

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Аналіз алгоритмів SCORE та Framingham оцінки
стану серцево-судинної системи людини»**

Виконала студентка групи ІПЗм-21-1
МОРГУН Анастасія Миколаївна

Керівник роботи:
ЧИЖМОТРЯ Олексій Володимирович

Рецензент:
ВАКАЛЮК Тетяна Анатоліївна

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Завідувач кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«18» вересня 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **МОРГУН Анастасія Миколаївна**

Керівник роботи: **ЧИЖМОТРЯ Олексій Володимирович**

Тема роботи: **«Аналіз алгоритмів SCORE та Framingham оцінки стану серцево-судинної системи людини»**,

затверджена Наказом закладу вищої освіти від **«18» вересня 2023 р. № 479/с**

Вихідні дані для роботи: Об'єктом дослідження є алгоритми SCORE і Framingham, які використовуються для оцінки ризику серцево-судинних захворювань. Предметом дослідження є порівняльний аналіз алгоритмів, їхніх переваг та недоліків, а також можливість реалізації за допомогою технік штучного інтелекту та створення веб-додатку, що буде прогнозувати ризик відповідно до вибраного алгоритму

Консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О. В.	22.09.2023р.	22.09.2023р.
2	Чижмотря О. В.	22.09.2023р.	22.09.2023р.
3	Чижмотря О. В.	22.09.2023р.	22.09.2023р.

РЕФЕРАТ

Кваліфікаційна робота магістра представляє собою ретельну оцінку та порівняльний аналіз алгоритмів SCORE і Framingham в контексті їхнього впливу на ефективність діагностики. В результаті аналізу був обраний кращий алгоритм, який має менший процент помилок при прогнозуванні ризику. Також було створено інноваційний веб-додаток «Система персоналізованої діагностики серцево-судинних захворювань». Пояснювальна записка до випускної роботи містить 89 сторінок, 39 ілюстрацій, 8 таблиць, 2 додатки та 29 використаних джерел.

Основною метою даної роботи є створення та впровадження високоточної та ефективної автоматизованої системи для персоналізованої діагностики та прогнозування ризику розвитку серцево-судинних захворювань.

Дослідження включало в себе формулювання та розв'язання ключових завдань розробки веб-проекту, проведення аналізу існуючих аналогів, формування функціональної моделі та технічних вимог до системи. Проведення ретельного аналізу алгоритмів SCORE і Framingham.

Для реалізації системи були вибрані передові технології веб-розробки, а алгоритми були вперше інтегровані з використанням інструментів машинного навчання для максимальної точності та надійності. Розроблений програмний продукт пройшов тестування, підтверджуючи його високу функціональність.

Ця робота є актуальною і важливою, оскільки розроблена система може служити інструментом для фахівців у галузі кардіології та допомагати пацієнтам у попередженні та ранньому виявленні серцево-судинних захворювань.

КЛЮЧОВІ СЛОВА: ВЕБ-ДОДАТОК, ПЕРСОНАЛІЗОВАНА ДІАГНОСТИКА, СЕРЦЕВО-СУДИННІ ЗАХВОРЮВАННЯ, АЛГОРИТМИ SCORE І FRAMINGHAM, ТЕХНОЛОГІЇ ВЕБ-РОЗРОБКИ, МАШИННЕ НАВЧАННЯ.

ABSTRACT

The master's thesis is a thorough assessment and comparative analysis of the SCORE and Framingham algorithms in the context of their impact on diagnostic efficiency. As a result of the analysis, the best algorithm was chosen, which has a lower percentage of errors in risk prediction. An innovative web application "System of personalized diagnosis of cardiovascular diseases" was also created. The explanatory note to the final thesis contains 89 pages, 39 illustrations, 8 tables, 2 appendices and 29 used sources.

The main goal of this work is to create and implement a highly accurate and efficient automated system for personalized diagnosis and prediction of the risk of developing cardiovascular diseases.

The research included formulating and solving the key tasks of developing a web project, conducting an analysis of existing analogues, forming a functional model and technical requirements for the system. Carrying out a thorough analysis of the SCORE and Framingham algorithms.

Advanced web development technologies were chosen to implement the system, and algorithms were integrated for the first time using machine learning tools for maximum accuracy and reliability. The developed software product has been tested, confirming its high functionality.

This work is relevant and important because the developed system can serve as a tool for specialists in the field of cardiology and help patients in the prevention and early detection of cardiovascular diseases.

KEY WORDS: WEB APPLICATION, PERSONALIZED DIAGNOSTICS, CARDIOVASCULAR DISEASES, SCORE AND FRAMINGHAM ALGORITHMS, WEB DEVELOPMENT TECHNOLOGIES, MACHINE LEARNING.

ЗМІСТ

РЕФЕРАТ	3
ВСТУП	6
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	9
РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ.....	10
1.1 Методика визначення оцінки.....	10
1.2 Аналіз аналогів дослідження та програмного продукту.....	14
1.3 Технічне завдання	20
1.4 Вибір засобів реалізації програмного продукту	23
Висновки до першого розділу.....	25
РОЗДІЛ 2 ПРАКТИЧНА РЕАЛІЗАЦІЯ АЛГОРИТМІВ	26
2.1 Побудова моделі аналізу і практична реалізація	26
2.2 Результати проведеного аналізу алгоритмів	31
2.3 Аналіз ризиків в залежності від ознак	33
2.4 Порівняльний аналіз отриманих результатів з іншими дослідженнями	35
Висновки до другого розділу	37
РОЗДІЛ 3. ПОРЯДОК РОБОТИ ТА ТЕСТУВАННЯ СИСТЕМИ	38
3.1 Визначення варіантів використання системи	38
3.2 Інтерфейс та порядок роботи з системою.....	40
3.3 Тестування роботи веб-додатку.....	52
Висновки до третього розділу.....	54
ВИСНОВКИ.....	56
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	58
ДОДАТКИ.....	62

ВСТУП

Актуальність теми. У сучасному світі серцево-судинні захворювання стали однією з найбільш серйозних загроз для здоров'я населення. Зростаюча кількість випадків цих захворювань вимагає невідкладних заходів та ефективних методів оцінки ризику, діагностики та профілактики. В цьому контексті порівняння алгоритмів SCORE і Framingham стає ключовим завданням у галузі медицини та медичної інформатики.

Обрані алгоритми використовуються для оцінки ризику виникнення серцево-судинних захворювань у пацієнтів, і це має вирішальне значення для ефективності діагностики та проведення профілактичних заходів. Особливо важливою стає ця проблема з огляду на тенденцію збільшення випадків серцево-судинних захворювань серед населення.

Обрані алгоритми SCORE і Framingham відзначаються схожістю у параметрах оцінки, що робить їх особливо цікавими для подальшого дослідження. Враховуючи особливості європейської популяції, ці алгоритми вважаються більш відповідними для використання в даному контексті.

Застосування інструментів програмування в дослідженні алгоритмів SCORE і Framingham сприятиме покращенню їхньої ефективності та розширенню можливостей їх використання в медичній практиці. Це може допомогти лікарям у прийнятті більш обґрунтованих рішень, а також сприяти збільшенню точності та швидкості діагностики серцево-судинних захворювань.

Узагальнюючи, порівняння алгоритмів SCORE і Framingham має стратегічне значення для підвищення якості медичної допомоги, що може значно покращити здоров'я та якість життя пацієнтів.

Мета даної роботи. Метою роботи є розробка та реалізація веб-додатка для персоналізованої діагностики пацієнтів в оцінці серцево-судинного ризику, на основі порівняння ефективності алгоритмів SCORE і Framingham, з використанням різних критеріїв для оцінки продуктивності та підходів до аналізу даних.

Об'єктом дослідження є алгоритми SCORE і Framingham, які використовуються для оцінки ризику серцево-судинних захворювань.

Предметом дослідження є порівняльний аналіз алгоритмів, їхніх переваг та недоліків, а також можливість реалізації за допомогою технік штучного інтелекту та створення веб-додатку, що буде прогнозувати ризик відповідно до вибраного алгоритму

Наукова новизна полягає в кількох ключових аспектах:

- Порівняльний аналіз алгоритмів SCORE і Framingham. В роботі проведено глибокий та всебічний порівняльний аналіз двох важливих алгоритмів оцінки серцево-судинного ризику, а саме SCORE і Framingham. Цей підхід дозволяє визначити переваги та недоліки кожного алгоритму з точки зору ефективності та застосовності в конкретних умовах.

- Використання інструментів програмування для підвищення ефективності алгоритмів. В даній роботі розглянута можливість застосування інструментів програмування, зокрема бібліотек та мов програмування для машинного навчання, для покращення ефективності алгоритмів SCORE і Framingham. Це відкриває нові перспективи для застосування технологій штучного інтелекту у сфері медичної діагностики та прогнозування.

- Розробка веб-додатка для персоналізованої діагностики. Створення веб-додатка для персоналізованої діагностики серцево-судинного ризику на основі кращого алгоритму є інноваційним кроком у напрямку забезпечення більш доступної та ефективної медичної допомоги. Цей веб-додаток може стати новаторським інструментом для лікарів та пацієнтів, дозволяючи здійснювати персоналізовану діагностику та вчасні профілактичні заходи.

Практичне значення. Розроблений веб-додаток для персоналізованої діагностики серцево-судинного ризику на основі кращого алгоритму володіє практичною значущістю. Цей додаток може слугувати ефективним інструментом для лікарів у процесі діагностики та прийняття рішень, а також забезпечити користувачам персоналізовані рекомендації щодо профілактики

серцево-судинних захворювань. Такий підхід дозволяє покращити якість медичної допомоги та зробити її більш доступною для широкого кола пацієнтів.

Апробація результатів роботи. Матеріали дослідження за тематикою кваліфікаційної роботи:

1. Легенчук С. Ф., Ковбасюк С. В., Моргун А. М. Інформаційна технологія діагностування стану серцево-судинної системи людини. Науковий журнал Вісник Хмельницького національного університету Серія: «Технічні науки», № 5 т. 2, 2023. – С. 60-65.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ССС (серцево-судинна система): Система, яка включає серце та судини, відповідальна за перекачування крові та подачу кисню та поживних речовин до різних органів та тканин.

ССЗ (серцево-судинні захворювання): Група захворювань, які впливають на серцево-судинну систему, включаючи ішемічну хворобу серця, інсульт, артеріальну гіпертензію тощо.

Framingham (алгоритм): Метод оцінки ризику серцево-судинних захворювань, розроблений у рамках дослідження серця в місті Фреймінгем, США.

ІХС (ішемічна хвороба серця): Стан, при якому м'яз серця отримує недостатню кількість кисню через обмежений кровопостачання

SCORE (Systematic Coronary Risk Evaluation): Методика систематичної оцінки коронарного ризику, що використовується для прогнозування ризику розвитку серцево-судинних подій.

MSE (Mean Squared Error): Середнє квадратичне відхилення, метрика, яка використовується для вимірювання середньої кількості квадратів відхилень між прогнозованими та спостережуваними значеннями.

MAE (Mean Absolute Error): Середнє абсолютне відхилення, метрика, яка визначає середню абсолютну різницю між прогнозованими та фактичними значеннями.

ROC (Receiver Operating Characteristic): Графік, який використовується для візуалізації ефективності класифікаційних моделей, відображаючи зв'язок між чутливістю та специфічністю при різних порогах класифікації.

РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1 Методика визначення оцінки

Оцінка серцево-судинного ризику є важливим етапом в сучасній медицині та медичній інформатиці. Одним із варіантів прогнозування ризику захворювання використовують шкалу SCORE. Оцінка серцево-судинного ризику рекомендована як практичний інструмент визначення оптимального ступеня втручання для корекції ризику конкретної людини [29]. З урахуванням зростання кількості випадків серцево-судинних захворювань у сучасному світі, важливість ефективних методів оцінки ризику та діагностики стає ключовою для підтримання здоров'я населення.

Основними двома передумовами для використання серцево-судинного ризику в лікарській практиці та охороні здоров'я є: обмеженість економічних ресурсів та необхідність їх використовувати найбільш ефективно; необхідність збалансувати користь та можливу шкоду від профілактичних втручань [29].

Історія розвитку шкали SCORE свідчить про важливість наукових досліджень та використання об'єктивних даних. Розробка систематичної оцінки коронарного ризику (SCORE) включала аналіз об'єднаних даних понад 250 000 осіб із 12 європейських досліджень. Початково розрахований у 2003 році, алгоритм шкали передбачав 10-річний ризик смерті від серцево-судинних захворювань, і в подальших версіях враховував загальний ризик серцево-судинних захворювань [29].

Калібрування для регіонів Європи з високим та низьким рівнем ризику було розроблено на основі статистики смертності в кожній країні. Крім того, у редакції 2012 р., опублікованій у рекомендаціях щодо ССЗ, випущених п'ятою спільною цільовою групою Європейського товариства кардіологів, було додано розрахунок віку серцево-судинного ризику. Розрахунок призначений для використання як засіб комунікації, особливо з молодшими людьми з низькими оцінками ризику [29].

Шкала SCORE (рис. 1.1) вражає своєю простотою та зрозумілістю. Пацієнтові достатньо обрати відповідну частину шкали, відповідно до його статі, віку та статусу куріння. Всередині таблиці визначається клітина, яка відображає індивідуальний рівень систолічного артеріального тиску та загального холестерину. Число в цій клітині показує 10-річний сумарний серцево-судинний ризик цього пацієнта.

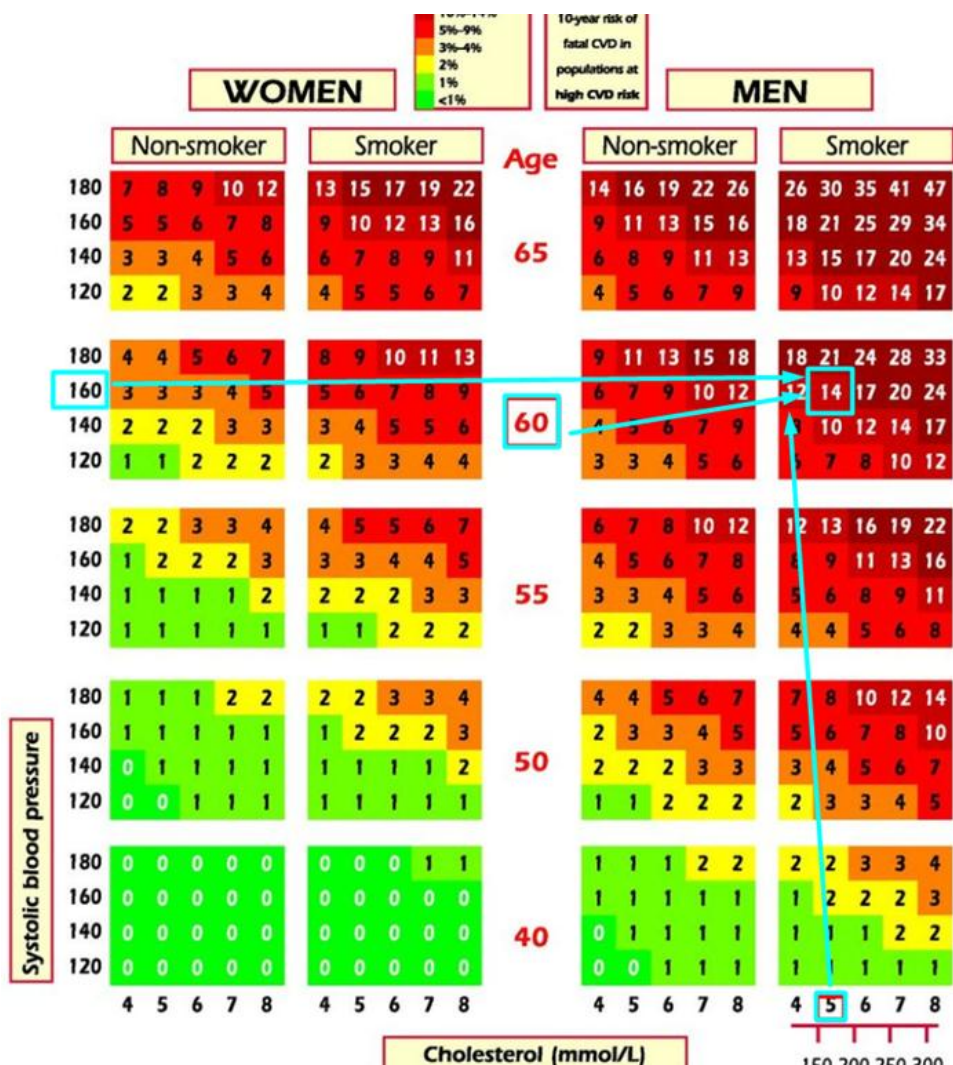


Рис. 1.1 – Шкала SCORE

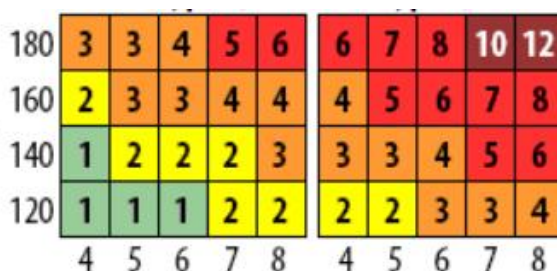


Рис. 1.2 – SCORE молодших осіб

Для молодших осіб (молодших за 40 років) використовується відносний сумарний ризик, розрахований за спеціальною шкалою (рис. 1.2). Ця інформація може бути особливо корисною для профілактичного консультування людей із низьким абсолютним, але високим відносним сумарним ризиком, сприяючи мотивації до здорового способу життя.

Процес оцінки ризику та корекції враховує різні аспекти, включаючи індивідуальні характеристики, і стає важливим інструментом для попередження серцево-судинних захворювань. Застосування шкали SCORE надає можливість ефективно оцінити ризик та вчасно розпочати профілактичні заходи для збереження здоров'я серця.

З другої Фремінгемська оцінка ризику є традиційною системою, що базується на інвентаризації факторів ризику, і використовує легкодоступні змінні для розрахунку 10-річного ризику для серцево-судинних подій.

Дослідження розпочалося в 1948 році з набору початкової когорти з 5209 чоловіків і жінок віком від 30 до 62 років з міста Фремінгем, штат Массачусетс, у яких ще не розвинулися явні симптоми серцево-судинних захворювань або які не перенесли серцевий напад чи інсульт. З того часу Дослідження додало когорту нащадків у 1971 році, когорту Опні у 1994 році, когорту третього покоління у 2002 році, когорту нових подружжя нащадків у 2003 році та когорту Опні другого покоління у 2003 році [17].

Протягом багатьох років ретельний моніторинг популяції Фремінгемського дослідження призвів до виявлення основних факторів ризику серцево-судинних захворювань, а також до отримання цінної інформації про вплив цих факторів, таких як артеріальний тиск, рівень тригліцеридів і холестерину в крові, вік, стать і психосоціальні питань. Фактори ризику для інших фізіологічних станів, таких як деменція, були та продовжують досліджуватися. Крім того, вивчається взаємозв'язок між фізичними рисами та генетичними моделями [17].

Оскільки оцінки ризику, такі як Фремінгемська оцінка ризику, дають вказівку на ймовірні переваги профілактики, вони корисні як для окремого пацієнта, так і для клініциста, допомагаючи прийняти рішення про зміну способу життя та профілактичне лікування, а також для навчання пацієнтів, ідентифікуючи чоловіків і жінки з підвищеним ризиком майбутніх серцево-судинних подій. Ризик ішемічної хвороби серця (ІХС) через 10 років у відсотках можна розрахувати за допомогою Фремінгемської оцінки ризику. Особи з низьким ризиком мають 10% або менше ризику ІХС через 10 років, з проміжним ризиком 10-20%, а з високим ризиком 20% або більше [17].

На рисунку 1.3 та рисунку 1.4 описано ризиковий бал Фреїмінгема для жінок та чоловіків. Цей бал використовується для оцінки ризику розвитку серцево-судинних захворювань протягом 10 років на основі різних факторів ризику. Бали, надані кожному фактору ризику, підсумовуються для визначення загального балу ризику, який потім відповідає відповідному оціненому відсотку розвитку серцево-судинних захворювань протягом 10 років.

Framingham Risk Score for Women

Age: 20–34 years: Minus 7 points. 35–39 years: Minus 3 points. 40–44 years: 0 points. 45–49 years: 3 points. 50–54 years: 6 points. 55–59 years: 8 points. 60–64 years: 10 points. 65–69 years: 12 points. 70–74 years: 14 points. 75–79 years: 16 points.

Total cholesterol, mg/dL:

- Age 20–39 years: Under 160: 0 points. 160–199: 4 points. 200–239: 8 points. 240–279: 11 points. 280 or higher: 13 points.
- Age 40–49 years: Under 160: 0 points. 160–199: 3 points. 200–239: 6 points. 240–279: 8 points. 280 or higher: 10 points.
- Age 50–59 years: Under 160: 0 points. 160–199: 2 points. 200–239: 4 points. 240–279: 5 points. 280 or higher: 7 points.
- Age 60–69 years: Under 160: 0 points. 160–199: 1 point. 200–239: 2 points. 240–279: 3 points. 280 or higher: 4 points.
- Age 70–79 years: Under 160: 0 points. 160–199: 1 point. 200–239: 1 point. 240–279: 2 points. 280 or higher: 2 points.

