**Міністерство освіти і науки України**

**Національний університет «Львівська політехніка»**

**Кафедра програмного забезпечення**

**Звіт**  
про переддипломну практику за темою бакалаврської кваліфікаційної роботи

«Система для пошуку геолокації та інформації про автоматичні дефібрилятори»

студента IV курсу групи ПЗ-42

Бойчука Тараса Віталійовича

**база практики** ФОП Сатурський Орест Андрійович

**термін практики** з «13 квітня 2020 року » до « 30 квітня 2020 року»  
 

**Керівники практики**

від кафедри доцент кафедри ПЗ Тушницький Р.Б.

(підпис)

від бази практики ФОП Сатурський Орест Андрійович

(підпис)       
 

**Керівник бакалаврської**  доцент кафедри ПЗ Коротєєва Т.О.

(підпис)

**кваліфікаційної роботи**

**Оцінка**\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Дата**\_\_\_\_\_\_\_\_\_\_\_\_\_

Львів 2020

**ЗМІСТ**

[**Опис місця практики 3**](#_Toc39664426)

[**Завдання від бази практики 4**](#_Toc39664427)

[**Виконання завдання від бази практики 6**](#_Toc39664428)

[**Завдання дипломного проекту 9**](#_Toc39664429)

[**РОЗДІЛ 1 10**](#_Toc39664430)

[**Небезпека проблеми раптової серцевої смерті, та важливість екстреної допомоги з використанням дефібрилятора 10**](#_Toc39664431)

[**1.1. Раптова серцева смерть. 10**](#_Toc39664432)

[**1.2. Важливість першої невідкладної допомоги. 10**](#_Toc39664433)

[**1.3. Автоматичний зовнішній дефібрилятор 12**](#_Toc39664434)

[**1.4. Принципи роботи та можливості існуючих систем для пошуку АЗД 15**](#_Toc39664435)

[**Висновки до розділу 1 18**](#_Toc39664436)

[**РОЗДІЛ 2 20**](#_Toc39664437)

[**Постановка задачі розробки системи пошуку геолокації та інформації про автоматичні дефібрилятори 20**](#_Toc39664438)

[**2.1. Загальна постановка завдання 20**](#_Toc39664439)

[**2.2. Специфікація вимог до програмного продукту 20**](#_Toc39664440)

[**2.2.1. Призначення та мета 20**](#_Toc39664441)

[**2.2.2. Продукти-аналоги 21**](#_Toc39664442)

[**2.2.3. Загальний опис 21**](#_Toc39664443)

[**2.2.4. Характеристики системи 22**](#_Toc39664444)

[**2.2.5. Вимоги зовнішніх інтерфейсів 27**](#_Toc39664445)

[**2.2.6. Інші нефункціональні вимоги 28**](#_Toc39664446)

[**2.2.7. Атрибути якості програмного продукту 29**](#_Toc39664447)

[**2.3. Вибір засобів для виконання завдання 29**](#_Toc39664448)

[**Висновок до розділу 2 30**](#_Toc39664449)

[**РОЗДІЛ 3 30**](#_Toc39664450)

[**Архітектура та проектування системи для пошуку геолокації та інформації про автоматичні дефібрилятори 30**](#_Toc39664451)

[**3.1 . Проектування поведінки розроблюваної системи 30**](#_Toc39664452)

[**3.2. Загальна архітектура системи 34**](#_Toc39664453)

[**3.3. Проектування бази даних 35**](#_Toc39664454)

[**3.4. Проектування серверної частини 37**](#_Toc39664455)

[**3.5. Проектування клієнтської частини 40**](#_Toc39664456)

[**Висновок до розділу 3 43**](#_Toc39664457)

[**Висновок до переддипломної практики 44**](#_Toc39664458)

[**СПИСОК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ 45**](#_Toc39664459)

# Опис місця практики

ФОП Сатурський Орест Андрійович - це спеціаліст у сфері мобільної та веб-розробки. Він закінчив Львівський національний університет імені Івана Франка, де здобув ступінь магістра за спеціальністю «Прикладна інформатика» та отримав кваліфікацію «Магістр прикладної інформатики». На протязі п’яти останніх років працює в компанії SoftServe. Орест має глибокі знання, та багато досвіду роботи з такими технологіями як:

* React
* React Native
* Node.js

Цей список можна продовжувати дуже довго, адже Орест на своїх проектах працює з величезним спектром технологій та бібліотек, проте саме React, React Native та Node.js Орест не тільки застосовує на проектах, але й також навчає ним студентів IT Academy SoftServe, на курсі WebUI&Node.js. Орест напряму взаємодіє з командою студентів, даючи їм на виконання вимоги, що до розробки тестового проекту. Під час розробки Орест слідкує за виконанням а також пропонує цікаві архітектурні та програмні рішення, які зарекомендували себе на реальних проектах. Мені пощастило поступити на даний курс, де експертом був саме Орест, тому його досвід та час приділений на навчання моєї команди, допоміг мені попрактикуватися в знайомих мені технологіях та вивчити нові, і в результаті реалізувати тестовий проект.

