували інтерфейс, проте при збільшенні масштабу користувач матиме змогу бачити іконки кожного дефібрилятору який входить до кластеру.

В правому верхньому куті карти розміщені три «екстрені кнопки» (Рисунок 4.2.).

При кліку на кнопку №2 карта на найближчому до користувача пристрої, і система проведе до нього маршрут.

При кліку на кнопку №3 буде відбуватися пошук наступного (в залежності від відстані до користувача, від найближчого до найдальшого) пристрою.

При кліку на кнопку №1 проведені раніше маршрути очищуються, та карта центрується на місці розташування користувача.

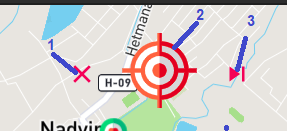


Рис. 4.2. Екстрені кнопки.

При кліку на іконку АЗД на карті, для користувача відкриється вікно, яке містить детальну інформацію про відповідний пристрій (Рисунок 4.3.).

При кліку на кнопку №1 система проведе маршрут до вибраного АЗД.

При кліку на кнопку №2 дане вікно закриється (воно також закриється якщо зумити карту або зробити клік поза цим вікном)

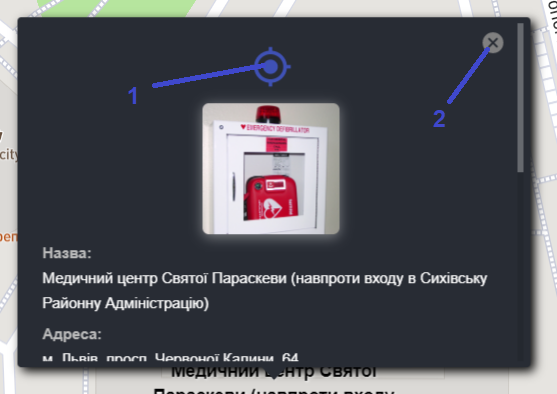


Рис. 4.3. Вікно з детальною інформацією про вибраний АЗД

В нижній частині вікна з детальною інформацією про АЗД розташована кнопка «Фотографії», при кліку на яку для користувача відкриється модальне вікно з фотографіями які прив’язані до цього пристрою (див. рисунок 4.4.). Якщо користувач буде авторизований як адміністратор, то для нього буде доступна можливість додавати нові фотографії, натиснувши на кнопку «Загрузити фото».

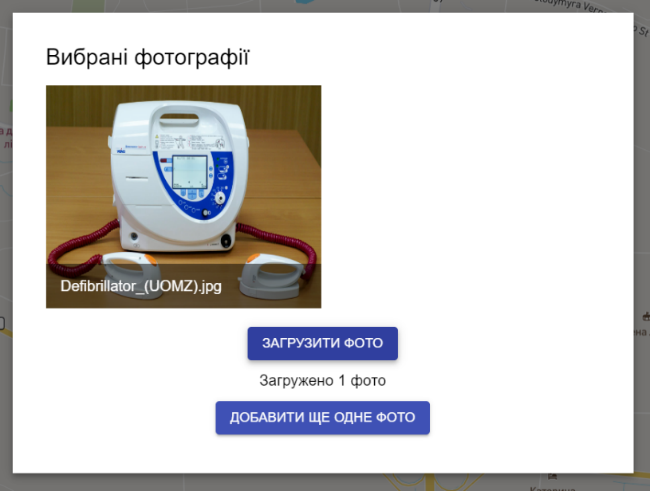


Рис. 4.4. модальне вікно з фотографіями вибраного АЗД

В бічному меню користувачу представлений список АЗД, в якому пристрої сортуються в залежності від відстані до локації користувача (чим вище АЗД в списку тим ближчий він до користувача). В списку реалізована "пагінація" - тобто на самому клієнті користувача при ініціалізації завантажуються тільки перші 9 АЗД, а при прокрутці вниз виконуються запити на сервер для загрузки в список інші пристроїв.

  const handleScroll = (event) => {

    const {scrolHeight, clientHeight, scrolTop } = event;

    if (

      totalCount >= page && scrollHeight - Math.ceil(scrollTop) <= clientHeight

    ) {

      fetchItems({ page, ...filter, ...search });

    }};

При кліку на один з елементів списку, карта відцентрується на відповідному пристрої, і елемент в списку буде відображатися як активний (буде підсвічуватись).

В бічному меню, над списком АЗД розміщене поле, в яке користувач може вводити адресу, і при введенні система динамічно буде відображатися в списку АЗД пристрої, адреса яких відповідає тому, що ввів користувач.

Зліва від кнопки для введення адреси міститься кнопка для відкриття фільтрів, натиснувши на яку буде відображено вікно з опціями для фільтрування (Рисунок 4.5), за якими користувач може фільтрувати список наявних в системі АЗД.

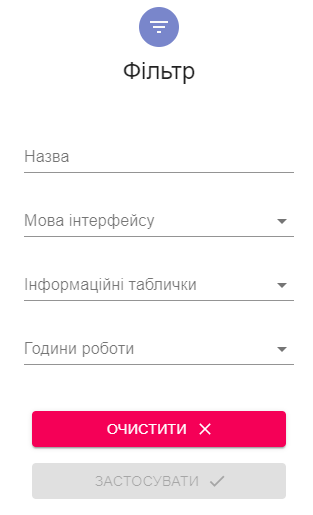


Рис. 4.5. вікно з опціями для фільтрування

У хедері бічного меню розміщена кнопка «Вхід», клікнувши на яку для користувача буде представлене модальне вікно для авторизації (див. рисунок 4.6.). Введені користувачем дані спочатку перевіряються на коректність на клієнті, а потім відправляються на сервер, де перевіряється наявність користувача з таким логіном і паролем в базі даних. У випадку знаходження відповідності, на сервері формується jwt token, який повертається на клієнт і використовується в подальшому для перевірки наявності прав у користувача на здійснення таких операцій як додавання, видалення чи редагування певного дефібрилятора. Для реалізації даного функціоналу на сервері було описано наступну кінцеву точку для запитів:

router.post(

  '/signin',

  async (req, res) => {

    const { email, password } = req.body;

    // Search document in table 'users' by email

    const user = await User.findOne({ email });

    if (candidate) {

      // Compare passwords using bcrypt

      const passwordResult = bcrypt.compareSync(

        password,

        user.password

      );

      if (passwordResult) {

        const { \_id, email, role } = user;

        // Create jwt based on id and email, expiration time = 1 hour

        const token = jwt.sign(

          { \_id, email, role },

          SECRET\_JWT\_KEY\_AUTH,

          { expiresIn: EXPIRE\_TIME\_JWT\_AUTH }

        );

        //OK - jwt and user information

        res

          .status(200)

          .set('Authorization', `Bearer ${token}`)

          .json({ \_id, email, role });

      } else {

        // Unauthorized - error message

        res.status(401).json({

          message:

            'Некоректна електронна адреса або пароль.\n Спробуйте ще раз.'

        });

      }

    } else {

      Unauthorized - error message

      res.status(401).json({

        message:

          'Некоректна електронна адреса або пароль.\n Спробуйте ще раз.'

      });

    }

  });

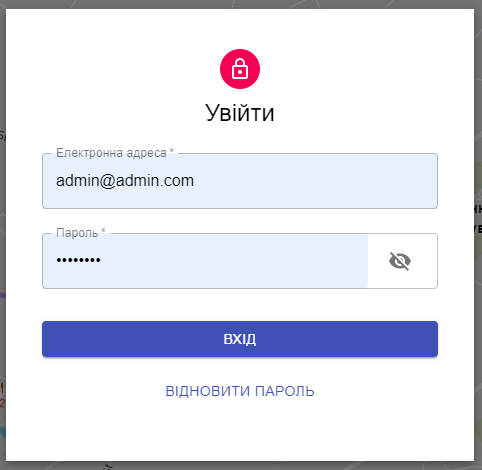


Рис. 4.6. модальне вікно для авторизації

Після успішної авторизації як адміністратор, бічне меню для користувача змінюється (див. рисунок 4.7.). Для нього з’являється можливість додавати новий АЗД в систему, клікнувши на кнопку «Додати дефібрилятор», яка знаходиться під списком АЗД. Також він може редагувати або видалити будь-який з АЗД, клікнувши на відповідну кнопку в елементі списку. Як можна замітити на рисунку в списку, для кожного дефібрилятора, доступна також третя кнопка з іконкою «блокування». Натиснувши на неї вона міняє колір з червоного на синій і навпаки. Якщо кнопка синя, то даний АЗД буде відображатися у списку для користувачів без прав адміністратора, якщо червона, то не буде.