If cigarette smoker: Age 20–39 years: 9 points. • Age 40–49 years: 7 points. • Age 50–59 years: 4 points. • Age 60–69 years: 2 points. • Age 70–79 years: 1 point.

All non smokers: 0 points.

HDL cholesterol, mg/dL: 60 or higher: Minus 1 point. 50–59: 0 points. 40–49: 1 point. Under 40: 2 points.

Systolic blood pressure, mm Hg: Untreated: Under 120: 0 points. 120–129: 1 point. 130–139: 2 points. 140–159: 3 points. 160 or higher: 4 points. • Treated: Under 120: 0 points. 120–129: 3 points. 130–139: 4 points. 140–159: 5 points. 160 or higher: 6 points.

10-year risk in %: Points total: Under 9 points: <1%. 9–12 points: 1%. 13–14 points: 2%. 15 points: 3%. 16 points: 4%. 17 points: 5%. 18 points: 6%. 19 points: 8%. 20 points: 11%. 21=14%, 22=17%, 23=22%, 24=27%, >25= Over 30%

Рис. 1.3 – Ризиковий бал Фреїмінгема для жінок

Framingham Risk Score for Men

Age: 20–34 years: Minus 9 points. 35–39 years: Minus 4 points. 40–44 years: 0 points. 45–49 years: 3 points. 50–54 years: 6 points. 55–59 years: 8 points. 60–64 years: 10 points. 65–69 years: 11 points. 70–74 years: 12 points. 75–79 years: 13 points.

Total cholesterol, mg/dL:

- Age 20–39 years: Under 160: 0 points. 160–199: 4 points. 200–239: 7 points. 240–279: 9 points. 280 or higher: 11 points.
- Age 40–49 years: Under 160: 0 points. 160–199: 3 points. 200–239: 5 points. 240–279: 6 points. 280 or higher: 8 points.
- Age 50–59 years: Under 160: 0 points. 160–199: 2 points. 200–239: 3 points. 240–279: 4 points. 280 or higher: 5 points.
- Age 60–69 years: Under 160: 0 points. 160–199: 1 point. 200–239: 1 point. 240–279: 2 points. 280 or higher: 3 points.
- Age 70–79 years: Under 160: 0 points. 160–199: 0 points. 200–239: 0 points. 240–279: 1 point. 280 or higher: 1 point.

If cigarette smoker: Age 20–39 years: 8 points. • Age 40–49 years: 5 points. • Age 50–59 years: 3 points. • Age 60–69 years: 1 point. • Age 70–79 years: 1 point.

All non smokers: 0 points.

HDL cholesterol, mg/dL: 60 or higher: Minus 1 point. 50–59: 0 points. 40–49: 1 point. Under 40: 2 points.

Systolic blood pressure, mm Hg: Untreated: Under 120: 0 points. 120–129: 0 points. 130–139: 1 point. 140–159: 1 point. 160 or higher: 2 points. • Treated: Under 120: 0 points. 120–129: 1 point. 130–139: 2 points. 140–159: 2 points. 160 or higher: 3 points.

10-year risk in %: Points total: 0 point: <1%. 1–4 points: 1%. 5–6 points: 2%. 7 points: 3%. 8 points: 4%. 9 points: 5%. 10 points: 6%. 11 points: 8%. 12 points: 10%. 13 points: 12%. 14 points: 16%. 15 points: 20%. 16 points: 25%. 17 points or more: Over 30%.^[24]

Рис. 1.4 – Ризиковий бал Фреймінгема для чоловіків

Важливо зазначити, що простота та зрозумілість застосування роблять алгоритм SCORE корисним інструментом для визначення індивідуального ризику та прийняття відповідних заходів.

Фремінгемська оцінка ризику полягає в наданні об'єктивної інформації про вплив факторів ризику на серцево-судинний стан.

Обидва підходи допомагають створити основу для вивчення стану серцево-судинної системи та розуміння індивідуального ризику виникнення захворювань. Враховуючи їхню простоту та доступність, ці методики можуть стати ефективним інструментарієм для лікарів та пацієнтів у прийнятті обґрунтованих рішень щодо здоров'я та профілактики захворювань.

1.2 Аналіз аналогів дослідження та програмного продукту

Існує безліч досліджень зі схожою тематикою. Кожне відрізняється один від одного підходами, висновками та періодом самого дослідження. Час грає важливу роль у всіх дослідженнях, минуло вже декілька років з останніх робіт, змінилися можливості технологій, підходи в медицині, тому використання сучасних даних та технік може призвести до нового відкриття. Далі буде розглянуто подібні дослідження.

Першим буде «Оцінка профілю серцево-судинного ризику: порівняльний аналіз між алгоритмом CUORE та шкалою ризику Framingham» [2]. В роботі були розглянуті дані про пацієнтів лікарів загальної практики, які працюють у провінції Рим, і загалом було оцінено 996 осіб обох статей. Відповідність регресійної моделі оцінювалася за допомогою значення R^2 . В результаті аналіз показав, що значення R^2 простої лінійної регресії між ризиком ІХС, розрахованим за методом CUORE (залежна змінна), та ризиком ІХС, розрахованим за методом Фремінгема (незалежна змінна), спочатку дорівнювало 0,350, а потім піднялося до 0,732, коли були додані незалежні змінні «Стать» і «Вік», що створило багатовимірну регресію [2].

Виявилося, що оцінка CUORE була менш надійною, ніж оцінка ризику Framingham.

Далі йде «Порівняння оцінок ризику серцево-судинних захворювань на основі Framingham та SCORE в учасників Німецького національного інтерв'ю та опитування щодо охорони здоров'я 1998 р.» [4]. Дані Німецького національного інтерв'ю та опитування щодо охорони здоров'я 1998 року використовувалися, щоб визначити, чи призводить SCORE до нижчих оцінок 10-річного абсолютного ризику летальних серцево-судинних захворювань та летальної ішемічної хвороби серця, ніж модель Фремінгема. Прогнозовану кількість подій порівнювали з приблизними даними на основі національної статистики смертності [4].

Критерії включення відповідали рекомендаціям щодо використання SCORE: вік від 30 до 69 років, відсутність попередньої історії серцево-судинних захворювань і відсутність помітно підвищених рівнів окремих факторів ризику (залишаючи 1811 чоловіків і 1955 жінок для аналізу) [4].

В результаті модель SCORE для регіонів з високим ризиком (SCORE-HIGH, яка рекомендована для Німеччини в очікуванні калібрування з національними даними) передбачила найбільшу кількість подій, за якою йдуть оцінки зі статистикою смертності, модель Framingham і SCORE-LOW.

Результати свідчать про те, що SCORE-HIGH може переоцінювати абсолютний ризик летальної ішемічної хвороби серця та серцево-судинних захворювань у Німеччині, і може знадобитися калібрування. Крім того, обмеження поточних інструментів прогнозування ризиків підкреслюють постійну потребу в комплексних, високоякісних і своєчасних європейських даних [4].

І в кінці дослідження «Порівняння моделей ризику Framingham і SCORE у прогнозуванні наявності та тяжкості ішемічної хвороби серця з урахуванням оцінки SYNTAX» [13]. Популяція обсерваційного перехресного дослідження складалася з 227 пацієнтів із середнім віком $63,3 \pm 9,2$ року. Пацієнтів розділили окремо на групи низького та високого ризику за моделями Framingham та SCORE. Після коронарографії пацієнтів було класифіковано на групи SYNTAX=0 (бал SYNTAX 0), низький SYNTAX (бал SYNTAX 1–22) і високий SYNTAX (бал SYNTAX >22) [13]. Зв'язок між моделями ризику та показником SYNTAX оцінювався за t-критерієм Стюдента, U-тестом Манна-Уїтні або тестом Краскела–Уолліса, а аналіз робочих характеристик приймача використовувався для виявлення здатності до розрізнення в прогнозуванні показника SYNTAX >0 і високого SYNTAX рахунок [13].

В результаті обидві моделі Framingham і SCORE виявилися ефективними для прогнозування наявності ішемічної хвороби серця, і жодна з двох моделей не мала переваги одна над одною. Під час аналізу підгруп виявилось, що модель SCORE краще прогнозує ступінь і тяжкість ішемічної хвороби серця в підгрупах чоловіків і діабетиків [13].

Далі розглянемо додатки для самодіагностики на основі цих алгоритмів, адже вони перший крок на шляху до покращення здоров'я та запобігання серцево-судинним захворюванням. Такі додатки можуть надати користувачам можливість самостійно визначати ризик серцево-судинних захворювань та вживати відповідні заходи для їхньої профілактики. Розглянемо такі додатки

Калькулятор для розрахунку за шкалою SCORE дозволяє оцінити ризик смерті людини від серцево-судинних захворювань протягом найближчих 10

років [1] (рис. 1.5). Це мобільний додаток, що має приємний інтерфейс та зручний у використанні, працює за таким же алгоритмом, що й майбутній веб-додаток.

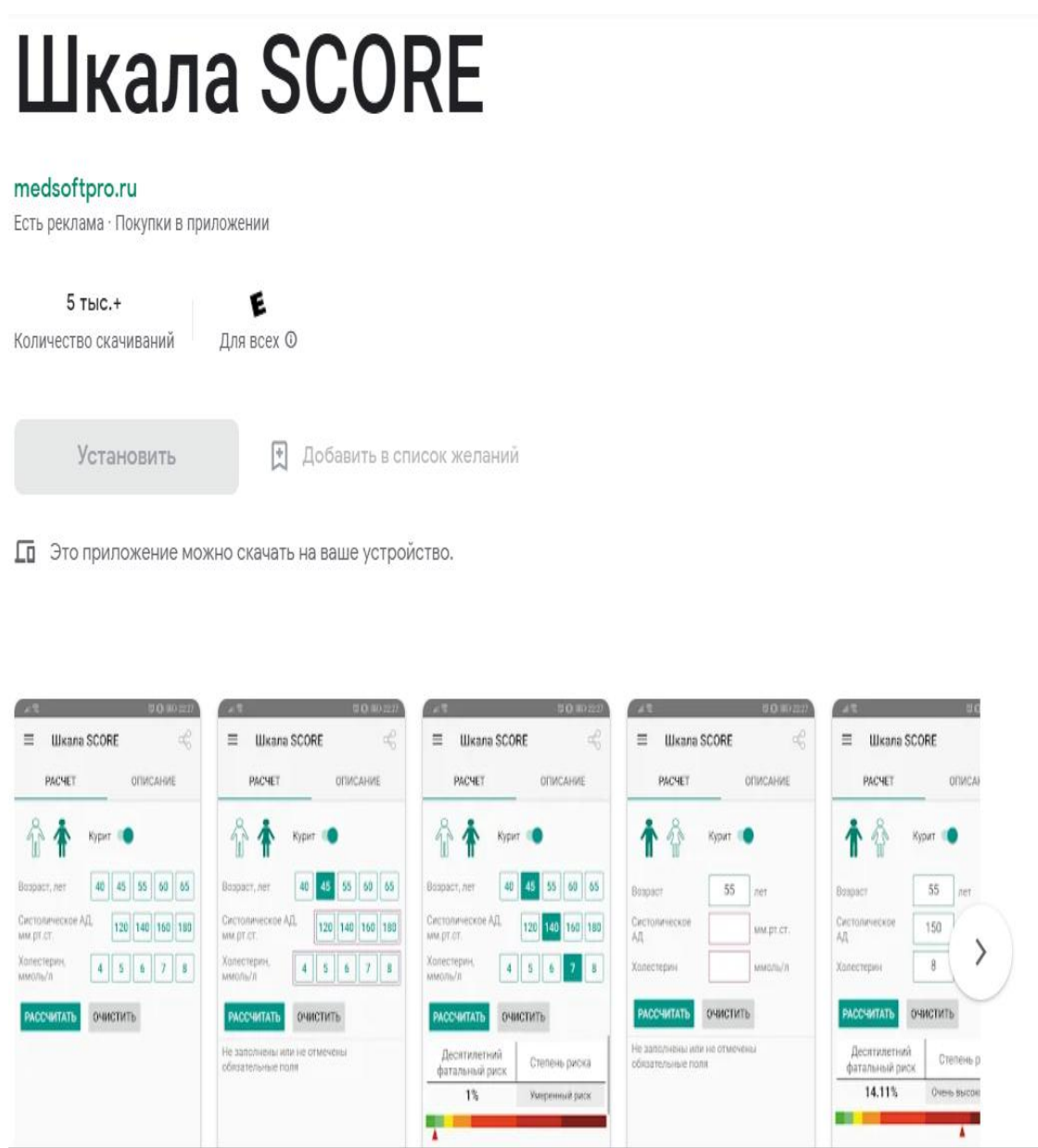


Рис. 1.5 – Шкала SCORE

Наступний додаток Framingham Calc (рис. 1.6) – калькулятор серцево-судинного ризику, заснований на когортному дослідженні Фрамінгема. Ця програма робить розрахунки на основі введених даних і дає відповідний результат. Результат полягає у відсотку ризику розвитку серцево-судинних захворювань у наступні 10 років і приблизний вік серця [9].

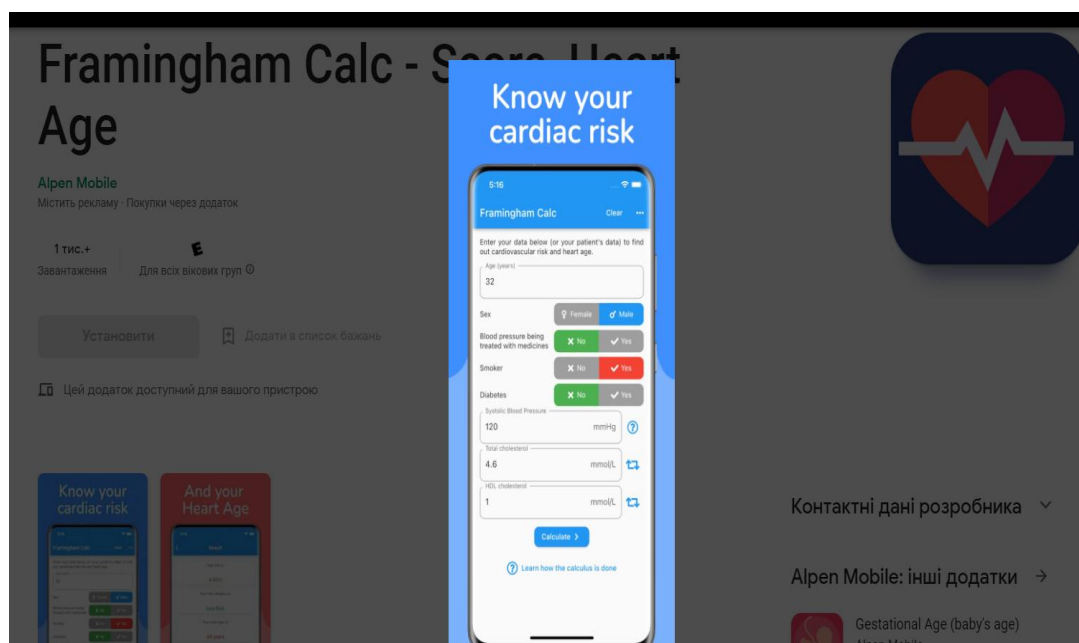


Рис. 1.6 – Framingham Calc

Таблиця 1.1

Порівняльна Таблиця додатків для самодіагностики SCORE та Framingham Calc

Характеристика	SCORE -Калькулятор	Framingham Calc
Плюси		
1.Легкість використання	Зручний мобільний додаток із приємним інтерфейсом.	Зручний мобільний додаток із приємним інтерфейсом.
2.Інформаційна доступність	Надає основні знання про серцево-судинний ризик.	Додаткові дані про відсоток ризику та вік серця.
3.Гнучкість в застосуванні	Можливість використання на мобільних пристроях.	Можливість використання на мобільних пристроях.
4. Інтерактивність	Простий калькулятор без додаткових функцій.	Деякі рекомендації стосовно ризикових факторів.
5. Визначення відсотку ризику	Результати представлені в числовій формі.	Вказано відсоток ризику та вік серця.
Мінуси		
1. Відсутність збереження результатів	Немає можливості зберегти попередні результати.	Відсутність історії та моніторингу результатів з часом.
2. Обмеженість взаємодії з користувачем	Мінімальна можливість взаємодії та залучення користувача.	Особисті рекомендації можуть бути загальними та неефективними.
3. Брак Функцій Авторизації та Реєстрації	Відсутність можливості створити обліковий запис для збереження даних.	Відсутність авторизації та можливості ведення статистики.

4. Неспроможність вести спостереження	Немає засобів для моніторингу змін в часі.	Відсутність функцій для ведення спостереження за показниками.
---------------------------------------	--	---

Загалом, звернення до аналогів та існуючих досліджень надає важливі вказівки для того, які можливості та функції будуть важливими для розроблення програмного продукту з оцінки серцево-судинного ризику:

1. Методи та Алгоритми: Обрати підходи та алгоритми для оцінки серцево-судинного ризику, враховуючи позитивні та негативні сторони підходів, які були використані в аналогічних дослідженнях.

2. Дані: Визначити, які дані буде використано для дослідження (клінічні дані).

3. Авторизація та реєстрація: Додаток може надати можливість користувачам створювати облікові записи, що дозволить зберігати та відстежувати їхні дані в часі. Це також забезпечить більш особистий підхід до оцінки ризику.

4. Введення даних: Важливо мати інтерфейс для введення основних медичних даних, таких як вік, стать, вага, артеріальний тиск, рівень холестерину тощо. Ці дані використовуватимуться для розрахунку ризику.

5. Алгоритми оцінки ризику: Базуючись на результаті порівняння алгоритмів Framingham та SCORE, додаток може здійснювати розрахунок ризику серцево-судинних захворювань для користувача. Важливо розробити точний та надійний алгоритм.

6. Візуалізація результатів: Додаток може відображати результати оцінки ризику в зручному графічному вигляді. Це може допомогти користувачам легше зрозуміти свій стан та поточний ризик.

7. Відстеження показників: Додаток може зберігати історію оцінок ризику користувача. Це дозволить користувачам спостерігати за змінами їхнього ризику серцево-судинних захворювань відповідно до змін в їхніх медичних даних та поведінці.

8. Поради та рекомендації: Засновано на результаті оцінки ризику, лікар може надавати користувачам конкретні поради щодо покращення їхнього здоров'я та зниження ризику серцево-судинних захворювань.

9. Безпека та конфіденційність: Важливо забезпечити високий рівень захисту особистих медичних даних користувачів.

10. Оновлення та покращення: Додаток повинен бути гнучким для внесення оновлень, покращень та додавання нових алгоритмів оцінки ризику з часом.

1.3 Технічне завдання

Виявлення ранніх змін функціонального стану серцево-судинної системи та прогнозування ризику розвитку захворювань при масових обстеженнях населення – одне з важливих завдань профілактичної медицини. Тому актуальним стає створення ефективних алгоритмів оцінки стану серцево-судинної системи, автоматизація цих алгоритмів з метою надання лікарю допомоги у прийнятті рішень та оптимізації його дій. Тобто визначити, який з алгоритмів має низький відсоток похибки при прогнозуванні ризику захворювання пацієнтів різних категорій (віку, статі), більш ефективний алгоритм може сприяти покращенню діагностики та профілактики серцево-судинних захворювань судинної, що позитивно впливає на здоров'я та життя пацієнтів.

В даній роботі потрібно:

1. Зібрати необхідні дані та параметри для обох алгоритмів.
2. Оптимізувати дані.
3. Реалізувати алгоритм персоналізованої діагностики.
4. Описати модель:
 - a. Параметри алгоритму SCORE;
 - b. Параметри алгоритму Фремінгема;
 - c. Властивості та особливості алгоритмів.

5. Тестування обох алгоритмів за допомогою наборів тестових даних, які відображають реальні клінічні випадки.

6. Порівняння результатів, отриманих за допомогою обох алгоритмів щодо точності та з результатами інших порівняльних аналізів.

7. Аналіз результатів за допомогою методів штучного інтелекту, вказати, який алгоритм дає кращі результати в контексті цілей аналізу.

8. Обрати більш ефективний алгоритм.

9. Реалізувати веб-додаток для персоналізованої діагностики.

10. Передбачити доступ для лікарів, щоб вони могли відстежувати стан пацієнтів.

11. Підготувати звіт з результатами дослідження.

Бізнес-вимоги:

1. Забезпечити надійну роботу системи для користувачів з урахуванням їхньої безпеки та конфіденційності.

2. Забезпечити можливість надання консультацій користувачам на основі індивідуального аналізу їх медичних даних.