# Завдання від бази практики

На початку проходження практики, для нас проводились тренінги, в кінці кожного з яких потрібно було виконати певне завдання:

* По використанню системи для менеджменту проектів – JIRA (кожен учень повинен був освоїти даний інструмент та описати тестовий проект в даній системі, розподіляючи завдання між командою та створюючи спрінти).
* По UML діаграмах (потрібно було досконало опрацювати всю типи діаграм та пройти тест)
* Тренін по React (в результаті якого потрібно було зробити міні-проект «ToDoList» із застосуванням React)
* Тренінг по «софт-скілам», на якому нас навчали як працювати в команді і які якості для цього потрібно в собі розвивати (в кінці тренінгу потрібно було пройти тест)

Потрібно було детально опрацювати такі технології як:

* Redux
* React
* MongoDB
* Node.js
* Google API
* Mapbox
* SQL (після опрацювання потрібно було пройти сертифікацію на сайті SoftServe Academy)
* GIT (також потрібно було пройти сертифікацію)
* Bush (також потрібно було пройти сертифікацію)

Також потрібно було реалізувати тестовий проект за наданими вимогами та рисунками проектування інтерфейсу. Вимоги були наступні.

Потрібно було реалізувати багатосторінковий, адаптивний веб-сайт, на спортивну тематику. Також повинна бути розроблений серверна частина на платформі Node.js, а також база даних на СУБД MongoDB з якою сервер буде взаємодіяти за допомогою бібліотеки mongoose. Веб сайт повинен містити наступні сторінки:

* *Головна сторінка,* на ній повинні бути розміщені відео та опис отанніх матчів.
* *Новини,* на ці сторінці повинні відображатися записи з колекції бази даних «news»
* *Коментарі,* на цій сторінці відображення записи з колекції «comments», а також мітиться форма, яка містить поля для додавання нового коментаря. Всі поля форми повинні валідуватися. Після додавання, коментар повинен відобразитися на сторінці та в базі даних. Якщо в користувача відсутнє підключення до мережі повинно передбачити запис в LocalStorage, а після того як мережа з’явиться коментар повинен бути доданий у базу даних і на сторінку.
* *Адміністратор,* на цій сторінці міститься форма для додавання нової новини в систему, серед полів форми міститься також поле для завантаження зображення яке буде відображатися в новині. Після додавання новина повинна бути відображена на сторінці *Новини* та в колекції бази даних «news». Поля форми повинні валідуватися.

# Виконання завдання від бази практики

Під час проходження практики було пройдено всі тренінги та виконано супутні завдання. Також багато уваги і часу було приділено вивченню зазначених в завданні від бази практики технологій. Найцікавіше було працювати з технологією для відображення та роботою з картами – Mapbox, а саме з бібліотекою для React - «react-mapbox-gl», оскільки моя тема дипломної роботи вимагає активної роботи з картами, набутий досвід та знання будуть дуже корисними.

Що до виконання тестового проекту, з ним я справився успішно і ресурси виконаної роботи ви зможете знайти за посиланням - <https://github.com/SaraTec/webTasks-New->.

Виконання роботи я розпочав з реалізації клієнтської частини, тому дані які повинні були надходити з сервера я імітував на клієнті. Завершивши розробку всіх сторінок за допомогою HTML та CSS (дивись рисунок 1.), я приступив до реалізації адаптивності та анімацій. Після цього, я розробив всі наявні на клієнті форми та реалізував логіку надсилання на сервер даних за допомогою http запиту з використанням інструменту axios та перевірку коректності введення полів.