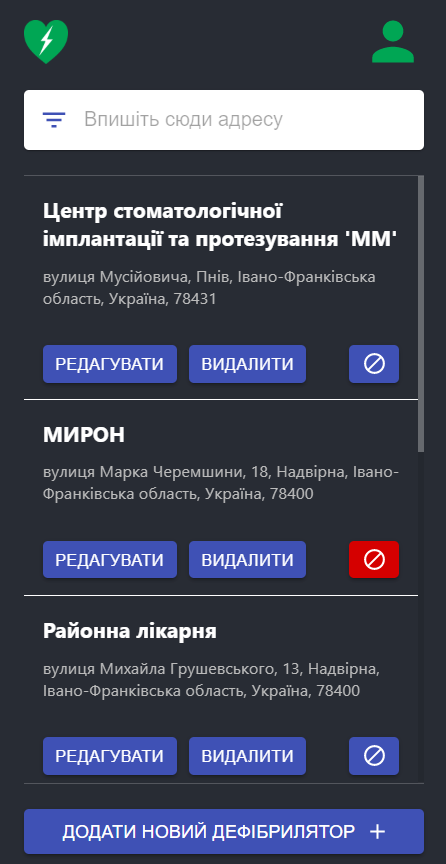


Рис. 4.7. вигляд бічного меню для користувача з правами адміністратора

Під час редагування або додавання дефібрилятора для користувача представлена форма яка зображена на рисунку 4.8. Під час додавання нового пристрою поля форми початково пусті, а під час редагування містять початкову інформацію про вибраний дефібрилятор.

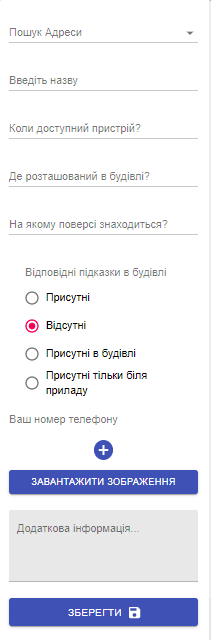


Рис. 4.8. Форма для додавання та редагування інформації про АЗД

Для реалізації цієї, та всіх інших форм у системі було використано бібліотеку Formik. За допомогою неї було описано правила для перевірки коректності введення в поля форми, а також поведінку при відправленні форми.

Поле «Пошук адреси» повинне містити адресу за якою знаходиться АЗД, і воно працює і виглядає аналогічно до поля пошуку в таких системах як "Google MapboxMaps" або "Yandex MapboxMaps", тобто при введенні кожного символу система здійснює асинхронний запит на Google MapboxMaps API, який повертає на клієнт варіанти співпадіння адрес.

useEffect(() => {

    if (value.length > 3) {

      (async () => {

        const countries = await getGeocodingOptions(value);

        setOptions(countries.data.predictions);

      })();

    } else {

      setOpen(false);

    }

  }, [value]);

Для додавання адреси в форму для додавання або редагування, користувач може здійснити подвійний клік по карті, після чого на карті буде встановлена мітка, яка свідчитиме про локацію, яка відповідає пристрою який редагується або додається, а також у полі «Пошук адреси» буде встановлено адресу яка відповідає цій локації. Для того, щоб реалізувати таку можливість, було додано функцію у компоненті MapHolder, яка викликається при подвійному кліку на карту, і яка встановлює у newPoint (значення з сховища Redux), координати місця на яке клікнув користувач. У компоненті AddAdressText, у випадку зміни значення newPoint, викликається наступна функція, яка здійснює запит на Google MapboxMaps API для отримання з наявних в змінній newPoint координат існуючої адреси.

  useEffect(() => {

    if (Object.keys(newPoint).length !== 0) {

      const result = await getReverseGeocoding(newPoint);

    const {lng,lat} = result.data.results[0].geometry.location;

    formik.setFieldValue(

      'address',

      result.data.results[0].formatted\_address

    );

    formik.setFieldValue('coordinates', [lng, lat]);

    }}, [newPoint]);

Для авторизованого як адміністратор користувача є можливість перейти в меню особистого кабінету (див. Рис. 4.9.), клікнувши на зелену іконку в правій частині хедера бічного меню. В даному меню є кнопки для реєстрації нового користувача, виходу з облікового запису та повернення на головну сторінку.

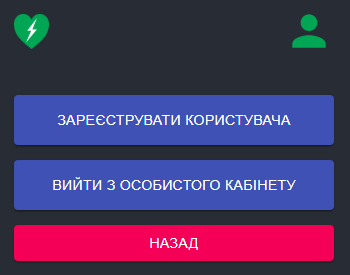


Рис. 4.9. Меню особистого кабінету

Користувач з правами адміністратора може зареєструвати в систему користувача з правами модератора, клікнувши на кнопку «Зареєструвати користувача», і в модальному вікні, що відкриється, ввести електронну пошту користувача якого реєструє. Після цього на вказану пошту система відправить повідомлення, яке містить посилання для продовження реєстрації. Користувач, який бажає зареєструватися, повинен перейти за цим посиланням. Після цього, йому буде представлена форма, яка містить поля для введення паролю для облікового запису. Ввівши пароль, і натиснувши кнопку «Реєстрація», користувач матиме змогу входити в систему з вказаним паролем та електронною поштою.

Мобільна версія веб-застосунку (див. рисунок 4.10.) сильно спрощена, тому в ній немає можливості працювати з бічним меню, а отже й адмініструвати інформацію про пристрої. Для користувачів що працюють з мобільною версією система також буде відображати на карті іконки, які визначають локацію користувача або пристроїв. Також залишається можливість користуватися «Екстреними кнопками» (див. рисунок 4.2.) які в даній версії розташованій внизу екрану користувача. Клікнувши на іконку АЗД, для користувача відкриється вікно з інформацією про пристрій, яке буде розміщуватись у верхній частині екрану та займатиме всю його ширину.

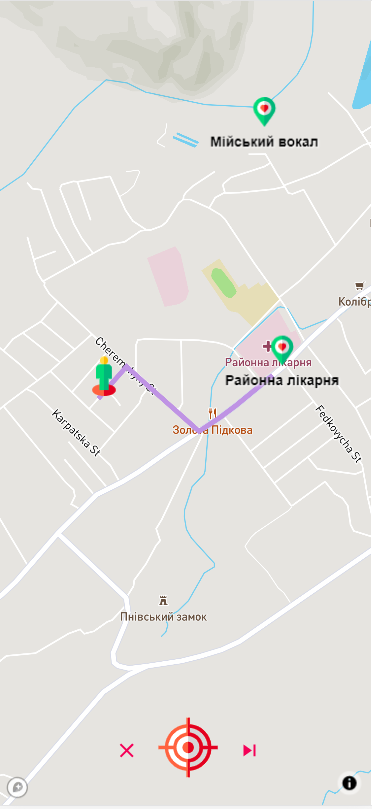


Рис. 4.10. Мобільна версія веб-застосунку

**4.3. Тестування програмного забезпечення**

Процес тестування є важливим етапом життєвого циклу ПЗ.

**4.3.1. Вступ**

В даному документі описуються результати тестування система пошуку геолокації та інформації про автоматичні дефібрилятори. Проводилось модульне, функціональне та тестування інтерфейсу. Дані види тестування дозволяють пересвідчитись у правильній реалізації функціонування програмного продукту на рівні коду та інтерфейсу користувача.

**4.3.2. Створення тестових випадків**

Тестові випадки було розбито на дві частини – для перевірки правильності виконання коду та тестування зручності використання.

**4.3.2.1. Модульне тестування**

Даний тип тестування проводився для перевірки правильності роботи окремих моделей, функцій та алгоритмів які містяться в системі. Для даного типу тестування в моєму проекті використано бібліотеку Jest, яка запускає тести, які розробник написав для перевірки функціональності, та аналізує певні модулі програми на відсоток покриття тестами. На клієнтській частині були сформовані тести для перевірки правильності роботи редюсерів, адже саме це модулі відповідають за більшість функціоналу на клієнті. На серверній ж частині були сформовані тести для перевірки коректності обробки запитів кінцевими точками які наявні в системі. Дані тести інтегровані в проект, тож розробник має змогу їх запустити викликавши в консолі наступні команди:

* yarn test-client – для запуску тестів для клієнтської частини.
* yarn test-server – для запуску тестів для серверної частини.