3. Реалізувати можливість представлення системи у вигляді веб-додатку для максимальної зручності використання.

Вимоги користувачів:

1. Забезпечити інтуїтивно зрозумілий та безпечний інтерфейс для реєстрації та авторизації користувачів.

2. Реалізувати можливість введення та зміни медичних даних користувача.

3. Забезпечити можливість перегляду статистики здоров'я за різними періодами та загальний звіт.

4. Реалізувати можливість отримання персоналізованого діагнозу та медичних рекомендацій.

5. Забезпечити обмежений, безпечний доступ лікарів до даних своїх пацієнтів.

Функціональні вимоги:

1. Реалізувати захищену систему авторизації та реєстрації користувачів.
2. Створити інтуїтивно зрозумілий інтерфейс для введення та збереження медичних даних користувачів.
3. Використати бібліотеки машинного навчання для створення моделей оцінки ризику за алгоритмами SCORE і Framingham.
4. Провести аналіз даних та здійснити порівняльний аналіз обох алгоритмів.
5. Реалізувати можливість отримання персоналізованого діагнозу та медичних рекомендацій.

Системні вимоги:

1. Система повинна відповідати мінімальним технічним вимогам, включаючи 128Мб пам'яті, процесор із тактовою частотою 600MHz, операційну систему Windows 10, сумісні браузері (Opera, Chrome, ...), а також використання JS та MongoDB.
2. Негайно виявляти та усувати всі технічні несправності.
3. Забезпечити конфіденційність даних користувачів та відсутність рекламних матеріалів в додатку.
4. Забезпечити безпеку даних, запобігаючи їх використання сторонніми сервісами.

Тестування системи:

1. Провести тестування алгоритмів на реальних медичних наборах даних.
2. Оцінити ефективність системи під час практичного використання.
3. Забезпечити адекватне тестування всіх функцій та форм.
4. Виконати тестування вразливостей системи.
5. Оцінити коректну роботу веб-додатка в усіх умовах використання.

1.4 Вибір засобів реалізації програмного продукту

Для реалізації алгоритмів штучного інтелекту та аналізу даних буде використано бібліотеки та інструменти з області машинного навчання, такі як:

1. Python: Вибір Python як основної мови програмування для реалізації алгоритмів штучного інтелекту є дуже популярним через його простоту та багатий екосистему бібліотек.

2. scikit-learn: Це одна з найбільш популярних бібліотек для машинного навчання в Python. Вона надає широкий спектр алгоритмів для класифікації, регресії, кластеризації, а також інструменти для оцінки моделей.

3. TensorFlow та PyTorch: Ці бібліотеки відомі своєю підтримкою глибокого навчання. Вони дозволяють створювати та навчати нейронні мережі для більш складних завдань аналізу даних та прогнозу ризику.

4. Pandas: Ця бібліотека надає зручні структури даних та інструменти для обробки та аналізу даних. Вона допоможе підготувати дані для навчання моделей.

5. Matplotlib та Seaborn: Ці бібліотеки дозволяють візуалізувати дані та результати аналізу, що спростить розуміння результатів.

6. Jupyter Notebook: Це інтерактивне середовище для програмування, яке дозволяє поєднувати код, текстові описи та графіку в одному документі. Завдяки ньому можна крок за кроком розробити та документувати аналіз.

Стек MEVN був обраний для створення потужного та динамічного веб-додатку з ефективним дизайном зовнішнього та внутрішнього інтерфейсу. Стек технологій — це набір інструментів і шаблонів, які використовуються під час розробки програмного продукту [24]. Ці фреймворки та інструменти були спеціально обрані, щоб максимізувати ефективність роботи програмного забезпечення.

MongoDB, Express.js, VueJS та Node.js є частиною MEVN.

- Node.js — це кросплатформа з відкритим вихідним кодом, яка дозволяє розробникам створювати будь-яке програмне забезпечення на JavaScript, що

працює на стороні сервера. Це середовище виконання, яке працює безпосередньо з операційною системою і не потребує браузера. Таким чином, середовище Node надає API операційної системи своїм програмам, щоб вони могли отримати доступ до файлової системи, системних бібліотек або запущених процесів, таких як HTTP-сервери. Node має багато переваг, включаючи високу продуктивність. Диспетчер пакетів Node (NPM) може мати доступ до сотень тисяч пакетів, які можна використати, це найкращий механізм дозволу залежностей у своєму класі, і він може використовуватися для автоматизації процесу побудови програмних інструментів.

- VueJS є потужним інструментом для створення динамічних веб-інтерфейсів за допомогою JavaScript. Цей фреймворк відкритого коду значно полегшує розробку веб-додатків і легко інтегрується в різні проекти. VueJS визначається не лише своєю ефективністю, але й простотою в освоєнні, роблячи його відмінним вибором для новачків і досвідчених розробників.

- Express – це потужний веб-фреймворк для Node.js, який забезпечує розробникам широкі можливості. Окрім створення обробників запитів та визначення маршрутів, він допомагає працювати з шаблонами та іншими аспектами веб-розробки. Express є відмінним вибором для створення масштабованих та продуктивних веб-додатків.

- CSS – це мова стилів, яка визначає зовнішній вигляд веб-сайтів. Використовуючи CSS, можна зручно керувати оформленням елементів і забезпечувати їхню правильну відображення на веб-сторінці. Завдяки CSS, можливо створювати красиві та користувацько-орієнтовані веб-інтерфейси.

- MongoDB – це система управління базами даних з документ-орієнтованим підходом. За допомогою форматів BSON і JSON документи можуть легко зберігатися та передаватися в системі. MongoDB є рішенням NoSQL, що підтримує ієрархію, фрагментацію та інші особливості для ефективного управління даними. Завдяки MongoDB, можна легко масштабувати свою базу даних та отримувати високу продуктивність у веб-додатках.

Висновки до першого розділу

У цьому розділі було сформульовано технічне завдання для створення програмного продукту, спрямованого на виявлення ранніх змін у функціональному стані серцево-судинної системи та прогнозування ризику розвитку захворювань при масових обстеженнях населення. Основні завдання проекту включають збір необхідних даних, оптимізацію даних, реалізацію алгоритму персоналізованої діагностики та порівняння ефективності двох алгоритмів: SCORE та Фремінгема.

Розглянуто алгоритми SCORE Framingham , їх ціль та виникнення, методика застосування.

Ключові етапи проекту охоплюють збір параметрів для обох алгоритмів, розробку та оптимізацію алгоритму персоналізованої діагностики, аналіз результатів та вибір найефективнішого алгоритму. Для цього використовуватимуться бібліотеки та інструменти машинного навчання, такі як scikit-learn, TensorFlow, PyTorch, та Pandas.

Окрім того, передбачено створення веб-додатку для персоналізованої діагностики та забезпечення його безпеки та зручності використання для користувачів.

Для реалізації веб-додатку обрано стек технологій MEVN (MongoDB, Express.js, VueJS, Node.js), який забезпечить потужність, динаміку та ефективний дизайн. Кожен компонент стеку виконує важливі функції: Node.js дозволяє створювати серверну частину, MongoDB служить базою даних, а VueJS забезпечить інтерактивний та легкий у використанні інтерфейс.

Зазначені засоби та технології мають на меті забезпечити надійність, ефективність та зручність використання програмного продукту, що відповідає поставленим завданням та вимогам. У наступних розділах планується детальніше розглянути кожен етап розробки та виконати необхідні тести для перевірки працездатності системи.

РОЗДІЛ 2 ПРАКТИЧНА РЕАЛІЗАЦІЯ АЛГОРИТМІВ

2.1 Побудова моделі аналізу і практична реалізація

В даному розділі розглянуто етап побудови моделі аналізу ризиків на основі розробленого коду. Мета цього етапу полягає в визначенні й класифікації ризиків за допомогою двох алгоритмів: Framingham та SCORE. Описані процеси включають завантаження даних, їхню обробку та подальший аналіз з використанням визначених алгоритмів.

Для аналізу було завантажено дані з CSV-файлу, які потім пройшли процес фільтрації. Непотрібні рядки та стовпці були видалені, забезпечуючи наявність лише необхідних даних для аналізу ризиків серцево-судинних захворювань.

```
# Зчитування датасету
data = pd.read_csv('/content/cardio_train.csv', delimiter=';')
column_names = data.columns
print(column_names)
data = data.dropna()
data = data[(data['ap_hi'] >= 100) & (data['ap_hi'] < 200)]
data = data[(data['gluc'] >= 1) & (data['gluc'] <= 10)]
data = data[(data['gender'] >= 1) & (data['gender'] <= 2)]
data = data[(data['smoke'] >= 0) & (data['smoke'] <= 1)]
data = data[(data['smoke'] >= 0) & (data['smoke'] <= 1)]
data = data[(data['alco'] >= 0) & (data['alco'] <= 1)]
```

Отриманий набір даних було поділено на дві частини: навчальний та тестовий. Навчальний набір використовувався для тренування моделей, а тестовий – для оцінки їхньої точності та ефективності.

Далі були реалізовані функції для обчислення ризиків за алгоритмами. Тут представлено частина коду, решту можна переглянути у Додатку А.

Спершу створили алгоритм Framingham:

```
# Функція для алгоритму Framingham
def calculate_simple_framingham_risk(days, gender, smoking, blood_pressure, cholesterol):
    days_in_year = 365.25
    age = round(days / days_in_year)
    female_coefs = [-7, -3, 0, 3, 6, 8, 10, 12, 14, 16]
    male_coefs = [-9, -4, 0, 3, 6, 8, 10, 11, 12, 13]
    # Коефіцієнти для total cholesterol
    total_cholesterol_coefs = [
        [0, 4, 8, 11, 13],
        [0, 3, 6, 8, 10],
        [0, 2, 4, 5, 7],
        [0, 1, 2, 3, 4],
        [0, 1, 1, 2, 2],
        [0, 4, 7, 9, 11],
        [0, 3, 3, 6, 8],
        [0, 2, 5, 4, 5],
        [0, 1, 1, 2, 3],
        [0, 0, 0, 1, 1]
    ]
```

```

# Коефіцієнти для куріння
smoking_coeffsF = [9, 7, 4, 2, 1]
smoking_coeffsM = [8, 5, 3, 1, 1]
# Коефіцієнти для systolic blood pressure
female_coeffs_sbp_coeffs = [0, 1, 2, 3, 4]
male_sbp_coeffs = [0, 0, 1, 1, 2]
smoke_points=0
sbp_points=0
chol_points=0
age_points=0
risk_percentage=0
# Визначення віку
if gender == 1: # Жінка
    age_coeffs = female_coeffs
    if 20<age <= 34:
        age_points=age_coeffs[0]
    ...
    age_points=age_coeffs[9]
    # Визначення балів за total cholesterol
    if age < 40:
        cholesterol_coeffs = total_cholesterol_coeffs[0]
        if smoking==1:
            smoke_points=smoking_coeffsF[0]
    elif 40 <= age < 50:
        cholesterol_coeffs = total_cholesterol_coeffs[1]
        if smoking==1:
            smoke_points=smoking_coeffsF[1]
    elif 50 <= age < 60:
        cholesterol_coeffs = total_cholesterol_coeffs[2]
    ...
    if blood_pressure < 120:
        sbp_points = female_coeffs_sbp_coeffs[0]
    elif 120<= blood_pressure <= 129:
        sbp_points = female_coeffs_sbp_coeffs[1]
if gender == 1: # Жінка
    if total_points<9:
        risk_percentage=1
    ...
else: # Чоловік
    ...
    elif total_points==15:
        risk_percentage=20
    elif total_points==16:
        risk_percentage=25
    elif 17<=total_points:
        risk_percentage=30

return risk_percentage

```

Далі реалізували алгоритм SCORE:

```

# Функція для алгоритму SCORE
def calculate_score_risk(days, gender, smoking, blood_pressure, cholesterol):
    # Коефіцієнти SCORE
    gen,age,ad,smoke,chol=0,0,0,0,0
    s0,s010,alpha,p,cs0,cs10,ncs0,ncs10,wc,wnc=0,0,0,0,0,0,0,0,0,0
    bchol,bsbp,bsm,cs,cs1,ncs,ncs1,r,r1=0,0,0,0,0,0,0,0,0
    age=round(days/366)
    gen=gender
    chol=cholesterol
    if gen==2:
        alpha=-21.0
        p=4.62;
    else:
        alpha=-28.7
        p=6.23;
    cs0=math.exp(-math.exp(alpha)*math.pow(age-20.0,p))
    cs10=math.exp(-math.exp(alpha)*math.pow(age-10.0,p))

```

```

if gen==2:
    alpha=-25.7
    p=5.47
else:
    alpha=-30.0
    p=6.42
ad=blood_pressure
ncs0=math.exp(-math.exp(alpha)*math.pow(age-20.0,p))
ncs10=math.exp(-math.exp(alpha)*math.pow(age-10.0,p));
bchol=0.24;
bsbp=0.018;
smoke=smoking;
bsm=smoke*0.71;
wc=bchol*(chol-6.0)+bsbp*(ad-120.0)+bsm;
bchol=0.02;
bsbp=0.022;
bsm=smoke*0.63;
wnc=bchol*(chol-6.0)+bsbp*(ad-120.0)+bsm
cs=math.pow(cs0,math.exp(wc))
cs1=math.pow(cs10,math.exp(wc))
ncs=math.pow(ncs0,math.exp(wnc));
ncs1=math.pow(ncs10,math.exp(wnc))
if cs != 0:
    cs1=cs1/cs
else:
    cs1 = 0

if ncs != 0:
    ncs1 = ncs1 / ncs
else:
    ncs1 = 0
r=1.0-cs1;
r1=1.0-ncs1;
score = 100.0 * (r + r1)
return score

```

Отримані ризики були класифіковані, використовуючи визначений поріг. Результати алгоритмів Framingham та SCORE були порівняні, щоб визначити, яка модель найкраще прогнозує ризик серцево-судинних захворювань.

Для оцінки ефективності моделей були використані різноманітні метрики, такі як точність, площа під кривою ROC, середньоквадратична помилка (MSE) та середня абсолютна помилка (MAE). Це дозволило отримати повну картину професійної придатності розроблених моделей.

Також були створенні додаткові моделі, які допомагають оцінити стан тиску та визначити чи відповідає він нормам конкретної особи.

Побудували модель для оціни тиску. Тут дані були розділені на ознаки та мітки. У ознакою виступає стовпець «sysBP», який представляє значення кров'яного тиску. Мітки зберігаються у стовпці «mark». Ознаки та мітки були збережені у змінні X та у відповідно. Для аналізу даних та прогнозу міток був вибраний алгоритм логістичної регресії.

Використовується клас `LogisticRegression` з бібліотеки `sklearn.linear_model`. Перед навчанням моделі, ознаки були масштабовані за допомогою класу `StandardScaler` з бібліотеки `sklearn.preprocessing`. Це допомагає забезпечити однаковий масштаб значень ознак. Навчання моделі проводилось за допомогою методу `fit`, передаючи тренувальні дані `X_train` та `y_train`.

Після навчання моделі було проведено оцінку її продуктивності на тестовому наборі даних. Для цього використовувалися метрики точності (`accuracy_score`) та F1-показника (`f1_score`) з бібліотеки `sklearn.metrics`. Значення метрик були виведені на екран.

Після успішної навчання та оцінки модель було збережено для подальшого використання. Використана функція `dump` з бібліотеки `joblib`, яка зберігає модель у форматі `Joblib`. Для завантаження моделі використовується функція `load`.

```
# Розділення даних на ознаки та мітки
X = data['sysBP'].values.reshape(-1, 1)
y = data['mark']
# Підготовка даних для навчання
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Масштабування ознак
from sklearn.preprocessing import StandardScaler
# Вибір та навчання моделі
model = LogisticRegression()
model.fit(X_train, y_train)
from joblib import dump
# Прогноз міток на тестовому наборі
y_pred = model.predict(X_test)
# Оцінка продуктивності моделі
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred, average='weighted')
print(f'Accuracy: {accuracy * 100}%')
print(f'F1 Score: {f1}')
# Використання навченої моделі для прогнозу на нових даних
new_data = pd.DataFrame({'sysBP': [152, 140, 134]})
new_pred = model.predict(new_data)
print(f'New predictions: {new_pred}')
```

Модель для оцінки норми тиску будувалася майже так само, як і попередня модель. Сперше завантаження набору даних з використанням функції `read_csv` з бібліотеки `pandas`. Далі сортування даних за зростанням віку для подальшого групування. Вибір вхідних змінних (вік та стать) та вихідної змінної (тиск). Розділення даних на тренувальний та тестовий набори за допомогою функції `train_test_split` з бібліотеки `sklearn.model_selection`. Об'єднання тренувальних даних в один `DataFrame` для подальшої обробки. Для побудови моделі аналізу

був використаний алгоритм групування даних та обчислення моди тиску для кожної групи. Опис кроків:

1. Групування тренувальних даних за віком та статтю.
2. Обчислення моди тиску для кожної групи за допомогою функції `agg` та `value_counts` з бібліотеки `pandas`.
3. Отримання `DataFrame` з групованими даними, що містить вік, стать та моду тиску для кожної групи.
4. Зберегти модель

```
# вхідні змінні: вік та стать
X = df[['age', 'male']]
# Вибрати вихідну змінну
y = df['sysBP']
# Розділити дані на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Об'єднати X_train та y_train в один DataFrame
train_data = pd.concat([X_train, y_train], axis=1)

# Групувати дані за віком та статтю та обчислити моду тиску для кожної групи
pressure_mode_by_group = train_data.groupby(['age', 'male'])['sysBP'].agg(lambda x:
x.value_counts().index[0]).reset_index()

joblib.dump(pressure_mode_by_group, 'pressure_model.pkl')

loaded_model = joblib.load('pressure_model.pkl')
# Заповнити відсутні значення середнім значенням тиску

pressure_mean = loaded_model.groupby(['age', 'male'])['sysBP'].mean().reset_index()
# Визначити норму тиску для кожного прогнозу
norms = []
for i in range(len(X_test)):
    age = X_test.iloc[i]['age']
    male = X_test.iloc[i]['male']
    pressure_mean_value = pressure_mean[(pressure_mean['age'] == age) &
(pressure_mean['male'] == male)]['sysBP']
    if pressure_mean_value.empty:
        # Якщо середнє значення не знайдено, використовуємо значення з найближчого
        # сусіднього віку та статі
        closest_age = pressure_mean['age'].iloc[(pressure_mean['age'] -
age).abs().argsort()[:2]].values
        closest_male = pressure_mean['male'].iloc[(pressure_mean['age'] -
age).abs().argsort()[:2]].values
        closest_pressure = pressure_mean[(pressure_mean['age'].isin(closest_age)) &
(pressure_mean['male'].isin(closest_male))]['sysBP'].values
        pressure_mean_value = closest_pressure.mean()
    else:
        pressure_mean_value = pressure_mean_value.values[0]
    norms.append(pressure_mean_value)
# поточний стан тиску
for i in range(len(norms)):
    print(f'Для віку {X_test.iloc[i]["age"]} та статі {X_test.iloc[i]["male"]} - Норма
АД: {norms[i]}')
```

2.2 Результати проведеного аналізу алгоритмів

На даному етапі було виконано побудову ROC-кривих для двох алгоритмів – Framingham та SCORE (рис. 2.1). Графіки надають візуальне представлення ефективності моделей у класифікації та визначають їхню здатність виявляти true positive та false positive результати при різних порогах класифікації. Площі під кривими (AUC-ROC) служать метрикою загальної ефективності моделей.