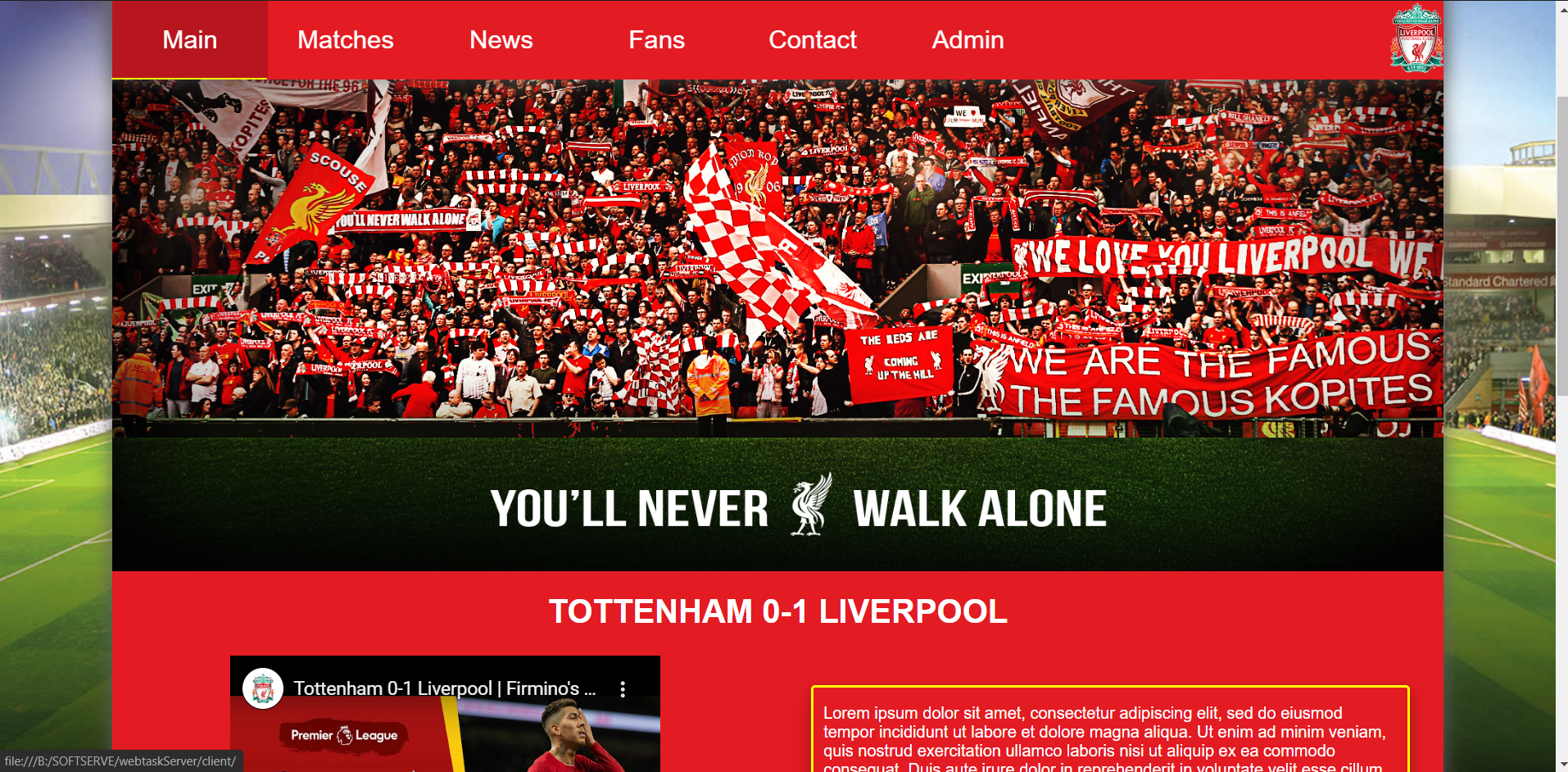


Рис 1. Інтерфейс Desktop версії проекту

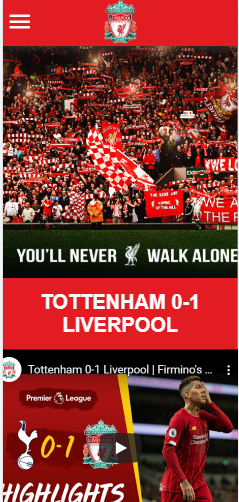


Рис 2. Інтерфейс мобільної версії проекту

Приступивши до реалізації серверної логіки я розпочав з проектування бази даних. Я створив на СУБД дві колекції «news» та «comments». Оскільки MongoDB це нереляційна СУБД, тож описувати схеми колекцій в ній напряму ми змоги не маємо, тому схеми я створив на нашому серверному застосунку за допомогою бібліотеки mongoose (дивись рисунок 2. 3.). В даних схемах я визначив поля колекцій та їхні типи. За допомогою цієї ж бібліотеки я здійснюю запити з мого серверного застосунку до бази даних.