**4.3.2.2. Функціональне тестування**

Під час виконання функціонального тестування було розроблено двадцять тестових випадків та проведене позитивне (з коректними вхідними даними), та негативне тестування (з помилковими вхідними даними).

Таблиця 4.1

Варіанти використання для функціонального тестування

|  |  |  |
| --- | --- | --- |
| **Варіанти використання** | **Тестові випадки** | **Тестові дані** |
| Пошук найближчого АЗД | 1 | 2 |
| Перегляд детальної інформації про АЗД | 2 | 6 |
| Пошук АЗД за адресом | 3 | 6 |
| Фільтрування пристроїв | 3 | 16 |
| Авторизація | 3 | 9 |
| Редагування АЗД | 2 | 9 |
| Додавання нового АЗД | 2 | 11 |
| Блокування АЗД | 1 | 4 |
| Реєстрація нового користувача | 4 | 8 |
| **Разом** | **20** | **71** |

**4.3.2.3. Тестування інтерфейсу**

Даний тип тестування проводиться для перевірки правильності переходів між сторінками системи, навігації по карті, відкритті нових вікон та коректність та інтуїтивність роботи форм. Сформовано набір тестових випадків. (див. таблиця 4.2.)

Таблиця 4.2

Варіанти використання для тестування інтерфейсу

|  |  |
| --- | --- |
| Варіант використання | Тестові дані |
| Перевірка карти | 17 |
| Перевірка вікна з інформацією про АЗД | 24 |
| Перевірка головної сторінки | 36 |
| Перевірка особистого кабінету | 5 |
| Перевірка хедера | 10 |
| Перевірка вікна для авторизації | 5 |
| Перевірка вікна для реєстрації | 8 |
| Перевірка форми для додавання/редагування АЗД | 34 |
| Загалом | 139 |

**4.3.3. Модульне тестування**

Після запуску, усі тести на клієнтській та серверній частині були виконані успішно. Модульне тестування вважається успішним.

Результати модульного тестування зображені в додатку Г.

**4.3.4. Функціональне тестування**

Усі тестові випадки пройдено успішно, отже функціональне тестування вважається успішним. Варіанти використання зображені в додатку В.

**4.3.5. Тестування інтерфейсу**

На даному етапі були певні проблеми з навігацією по карті, зокрема після того, як користувач закінчував переміщати карту, відбувалася анімація збільшення масштабу. Також під час пошуку наступного дефібрилятора, шлях до найближчого не зникав. Вікно інформації про АЗД некоректно відображалося на мобільних пристроях. Дані проблеми були усунуті шляхом редагування компоненти клієнтської частини яка містять карту.

**4.3.6. Критерії успішності виконання тестування**

При проведенні тестування програмного продукту потрібно було забезпечити наступні умови:

* Усі алгоритми, функції та моделі програми працюють правильно.
* Всі вимоги представлені у специфікації, реалізовані у системі покриті тестами.
* Виконано тестування інтерфейсу користувача та перевірка його відповідності всім вимогам.

Беручи до уваги пункти які описують результати тестувань, можна зробити висновок, що тестування було успішно.

**Висновок до розділу 4**

В даному розділі було описано реалізації клієнтської та серверної частини системи, а також специфіку роботи та способи взаємодії з продуктом. Були представлені результати програмної реалізації ключових модулів, наявних в системі. Для запевнення в коректності роботи системи, проводилися три типи тестування: модульне, функціональне та тестування інтерфейсу. Всі тестові випадки були пройдені успішно. Було створено інструкцію користувача (див. додаток Д).

**РОЗДІЛ 5**

**Економічне обґрунтування доцільності роботи**

**5.1. Економічна характеристика проектного рішення (програмного продукту)**

Програмна система для пошуку геолокації та інформації про автоматичні зовнішні дефібрилятори, яку я розробляв для бакалаврської кваліфікаційної роботи, повинна дозволити будь-яким користувачам, які стали свідком того, що людина, яка знаходиться поряд, потерпає від раптової зупинки серця, швидко знаходити найближчий АЗД та шлях до нього, а також дозволить певним медичним структурам адмініструвати стан, інформацію та розташування цих пристроїв.

Основний функціонал програмного продукту – надання користувачам можливості отримати швидкий доступ до детального розташування та інформації про найближчий АЗД. Цей функціонал буде доступний користувачам без авторизації, без сплати та з будь-якого пристрою, на якому наявний веб-оглядач. Була вибрана така стратегія, адже у випадку екстреної ситуації, не буде доречним просити користувача здійснити авторизацію чи оплату, тому, що це не гуманно, та займе час (який є критично важливим при настанні раптової зупинки серця).

Міністерству охорони здоров’я потрібен інструмент для зручного адміністрування розміщення та стану цих пристроїв, адже саме вдале розміщення АЗД буде гарантувати найшвидший доступ до пристроїв Саме тому дана система може надати змогу органам охорони здоров’я адмініструвати систему, візуалізувати ситуацію з розміщенням АЗД для впровадження найбільш гармонічного розташування пристроїв, а також децентралізувати відповідальність за певні райони шляхом створення користувачів, які будуть слідкувати за станом дефібриляторів тільки у своєму районі зможуть редагувати інформацію тільки про АЗД, за належний стан і розташування яких вони несуть відповідальність.

В Україні на даний час не існує програмного продукту який би забезпечував вище згаданий функціонал ,отже розробка такого програмного забезпечення є економічно вигідною, оскільки вона здобуде підтримку серед суспільства та органів охорони здоров’я.

**5.2. Інформаційне забезпечення та формування гіпотези щодо потреби розроблення проектного рішення (програмного продукту)**

У всіх розвинутих країнах досліджена небезпека раптової зупинки серця, та визначений найбільш ефективний спосіб допомогти потерпілому у разі її настання – екстрена дефібриляція та серцево-легенева реанімація. З цієї причини, у розвинутих країнах, у місцях скупчення людей встановлені АЗД (автоматичний зовнішній дефібрилятор). В Україні тенденція на масове встановлення АЗД розпочалася нещодавно, тому програмний продукт який я розробив здатен оптимізувати впровадження та розміщення пристроїв, а також зробити їхній пошук максимально швидким і простим.

Для даного продукту не існує конкурентів на Українському ринку, адже, як я згадував в попередньому розділі, в Україні не існує програмної системи, яка б дозволяла шукати АЗД, чи адмініструвати їх. Навіть такий сервіс як Google maps, не відповідає всім вимогам, тому, що в цілях адміністрування потрібна закрита система з доступною авторизацією, яка буде доступна тільки користувачам, відповідальним за розміщення та стан дефібриляторів в певному районі, і які зможуть адмініструвати інформацію про пристрої, за які несуть відповідальність.

Для аналізу аналогічного продукту можна взяти мобільний додаток «AED Locations», який на сьогоднішній день ще не вийшов на глобальний ринок, і обслуговується тільки для одної країни, а саме Нової Зеландії, тому в даному додатку локації АЗД ви можете знайти тільки там. В даному програмному продукті не оптимізований процес адміністрування системи, тому, щоб доданий новий пристрій в систему потрібно відіслати дані про нього на вказану в додатку пошту, проте пошук функціонал для пошуку АЗД реалізований відмінно. Даний продукт фінансується приватними та державними медичними закладами.

Проаналізувавши вимоги на ринку, та функціональні можливості які забезпечує програмний продукт, можна зробити висновок, що дана система буде активно використовуватись людьми у випадку екстреної ситуації та структурами охорони здоров’я для адміністрування інформації про пристрої, тож проект має великі шанси здобути фінансування з державного бюджету виділеного Міністерством Охорони Здоров’я.