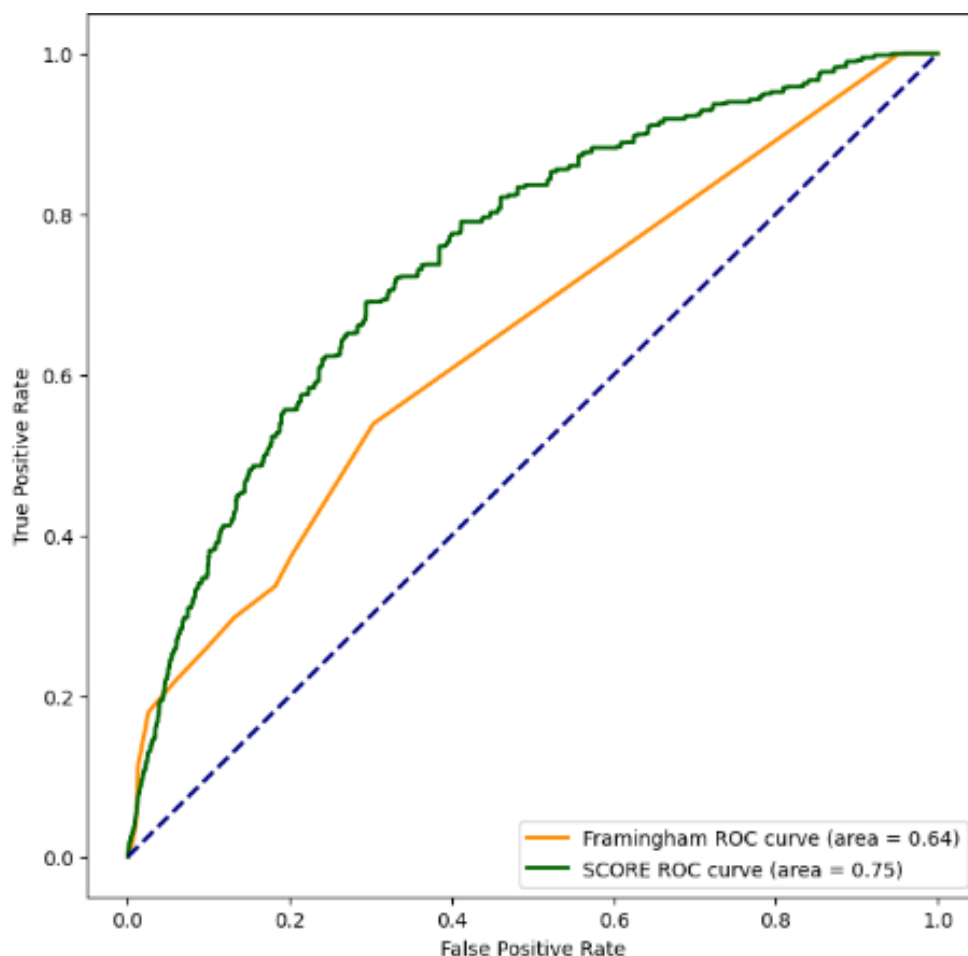


Рис. 2.1 - ROC-крива

На графіках ROC-кривих (рис. 2.1) видно, що алгоритм SCORE (AUC-ROC 75.05%) перевершує алгоритм Framingham (AUC-ROC 64.41%) у виявленні true positive та уникненні false positive результатів. Це свідчить про кращу здатність SCORE в класифікації пацієнтів з серцево-судинними захворюваннями.

Для більш об'єктивної оцінки точності прогнозів обох алгоритмів використовувалися метрики Mean Squared Error (MSE) (рис. 2.2) та Mean

Absolute Error (MAE) (рис. 2.3). MSE метрика використовується для задач регресії і вимірює середню квадратичну відстань між фактичними та передбаченими значеннями. Мінімізація MSE вказує на те, що модель добре відтворює фактичні дані. MAE – метрика, що також використовується для задач регресії і вимірює середню абсолютну відстань між фактичними та передбаченими значеннями. Вона менш чутлива до великих відхилень, порівняно з MSE, що робить її корисною у випадках, коли великі відхилення важливі. Графіки порівняння MSE та MAE для Framingham та SCORE дозволяють визначити, який алгоритм краще відтворює реальні значення цільової змінної.

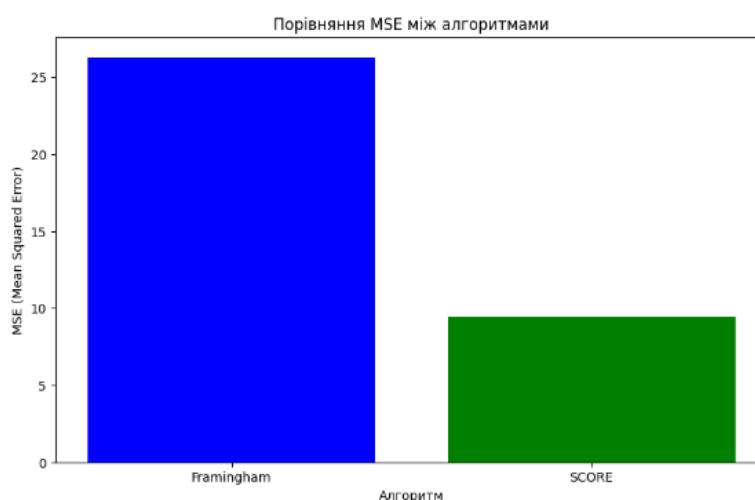


Рис. 2.2 - MSE

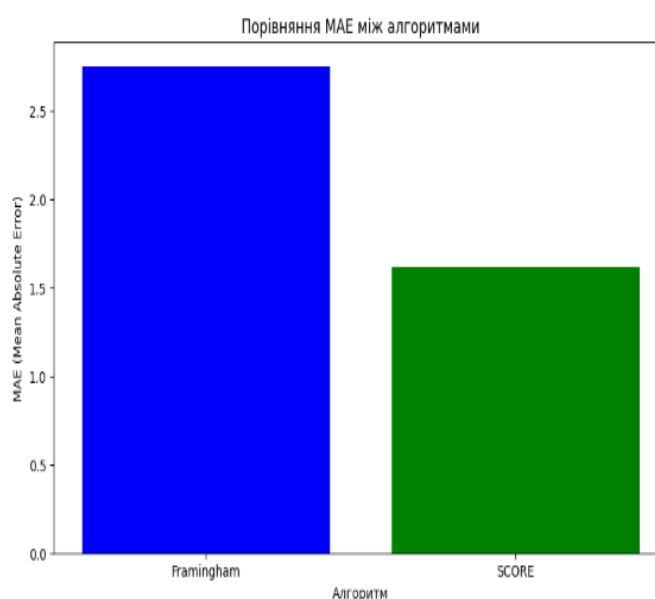


Рис. 2.3 - MAE

З аналізу графіків MSE (рис. 2.2) та MAE (рис. 2.3) видно, що алгоритм SCORE має значно менше значення MSE (939.52 проти 2629.14 у Framingham) та MAE (161.89 проти 275.07 у Framingham). Це свідчить про те, що SCORE надає більш точні прогнози щодо ризику серцево-судинних захворювань.

Зведені результати показують, що алгоритм SCORE виявляється більш ефективним, з точністю 65.92% та AUC-ROC 75.05%, порівняно з Framingham, який має точність 52.97% та AUC-ROC 64.41%. MSE та MAE також підтверджують вищу точність SCORE в порівнянні з Framingham.

2.3 Аналіз ризиків в залежності від ознак

Оцінка ризиків в залежності від різних ознак є важливим етапом у вивченні ефективності алгоритмів SCORE та Framingham. Нижче подано докладний аналіз взаємозв'язків між ризиком та різними факторами.

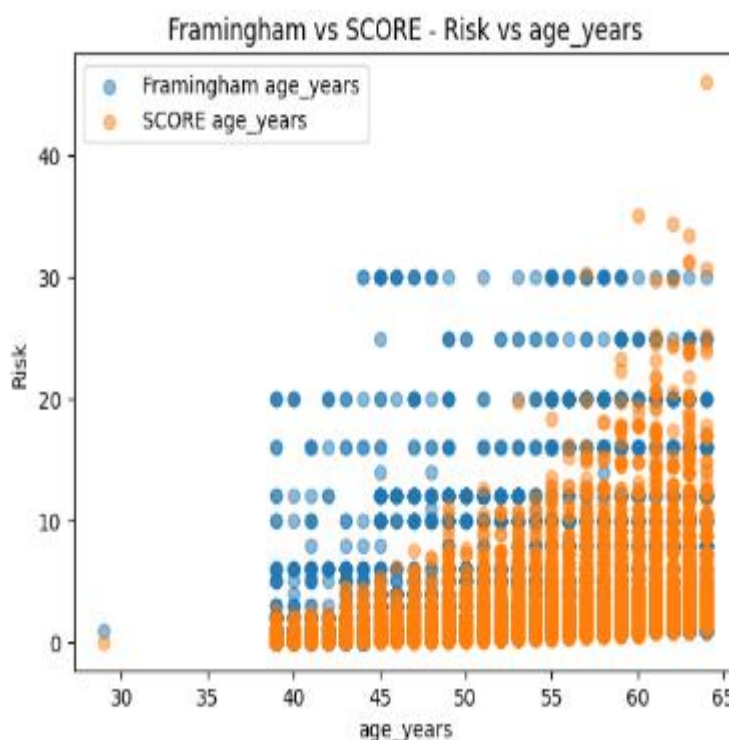


Рис. 2.4- Ризик vs Вік

На графіку взаємодії ризику та віку (рис. 2.4) видно, що обидва алгоритми виявляють збільшення ризику із зростанням віку. Важливо відзначити, що SCORE схильний видавати менше ризиковані прогнози в порівнянні з

Framingham. Це може свідчити про те, що SCORE може бути консервативнішим у визначенні ризику серцево-судинних подій для пацієнтів похилого віку.

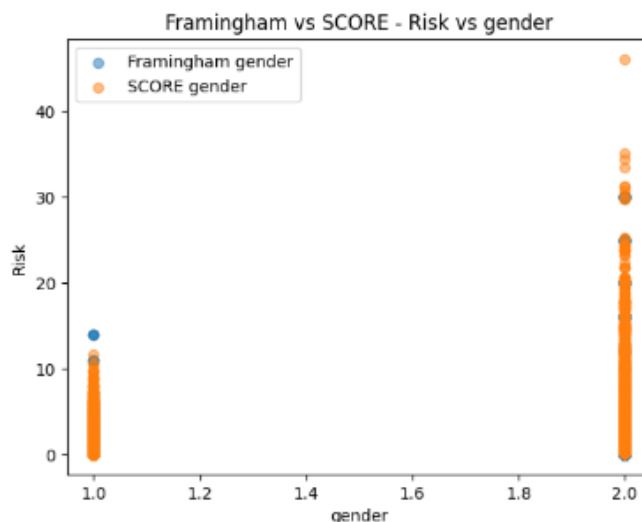


Рис. 2.5 - Ризик vs Стать

Порівняльний аналіз ризиків за статевою (рис. 2.5) ознакою показав, що SCORE та Framingham подавали схожі результати у визначенні ризику в залежності від статі, при тому ризик для чоловіків був більший. Це підтверджує консистентність обох алгоритмів у врахуванні цього фактору.

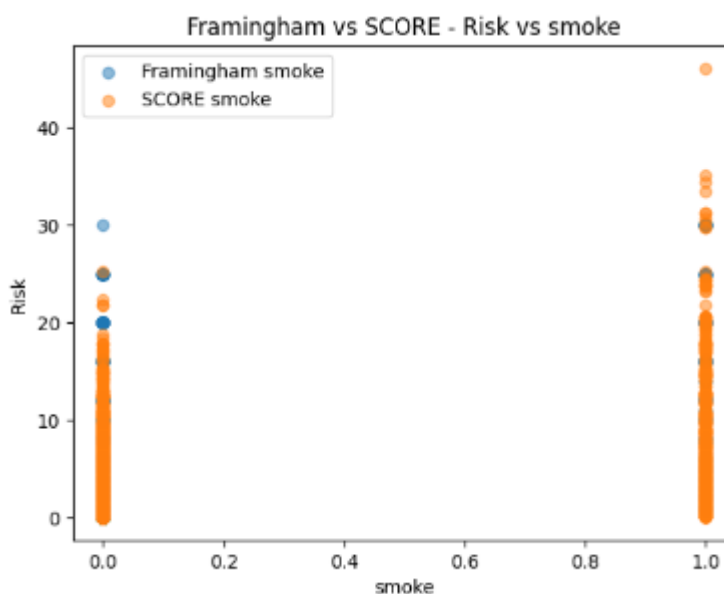


Рис. 2.6 - Ризик vs Куріння

У випадку куріння (рис. 2.6) обидва алгоритми виявляли збільшений ризик для курців. Однак SCORE відзначався більшою точністю у визначенні ризику

для цієї групи пацієнтів. Це може вказувати на те, що SCORE більше чутливий до впливу куріння на серцево-судинний ризик.

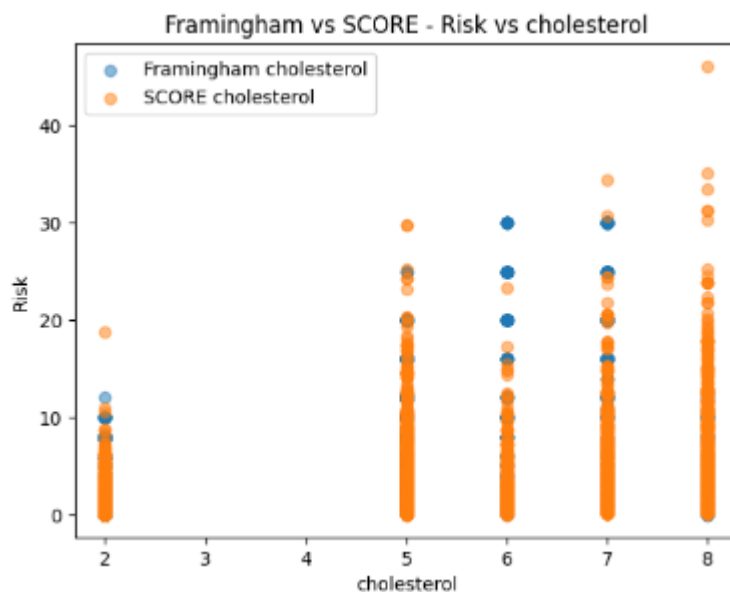


Рис. 2.7 - Ризик vs Холестерин

SCORE виявився більш чутливим до змін рівня холестерину (рис. 2.7) порівняно з Framingham. Це робить SCORE ефективнішим у визначенні ризику для пацієнтів з високими значеннями цих показників. Цей аспект важливий при індивідуалізованому підході до управління серцево-судинним ризиком.

Загальною висновком є те, що детальний аналіз взаємозв'язків між ризиком та різними факторами дозволяє краще зрозуміти, які аспекти життєвого стилю та фізичного стану можуть впливати на результати алгоритмів SCORE та Framingham. Ці відомості мають важливе значення для розробки індивідуалізованих стратегій управління серцево-судинним ризиком та попередженням серцево-судинних захворювань.

2.4 Порівняльний аналіз отриманих результатів з іншими дослідженнями

У цьому розділі розглянуто порівняльний аналіз результатів поточного дослідження з важливими висновками інших наукових робіт, що стосуються оцінки серцево-судинного ризику за допомогою алгоритмів Framingham та SCORE.

Результати дослідження «Оцінка профілю серцево-судинного ризику: порівняльний аналіз між алгоритмом CUORE та шкалою ризику Framingham» [2]:

- У роботі вказано, що оцінка CUORE виявилася менш надійною, ніж оцінка ризику за алгоритмом Framingham. Наше дослідження, де SCORE продемонстрував кращу ефективність порівняно з Framingham, розширює усвідомлення про альтернативні методи оцінки ризику.

Результати дослідження «Порівняння оцінок ризику серцево-судинних захворювань на основі Framingham та SCORE в учасників Німецького національного інтерв'ю» [4]:

- В роботі зазначено, що SCORE-HIGH може переоцінювати ризик серцево-судинних захворювань, порівняно з моделлю Framingham. У нашому дослідженні ми виявили, що SCORE виявився досить ефективним, особливо при визначенні ризику для пацієнтів з високим артеріальним тиском та високим холестеринем.

Результати дослідження «Порівняння моделей ризику Framingham і SCORE у прогнозуванні наявності та тяжкості ішемічної хвороби серця з урахуванням оцінки SYNTAX» [13]:

- Дослідження підтверджує ефективність обох моделей Framingham і SCORE в прогнозуванні ішемічної хвороби серця. У нашому дослідженні SCORE також виявився досить ефективним, але протилежно до цього дослідження, ми визначили, що SCORE може бути кращим у визначенні ризику для певних груп пацієнтів.

Загальний порівняльний аналіз результатів вказує на те, що ефективність алгоритмів оцінки серцево-судинного ризику може значно варіюватися в залежності від контексту, популяції та умов дослідження. Результати нашого дослідження, де SCORE виявився більш ефективним у визначенні ризику для певних груп пацієнтів, доповнюють існуючий корпус знань і підкреслюють необхідність індивідуалізованого підходу до оцінки серцево-судинного ризику.

Висновки до другого розділу

У другому розділі дослідження була проведена практична реалізація алгоритмів для аналізу ризиків серцево-судинних захворювань на основі розробленого коду. Отримані результати та аналіз дозволяють зробити наступні висновки:

1. Ефективність алгоритмів ROC-кривих: Здійснено побудову ROC-кривих для алгоритмів Framingham та SCORE, які визначають їхню здатність виявляти true positive та false positive результати при різних порогах класифікації. SCORE продемонстрував кращу ефективність, маючи більшу площу під кривою AUC-ROC (75.05% проти 64.41% у Framingham).

2. Оцінка точності прогнозів за MSE та MAE: Використано метрики Mean Squared Error (MSE) та Mean Absolute Error (MAE) для порівняння точності алгоритмів. SCORE виявився точнішим, маючи менші значення MSE (939.52 проти 2629.14 у Framingham) та MAE (161.89 проти 275.07 у Framingham).

3. Аналіз ризиків в залежності від ознак: Вивчено взаємозв'язки між ризиком серцево-судинних захворювань та різними факторами, такими як вік, стать, куріння, рівень холестерину. SCORE виявився ефективнішим у визначенні ризику для певних груп пацієнтів, таких як ті, хто має високий високий рівень холестерину.

4. Порівняльний аналіз з іншими дослідженнями: Результати дослідження порівняні з іншими науковими роботами, що досліджують алгоритми Framingham та SCORE. Дані вказують на важливість індивідуалізованого підходу до оцінки серцево-судинного ризику та відзначають SCORE як потенційно ефективний метод.

Загальною висновком з розділу 2 є те, що алгоритм SCORE виявився більш точним та ефективним у класифікації ризиків серцево-судинних захворювань порівняно з алгоритмом Framingham. Отже веб-додаток буде використовувати алгоритм SCORE.

РОЗДІЛ 3. ПОРЯДОК РОБОТИ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1 Визначення варіантів використання системи

У цьому підрозділі розглядається, як різні типи користувачів взаємодіють із системою, враховуючи їхні ролі та функціональні можливості. Діаграма варіантів використання надає візуальне відображення можливих сценаріїв взаємодії користувачів із системою, враховуючи різні функціональні можливості, які вона пропонує. Зазначені нижче сценарії відображають різні дії, які користувач може виконувати під час використання системи.

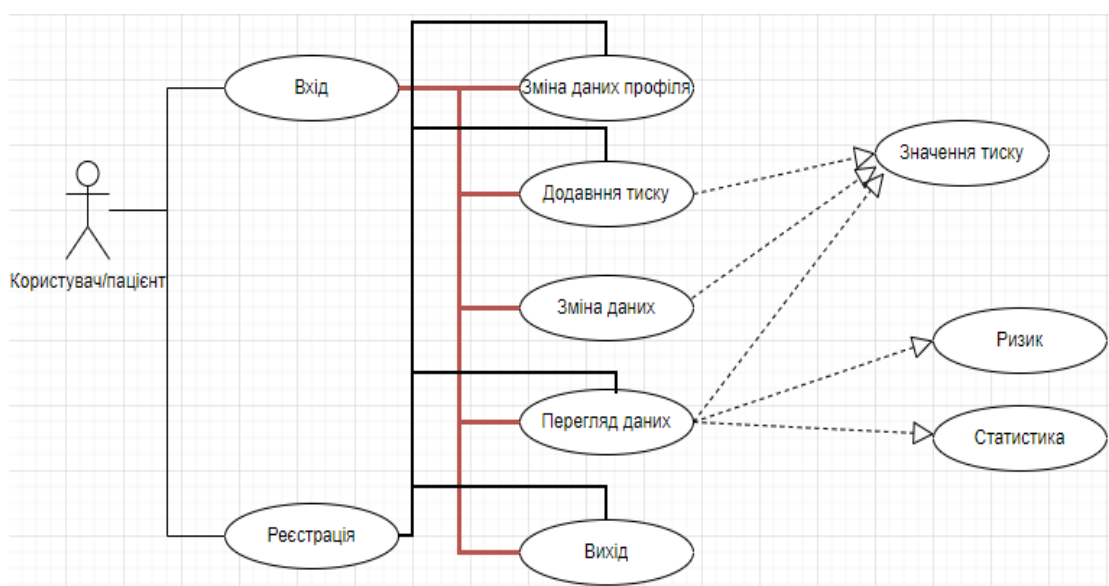


Рис. 3.1 - Користувач/пацієнт

На рисунку 3.1 представлена діаграма варіантів використання для користувача (пацієнта), він має такі можливості:

1. Авторизація в системі: Користувач може використовувати свої облікові дані для входу в систему та отримання доступу до особистого кабінету.
2. Перегляд медичних даних: Після входу в систему, користувач може переглядати свої медичні дані (такі як тиск та ризик захворювання).
3. Введення та оновлення медичних даних: Користувач має можливість вводити та оновлювати свої медичні дані, щоб система завжди мала актуальну інформацію.
4. Отримання рекомендацій від лікаря: Система надає користувачеві персоналізовані рекомендації від лікаря кожного місяця.