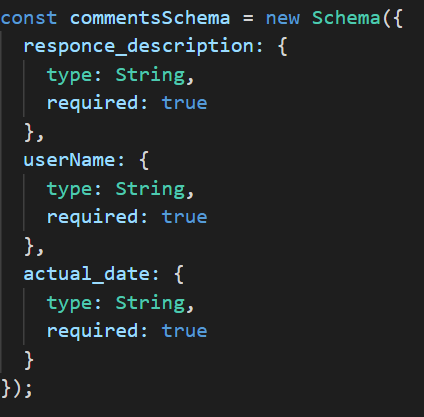


Рис 3. Схема колекції «comments»

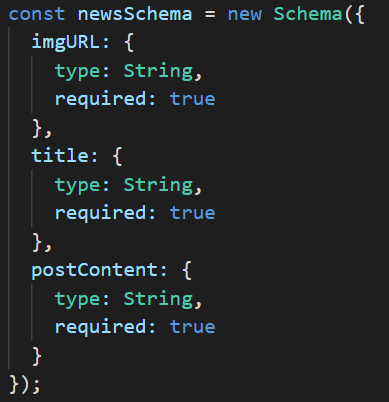


Рис 4. Схема колекції «news»

Наступним кроком була реалізація на сервері кінцевих точок на які клієнт здійснюватиме запити. Для цього я використав фреймворк express, за допомогою якого я легко визначив кінцеві точки, та описав функціонал який повинен виконуватися коли на кожну з них здійснюється запити.

В результаті я реалізував проект який задовольняє всім визначеним вимогам.

# Завдання дипломного проекту

Під час проходження практики мені потрібно було виконати частину дипломного проекту, а саме:

* Проаналізувати предметну область, дослідивши різноманітні електронні та літературні ресурси.
* Проаналізувати наявні на ринку програми-аналоги, та визначити який функціонал та якості варто наслідувати в своєму проекті, а чого в цих проектах не вистачає.
* Провести аналіз вимог та створити специфікацію вимог.
* Спроектувати поведінку ПЗ, та розробити діаграму прецидентів.
* Визначити загальну архітектуру ПЗ та створити діаграму розгортання.
* Спроектувати серверну частину, та відобразити сценарій взаємодії клієнта з сервером за допомогою діаграми послідовності.
* Спроектувати базу даних та відобразити схеми колекцій за допомогою фізичної діаграми бази даних.
* З отриманих даних сформувати оглядовий, постановчий та проектний розділи записки до дипломної роботи

Окрім цього, потрібно також зробити прототип програми дипломного проекту та задеплоїти її на сервісі Heroku. Базу даних для проекту також слід перенести на хмарний сервіс MongoDB Atlas.

**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]

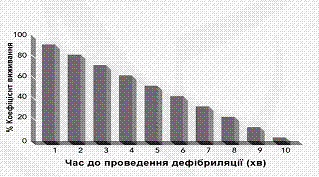
* 1. **Важливість першої невідкладної допомоги.**

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 ми зможемо 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 1.3.1*

**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 ми можемо 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-аплікаця 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 maps 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, я проаналізую його і визначу 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 ми бачимо 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.

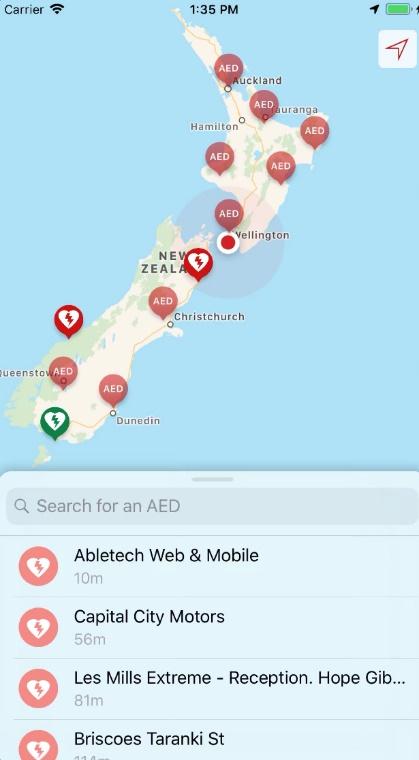


Рис. 1.2. Скріншот з додатку AED Locations №1

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 ми можемо бачити на рисунку 1.3.