**5.3. Оцінювання та аналізування факторів зовнішнього та внутрішнього середовищ**

В даному розділі оцінено та проаналізовано фактори зовнішнього та внутрішнього середовищ групою експертів. Для оцінки зовнішніх факторів використовується шкала [-5; 5], де -5 — це негативний вплив на організацію, 5 — це позитивний вплив, а 0 — нейтральний вплив. Фактори внутрішнього середовища оцінюються за шкалою [0;5], де 0 демонструє низький розвиток, відсутність чи катастрофічний стан фактору внутрішнього середовища, оцінка 5 демонструє високий рівень розвитку даного фактору. Рівень вагомості для кожного фактору визначається за допомогою коефіцієнтів, сума всіх коефіцієнтів дорівнює одиниці. Зважений рівень впливу факторів дорівнює добутку впливу фактору у балах та рівня вагомості. Результати експертних оцінок впливу факторів зовнішнього середовища наведено у табл. 5.1.

Таблиця 5.1

**Результати експертного оцінювання впливу факторів зовнішнього та внутрішнього середовищ**

|  |  |  |  |
| --- | --- | --- | --- |
| **Фактори** | **Середня експертна оцінка, бали** | **Середня вагомість факторів** | **Зважений рівень впливу, бали** |
| *Фактори зовнішнього середовища* | | | |
| Споживачі | 4 | 0,11 | 0,44 |
| Постачальники | 0 | 0,1 | 0 |

Продовження табл. 5.1

|  |  |  |  |
| --- | --- | --- | --- |
| Конкуренти | 3 | 0,1 | 0,3 |
| Державні органи влади | 2 | 0,05 | 0,1 |
| Інфраструктура | 4 | 0,06 | 0,24 |
| Законодавчі акти | 4 | 0,1 | 0,4 |
| Профспілки, партії та інші громадські організації | 5 | 0,05 | 0,25 |
| Система економічних відносин в державі | 2 | 0,06 | 0,12 |
| Організації-сусіди | 0 | 0,01 | 0 |
| Міжнародні події | 4 | 0,01 | 0,04 |
| Міжнародне оточення | 4 | 0,03 | 0,12 |
| Науково-технічний прогрес | 5 | 0,07 | 0,35 |
| Політичні обставини | 2 | 0,06 | 0,12 |
| Соціально-культурні обставини | 5 | 0,05 | 0,25 |
| Рівень техніки та технологій | 4 | 0,04 | 0,16 |
| Особливості міжнародних економічних відносин | -2 | 0,02 | -0,04 |
| Стан економіки | -3 | 0,08 | -0,24 |
| Загальна сума |  | 1 | 2,33 |
| *Фактори внутрішнього середовища* | | | |
| Цілі | 5 | 0,11 | 0,55 |
| Структура | 3 | 0,16 | 0,48 |
| Завдання | 4 | 0,07 | 0,28 |
| Технологія | 4 | 0,2 | 0,8 |
| Працівники | 2 | 0,21 | 0,42 |
| Ресурси | 2 | 0,25 | 0,50 |
| Загальна сума |  | 1 | 3,03 |

Отже, зважений рівень впливу зовнішнього середовища є позитивним, та доволі значним, та становить 2,33. Найбільш позитивно на продукт впливають такі фактори як Законодавчі акти (Міністерство охорони здоров’я сприяє введенню законів про обов’язкову наявність АЗД в місцях скупчення людей), Соціально-культурні обставини (здоров’я та методи його забезпечення стають надзвичайно важливими для суспільства), Науково-технічний прогрес(практично в кожної особи є при собі мобільний пристрій, за допомогою якого вона зможе скористатися програмною системою у разі необхідності), Міжнародні події (досвід більш економічно-розвинутих країн доводить важливість пристроїв АЗД та методів швидкого доступу до них), Конкуренція (на ринку України не існує аналогів). Найбільшим негативним фактором є економічна система в державі через недофінансування та малу підтримку закладів та методів охорони здоров’я. У внутрішньому середовищі сумарний зважений рівень впливу становить 3,03.

Вплив зовнішнього та внутрішнього середовища є позитивним, і тому вихід на ринок є можливим без високих ризиків.

**5.4. Формування стратегічних альтернатив**

Формування стратегічних альтернатив є дуже важливим при становленні цілей організації, а також допомагає обрати найоптимальніший варіант з урахування її можливостей, недоліків та переваг. Зазвичай підприємству доводиться обирати стратегію серед кількох можливих варіантів. На рисунку 5.1. зображена перша група стратегічних альтернатив, критеріями поділу якої є існуючий та новий продукт (програмний засіб), а також відповідні послуги.

Існуюче

Нове

Стратегія розроблення нового продукту

Стратегія розвитку існуючого продукту

Продукт (проектне рішення)

Додаткові послуги **послуги**

Наявні

Відсутні

Стратегія нового продукту з супутніми послугами

Стратегія розвитку існуючого продукту

з супутніми послугами

Рис. 5.1 Стратегічні альтернативи першої групи

**Стратегія розроблення нового продукту (програмного засобу)** означає створенням нового програмного забезпечення, яке дає змогу вирішити новоутворені потреби людини, економіки, певного колективу або суспільства загалом тощо.

**Стратегія розвитку існуючого продукту (програмного засобу)** означає модифікацію програмного забезпечення а також його характеристик.

**Стратегія розвитку існуючого продукту (програмного засобу) з супутніми послугами** означає представлення на ринок модифікованого програмного забезпечення із додатковими послугами (встановлення, коригування, супроводження, адаптування до специфіки конкретного підприємства).

**Стратегія нового продукту (програмного засобу) з супутніми послугами** означає розроблення нового програмного забезпечення та пропонування додаткових послуг при його експлуатації.

На рисунку 5.2. зображена друга група стратегічних альтернатив, критеріями поділу якої є існуючий ринок та продукт (програмний засіб), новий ринок та продукт (програмний засіб).

Існуючий

Новий

Диверсифікація

Стратегія глибокого проникнення продукту

Продукт (проектне рішення)

**Ринок**

Новий

Існуючий

Стратегія розвитку продукту

Стратегія розвитку ринку

Рис. 5.2 Стратегічні альтернативи другої групи

**Глибоке проникнення на ринок** - це використання існуючого продукту для збільшення частки на існуючому ринку. При достатніх запасах ресурсів та великій потужності ця стратегія є найменш ризикованою, проте при активному зростанні на ринку збільшується конкуренція. Стратегія буде успішною якщо конкуренти будуть обмежені у ресурсах та потужностях або якщо ринок буде надзвичайно стрімко розвиватися.

**Стратегія розвитку ринку** – це стратегія яка полягає в використанні існуючого продукту або його модифікації для виходу на новий сегмент ринку. Ця стратегія має дуже великий ризик пов’язаний з тим, що успішність продукту даного типу на новому ринку ще не перевірена.

**Стратегія розвитку продукту** – це стратегія яка полягає у створенні нового продукту для існуючого сегменту ринку. Так як ця стратегія вимагає створення нового продукту для сформованого сегменту споживачів, то ризик є доволі високим. Однак, якщо ринок починає зменшувати обсяги та існуючий продукт є на етапі зрілості, тоді цю стратегію варто використовувати.

**Стратегія диверсифікації** – це стратегія, яка реалізується шляхом виходу на нові сфери бізнесу.

Проаналізувавши першу групу стратегічних альтернатив, я обрав стратегію нового продукту (програмного застосунку) з супутніми послугами, адже я реалізовую нову систему, для пошуку та адміністрування інформації про АЗД, яка не має абсолютних аналогів в даному сегменті ринку. І хоч на ринку і наявний такий гігант як Google maps, яким містить інформацію про деякі АЗД, за допомогою нього не можливо якісно адмініструвати інформацію про стан всіх пристроїв, а також в ньому не вистачає іншого функціоналу який наявний в системі яку я розробляю і який необхідний для поставлених в специфікації вимог задач.

З другої групи стратегічних альтернатив, було вибрано стратегія розвитку продукту з огляду на те, що розроблюване програмне забезпечення не можна вважати аналогічним до інших програмних продуктів, що вже існують на ринку, а передбачає багато нововведень та корисного функціоналу.

**5.5. Бюджетування**

Бюджетування є комплексно-обґрунтованою системою розрахунку витрат, які пов’язані з виготовленням та реалізацією продукту, що здійснює аналіз витрат та допомагає розробити заходи, щодо підвищення рентабельності виробництва. На даному етапі необхідно визначити собівартість продукту та забезпечити комерційну та виробничу діяльність підприємства.

Бюджет матеріальних витрат зображено в таблиці 5.2.