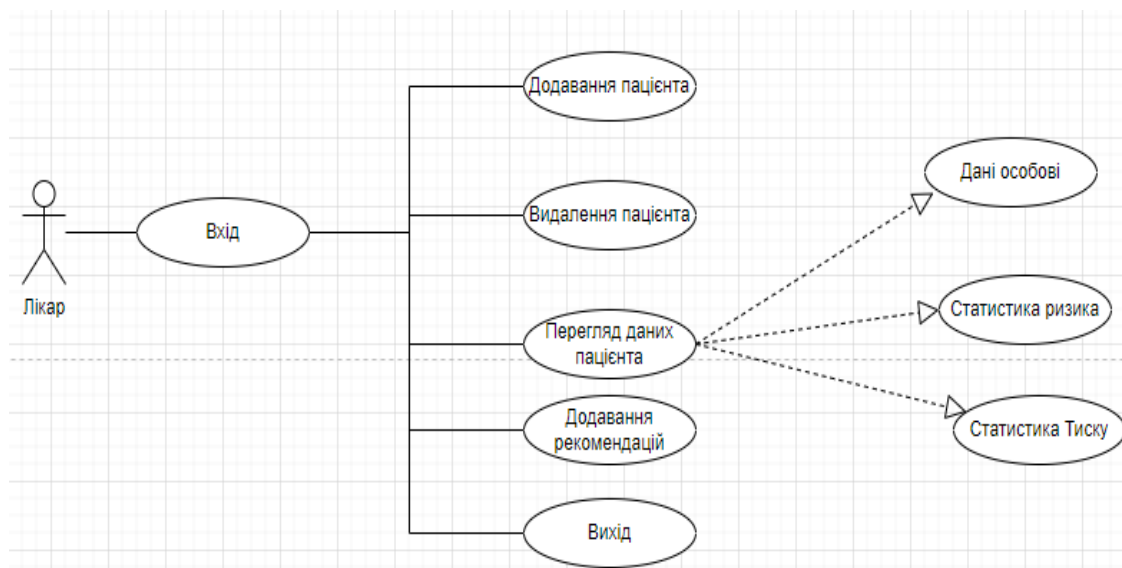


Рис. 3.2 - Лікар

На рисунку 3.2 представлена діаграма варіантів використання для лікаря, він має такі можливості:

1. Авторизація в системі: Лікар повинен увійти в систему щоб отримати доступ до даних та можливостей додатку.
2. Перегляд даних: Після входу в систему, лікар може обрати зі списку свої пацієнтів, конкретну особу та переглянути його дані особові (вік, холестерин, стать і т.д.)
3. Перегляд статистики: Обравши рік і місяць та пацієнта лікар може переглянути, яка була статистика здоров'я у пацієнта, В нього є можливість обрати тип статистики за тиском чи за ризиком захворювання
4. Видалення пацієнтів: Лікар може усунути за потреби відповідного пацієнта, наприклад якщо він завершив лікування.
5. Додавання пацієнтів: Лікар може додавати собі нових пацієнтів.
6. Виписка рекомендацій: Якщо існує необхідність лікар може виписати пацієнту необхідні рекомендації.

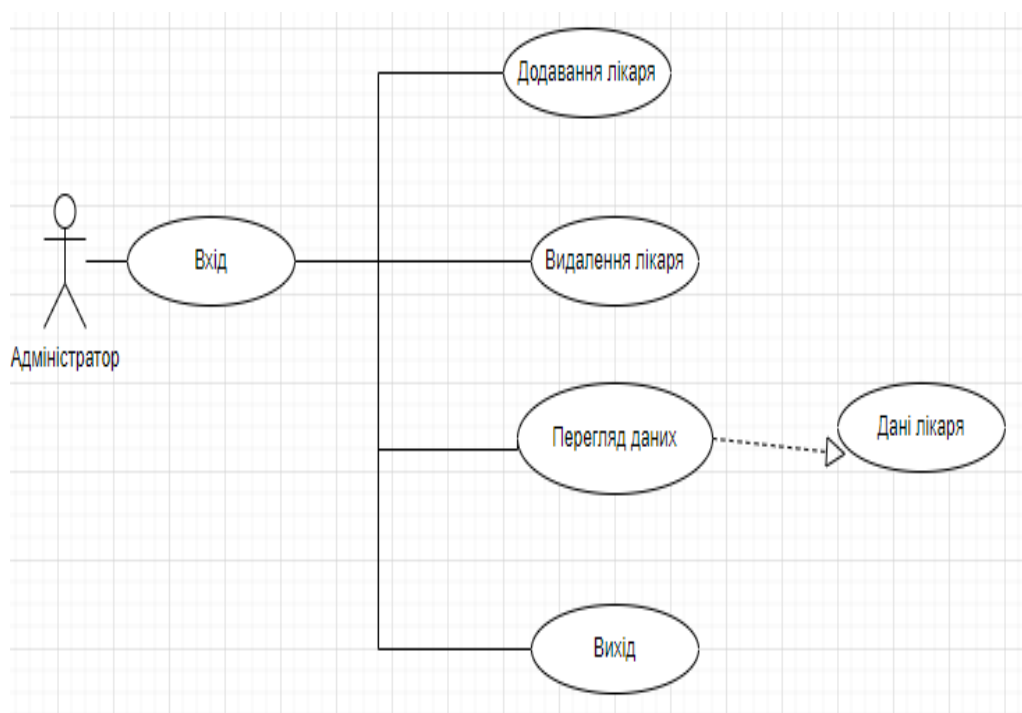


Рис. 3.3 - Адміністратор

На рисунку 3.3 представлена діаграма варіантів використання для адміністратора, він має такі можливості:

1. Авторизація: Спершу він повинен ввійти в систему для доступу
2. Додавання лікарів: Тільки адміністратор може додавати лікарів до системи
3. Видалення лікарів: Адміністратор може видаляти лікарів із системи, наприклад якщо лікар не має пацієнтів
4. Перегляд даних: Адміністратор може переглядати та перевіряти дані лікаря

3.2 Інтерфейс та порядок роботи з системою

Під час першого відвідування сайту вас вітає сторінка авторизації (рис. 3.4). Тут ви маєте можливість увійти за допомогою введення номера телефону та пароля. У разі некоректного вводу, система повідомить вас про помилки (рис. 3.5).

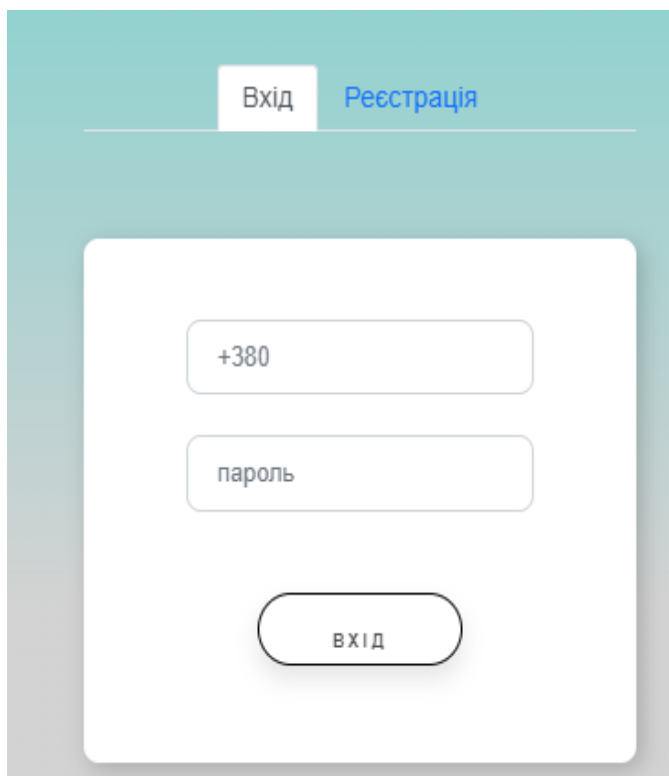


Рис. 3.4 – Сторінка авторизації пацієнта

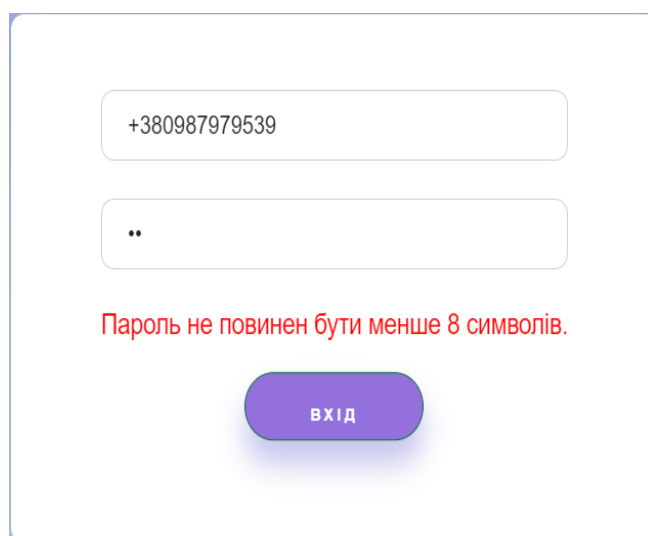


Рис. 3.5 – Перевірка на помилки

Далі розглянемо форму реєстрації (рис. 3.6). Вона складається з двох етапів та має перевірку полів. У разі необхідності ви можете повернутися до попереднього кроку.

Вхід Реєстрація

Ім'я

Фамілія

+380

пароль

ДАЛІ

Вік

Зріст

Вага

Стать

Паління

Холестерин

ГОТОВО НАЗАД

Рис. 3.6 – Форма реєстрації

Після успішної авторизації основна сторінка веб-додатка відобразиться перед вами (рис. 3.7).

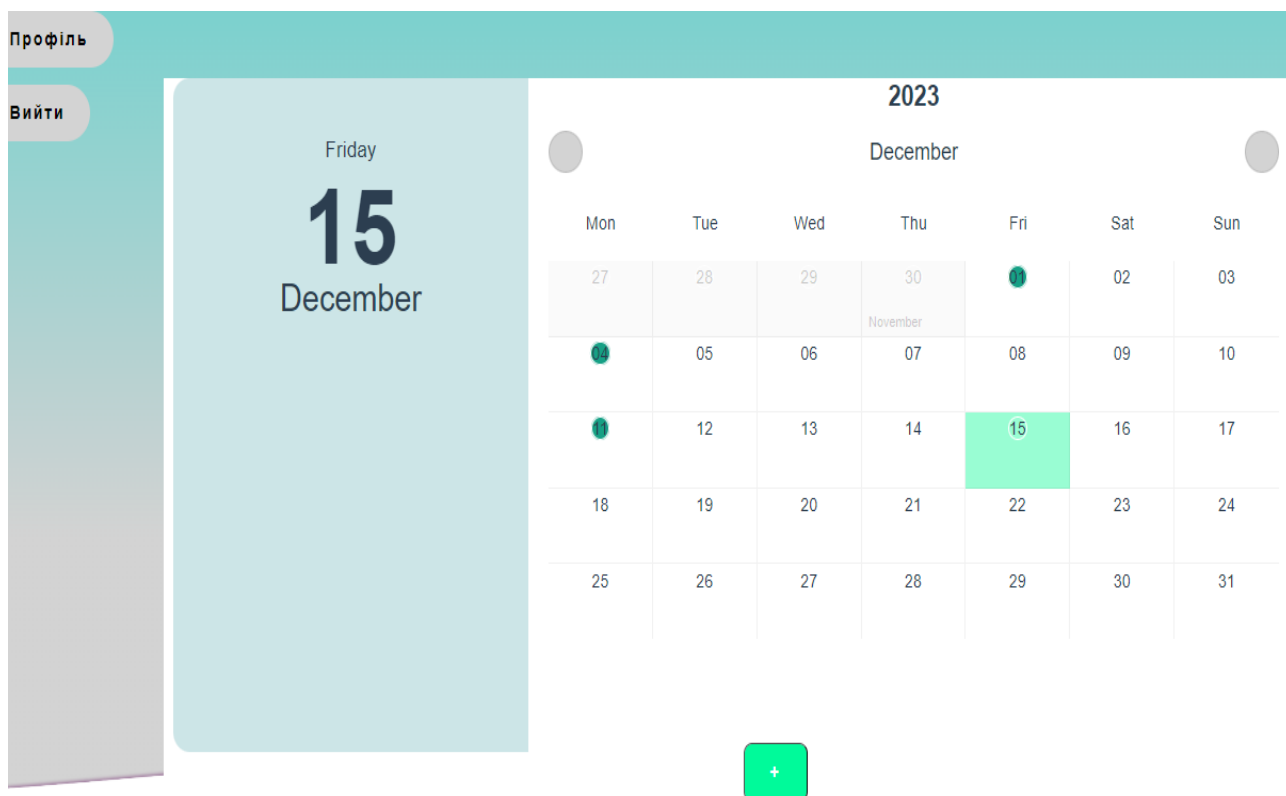


Рис. 3.7 – Основна сторінка

Обравши день на календарі, ви можете переглядати свої медичні дані за обраною датою (рис. 3.8)

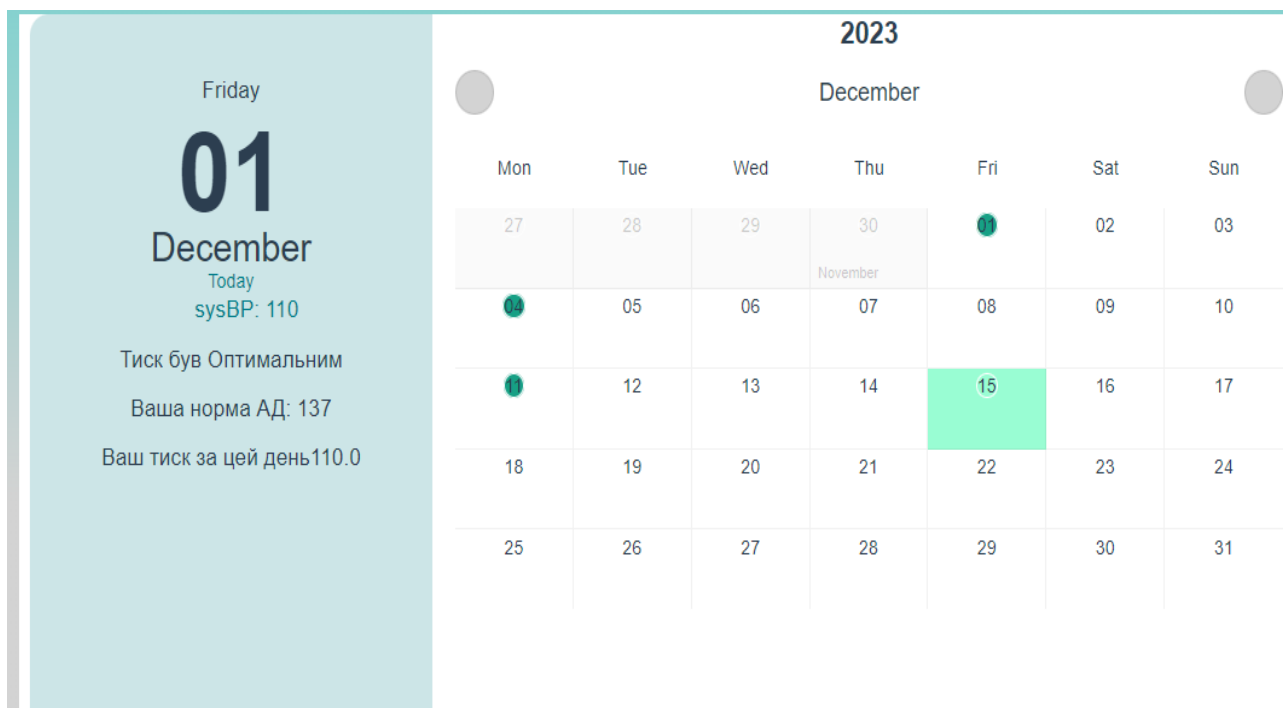


Рис. 3.8 – Перегляд даних

Натиснувши на кнопку «+», ви можете додати нові показники, відкриваючи форму для їх введення (рис. 3.9). Після додавання можна спостерігати за змінами у ваших показниках (рис. 3.10).

27 28 29 30

04 07

11 14

18 21

25 26 27 28

Сист.АД,мм.рт.ст.

Додати

Закрити

Рис. 3.9 – Додавання даних

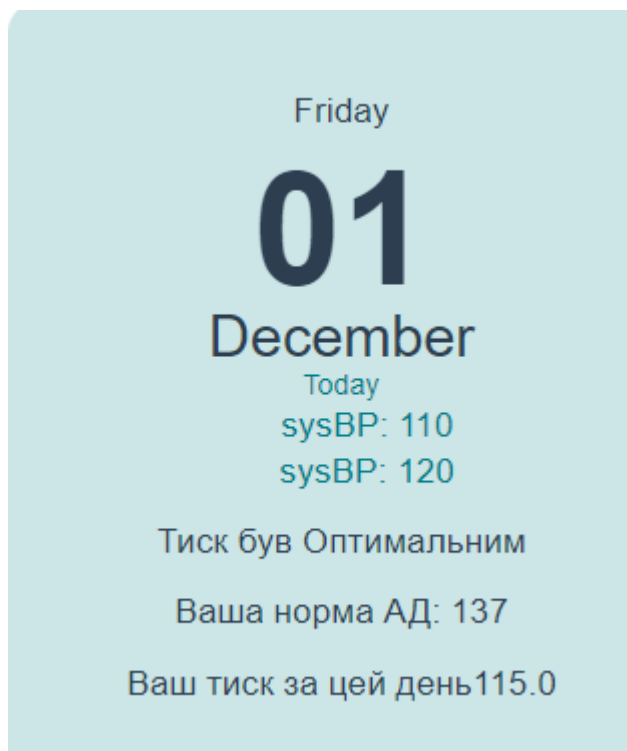


Рис. 3.10 – Оновлення даних

Також можливо змінити дані, обравши дату та параметр, і натиснувши «Змінити». З'явиться форма для редагування (рис. 3.11)

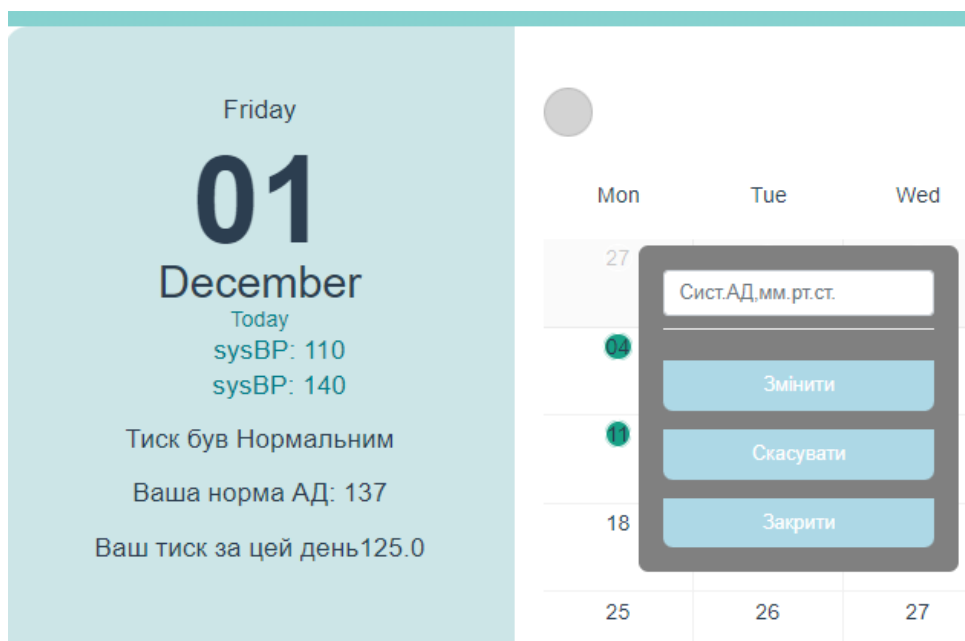


Рис. 3.11 – Зміна даних

Перейшовши у минулий місяць, ви можете переглядати статистику та ризик захворювання. Для цього потрібно натиснути на кнопку, яка показана на рисунку 3.12

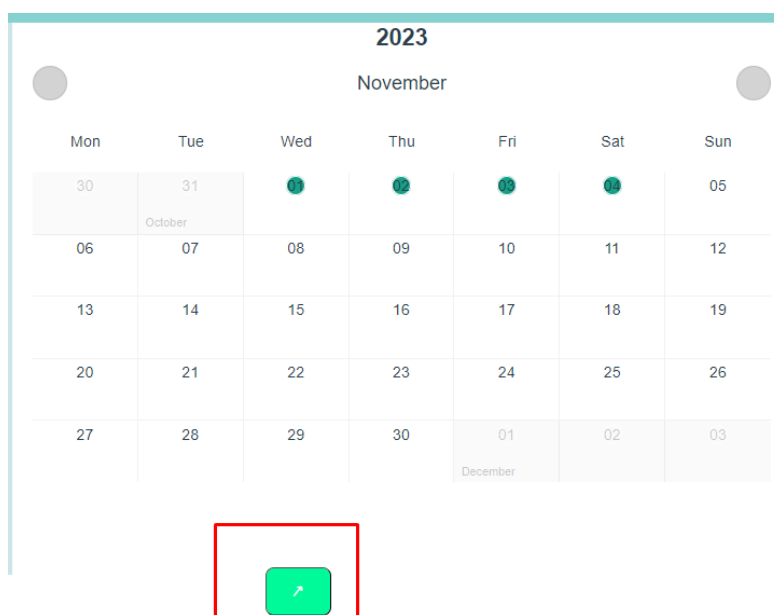


Рис. 3.12 –Кнопка статистики

Тепер відображається графік статистики за обраний місяць та процент ризику за алгоритмом SCORE, а також рекомендації від лікаря, якщо вони були (рис. 3.13).