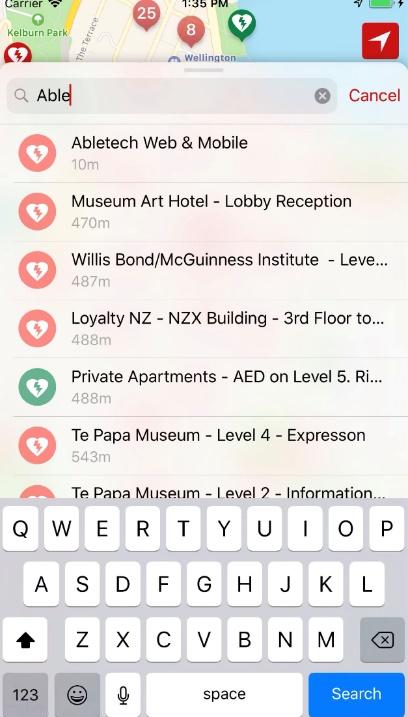


Рис. 1.3. Скріншот з додатку AED Locations №2

Після кліку на один зі значків на карті, що символізують про розміщення в тому місці АЗД, даний значок збільшиться, а в нижній панелі ми отримаємо 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 це ми можемо бачити на рисунку 1.4. Також ми можемо 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.

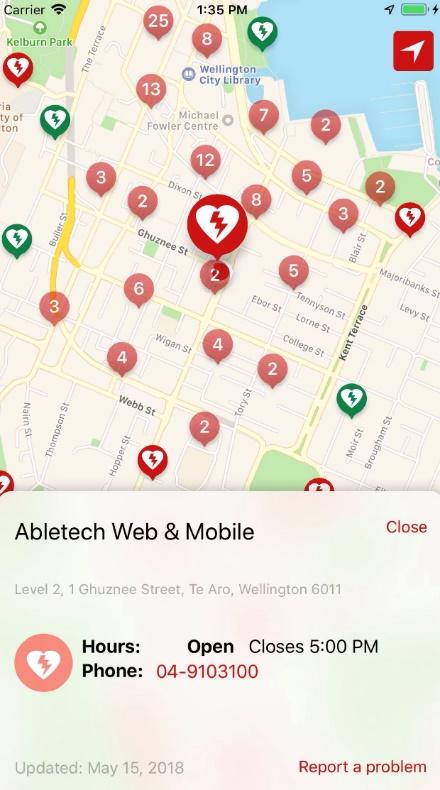


Рис. 1.4. Скріншот з додатку AED Locations №3

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 я описав поширену сьогодні проблему раптової серцевої смерті, а також важливість методів екстреної медичної допомоги потерпілому у випадках її настання. Опрацьовуючи матеріали по цій темі, я визначив, 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**

Опрацювавши велику кількість ресурсів, я визначив, що найпоширенішим лихом в розвинутих країнах є раптова серцева смерть, і одним з найбільш ефективних способів врятувати людину від раптової зупинки серця є надання їй екстреної допомоги з використанням автоматичного зовнішнього дефібрилятора (АЗД). Саме тому такі пристрої розташовані у всіх місцях скупчення людей у більшості країн Європи, проте в Україні активні кроки по встановленню АЗД у відповідних місцях здійснюються тільки зараз, тому є доцільним знайти спосіб, щоб зробити пошук цих пристроїв у екстрених випадках максимально швидким і ефективним, а розміщення пристроїв в максимальній степені зручним і гармонійним.

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. Одну з таких систем я проаналізував та описав 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 NodeJS 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 «налазять» 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 емейлу 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 яку я розробляю складається з клієнтської та серверної частини. Для розробки серверної частини я використовуватиму 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. Я буду використовувати його можливості, 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.

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

Для збереження даних я буду використовувати не реляційну базу даних - MongoDB, а також MongoDB Compass як графічний інтерфейс для роботи з моїми колекціями даних.

Для розробки клієнтської частини я використовуватиму бібліотеку React, яка ідеально підходить для управління станом всіх компонент інтерфейсу.

Оскільки в моїй системі буде велика кількість компонент, я також використовуватиму 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 я використовуватиму 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, я використовуватиму 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 я використовую бібліотеку Formik яка забезпечить нас великою кількістю методів для наших 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 я визначив 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 я опишу 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 я в подальшому буду 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.