Таблиця 5.2

**Бюджет витрат матеріалів та комплектуючих виробів**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва матеріалів та комплектуючих | Марка, тип, модель | Фактична кількість, шт. | Ціна за одиницю, грн. | Разом, грн. |
| 1 | 2 | 3 | 4 | 5 |
| Папір | XEROX, А4, 80г/м2, 400арк | 1 | 80,0 | 80,0 |
| Флешка | Kingston DataTraveler SE9 16GB | 1 | 180,0 | 180,0 |
| Програмне забезпечення | Microsoft Office | 1 | 1150,0 | 1150,0 |
| Разом: | | | | 1410,0 |

Систему будуть розробляти троє осіб. Бюджет витрат на оплату праці наведений у таблиці 5.3*.*

Таблиця 5.3

**Бюджет витрат на оплату праці виробничих працівників**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Посада,  спеціальність | Кількість працівників, осіб | Час роботи, дні | Денна заробітна плата працівників, грн. | Сума витрат на оплату праці, грн. |
| *Основна заробітна плата* | | | | |
| Розробник клієнтської частини | 1 | 40 | 450,0 | 18000,0 |
| Розробник серверної частини | 1 | 20 | 500,0 | 10000,0 |
| Тестувальник | 1 | 9 | 400,0 | 3600,0 |
| Разом: | | | | 31600,0 |

Бюджет обов’язкових відрахувань та податків наведений в таблиці 5.4.

Визначаємо суму військового збору для кожної категорії працівників:

= 18000,0 \* 0,015 = 270,0 (грн)

= 10000,0 \* 0,015 = 150,0 (грн)

= 3600,0 \* 0,015 = 54,0 (грн)

Визначаємо суму податку з доходів фізичних осіб:

= 18000,0 \* 0,18 = 3240,0 (грн)

= 10000,0 \* 0,18 = 1800,0 (грн)

= 3600,0 \* 0,18 = 648,0 (грн)

Таблиця 5.4

**Бюджет обов’язкових відрахувань та податків**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Посада,  спеціальність | Сума  основної заробітної плати | Сума додаткової заробітної плати | Разом витрат на оплату праці | Сума податку з доходів фізичних осіб, грн. | Сума військового збору, грн.  (1,5%) |
| Розробник клієнтської частини | 18000,0 | - | 18000,0 | 3240,0 | 270,0 |
| Розробник серверної частини | 10000,0 | - | 10000,0 | 1800,0 | 150,0 |
| Тестувальник | 3600,0 | - | 3600,0 | 648,0 | 54,0 |
| Разом: | 31600,0 | - | 31600,0 | 5688,0 | 474,0 |

Бюджет загальновиробничих витрат зображено в таблиці 5.5.

'Таблиця 5.5

**Бюджет загальновиробничих витрат**

|  |  |
| --- | --- |
| Статті витрат | Сума, грн. |
| *Змінні загальновиробничі витрати, у т.ч.:* | |
| - заробітна плата допоміжного персоналу; | 2000,0 |
| - нарахування на заробітну плату (ЄСВ 22%) | 440,0 |
| - витрати на МШП; | 300,0 |
| - витрати на електроенергію; | 800,0 |
| - витрати на ремонт; | 900,0 |
| - інші змінні витрати; | 300,0 |
| Разом змінних витрат: | 4740,0 |
| *Постійні загальновиробничі витрати, у т.ч.:* | |
| - комунальні послуги; | 1000,0 |
| - витрати на оренду; | 3500,0 |
| - інші постійні витрати (амортизація); | 1500,0 |
| Разом постійних витрат: | 6000,0 |
| *Разом загальновиробничих витрат:* | 10740,0 |

Бюджет адміністративних витрат та витрат на збут наведено в таблиці 5.6.

Таблиця 5.6

**Бюджет адміністративних витрат та витрат на збут**

|  |  |
| --- | --- |
| Статті витрат | Сума, грн. |
| *Адміністративні витрати, у т.ч.:* | |
| - заробітна плата адміністративного персоналу; | 2500,0 |
| - нарахування на заробітну плату (ЄСВ 22%) | 550,0 |
| - витрати на МШП; | 300,0 |
| - витрати на відрядження; | 400,0 |

Продовження табл. 5.6

|  |  |
| --- | --- |
| - облаштування адміністративного обладнання; | 250,0 |
| - інші адміністративні витрати; | 300,0 |
| Разом адміністративних витрат: | 4300 |
| *Витрати на збут, у т.ч.:* | |
| - заробітна плата менеджерів зі збуту; | 3000,0 |
| - нарахування на заробітну плату (ЄСВ 22%) | 660,0 |
| - витрати на відрядження; | 500,0 |
| - витрати на гарантійне обслуговування; | 500,0 |
| - витрати на налагодження і експлуатацію; | 200,0 |
| - витрати на рекламу; | 900,0 |
| - інші витрати на збут; | 0,0 |
| Разом витрат на збут: | 5760,0 |

Витрати на виробництво продукції у вартісному виразі формують її виробничу собівартість. Цей показник є одним з найважливіших економічних показників господарської діяльності підприємства, у якому дістають відображення зростання продуктивності праці, економія ресурсів, технічний прогрес. Зведений кошторис витрат на розробку проектного рішення наведено в таблиці 5.7.

Таблиця 5.7

**Зведений кошторис витрат на розробку проектного рішення (продукту)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Статті витрат | Одиниці виміру | Фактична кількість, шт. | Ціна одиниці, грн. | Разом, грн. |
| Сировина і матеріали | шт. | 1 | 1410,0 | 1410,0 |
| Купівельні напівфабрикати та комплектуючі вироби | грн. | - | - | - |
| Зворотні відходи (вираховуються) | грн. | - | - | - |

Продовження табл. 5.7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Основна заробітна плата | грн. | - | - | 31600,0 |
| Додаткова заробітна плата | грн. | - | - | 0,0 |
| Відрахування на соціальне страхування (ЄСВ 22%) | грн. | - | - | 6952,0 |
| Витрати на утримання й експлуатацію устаткування | грн. | 3 | 80,0 | 240,0 |
| Загальновиробничі витрати, у т.ч.: | | | | |
| - змінні; | грн. | - | - | 4740,0 |
| - постійні; | грн. | - | - | 6000,0 |
| ***Разом виробничих витрат:*** | **грн.** | **-** | **-** | **50942,0** |
| Адміністративні витрати | грн. | - | - | 4300,0 |
| Витрати на збут | грн. | - | - | 5760,0 |
| Інші операційні витрати | грн. | - | - | - |
| ***Разом виробничих і операційних витрат:*** | **грн.** | **-** | **-** | **61002,0** |

Ціну продукту визначаємо за формулою:

Ц = СБ + СБ \* Р,

де Ц – ціна програмного продукту, грн.

СБ – собівартість програмного продукту, грн.Р – рентабельність, 23 %.

Ц = 50942,0 \* 0,23 + 50942,0‬‬ = 62658,66 грн.

Дохід від реалізації продукції = Ц \* к-сть реалізованої продукції = 62658,66 \* 1 = 62658,66 грн.

Валовий прибуток = Чистий дохід від реалізації продукції – Собівартість реалізованої продукції = 62658,66 – 50942,0 ‬‬ = 11716,66грн.

Фінансовий результат від операційної діяльності = Валовий прибуток - адміністративні витрати - витрати на збут - інші операційні витрати =

11716,66 – 4300,0– 5760,0 = 1656,66 грн.

Податок на прибуток = Фінансовий результат від операційної діяльності \* 18% = 1656,66 \* 0,18 = 298,19 грн.

Чистий прибуток (збиток) = Фінансовий результат від операційної діяльності - Податок на прибуток = 1656,66 – 298,19 = 1358,47 грн.

Бюджет фінансових результатів наведено в таблиці 5.8.