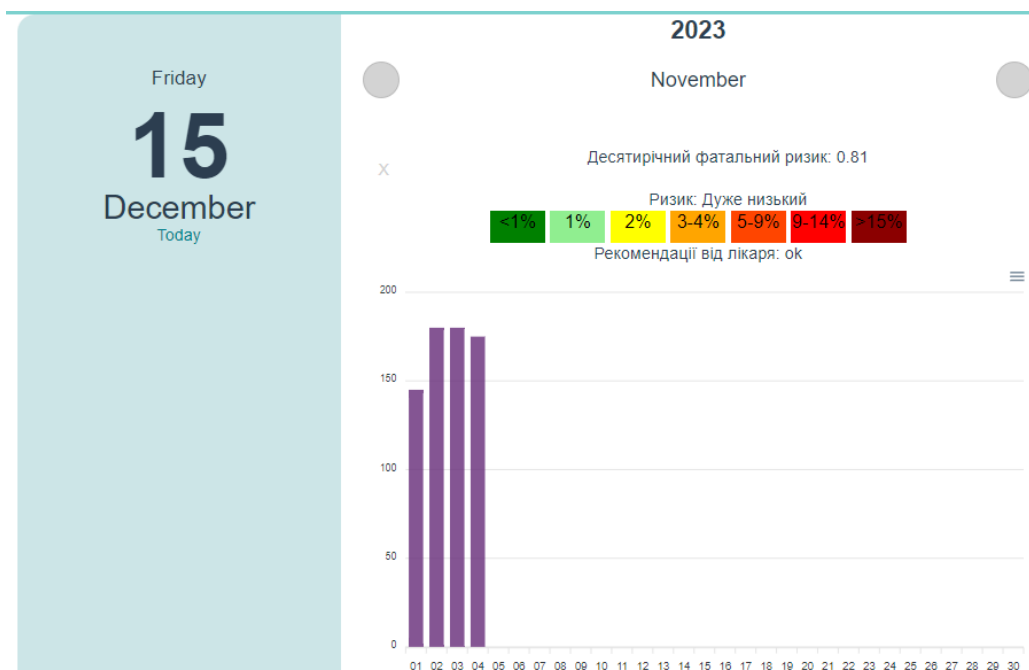
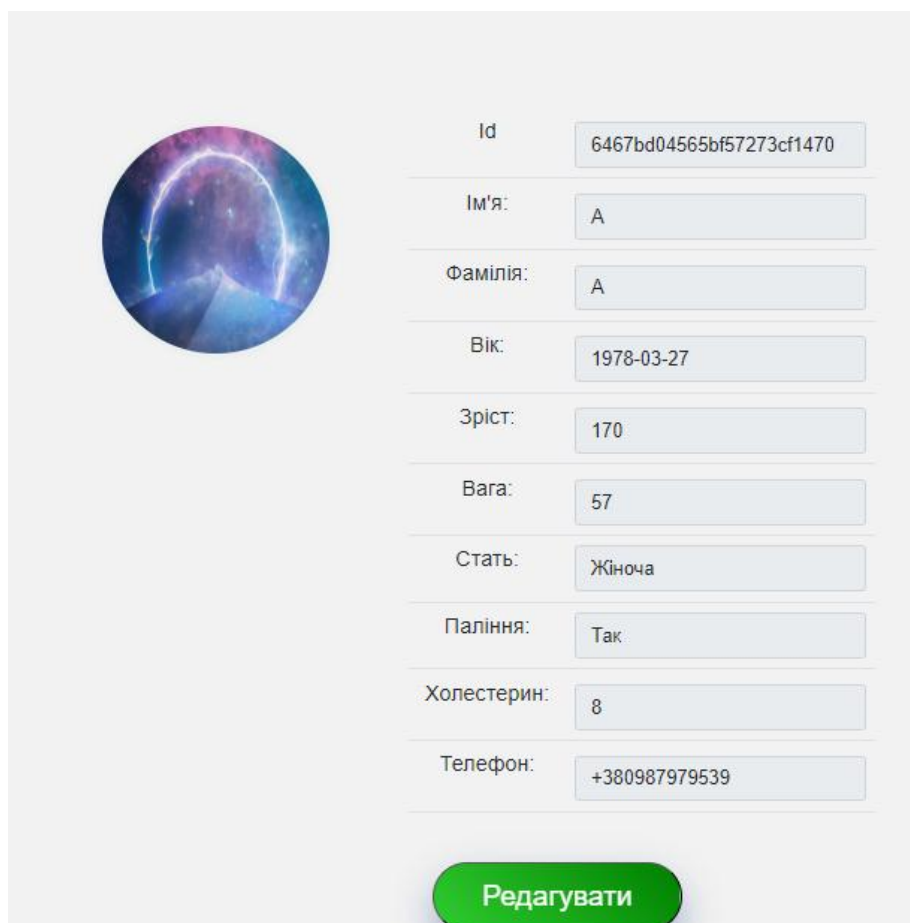


Рис. 3.13 – Статистика і ризик захворювання

Натиснувши на кнопку «Профіль» у верхньому кутку екрана, ви перейдете на сторінку користувача (рис. 3.14). Тут ви можете переглядати особисті дані та редагувати їх, натискуючи кнопку «Редагування» (рис. 3.15).

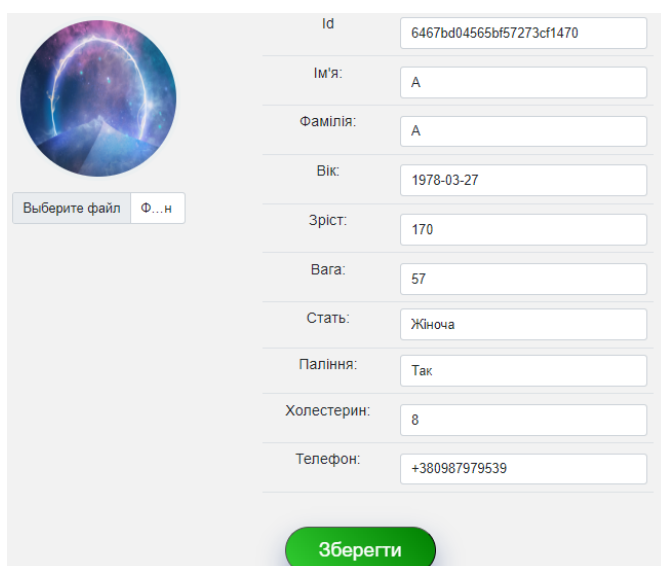


The image shows a user profile page. On the left is a circular profile picture of a person with their arms raised in a blue and purple nebula. To the right is a form with the following fields: Id (6467bd04565bf57273cf1470), Ім'я: (A), Фамілія: (A), Вік: (1978-03-27), Зріст: (170), Вага: (57), Стать: (Жіноча), Паління: (Так), Холестерин: (8), and Телефон: (+380987979539). At the bottom is a green button labeled «Редагувати».

Id	6467bd04565bf57273cf1470
Ім'я:	A
Фамілія:	A
Вік:	1978-03-27
Зріст:	170
Вага:	57
Стать:	Жіноча
Паління:	Так
Холестерин:	8
Телефон:	+380987979539

Редагувати

Рис. 3.14 – Профіль



The image shows the same user profile page as in Figure 3.14, but with the edit button labeled «Зберегти» at the bottom. The profile picture is the same. The form fields are: Id (6467bd04565bf57273cf1470), Ім'я: (A), Фамілія: (A), Вік: (1978-03-27), Зріст: (170), Вага: (57), Стать: (Жіноча), Паління: (Так), Холестерин: (8), and Телефон: (+380987979539). Below the profile picture is a button labeled «Виберіть файл» and a text input field labeled «Ф...н».

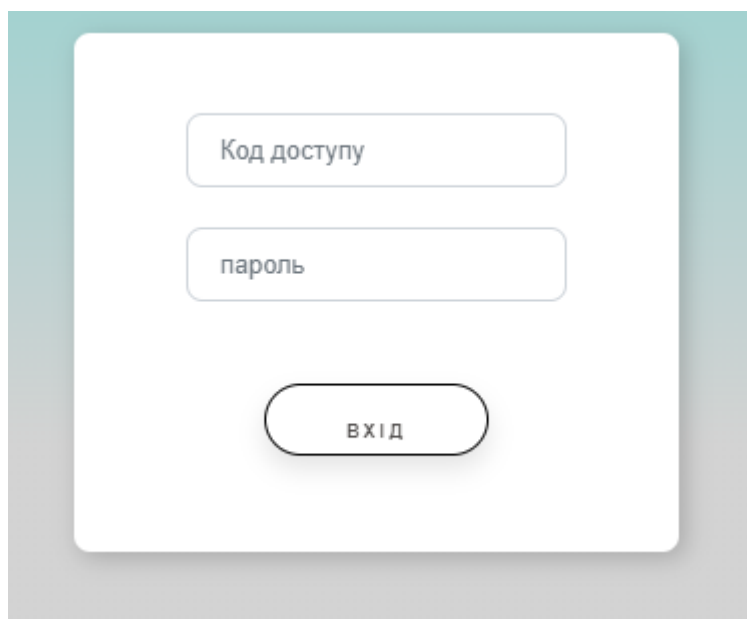
Id	6467bd04565bf57273cf1470
Ім'я:	A
Фамілія:	A
Вік:	1978-03-27
Зріст:	170
Вага:	57
Стать:	Жіноча
Паління:	Так
Холестерин:	8
Телефон:	+380987979539

Виберіть файл Ф...н

Зберегти

Рис. 3.15 – Редагування даних

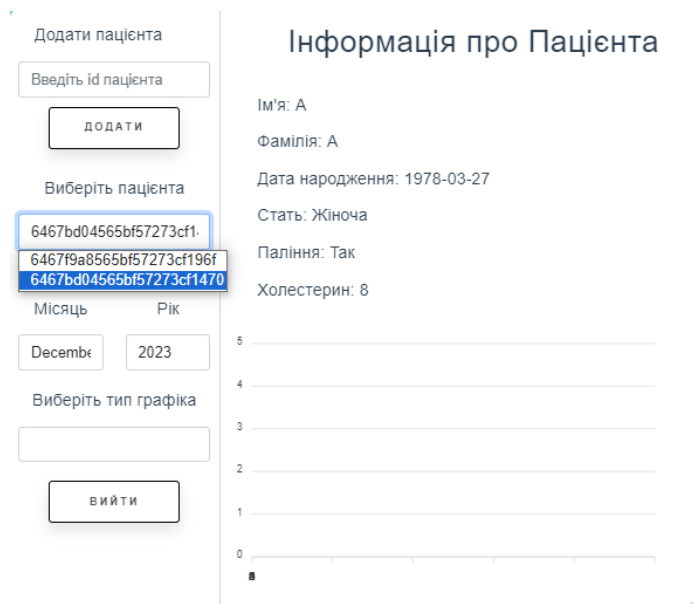
Лікарі мають зручний та функціональний інтерфейс для ефективного управління медичною інформацією пацієнтів. На рисунку 3.16 представлена форма для авторизації



The image shows a login form for doctors. It consists of a white rounded rectangle on a light blue background. Inside the rectangle, there are two input fields: the top one is labeled 'Код доступу' (Access code) and the bottom one is labeled 'пароль' (password). Below these fields is a button labeled 'ВХІД' (Login).

Рис. 3.16 –Авторизація лікарів

Після входу в систему лікар може обрати зі списку своїх пацієнтів конкретну особу та переглянути її особисті дані, такі як вік, рівень холестерину, стать тощо (рис. 3.17).



The image shows a patient management interface. On the left, there is a sidebar with the title 'Додати пацієнта' (Add patient). It contains a text input 'Введіть id пацієнта' (Enter patient ID), a 'ДОДАТИ' (Add) button, a 'Виберіть пацієнта' (Select patient) section with a list of IDs (6467bd04565bf57273cf1, 6467f9a8565bf57273cf196f, 6467bd04565bf57273cf1470), a date picker for 'Місяць' (Month) and 'Рік' (Year) (showing 'Деcember' and '2023'), a 'Виберіть тип графіка' (Select chart type) dropdown, and a 'ВИЙТИ' (Exit) button. On the right, there is a main section titled 'Інформація про Пацієнта' (Patient Information) showing details for a patient named 'А': 'Ім'я: А', 'Фамілія: А', 'Дата народження: 1978-03-27', 'Стать: Жіноча' (Gender: Female), 'Паління: Так' (Smoking: Yes), and 'Холестерин: 8'. Below this information is a line graph with a y-axis from 0 to 5 and an x-axis labeled 'В' (Time).

Рис. 3.17 – Перегляд пацієнта

Після того як лікар вибрав пацієнта, він може додати йому якісь поради щодо здоров'я за обраний місяць (рис. 3.18)

Виберіть пацієнта

6467bd04565bf57273cf1470

ВИДАЛИТИ ПАЦІЄНТА

Місяць Рік

November 2023

Виберіть тип графіка

РЕКОМЕНДАЦІЯ

ВИЙТИ

Дата народження: 1978-03-27

Стать: _____

Підприємство: _____

Холестерин: _____

Рекомендації

Додати

Закрити

5

4

3

2

1

0

Рис. 3.18 – Рекомендації

Лікар може видаляти пацієнтів за необхідності (рис. 3.20), наприклад, якщо пацієнт завершив лікування. Також, він може додавати нових пацієнтів до системи (рис. 3.19).

Додати пацієнта

Введіть id пацієнта

ДОДАТИ

Виберіть пацієнта

ВИДАЛИТИ ПАЦІЄНТА

Місяць Рік

December 2023

Виберіть тип графіка

ВИЙТИ

Інформація про Пацієнта

Ім'я: _____

Фамілія: _____

Дата народження: _____

Стать: _____

Паління: _____

Холестерин: _____

5

4

3

2

1

0

Рис. 3.19 – Додавання пацієнтів

Додати пацієнта

Введіть ід пацієнта

ДОДАТИ

Виберіть пацієнта

6467bd04565bf57273cf1470

ВИДАЛИТИ ПАЦІЄНТА

МісяцьРік

December2023

Виберіть тип графіка

РЕКОМЕНДАЦІЯ

ВИЙТИ

Інформація про Пацієнта

Ім'я: А

Фамілія: А

Дата народження: 1978-03-27

Стать: Жіноча

Паління: Так

Холестерин: 8

5

4

3

2

1

0

Рис. 3.20 –Видалення пацієнтів

Лікар має можливість переглядати статистику здоров'я пацієнта за обраний період, вибираючи тип статистики за тиском (рис. 3.22) чи ризиком захворювання (рис. 3.21).

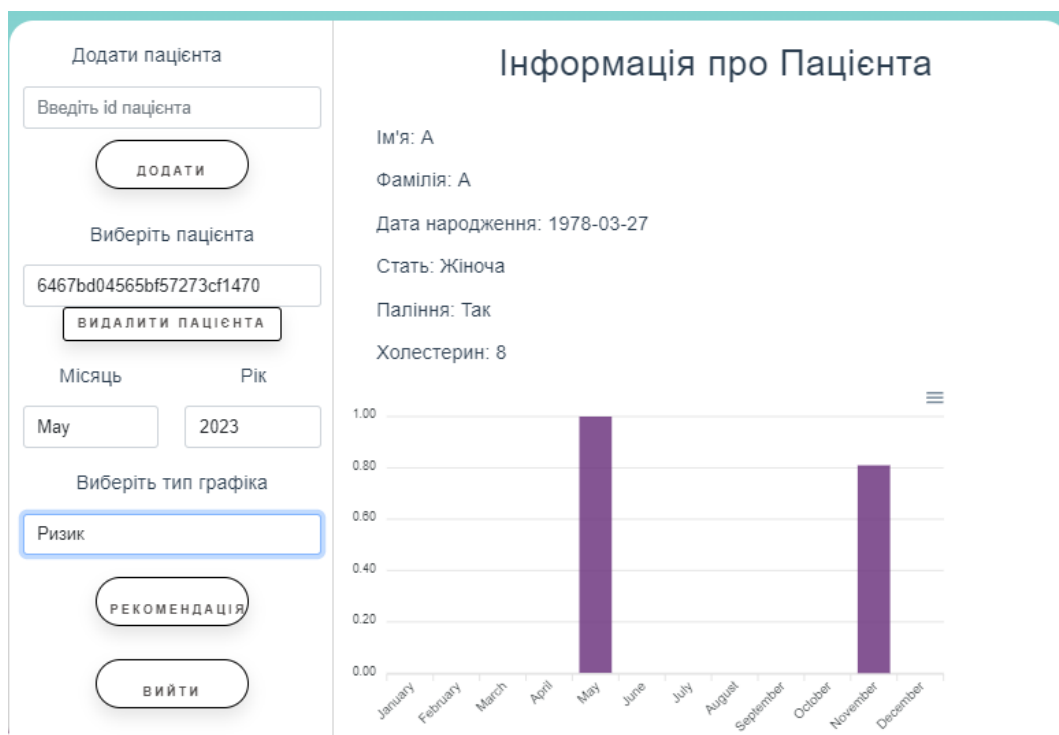


Рис. 3.21 –Статистика ризику

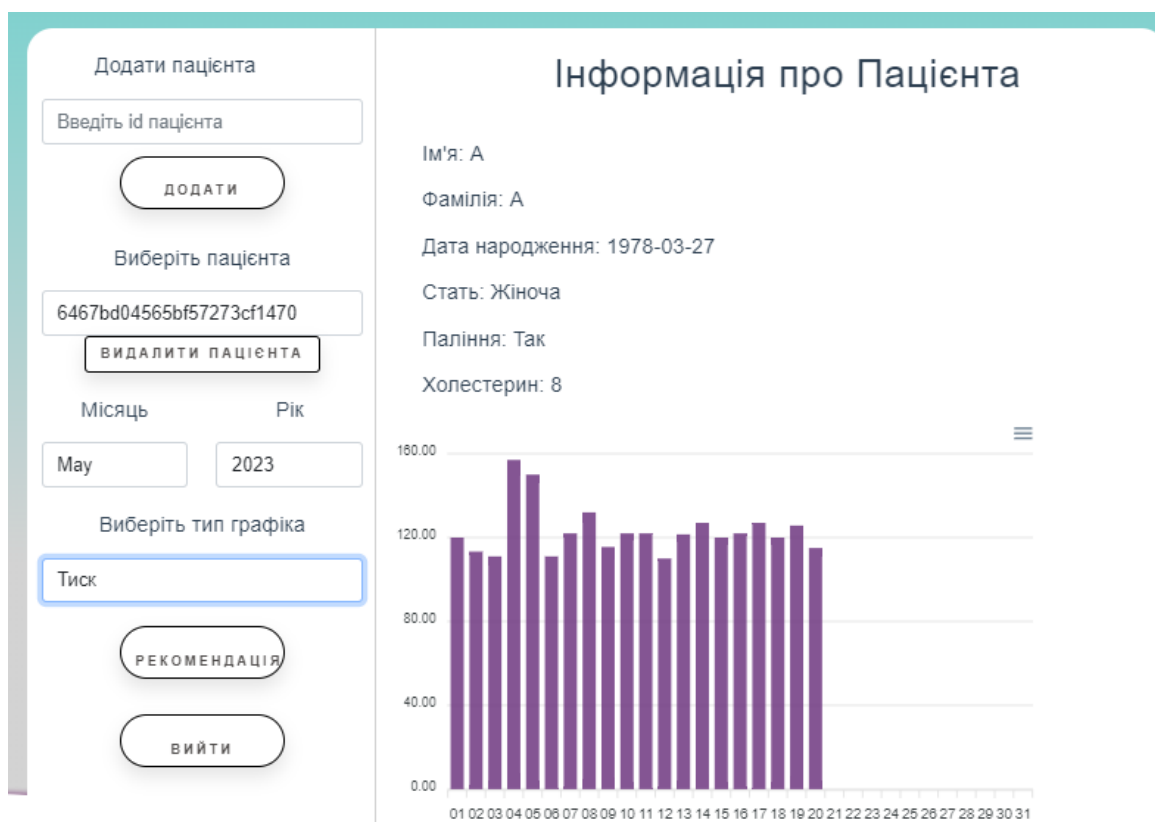


Рис. 3.22 –Статистика тиску

Адміністратор системи має власний інтерфейс, який починається з авторизації в системі (рис. 3.23).