На рисунку 3.1. зображена діаграма прецедентів, на якій розміщені актори, які є класами користувачів, а також функціонал для кожного з наявних в системі користувачів.



Рис. 3.1. Діаграма прецедентів

**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 ми створимо REST API, за допомогою програмного забезпечення на платформі Node.js – це буде нашою серверною частиною. На ній ми використовуватимемо фреймворк Express.js, щоб специфікувати «роути», на які клієнт здійснюватиме запити за допомогою протоколу HTTP, а також опишемо як наш сервер буде опрацьовувати ці запити, і що він буде повертати на клієнт. Для того, щоб дати можливість серверу працювати з базою даних ми використовуватимемо бібліотеку mongoose, яка дозволить нам проводити операції з базою даних за допомогою протоколу SSL. Базою даних для нашої 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 3.2.



Рис. 3.2. Діаграма розгортання

**3.3. Проектування бази даних**

Для формування нашої бази даних ми вибрали 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 ми сформуємо певні схеми для кожної колекції бази даних саме на нашому серверному застосунку, за допомогою бібліотеки Mongoose.js. В кожній схемі які відповідатимуть певній колекції ми зазначимо поля які наявні в колекції та тип даних цього поля. В була розроблена фізичні модель нашої бази даних (дивись рисунок 3.3.).

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. 3.3. Фізична модель бази даних

**3.4. Проектування серверної частини**

Як ми вже визначили 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 нам швидко описати всі шляхи (API endpoints), на які будуть поступати запити, які ми будемо обробляти. Також до даної платформи ми матимемо змогу додати ще декілька дуже корисних модулів, зокрема бібліотеку PassportJS. За допомогою даної бібліотеки ми здійснюватимемо авторизацію наших користувачів, і будемо перевіряти наявність прав в користувача на виконання тої ти іншої операції.

Для формування нашої бази даних ми вибрали MongoDB. Тому для зручності роботи нашої серверної частини з MongoDB я використовуватиму бібліотеку Mongoose.js – це «обгортка», яка дозволяє нам на 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/gmap – 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 хіаграми послідовності (дивись рисунок 3.4.).



Рис. 3.4. Діаграма послідовності

**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 ми зможемо описати спосіб, та сценарій маніпулювання глобального стану (сховища). Тобто в будь-які ми зможемо 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 3.5.):

* 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.

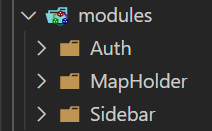


Рис. 3.5.Структура верхнього рівня клієнтського застосування

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 3.6.), тому. якщо ми хочемо використати один з них у своїй компоненті, достатньо просто імпортувати цей модуль на початку файлу, за допомогою утиліти import.

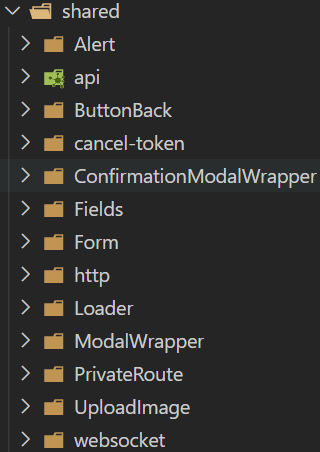


Рис. 3.6.Структура папки для збереження модулів які часто використовуються в системі

**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.

# Висновок до переддипломної практики

Під час проходження практики я вдосконалив свої знання та навички у вмінні працювати з багатьма популярними технологіями та архітектурними рішеннями які активно використовуються в реальних клієнт-серверних програмних продуктах, де на клієнті головним інструментом є React а на серверній частині Node.js.

Усі завдання які були поставлені керівником практики від підприємства, були успішно виконані. Був розроблений тестовий проект – клієнт-серверний веб-застосунок для перегляду інформації про футбольний клуб «Ліверпуль». Ресурси даного проекту ви можете знайти за посиланням - <https://github.com/SaraTec/webTasks-New->.

Також були виконані завдання, поставлені мені керівником бакалаврської роботи. Результатом виконання стали сформовані три розділи до пояснювальної записки: оглядовий, постановчий, проектування. До розділу проектування було розроблено діаграми, які описують архітектуру системи та її поведінку. Також був розроблений, та розміщений на хмарних сервісах прототип дипломного проекту, який ви можете переглянути за посиланням - <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. 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/>
7. 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>

Що таке раптова смерть - [Електронний ресурс] – Режим доступу : <http://www.diagnoz.lutsk.ua/news/shcho-take-raptova-sertseva-smert>