Таблиця 5.8

**Бюджет фінансових результатів**

|  |  |
| --- | --- |
| Показники | Сума, грн. |
| Дохід від реалізації продукції | 62658,66 |
| Податок на додану вартість (0 %) | 0,0 |
| Чистий дохід від реалізації продукції | 62658,66 |
| Собівартість реалізованої продукції | 50942,0 |
| Валовий прибуток | 11716,66 |
| Операційні витрати: | |
| - адміністративні витрати | 4300,0 |
| - витрати на збут | 5760,0 |
| - інші операційні витрати | 0,0 |
| Фінансовий результат від операційної діяльності | 1656,66 |
| Податок на прибуток (18 %) | 298,19 |
| Чистий прибуток (збиток) | 1358,47 |

**5.6. Вибір стратегії**

На основі отриманих результатів аналізу можна зробити висновок, що рівень впливу зовнішнього середовища є позитивним, та доволі значним (становить 2,33). Найбільш позитивно на продукт впливають такі фактори як законодавчі акти, соціально-культурні обставини, науково-технічний прогрес, міжнародні події та конкуренція. Найбільшим негативним фактором є економічна система в державі через недофінансування та малу підтримку закладів та методів охорони здоров’я. Серед внутрішніх факторів найбільш розвинутими є цілі та технології, сумарний зважений рівень впливу у внутрішньому середовищі становить 3,03. Беручи до уваги фактори зовнішнього та внутрішнього середовища, можна з стверджувати, що суттєві ризики під час розробки та впровадження програмного продукту відсутні.

Обрано стратегію нового продукту з супутніми послугами та його подальшим розвитком, тому що реалізовується нова система для пошуку геолокації та детальної інформації про АЗД, з можливістю адміністрування.

Конкуренція для даного програмного продукту практично відсутня, адже сервіси які доступні для використання в Україні не покривають необхідний функціонал для системи даного типу.

Проаналізувавши вимоги на ринку, та функціональні можливості які забезпечує програмний продукт, можна зробити висновок, що дана система буде активно використовуватись людьми у випадку екстреної ситуації та структурами охорони здоров’я для адміністрування інформації про пристрої, тож проект має великі шанси здобути фінансування з державного бюджету виділеного Міністерством Охорони Здоров’я.

**Висновки до розділу 5**

В економічній частині бакалаврської кваліфікаційної роботи обґрунтовано доцільність розробки існуючого продукту з супутніми послугами, розраховано всі його економічні характеристики. Період, протягом якого можливо здійснити розробку програмного продукту та запровадження його в дію, складає 50 днів. Загальна сума виробничих витрат 50942,0 грн. Також під час розробки наявні адміністративні витрати – 4300,0 грн та витрати на збут – 5760,0 грн. Витрати на збут виявились настільки значні, адже велика частка коштів буде виділена на рекламу, для того, щоб проінформувати населення про її існування, адже чим більше людей знатимуть про даний продукт – тим більше скористаються, та можливо врятують людське життя, вчасно знайшовши АЗД. Чистий прибуток склав тільки – 1358 грн, через низький відсоток рентабельності та великі виробничі витрати.

# ВИСНОВКИ

Під час виконання даної бакалаврської кваліфікаційної роботи було досліджено небезпеку, та поширеність такого явища як серцево судинна смерть. Було визначено, що ключовим аспектом, який збільшує шанси потерпілого на виживання, є наявність та максимально швидка доступність до АЗД. Був здійснений аналіз ринку, на наявність продуктів, які б могли оптимізувати пошук та адміністрування АЗД. Після аналізу було визначено, що на ринку наявна дуже невелика кількість систем які покривають даний функціонал , а ті, що існують, не обслуговують інформацію про АЗД в Україні та не відповідають всім вимогам. Саме тому було прийняте рішення про розробку програмної системи для пошуку геолокації та інформації про АЗД.

Першим етапом розробки стала постановка задачі, під час якої було здійснено специфікацію вимог, та описано майбутній функціонал яким повинна володіти система. Для виконання поставлених задач було вирішено, що для розробки буде використовуватися платформа Node.js а також такі бібліотеки як Mongoose, Passport.js та Express.js. Клієнтську частину було вирішено розробляти з використанням бібліотеки React, та великої кількості суміжних пакетів для роботи з формами та картами.

Під час розробки архітектури системи, та проектування були створені діаграми прецедентів, використання та розгортання. Було спроектовано фізичну модель бази даних, яка в подальшому використовувалась для опису схем бази даних за допомогою бібліотеки Mongoose. Також було описано загальну структуру модулів клієнтської та серверної частини.

Під час етапу розробки, був написаний вихідний код для серверної та клієнтської частини. Були проведені, та успішно пройдені три типи тестування: модульне, функціональне та тестування інтерфейсу. Завершений веб-застосунок був розміщений на хмарному сервісі Heroku, а база даних на сервісі MongoDB Atlas. Посилання на робочу версію програмного продукту - <https://doplom.herokuapp.com/>

**СПИСОК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ**

1. Як користуватися АЗД - [Електронний ресурс] – Режим доступу : http://www.ucnovator.com.ua/%D0%B4%D0%B5%D1%84%D1%96%D0%B1%D1%80%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80.
2. Підручник з Захисту Вітчизни. Застосування АЗД - [Електронний ресурс] – Режим доступу : <https://history.vn.ua/pidruchniki/gydima-national-defense-bases-medical-knowledge-girls-10-class-2018/8.php>.
3. Петиція для встановлення АЗД в місці Чернівці - [Електронний ресурс] – Режим доступу : <https://gb.city.cv.ua/projects/4>
4. Факти про АЗД - [Електронний ресурс] – Режим доступу : <http://www.motodopomoga.lviv.ua/archives/535>.
5. Серцево-легенева реанімація - [Електронний ресурс] – Режим доступу : <https://ns-plus.com.ua/2019/06/20/sertsevo-legeneva-reanimatsiya/>.
6. Miles R. Learning UML 2.0: A Pragmatic Introduction to UML, 2006.–292 с.
7. Шлуночкові порушення ритму серця та раптова серцева смерть - [Електронний ресурс] – Режим доступу : <https://compendium.com.ua/uk/clinical-guidelines-uk/cardiology-uk/section-13-uk/glava-6-shlunochkovi-porushennya-ritmu-sertsya-ta-raptova-sertseva-smert/>.
8. Верчено П.І., Вітлінський В.Р., Навчально методичний посібник для самостійного вивчення дисципліни. Аналіз, моделювання та управління економічним ризиком:– К.: КНЕУ, 2003. - 290с.
9. Документація бібліотеки React - [Електронний ресурс] – Режим доступу :  [https://uk.reactjs.org/docs/getting-started.html/](https://compendium.com.ua/uk/clinical-guidelines-uk/cardiology-uk/section-13-uk/glava-6-shlunochkovi-porushennya-ritmu-sertsya-ta-raptova-sertseva-smert/).
10. Документація бібліотеки MapboxMapbox GL JS - [Електронний ресурс] – Режим доступу : https://www.MapboxMapbox.com/.
11. Документація по роботі з фреймворком Express.js - [Електронний ресурс] – Режим доступу : https://expressjs.com/ru/guide/routing.html.
12. Програмна продукція та ПДВ - [Електронний ресурс] – Режим доступу : http://www.visnuk.com.ua/uk/publication/100008243-programna-produktsiya-ta-pdv.
13. Документація по роботі з бібліотекою Mongoose - [Електронний ресурс] – Режим доступу : https://metanit.com/web/nodejs/6.6.php.
14. Документація по роботі з бібліотекою Passport.js - [Електронний ресурс] – Режим доступу : http://www.passportjs.org/docs/.
15. Автоматичні зовнішні дефібрилятори та кардіовертери-дефібрилятори: значення і перспективи широкого застосування - [Електронний ресурс] – Режим доступу : <http://www.journal.ukrcardio.org/cardio_archive/2005/4D/kovalenko2.htm>.
16. Що таке раптова смерть - [Електронний ресурс] – Режим доступу : http://www.diagnoz.lutsk.ua/news/shcho-take-raptova-sertseva-smert.