The login interface is a simple form with a light blue background. It contains three input fields: 'Код доступу' (Access code), 'пароль' (password), and a 'ВХІД' (Login) button at the bottom.

Рис. 3.23 – Авторизація адміністратора

Коли адміністратор входить в систему, то може бачити перед собою опції представлені на рисунку 3.24

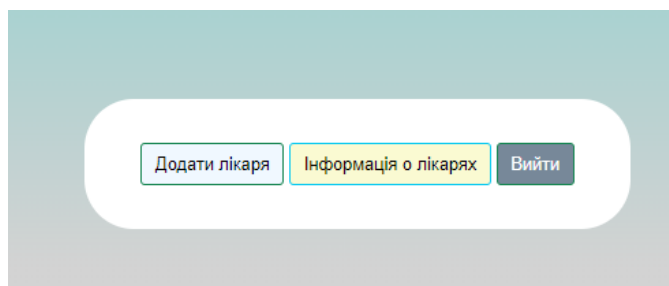


Рис. 3.24 – Головна сторінка адміністратора

Лише адміністратор може додавати лікарів до системи. Для цього він повинен заповнити форму представлену на рисунку 3.25

Рис. 3.25 – Додавання лікаря

Адміністратор може переглядати та перевіряти дані лікаря для забезпечення правильного функціонування системи. Також адміністратор має право видаляти лікарів із системи, наприклад, якщо лікар не має пацієнтів (рис. 3.26).

Рис. 3.26 – Дані лікаря

3.3 Тестування роботи веб-додатку

Тестування захищеності шляхів системи включає проведення ряду випробувань для забезпечення безпеки та коректної роботи системи.

Таблиця 3.1

Тестування форми логіну

Тест №	Очікуваний результат	Фактичний результат	Пройдено/не пройдено
1	Пусті поля не зчитуються	Пусті поля не зчитуються	Пройдено
2	Перевірка формату	Форма не відправляється, доки не виправлені помилки	Пройдено
3	Повідомлення про неправильно введені дані	Виводиться повідомлення про неправильно введені дані	Пройдено

Таблиця 3.2

Тестування форми реєстрації

Номер №	Очікуваний результат	Фактичний результат	Пройдено/не пройдено
1	Пусті поля не зчитуються	Пусті поля не зчитуються	Пройдено
2	Перевірка формату	Форма не відправляється, доки не виправлені помилки	Пройдено
3	Створення облікового запису з правильними даними	Обліковий запис створено успішно	Пройдено

Таблиця 3.3

Тестування перегляду статистики

Номер №	Очікуваний результат	Фактичний результат	Пройдено/не пройдено
1	Статистика відображається коректно	Статистика відображається коректно	Пройдено
2	Вибір періоду відображення статистики	Статистика відображається за обраним періодом	Пройдено

Таблиця 3.4

Тестування введення та оновлення медичних даних

Номер №	Очікуваний результат	Фактичний результат	Пройдено/не пройдено
1	Успішне внесення нових даних	Користувач може додавати та оновлювати свої медичні дані	Пройдено
2	Перевірка формату введення даних	Система не дозволяє вводити дані неправильного формату	Пройдено

Таблиця 3.5

Тестування можливостей лікаря

Номер №	Очікуваний результат	Фактичний результат	Пройдено/не пройдено
1	Успішна авторизація лікаря в системі	Лікар може увійти в систему за допомогою своїх облікових даних	Пройдено
2	Перегляд особистих даних пацієнтів	Лікар може обрати пацієнта та переглядати його особисті медичні дані	Пройдено
3	Перегляд статистики здоров'я пацієнта	Лікар може переглядати статистику здоров'я пацієнта за обраним періодом	Пройдено
4	Видалення пацієнта з системи	Лікар може видалити пацієнта за потреби	Пройдено
5	Додавання нового пацієнта	Лікар може додати нового пацієнта в систему	Пройдено
6	Виписка рекомендацій пацієнту	Лікар може виписати необхідні рекомендації пацієнту	Пройдено

Таблиця 3.6

Тестування можливості адміністратора

Номер №	Очікуваний результат	Фактичний результат	Пройдено/не пройдено
1	Успішна авторизація адміністратора в системі	Адміністратор може увійти в систему за допомогою своїх облікових даних	Пройдено
2	Додавання лікарів в систему	Адміністратор може додавати нових лікарів в систему	Пройдено
3	Видалення лікарів з системи	Адміністратор може видаляти лікарів з системи за потреби	Пройдено
4	Перегляд та перевірка даних лікарів	Адміністратор може переглядати та перевіряти дані лікарів	Пройдено

Таблиця 3.7

Тестування захищеності шляхів та функціоналу сайту

Номер №	Очікуваний результат	Фактичний результат	Пройдено/не пройдено
1	Відсутність доступу до конфіденційних даних без авторизації	Спроба отримання конфіденційних даних без авторизації не повинна бути успішною	Пройдено

2	Відсутність доступу до конфіденційних даних інших користувачів	Користувач не повинен мати доступ до даних інших користувачів	Пройдено
3	Перевірка валідації введених URL-параметрів	Система вірно обробляє та перевіряє URL-параметри	Пройдено

Висновки до третього розділу

У розділі 3 надано детальний опис варіантів використання для різних типів користувачів, а також описано інтерфейс та порядок роботи з системою. Зазначені ключові етапи тестування та перевірки функціоналу веб-додатку. Нижче представлені висновки з кожного підрозділу розділу 3:

3.1 Визначення варіантів використання системи

1. Користувач/пацієнт:

- Успішна реалізація авторизації та введення особистих медичних даних.
- Перегляд медичних даних та отримання персоналізованих рекомендацій.

2. Лікар:

- Ефективна авторизація лікаря та перегляд даних своїх пацієнтів.
- Можливість видалення та додавання пацієнтів, а також виписка рекомендацій.

3. Адміністратор:

- Успішна авторизація адміністратора та можливість додавання/видалення лікарів.

3.2 Інтерфейс та порядок роботи з системою

- Система володіє зручним інтерфейсом, який дозволяє користувачам легко авторизуватися та здійснювати реєстрацію. Інтуїтивно зрозуміла основна сторінка та простий у використанні календар забезпечують зручний перегляд і введення медичних даних. Графіки та статистика надають чітку інформацію про стан здоров'я та ризик захворювання.

3.3 Тестування роботи веб-додатку

1. Форми логіну та реєстрації:

- Успішно пройдені тести на валідацію та відображення повідомлень про помилки.

2. Перегляд статистики та введення медичних даних:

- Спроектовані та успішно протестовані зручні механізми введення та перегляду медичних даних.

3. Функціонал лікаря та адміністратора:

- Лікар та адміністратор мають зручні та ефективні інструменти для управління пацієнтами та лікарями відповідно.

4. Захищеність шляхів та функціоналу:

- Успішно пройдені тести на доступ до конфіденційних даних та перевірку введених URL-параметрів.

Загалом, розділ 3 демонструє вдале планування та розробку системи, а також її ефективність у реальних умовах використання. Описані функції та інтерфейс відповідають вимогам користувачів та вимогам безпеки.

ВИСНОВКИ

В ході виконання магістерської роботи «Аналіз алгоритмів SCORE та Framingham оцінки стану серцево-судинної системи людини» було проведено дослідження з метою виявлення ранніх змін у функціональному стані серцево-судинної системи та прогнозування ризику розвитку серцево-судинних захворювань. Робота складалася з трьох розділів, які включали в себе аналіз проблематики та визначення методів її вирішення, практичну реалізацію алгоритмів, а також розробку та тестування веб-додатку.

У розділі 1 було визначено технічне завдання для розробки програмного продукту, здійснено аналіз аналогів, визначено методику визначення оцінки та вибрано засоби реалізації програмного продукту. В результаті було сформульовано завдання для подальшого порівняння алгоритмів SCORE та Framingham.

Розділ 2 присвячено практичній реалізації алгоритмів та аналізу отриманих результатів. Проведено порівняльний аналіз ефективності алгоритмів за допомогою ROC-кривих, визначено точність прогнозування за допомогою метрик MSE та MAE, а також проведено аналіз ризиків в залежності від ознак.

У розділі 3 детально описано варіанти використання системи для різних типів користувачів, представлено інтерфейс та порядок роботи з веб-додатком, а також проведено тестування його функціоналу.

Загальною метою роботи було визначення найбільш ефективного алгоритму для визначення ризику серцево-судинних захворювань. На основі результатів дослідження встановлено, що алгоритм SCORE виявився більш точним та ефективним порівняно з алгоритмом Framingham. Веб-додаток, розроблений для реалізації алгоритму SCORE, відповідає всім вимогам та успішно пройшов тестування.

Отже, отримані результати дозволяють зробити висновок про доцільність використання алгоритму SCORE для виявлення ризику розвитку серцево-

судинних захворювань, а розроблений веб-додаток надає зручний та безпечний інструмент для використання цього алгоритму користувачами різних категорій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шкала SCORE [Електронний ресурс]. – Режим доступу: <https://play.google.com/store/apps/details?id=ua.medsoftpro.medcalc&hl=uk&gl=US>
2. Evaluation of cardiovascular risk profile: a comparative analysis between CUORE algorithm and the Framingham risk scores [Електронний ресурс]. – Режим доступу: <https://pubmed.ncbi.nlm.nih.gov/29762858/>
3. Артеріальний тиск людини – норма за віком, таблиця [Електронний ресурс]. – Режим доступу: <https://diagnoz.in.ua/kardiologiya/arterialniy-tisk-lyudini-norma-za-vik/>
4. A comparison of Framingham and SCORE-based cardiovascular risk estimates in participants of the German National Health Interview and Examination Survey 1998 [Електронний ресурс]. – Режим доступу: <https://pubmed.ncbi.nlm.nih.gov/16210930/>
5. Артеріальна гіпертензія [Електронний ресурс]. – Режим доступу: <https://phc.org.ua/kontrol-zakhvoryuvan/neinfekciyni-zakhvoryuvannya/sercevo-sudinni-zakhvoryuvannya/arterialna-gipertenziya>
6. Efficient Data-Driven Machine Learning Models for Cardiovascular Diseases Risk Prediction [Електронний ресурс]. – Режим доступу: <https://pubmed.ncbi.nlm.nih.gov/36772201/>
7. Таблиця SCORE [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/figure/SCORE-chart-10-year-risk-of-fatal-cardiovascular-disease-in-populations-of-countries-at_fig3_303515610
8. Оцінювання стану серцево-судинної системи людини на основі аналізу добового серцевого ритму [Електронний ресурс] – Режим доступу: http://elartu.tntu.edu.ua/bitstream/lib/27040/2/VII_MNTK_2018v2_Khvesik_V_M-Evaluation_of_the_human_cardiovascular_187.pdf
9. Framingham Calc [Електронний ресурс]. – Режим доступу: <https://play.google.com/store/apps/details?id=com.danproductions.medical.cardiocalc&hl=uk&gl=US>

10. Побудова алгоритмів кількісної оцінки стану серцево-судинної системи під час масових обстежень населення [Електронний ресурс]. – Режим доступу: <https://www.dissercat.com/content/postroenie-algoritmov-kolichestvennoi-otsenki-sostoyaniya-serdechno-sosudistoi-sistemy-pri-m>

11. Рекомендації щодо контролю ліпідів [Електронний ресурс]. – Режим доступу: <https://emedicine.medscape.com/article/2500032-overview#a1>

12. The Framingham Heart Study and the epidemiology of cardiovascular disease: a historical perspective [Електронний ресурс]. – Режим доступу: <https://www.sciencedirect.com/science/article/pii/S0140673613617523?via%3Dihub>

13. Comparison of the Framingham risk and SCORE models in predicting the presence and severity of coronary artery disease considering SYNTAX score [Електронний ресурс]. – Режим доступу: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5331373/>

14. Інформативність шкал-калькуляторів оцінки 10-річного ризику серцево-судинних захворювань у жінок, хворих на ревматоїдний артрит [Електронний ресурс]. – Режим доступу: <https://www.rheumatology.kiev.ua/article/10803/informativnist-shkal-kalkulyatoriv-ocinki-10-richnogo-riziku-sercevo-sudinnix-zaxvoryuvan-u-zhinok-xvorix-na-revmatoidnij-artrit>

15. Comparison of the Framingham Risk Score, SCORE and WHO/ISH cardiovascular risk prediction models in an Asian population [Електронний ресурс]. – Режим доступу: <https://www.sciencedirect.com/science/article/pii/S0167527314012583>

16. Systematic Coronary Risk Evaluation (SCORE): JACC Focus Seminar 4/8 [Електронний ресурс]. – Режим доступу: <https://www.sciencedirect.com/science/article/pii/S0735109721048233>

17. Framingham Heart Study: JACC Focus Seminar, 1/8 [Електронний ресурс]. – Режим доступу: <https://www.sciencedirect.com/science/article/pii/S0735109721010962>

18. Landmark studies in coronary heart disease epidemiology. The Framingham Heart Study after 70 years and the Seven Countries Study after 60 years [Электронный ресурс]. – Режим доступа: <https://pubmed.ncbi.nlm.nih.gov/30828782/>

19. Before the heart attack [Электронный ресурс]. – Режим доступа: <https://www.nature.com/articles/s41591-022-01685-7>

20. Perceived Risk of Heart Attack: A Function of Gender? [Электронный ресурс]. – Режим доступа: <https://onlinelibrary.wiley.com/doi/10.1111/j.0029-6473.2004.00018.x>

21. Monitoring Cardiovascular Problems in Heart Patients Using Machine Learning [Электронный ресурс]. – Режим доступа: <https://www.hindawi.com/journals/jhe/2023/9738123/>

22. Cardiovascular diseases prediction by machine learning incorporation with deep learning [Электронный ресурс]. – Режим доступа: <https://www.frontiersin.org/articles/10.3389/fmed.2023.1150933/full>

23. Machine learning and atherosclerotic cardiovascular disease risk prediction in a multi-ethnic population [Электронный ресурс]. – Режим доступа: <https://www.nature.com/articles/s41746-020-00331-1>

24. Artificial Intelligence and Cardiovascular Risk Prediction: All That Glitters is not Gold [Электронный ресурс]. – Режим доступа: <https://www.ecrjournal.com/articles/artificial-intelligence-and-cardiovascular-risk-prediction-all-glitters-not-gold>

25. 2016 European Guidelines on cardiovascular disease prevention in clinical practice: The Sixth Joint Task Force of the European Society of Cardiology and Other Societies on Cardiovascular Disease Prevention in Clinical Practice (constituted by representatives of 10 societies and by invited experts) Developed with the special contribution of the European Association for Cardiovascular Prevention & Rehabilitation (EACPR) [Электронный ресурс]. – Режим доступа: <https://academic.oup.com/eurheartj/article/37/29/2315/1748952>

26. Validity of cardiovascular risk prediction models in Latin America and among Hispanics in the United States of America: a systematic review [Електронний ресурс]. – Режим доступу: <https://pubmed.ncbi.nlm.nih.gov/23099874/>

27. Cardiovascular disease risk prediction using automated machine learning: A prospective study of 423,604 UK Biobank participants [Електронний ресурс]. – Режим доступу: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0213653>

28. Risk prediction of cardiovascular disease using machine learning classifiers[Електронний ресурс]. – Режим доступу: <https://www.degruyter.com/document/doi/10.1515/med-2022-0508/html>

29. Легенчук С.Ф., Ковбасюк С.В., Моргун А.М. Інформаційна технологія діагностування стану серцево-судинної системи людини. Науковий журнал Вісник Хмельницького національного університету Серія: «Технічні науки», № 5 т. 2, 2023. – С. 60-65.

ДОДАТКИ

```

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, roc_auc_score
import math
# Зчитування датасету
data = pd.read_csv('/content/cardio_train.csv', delimiter=';')
column_names = data.columns
print(column_names)
data = data.dropna()
data = data[(data['ap_hi'] >= 100) & (data['ap_hi'] < 200)]
data = data[(data['gluc'] >= 1) & (data['gluc'] <= 10)]
data = data[(data['gender'] >= 1) & (data['gender'] <= 2)]
data = data[(data['smoke'] >= 0) & (data['smoke'] <= 1)]
data = data[(data['smoke'] >= 0) & (data['smoke'] <= 1)]
data = data[(data['alco'] >= 0) & (data['alco'] <= 1)]
X = data[['age', 'gender', 'smoke', 'ap_hi', 'cholesterol']]
y = data['cardio']

# Розділення на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Функція для алгоритму Framingham
def calculate_simple_framingham_risk(days, gender, smoking, blood_pressure, cholesterol):
    days_in_year = 365.25
    age = round(days / days_in_year)
    female_coefs = [-7, -3, 0, 3, 6, 8, 10, 12, 14, 16]
    male_coefs = [-9, -4, 0, 3, 6, 8, 10, 11, 12, 13]
    # Коефіцієнти для total cholesterol
    total_cholesterol_coefs = [
        [0, 4, 8, 11, 13],
        [0, 3, 6, 8, 10],
        [0, 2, 4, 5, 7],
        [0, 1, 2, 3, 4],
        [0, 1, 1, 2, 2],
        [0, 4, 7, 9, 11],
        [0, 3, 3, 6, 8],
        [0, 2, 5, 4, 5],
        [0, 1, 1, 2, 3],
        [0, 0, 0, 1, 1]
    ]

    # Коефіцієнти для куріння
    smoking_coefsF = [9, 7, 4, 2, 1]
    smoking_coefsM = [8, 5, 3, 1, 1]
    # Коефіцієнти для systolic blood pressure
    female_coefs_sbp_coefs = [0, 1, 2, 3, 4]
    male_sbp_coefs = [0, 0, 1, 1, 2]
    smoke_points=0
    sbp_points=0
    chol_points=0
    age_points=0
    risk_percentage=0
    # Визначення віку
    if gender == 1: # Жінка
        age_coefs = female_coefs
        if 20 < age <= 34:
            age_points=age_coefs[0]
        elif 35 <= age <= 39:
            age_points=age_coefs[1]
        elif 40 <= age <= 44:
            age_points=age_coefs[2]
        elif 45 <= age <= 49:
            age_points=age_coefs[3]
        elif 50 <= age <= 54:
            age_points=age_coefs[4]

```

```

elif 55<=age <= 59:
    age_points=age_coeffs[5]
elif 60<=age <= 64:
    age_points=age_coeffs[6]
elif 65<=age <= 69:
    age_points=age_coeffs[7]
elif 70<=age <= 74:
    age_points=age_coeffs[8]
elif 75<=age <= 79:
    age_points=age_coeffs[9]
    # Визначення балів за total cholesterol
if age < 40:
    cholesterol_coeffs = total_cholesterol_coeffs[0]
    if smoking==1:
        smoke_points=smoking_coeffsF[0]
elif 40 <= age < 50:
    cholesterol_coeffs = total_cholesterol_coeffs[1]
    if smoking==1:
        smoke_points=smoking_coeffsF[1]
elif 50 <= age < 60:
    cholesterol_coeffs = total_cholesterol_coeffs[2]
    if smoking==1:
        smoke_points=smoking_coeffsF[2]
elif 60 <= age < 70 :
    cholesterol_coeffs = total_cholesterol_coeffs[3]
    if smoking==1:
        smoke_points=smoking_coeffsF[3]
if cholesterol< 4:
    chol_points = cholesterol_coeffs[0]
elif 4<cholesterol<=5:
    chol_points = cholesterol_coeffs[1]
elif 5<cholesterol<=6:
    chol_points = cholesterol_coeffs[2]
elif 6<cholesterol<=7:
    chol_points = cholesterol_coeffs[3]
if blood_pressure < 120:
    sbp_points = female_coeffs_sbp_coeffs[0]
elif 120<= blood_pressure <= 129:
    sbp_points = female_coeffs_sbp_coeffs[1]
elif 130<= blood_pressure <= 139:
    sbp_points = female_coeffs_sbp_coeffs[2]
elif 140<= blood_pressure <= 159:
    sbp_points = female_coeffs_sbp_coeffs[3]
elif blood_pressure >160:
    sbp_points = female_coeffs_sbp_coeffs[4]
else: # Чоловік
    age_coeffs = male_coeffs
    if 20<age <= 34:
        age_points=age_coeffs[0]
    elif 35<=age <= 39:
        age_points=age_coeffs[1]
    elif 40<=age <= 44:
        age_points=age_coeffs[2]
    elif 45<=age <= 49:
        age_points=age_coeffs[3]
    elif 50<=age <= 54:
        age_points=age_coeffs[4]
    elif 55<=age <= 59:
        age_points=age_coeffs[5]
    elif 60<=age <= 64:
        age_points=age_coeffs[6]
    elif 65<=age <= 69:
        age_points=age_coeffs[7]
    elif 70<=age <= 74:
        age_points=age_coeffs[8]
    elif 75<=age <= 79:
        age_points=age_coeffs[9]
    if age < 40:
        cholesterol_coeffs = total_cholesterol_coeffs[4]