# ДОДАТОК А. Варіанти використання



# ДОДАТОК Б. Діаграма прецедентів

# 

# ДОДАТОК В. Варіанти використання

FT1. Пошук найближчого АЗД.

|  |  |  |
| --- | --- | --- |
|  | Дія | Очікуваний результат |
|  | Користувач клікає на кнопку для пошуку найближчого дефібрилятора | Визначається найближчий дефібрилятор, карта центрується на його локації, та проводиться найкоротший шлях від локації користувача до локації визначеного пристрою. |

FT2. Перегляд детальної інформації про АЗД.

|  |  |  |
| --- | --- | --- |
|  | Дія | Очікуваний результат |
|  | Користувач клікає на іконку АЗД на карті | Над іконкою на яку клікнув користувач з’являється вікно з інформацією про пристрій та кнопкою для проведення шляху до нього. |

FT3. Пошук АЗД за адресом.

|  |  |  |
| --- | --- | --- |
|  | Дія | Очікуваний результат |
|  | Користувач вводить адрес в полі для введення адресу над списком дефібриляторів. | При введенні кожного символу, в списку дефібриляторів виводяться тільки ті елементи які містять в атрибуті адреси значення введене користувачем. |

FT4. Застосування опцій фільтрації до списку пристроїв.

|  |  |  |
| --- | --- | --- |
|  | Дія | Очікуваний результат |
|  | Користувач клікає на копку «Відкрити фільтр» | Відкривається вікно з опціями фільтрації |
|  | Користувач вибирає певні фільтри та натискає на кнопку «Застосувати» | В списку АЗД відображаються тільки ті пристрої, які відповідають зазначеними користувачем фільтрами. |

FT5. Авторизація в систему з коректними даними

|  |  |  |
| --- | --- | --- |
|  | Дія | Очікуваний результат |
|  | Користувач клікає на копку «Вхід», яка знаходиться в хедері бічного меню. | Відкривається модальне вікно авторизації. |
|  | Користувач вводить коректні дані для входу в систему. | Модальне вікно зникає, в хедері на місці кнопки «Вхід» з’являється іконка «Особистий кабінет». В залежності від ролі користувача, в бічному меню з’являються кнопки для адміністрування АЗД. |

FT6. Редагування вибраного дефібрилятора

|  |  |  |
| --- | --- | --- |
|  | Дія | Очікуваний результат |
|  | Авторизований користувач клікає на копку «редагувати», яка знаходиться в елементі списку АЗД, який відповідає пристрою, інформацію про який користувач бажає змінити. | В бічному меню відкривається форма, в полях якої міститься поточна інформація про вибраний пристрій. |
|  | Користувач некоректно заповнює поля (необхідні поля залишає пустими, в полі адреси вписує неіснуючу адресу, або ж вводить номер в некоректному форматі), та натискає на кнопку «Зберегти» | Навіть під час того, як користувач закінчивши редагування певного поля, залишив його в некоректному форматі, система підсвітить поле червоним кольором, та виведе повідомлення про помилку. Якщо користувач проігнорує помилки, та натисне кнопку «Зберегти», буде виведено повідомлення в модальному вікні з текстом «Дані полів введені некоректно». |
|  | Користувач коректно заповнює поля та натискає на кнопку «Зберегти». | Буде виведено повідомлення в модальному вікні з текстом «Додавання пройшло успішно», користувача направить на головну сторінку та оновлена інформація про дефібрилятор буде відображатися в системі. |

FT7. Додавання нового дефібрилятора

|  |  |  |
| --- | --- | --- |
|  | Дія | Очікуваний результат |
|  | Авторизований користувач клікає на копку «Додати новий дефібрилятор», яка знаходиться в під списком АЗД. | В бічному меню відкривається форма додавання, поля якої пусті. |
|  | Користувач некоректно заповнює поля (необхідні поля залишає пустими, в полі адреси вписує неіснуючу адресу, або ж вводить номер в некоректному форматі), та натискає на кнопку «Зберегти» | Навіть під час того, як користувач закінчивши редагування певного поля, залишив його в некоректному форматі, система підсвітить поле червоним кольором, та виведе повідомлення про помилку. Якщо користувач проігнорує помилки, та натисне кнопку «Зберегти», буде виведено повідомлення в модальному вікні з текстом «Дані полів введені некоректно». |
|  | Користувач коректно заповнює поля та натискає на кнопку «Зберегти». | Буде виведено повідомлення в модальному вікні з текстом «Додавання пройшло успішно», користувача направить на головну сторінку та новий пристрій буде відображатися на карті та у списку. |

FT8. Блокування АЗД

|  |  |  |
| --- | --- | --- |
|  | Дія | Очікуваний результат |
|  | Користувач з правами адміністратора клікає на копку «Блокування», яка знаходиться в елементі списку АЗД, який відповідає пристрою, який адміністратор бажає заблокувати. | Виводиться модальне вікно для підтвердження блокування. |
|  | В модальному вікні користувач натискає на кнопку «Так». | Модальне вікно зникає, кнопка для блокування на яку клікав користувач міняє колір на червоний. Заблокований дефібрилятор не буде відображатися для користувачів без прав адміністратора. |

FT9. Реєстрація користувача

|  |  |  |
| --- | --- | --- |
|  | Дія | Очікуваний результат |
|  | Користувач з правами адміністратора клікає на копку «Особистий кабінет» в хедері бічного меню | В бічному меню відображається три кнопки:   * «Зареєструвати користувача». * «Вийти з особистого кабінету». * «Назад». |
|  | Користувач вводить в поле для електронної адреси некоректне значення. | Поле підсвічується червоним кольором, та під ним виводиться повідомлення «Некоректно введена електронна адреса» |
|  | Користувач вводить в поле для електронної адреси значення, яке відповідає користувачу який вже наявний в системі. | Поле підсвічується червоним кольором, та під ним виводиться повідомлення «Ця електронна адреса вже комусь належить» |
|  | Користувач вводить в поле для електронної адреси коректне значення, та натискає на кнопку «Реєстрація». | Виводиться повідомлення про успішність виконання операції, на вказану електронну пошту відправляється повідомлення дл продовження реєстрації. |

# ДОДАТОК Г. Звіт про модульне тестування

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тестування серверної частини    Тестування кнієнтської частини | | | | | |
| Виконав |  | Підпис |  | Звіт про модульне тестування | Сторінка |
| Перевірив |  | Підпис |  | 1 з 1 |
| ІКНІ кафедра ПЗ П3-41 | | | |

# ДОДАТОК Д. Інструкція користувача

1. **Компоненти ПЗ**

Програмна система розроблена з використанням платформи Node.js для серверної частини, та бібліотеки React для клієнтської. Середовищем в якому розроблялась система є засіб для створення і редагування програм - Visual Studio Code. Продукт можна розгорнути на комп’ютері з будь-якою операційною системою, процесором не менше 1.6 GHz та оперативною пам’яттю від 512 Mb. На машині також повинен бути встановлений Node.js. Для роботи системи необхідний доступ до бази даних Mongodb, яка може розміщуватись на зовнішньому сервері.

1. **Встановлення ПЗ**

Робоча версія веб-застосунку розташована на хмарному сервісі Heroku, тому можна перейти за посиланням - <https://doplom.herokuapp.com/>, для того, щоб користуватися продуктом.

Якщо є необхідність розгорнути веб-застосунок на локальній машині, необхідно спочатку завантажити папку з проектом на локальний диск. Першочергово необхідно перейти в режим командного рядка, де перейти в кореневу папку проекту та ввести команду «yarn install», після в проект будуть встановлені всі необхідні пакети. Потім слід ввести команду 'yarn start', після чого дочекатися повідомлення про те, що сервер запущений на порту 3012. В будь якому веб-оглядачі (бажано в Google Chrome) перейти за посиланням - <http://localhost:3012/>. При першому веб-застосунку необхідно дозволити програмі відстежувати ваше місцезнаходження (без даного кроку основний функціонал системи буде недоступний).

Запускати та конфігурувати клієнтську частину не потрібно, адже в папці client/build/ міститься заздалегіть скомпільована версія, яку використовує сервер при роботі. Для під'єднання та конфігурації бази даних ніяких маніпуляцій проводити не потрібно, адже база даних була розміщена на хмарному сервісі MongoDB Atlas, та заздалегідь приєднав до проекту. Якщо виникла необхідність приєднати до проекту іншу сутність бази даних, необхідно перейти у файл db.js, який також знаходиться в кореневій папці проекту, де вказати дані для підключення зовнішньої або внутрішньої сутності бази даних MongoDB. У файлі package.json описані міграційні скріпти, які можуть бути корисними, якщо є необхідність створити необхідні для роботи системи колекції в базі даних, а також адміністраторів системи.

1. **Базові функції**

Будь який користувач який володіє сучасним веб-оглядачем може, не авторизуючись, отримати доступ до наступного функціоналу:

* Пошук найближчого до них пристрою АЗД та побудова до нього найкоротшого шляху (потрібно клікнути на «Екстрену кнопку»).
* Перегляд детальної інформації про вибраний дефібрилятор (потрібно клікнути іконку про наявність АЗД на карті).
* Перегляд наявних в системі пристроїв з можливістю фільтрування за адресом, назвою, мовою інтерфейсу та годинами роботи (засоби для фільтрації знаходяться над списком дефібриляторів у бічному меню).

Веб-застосунок містить можливість авторизуватися як адміністратор, для того, щоб адмініструвати інформацію про наявні дефібрилятори в системі (додавати новий дефібрилятор, видаляти, блокувати чи редагувати наявний) або реєструвати нових користувачів (потрібно клікнути на іконку «Особистий кабінет» в хедері бічного меню, і потім натиснути на кнопку «Зареєструвати користувача») які володітимуть обмеженими правами – модераторів. Модератори можуть додавати дефібрилятори в систему, проте редагувати вони можуть лиш ті пристрої які самі ж додали. Функція блокування для користувачів з правами модератора відсутня.

Для того, щоб авторизуватися як адміністратор, потрібно в хедері бічного меню натиснути на кнопку «Вхід», після чого в модальному вікні яке відкриється ввести наступні дані:

Електронна адреса: [admin@admin.com](mailto:admin@admin.com)

Пароль: qwe123Q!

1. **Аналіз помилок**

Дана система виводить інформаційні повідомлення у випадку будь-яких помилок, або відсутності підключення до бази даних чи інтернету. Код та опис помилок які перехоплюються системою будуть виведені у терміналі з якого запущено програмну систему.

# ДОДАТОК Е. Компонента що містить карту, та її функціонал

**MapHolder.js**

import React, {useState,useEffect} from "react";

import { geolocated } from "react-geolocated";

import { makeStyles } from "@material-ui/core/styles";

import { connect } from "react-redux";

import { MapboxMapContainer, showMenuIcon } from "./styles";

import Media from "react-media";

import {Button,Tooltip} from "@material-ui/core";

import ReactMapboxMapboxGl, {

 GeoJSONLayer

} from "react-MapboxMapbox-gl";

import PopupHolder from "./components/PopupHolder";

import {

 setMapboxMapCenter,

 setMapboxMapZoom,

 NewPoint,

 selectOndeDeff

} from "../actions/MapboxMapState";

import SearchBar from "./components/SearchBar";

import { hidePopup } from "../actions/DisplayPopup";

import DefibrillatorPinLayer from "./layers/DefibrillatorPinLayer";

import PinUser from "./layers/PinUser";

import AddedPin from "./layers/AddedPin";

import { sidebarWidth } from ".../Sidebar/styleConstants";

import ChevronRightIcon from "@material-ui/icons/ChevronRight";

const useSearchBarStyle = makeStyles(

 () => ({desktopStyle: desktopStyle},

  mobileStyle: mobileStyle

);

const useStyles = makeStyles(() => ({

 MapboxMap: MapboxMap,

 MapboxMapContainer: ({ visible }) => MapboxMapContainer,

 showSidebarIcon: showSidebarIcon,

 showMenuIcon: ({ visible }) => showMenuIcon)

const MapboxMap = ReactMapboxMapboxGl({

 accessToken:"TOKEN"});

const MapHolder = ({

 MapboxMapState,

 newPoint,

 visible,

 selectedDeff,

 setMapboxMapCenter,

 setVisible,

 cords,

NewPoint,

 hidePopup,

}) => {

 const classes = useStyles({ visible });

 const SearchBarStyle = useSearchBarStyle();

 const [MapboxMap, setLocalMapboxMap] = useState(

  null

 );

 const [geoJSON, setGeoJSON] = useState(

  {}

 );

 useEffect(() => {

  if (coords) {

   setMapboxMapCenter({

    lng: coords.longitude,

    lat: coords.latitude

   });

  }

 }, [coords]);

 useEffect(() => {

  if (

   coords &&

   Object.keys(selectedDeff).length !==

    0

  ) {

   document.addEventListener(

    "click",

    handlePopupClose

   );

   const {

    longitude,

    latitude

   } = coords;

   const { lng, lat } = selectedDeff;

   fetch(

    `https://api.MapboxMapbox.com/directions/v5/MapboxMapbox/driving/${longitude},${latitude};${lng},${lat}?access\_token=pk.eyJ1Ijoib3Nrb3ZiYXNpdWsiLCJhIjoiY2s1NWVwcnhhMDhrazNmcGNvZjJ1MnA4OSJ9.56GsGp2cl6zpYh-Ns8ThxA&geometries=geojson`

   )

    .then(response => {

     return response.json();

    })

    .then(data => {

     setGeoJSON({

      type: "FeatureCollection",

      features: [

       {

        type: "Feature",

        geometry:

         data.routes[0].geometry

       }

      ]

     });

    });

  }

  return () => {

   document.removeEventListener(

    "click",

    handlePopupClose

   );

  };

 }, [selectedDeff, coords]);

 const loadMapboxMap = MapboxMapRaw => {

  if (MapboxMapRaw) {

   setLocalMapboxMap(MapboxMapRaw);

  }

 };

 const { lng, lat, zoom } = MapboxMapState;

 const changeMapboxMapCenterCoords = event => {

  setMapboxMapCenter(event.getCenter());

 };

 const hideSidebar = () => {

  if (MapboxMap) {

   setVisible(prev => !prev);

   setTimeout(() => {

    MapboxMap.resize();

   }, 150);

  }

 };

 useEffect(() => {

  if (

   Object.keys(newPoint).length !== 0

  ) {

   const { lng, lat } = newPoint;

   setMapboxMapCenter({ lng, lat });

  }

 }, [newPoint]);

 const onDblClickMapboxMap = (\_, event) => {

  const myRoute =window.location.pathname;

  if (

   myRoute == "/add-form" ||

   myRoute.includes("/edit-form")

  ) {

   const { lng, lat } = event.lngLat;

   NewPoint({ lng, lat });

  }

 };

 const linePaint = {

  "line-color": "#BF93E4",

  "line-width": 5

 };

 return (

  <div className={classes.MapboxMapContainer}>

   <Media >

    {matches =>

     matches ? (

      <Button

       className={classes.showSidebarIcon}

       color="primary"

       onClick={hideSidebar}

       size="small"

      >

       {}

       <Tooltip title={tooltipMessage}>

        <ChevronRightIcon

         className={

          classes.showMenuIcon

         }

        />

       </Tooltip>

      </Button>

     ) : (

      <></>

     )

    }

   </Media>

   <Media>

    {matches =>

     matches ? (

      <SearchBar

       style={

        SearchBarStyle.desktopStyle

       }

      />

     ) : (

      <SearchBar

       style={

        SearchBarStyle.mobileStyle

       }

      />

     )

    }

   </Media>

   <MapboxMap

    zoom={[zoom]}

    className={classes.MapboxMap}

    center={[lng, lat]}

    onStyleLoad={loadMapboxMap}

style="\*\*"

    onZoomStart={onZoomStarted}

    onRotateEnd={changeMapboxMapCenterCoords}

    onDragEnd={changeMapboxMapCenterCoords}

    onDblClick={onDblClickMapboxMap}

   >

     <AddedPin coordinates={newPoint} />

    {MapboxMap && (<DefibrillatorPinLayer MapboxMap={MapboxMap} />

    )}

    {coords && (

     <PinUser

      coordinates={{

       lng: coords.longitude,

       lat: coords.latitude

      }}

     />

    )}

    {Object.keys(geoJSON).length !==

     0 &&

     Object.keys(selectedDeff)

      .length !== 0 && (

      <GeoJSONLayer data={geoJSON} linePaint={linePaint}/>

     )}

    <PopupHolder />

   </MapboxMap>

  </div>

 );

};

export default geolocated({

watchPosition: false,

 positionOptions: {

  enableHighAccuracy: false

 },

  userDecisionTimeout: 7000

})(connect(

  state => ({

   MapboxMapState: state.MapboxMapState,

   newPoint: state.newPoint,

   defsState: state.defs,

   selectedDeff: state.selectedDeff

  }),

  dispatch => ({

   setMapboxMapCenter: MapboxMap =>

    dispatch(setMapboxMapCenter(MapboxMap)),

   setMapboxMapZoom: zoom =>

    dispatch(setMapboxMapZoom(zoom)),

   selectOndeDeff: selectedDeff =>

    dispatch(selectOndeDeff(selectedDeff)),

   hidePopup: () =>

    dispatch(hidePopup())

   NewPoint: newPoint =>

    dispatch(NewPoint(newPoint)),

  })

 )(MapHolder)

);