```

```

        if smoking==1:
            smoke_points=smoking_coeffsM[0]
    elif 40 <= age < 50:
        cholesterol_coeffs = total_cholesterol_coeffs[5]
        if smoking==1:
            smoke_points=smoking_coeffsM[1]
    elif 50 <= age < 60:
        cholesterol_coeffs = total_cholesterol_coeffs[6]
        if smoking==1:
            smoke_points=smoking_coeffsM[2]
    elif 60 <= age < 70 :
        cholesterol_coeffs = total_cholesterol_coeffs[7]
        if smoking==1:
            smoke_points=smoking_coeffsM[3]
    if cholesterol< 4:
        chol_points = cholesterol_coeffs[0]
    elif 4<cholesterol<=5:
        chol_points = cholesterol_coeffs[1]
    elif 5<cholesterol<=6:
        chol_points = cholesterol_coeffs[2]
    elif 6<cholesterol<=7:
        chol_points = cholesterol_coeffs[3]
    if blood_pressure < 120:
        sbp_points = male_sbp_coeffs[0]
    elif 120<= blood_pressure <= 129:
        sbp_points = male_sbp_coeffs[1]
    elif 130<= blood_pressure <= 139:
        sbp_points = male_sbp_coeffs[2]
    elif 140<= blood_pressure <= 159:
        sbp_points = male_sbp_coeffs[3]
    elif blood_pressure >160:
        sbp_points = male_sbp_coeffs[4]
total_points = age_points + chol_points + smoke_points + sbp_points
if gender == 1: # Жінка
    if total_points<9:
        risk_percentage=1
    elif 9<=total_points<=12:
        risk_percentage=1
    elif 13<=total_points<=14:
        risk_percentage=2
    elif total_points==15:
        risk_percentage=3
    elif total_points==16:
        risk_percentage=4
    elif total_points==17:
        risk_percentage=5
    elif total_points==18:
        risk_percentage=6
    elif total_points==19:
        risk_percentage=8
    elif total_points==20:
        risk_percentage=11
    elif total_points==21:
        risk_percentage=14
    elif total_points==22:
        risk_percentage=17
    elif total_points==23:
        risk_percentage=22
    elif total_points==24:
        risk_percentage=27
    elif 25<=total_points:
        risk_percentage=30
    # risk_percentage = min(30, max(1, total_points))
else: # Чоловік
    # risk_percentage = min(30, max(1, total_points - 10))
    if total_points==0:
        risk_percentage=0
    elif 1<=total_points<=4:
        risk_percentage=1

```

```

        elif 5<=total_points<=6:
            risk_percentage=2
        elif total_points==7:
            risk_percentage=3
        elif total_points==8:
            risk_percentage=4
        elif total_points==9:
            risk_percentage=5
        elif total_points==10:
            risk_percentage=6
        elif total_points==11:
            risk_percentage=8
        elif total_points==12:
            risk_percentage=10
        elif total_points==13:
            risk_percentage=12
        elif total_points==14:
            risk_percentage=16
        elif total_points==15:
            risk_percentage=20
        elif total_points==16:
            risk_percentage=25
        elif 17<=total_points:
            risk_percentage=30

    return risk_percentage

# Функція для алгоритму SCORE
def calculate_score_risk(days, gender, smoking, blood_pressure, cholesterol):
    # Коефіцієнти SCORE
    gen, age, ad, smoke, chol=0,0,0,0,0
    s0,s010,alpha,p,cs0,cs10,ncs0,ncs10,wc,wnc=0,0,0,0,0,0,0,0,0,0
    bchol,bsbp,bsm,cs,cs1,ncs,ncs1,r,r1=0,0,0,0,0,0,0,0,0,0
    age=round(days/366)
    gen=gender
    chol=cholesterol
    if gen==2:
        alpha=-21.0
        p=4.62;
    else:
        alpha=-28.7
        p=6.23;

    cs0=math.exp(-math.exp(alpha)*math.pow(age-20.0,p))
    cs10=math.exp(-math.exp(alpha)*math.pow(age-10.0,p))

    if gen==2:
        alpha=-25.7
        p=5.47
    else:
        alpha=-30.0
        p=6.42
    ad=blood_pressure
    ncs0=math.exp(-math.exp(alpha)*math.pow(age-20.0,p))
    ncs10=math.exp(-math.exp(alpha)*math.pow(age-10.0,p));
    bchol=0.24;
    bsbp=0.018;
    smoke=smoking;
    bsm=smoke*0.71;
    wc=bchol*(chol-6.0)+bsbp*(ad-120.0)+bsm;
    bchol=0.02;
    bsbp=0.022;
    bsm=smoke*0.63;
    wnc=bchol*(chol-6.0)+bsbp*(ad-120.0)+bsm
    cs=math.pow(cs0,math.exp(wc))
    cs1=math.pow(cs10,math.exp(wc))
    ncs=math.pow(ncs0,math.exp(wnc));
    ncs1=math.pow(ncs10,math.exp(wnc))
    if cs != 0:

```

```

        cs1=cs1/cs
    else:
        cs1 = 0

    if ncs != 0:
        ncs1 = ncs1 / ncs
    else:
        ncs1 = 0
    r=1.0-cs1;
    r1=1.0-ncs1;
    score = 100.0 * (r + r1)
    return score

# return score

# Обчислення ризиків за обома алгоритмами
framingham_risks = X_test.apply(lambda row: calculate_simple_framingham_risk(*row),
axis=1)
score_risks = X_test.apply(lambda row: calculate_score_risk(*row), axis=1)

# Визначення порогу для класифікації
threshold = 0.5

# Класифікація за порогом
framingham_predictions = (framingham_risks > threshold).astype(int)
score_predictions = (score_risks > threshold).astype(int)

# Оцінка ефективності
framingham_accuracy = accuracy_score(y_test, framingham_predictions)
score_accuracy = accuracy_score(y_test, score_predictions)

framingham_auc = roc_auc_score(y_test, framingham_risks)
# print(y_test)
# score_risks = score_risks.fillna(score_risks.mean())
print(np.isnan(score_risks).any())
# score_risks = score_risks.fillna(score_risks.mean())
score_auc = roc_auc_score(y_test, score_risks)
from sklearn.metrics import mean_squared_error, mean_absolute_error

# Оцінка ефективності для регресії
mse_framingham = mean_squared_error(y_test, framingham_risks)
mae_framingham = mean_absolute_error(y_test, framingham_risks)

mse_score = mean_squared_error(y_test, score_risks)
mae_score = mean_absolute_error(y_test, score_risks)

# Виведення результатів
print("Framingham Accuracy:", framingham_accuracy*100)
print("Framingham AUC-ROC:", framingham_auc*100)
print("\nSCORE Accuracy:", score_accuracy*100)
print("SCORE AUC-ROC:", score_auc*100)
print("Framingham MSE:", mse_framingham*100)
print("Framingham MAE:", mae_framingham*100)
print("\nSCORE MSE:", mse_score*100)
print("SCORE MAE:", mae_score*100)
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc, confusion_matrix

# Отримання значень FPR, TPR та порогів для ROC-кривих
fpr_framingham, tpr_framingham, _ = roc_curve(y_test, framingham_risks.values)
fpr_score, tpr_score, _ = roc_curve(y_test, score_risks.values)

# Обчислення площі під кривими
roc_auc_framingham = auc(fpr_framingham, tpr_framingham)
roc_auc_score = auc(fpr_score, tpr_score)

# Побудова ROC-кривих
plt.figure(figsize=(8, 8))

```

```

plt.plot(fpr_framingham, tpr_framingham, color='darkorange', lw=2, label=f'Framingham ROC curve (area = {roc_auc_framingham:.2f})')
plt.plot(fpr_score, tpr_score, color='darkgreen', lw=2, label=f'SCORE ROC curve (area = {roc_auc_score:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Характеристика роботи приймача (ROC) крива')
plt.legend(loc='lower right')
plt.show()

import matplotlib.pyplot as plt

# Дані для графіків
algorithms = ['Framingham', 'SCORE']
mse_values = [mse_framingham, mse_score]
mae_values = [mae_framingham, mae_score]

# Графік MSE
plt.figure(figsize=(10, 6))
plt.bar(algorithms, mse_values, color=['blue', 'green'])
plt.xlabel('Алгоритм')
plt.ylabel('MSE (Mean Squared Error)')
plt.title('Порівняння MSE між алгоритмами')
plt.show()

# Графік MAE
plt.figure(figsize=(10, 6))
plt.bar(algorithms, mae_values, color=['blue', 'green'])
plt.xlabel('Алгоритм')
plt.ylabel('MAE (Mean Absolute Error)')
plt.title('Порівняння MAE між алгоритмами')
plt.show()
import matplotlib.pyplot as plt

# Перетворення age в роки
X_test['age_years'] = round(X_test['age']) // 365.25

# Функція для побудови графіка ризику відносно кожної ознаки
def plot_risk_vs_feature(algorithm1_risks, algorithm2_risks, feature_name, algorithm1_name, algorithm2_name):
    plt.scatter(X_test[feature_name], algorithm1_risks, alpha=0.5, label=f'{algorithm1_name} {feature_name}')
    plt.scatter(X_test[feature_name], algorithm2_risks, alpha=0.5, label=f'{algorithm2_name} {feature_name}')
    plt.title(f'{algorithm1_name} vs {algorithm2_name} - Risk vs {feature_name}')
    plt.xlabel(feature_name)
    plt.ylabel('Risk')
    plt.legend()
    plt.show()

# Побудова графіків для обох алгоритмів для кожної ознаки
plot_risk_vs_feature(framingham_risks, score_risks, 'age_years', 'Framingham', 'SCORE')

plot_risk_vs_feature(framingham_risks, score_risks, 'gender', 'Framingham', 'SCORE')

plot_risk_vs_feature(framingham_risks, score_risks, 'smoke', 'Framingham', 'SCORE')

plot_risk_vs_feature(framingham_risks, score_risks, 'ap_hi', 'Framingham', 'SCORE')

plot_risk_vs_feature(framingham_risks, score_risks, 'cholesterol', 'Framingham', 'SCORE')

# Видалення додаткової колонки
X_test = X_test.drop('age_years', axis=1)
# *****норма тиску*****
import pandas as pd
from sklearn.model_selection import train_test_split
import joblib

```

```

# Завантажити набір даних
df = pd.read_csv('/content/cardio.csv')
df = df[df['TenYearCHD'] == 0]
df = df.sort_values(by='age')
# вхідні змінні: вік та стать
X = df[['age', 'male']]
# Вибрати вихідну змінну
y = df['sysBP']
# Розділити дані на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Об'єднати X_train та y_train в один DataFrame
train_data = pd.concat([X_train, y_train], axis=1)

# Групувати дані за віком та статтю та обчислити моду тиску для кожної групи
pressure_mode_by_group = train_data.groupby(['age', 'male'])['sysBP'].agg(lambda x:
x.value_counts().index[0]).reset_index()

joblib.dump(pressure_mode_by_group, 'pressure_model.pkl')

loaded_model = joblib.load('pressure_model.pkl')
# Заповнити відсутні значення середнім значенням тиску

pressure_mean = loaded_model.groupby(['age', 'male'])['sysBP'].mean().reset_index()
# Визначити норму тиску для кожного прогнозу
norms = []
for i in range(len(X_test)):
    age = X_test.iloc[i]['age']
    male = X_test.iloc[i]['male']
    pressure_mean_value = pressure_mean[(pressure_mean['age'] == age) &
(pressure_mean['male'] == male)]['sysBP']
    if pressure_mean_value.empty:
        # Якщо середнє значення не знайдено, використовуємо значення з найближчого
        # сусіднього віку та статі
        closest_age = pressure_mean['age'].iloc[(pressure_mean['age']-
age).abs().argsort()[2]].values
        closest_male = pressure_mean['male'].iloc[(pressure_mean['age']-
age).abs().argsort()[2]].values
        closest_pressure = pressure_mean[(pressure_mean['age'].isin(closest_age)) &
(pressure_mean['male'].isin(closest_male))]['sysBP'].values
        pressure_mean_value = closest_pressure.mean()
    else:
        pressure_mean_value = pressure_mean_value.values[0]
    norms.append(pressure_mean_value)
# поточний стан тиску
for i in range(len(norms)):
    print(f'Для віку {X_test.iloc[i]["age"]} та статі {X_test.iloc[i]["male"]} - Норма
АД: {norms[i]}')

# Оцінка продуктивності моделі
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test, norms)
print(f'Mean Squared Error: {mse}')
# Оцінка тиску
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, f1_score

# Завантаження даних з файлу
data = pd.read_csv('/content/MOCK_DATA.csv')
print('Кількість записів:', data.shape[0], '\nКількість ознак:', data.shape[1])
print(data.isnull().sum())
# Розділення даних на ознаки та мітки
X = data['sysBP'].values.reshape(-1, 1)
y = data['mark']

# Підготовка даних для навчання

```

```

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Масштабування ознак
from sklearn.preprocessing import StandardScaler

# Вибір та навчання моделі
model = LogisticRegression()
model.fit(X_train, y_train)
from joblib import dump
# Прогноз міток на тестовому наборі
y_pred = model.predict(X_test)
# Оцінка продуктивності моделі
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred, average='weighted')

print(f'Accuracy: {accuracy * 100}%')
print(f'F1 Score: {f1}')
# Використання навченої моделі для прогнозу на нових даних
new_data = pd.DataFrame({'sysBP': [152, 140, 134]})
new_pred = model.predict(new_data)

print(f'New predictions: {new_pred}')
dump(model, 'model4.h5')

```

Головна сторінка адміністратора

```

<template>
  <b-container class="mt-3" style="background-color: white; border-radius: 45px">
    <b-row>
      <b-col md="12" style="padding: 40px">
        <b-button variant="success" @click="showAddForm" style="background-color:
        aliceblue; color: black">Додати лікаря</b-button>
        <b-button variant="info" @click="showSelectForm" style="background-color:
        lightgoldenrodyellow">Інформація о лікарях</b-button>
        <b-button variant="success" @click="Logout()" style="background-color:
        lightslategrey">Вийти</b-button>
      </b-col>
      <b-col md="12">
        <div v-if="showAdd" style="position:center">
          <div style="position: center; margin:30px;">
            <h2 class="mb-3">Додавання лікаря:</h2>
            <b-input v-model="name" placeholder="Ім'я"></b-input>
            <b-input v-model="surname" placeholder="Фамілія"></b-input>
            <b-input v-model="key" placeholder="Код доступу"></b-input>
            <b-input v-model="password" placeholder="Пароль"></b-input>
            <b-button variant="success" style="background-color: mediumaquamarine"
            @click="Add">Додати</b-button>
          </div>
        </div>

        <div v-if="showSelect">
          <div>
            <label for="selectedPacient" class="form-label">Оберіть лікаря</label>
            <b-select v-model="selectedPacient" class="form-select" >
              <option v-for="option in doctor" :key="option._id"
              :value="option._id">{{ option._id }}</option>
            </b-select>
            <b-button variant="success" @click="removeSelectedPacient"
            style="background-color: lightcoral">Видалити</b-button>
          </div>
          <div v-if="selectedPacient">
            <h2 class="mb-3">Інформація про лікаря:</h2>
            <div style="margin-left: 300px">
              <p style="text-align: left"><strong>Ім'я:</strong> {{
              selectedDoctorObject.name }}</p>
              <p style="text-align: left"><strong>Прізвище:</strong> {{
              selectedDoctorObject.surname }}</p>
              <p style="text-align: left"><strong>Код доступу:</strong> {{
              selectedDoctorObject.key }}</p>
            </div>
          </div>
        </div>
      </b-col>
    </b-row>
  </b-container>
</template>

<script>
import axios from 'axios'
import ProfilService from '../services/ProfilService'
// eslint-disable-next-line import/no-duplicates
import News from '../services/New'
// eslint-disable-next-line import/no-duplicates
import TaskS from '../services/New'
axios.defaults.headers.common['Authorization'] = localStorage.getItem('jwtToken')

export default {
  name: 'AdminMain',
  data () {
    return {

```

```

        showAdd: false,
        showSelect: false,
        selectedPacient: null,
        doctor: [],
        id: '',
        name: '',
        surname: '',
        key: '',
        password: ''
    }
},
mounted () {
    this.getMe()
},
computed: {
    selectedDoctorObject () {
        return this.doctor.find(doctor => doctor._id === this.selectedPacient) || {}
    }
},
methods: {
    showAddForm () {
        this.showAdd = true
        this.showSelect = false
    },
    showSelectForm () {
        this.showAdd = false
        this.showSelect = true
    },
    async removeSelectedPacient () {
        await News.deleteDoctor({
            id: this.selectedPacient
        })

        this.selectedPacient = null
        const response = await TaskS.fetchDoctors()
        this.doctor = response.data.doctor
    },
    async getMe () {
        this.doctor = ''
        axios.defaults.headers.common['Authorization'] = localStorage.getItem('jwtToken')
        // eslint-disable-next-line no-unused-expressions
        localStorage.id
        const response = await TaskS.fetchDoctors()
        this.doctor = response.data.doctor
    },
    async Add () {
        axios.defaults.headers.common['Authorization'] = localStorage.getItem('jwtToken')
        await News.addDoctor({
            name: this.name,
            surname: this.surname,
            key: this.key,
            password: this.password
        })
        const response = await TaskS.fetchDoctors()
        this.doctor = response.data.doctor
    },
    async Logout () {
        await ProfilService.LogOutAdmin()
        localStorage.clear()
        // storage.clear();
        await this.$router.push({name: 'AdminLog'})
    }
}
}
</script>

<style scoped>

</style>

```

Головна сторінка лікаря

```

<template>
  <b-container style="background-color: white; border-radius: 22px; margin-top: -20px">
    <div>
      <div style="float: left; width: 30%; padding-right: 10px">
        <p class="col-md-10" style="margin-top: 20px">Додати пацієнта</p>
        <Rec v-if="show"
          @close="show = false"
          @done="addRec"/>
        <b-input class="mb-6" v-model="pacient" placeholder="Введіть id пацієнта"></b-
input>
        <b-button style="position: center" variant="success" @click="Add"
class="floating-button">Додати</b-button>
        <div class="col-md-12">
          <p style="margin-top: 20px">Виберіть пацієнта</p>
          <select v-model="selectedPacient" class="form-control">
            <option v-for="option in IDpacient" :key="option">{{ option }}</option>
          </select>
          <b-button variant="success" @click="removeSelectedPacient" class="delete-
button">Видалити пацієнта</b-button>
        </div>
        <div class="row">
          <div class="col-md-6">
            <p style="margin-top: 20px">Місяць</p>
            <select v-model="selectedMonth" class="form-control">
              <option v-for="month in months" :key="month" :value="month">{{ month
}}</option>
            </select>
          </div>
          <div class="col-md-6">
            <p style="margin-top: 20px">Рік</p>
            <select v-model="selectedYear" class="form-control">
              <option v-for="year in years" :key="year" :value="year">{{ year
}}</option>
            </select>
          </div>
        </div>
        <div class="col-md-12">
          <p style="margin-top: 20px">Виберіть тип графіка</p>
          <select v-model="selectedVariant" class="form-control">
            <option v-for="option in list1" :key="option.id" v-text="option.Title"
:value="option.id"></option>
          </select>
        </div>
        <b-button v-if="selectedPacient" style="margin-top: 20px" variant="success"
@click="Recomend" class="floating-button">Рекомендація</b-button>
        <b-button style="margin-top: 20px" variant="success" @click="LogOut"
class="floating-button">Вийти</b-button>
      </div>
      <div style="float: right; width: 70%; border-left: 2px lightgray solid;" >
        <div style="text-align: left; margin-left: 39px">
          <h2 class="mb-3" style="text-align: center; margin-top: 20px">Інформація про
Пацієнта</h2>
          <p v-if="selectedDoctorObject">{{ selectedDoctorObject.name }}</p>
          <p v-if="selectedVariantObject">{{ selectedVariantObject.name }}</p>
          <p v-if="this.info!=null"><strong>Ім'я:</strong> {{
this.info.data.name}}</p>
          <p v-if="this.info!=null"><strong>Фамілія:</strong> {{
this.info.data.surname}}</p>
          <p v-if="this.info!=null"><strong>Дата народження:</strong> {{
this.info.data.vik}}</p>
          <p v-if="this.info!=null"><strong>Стать:</strong> {{ this.sex}}</p>
          <p v-if="this.info!=null"><strong>Паління:</strong> {{
this.zvyh}}</p>
          <p v-if="this.info!=null"><strong>Холестерин:</strong> {{

```

```

this.info.data.hol}}</p>
</div>
<apexchart
  style="background-color: white; color: black; z-index: 9999999;"
  type="bar"
  :options="options"
  :series="series"
></apexchart>
</div>
<div style="clear: both;"></div>
</div>
</b-container>
</template>

<script>
import axios from 'axios'
import Rec from './Recomend'
import ApexCharts from 'apexcharts'
import ProfilService from '../services/ProfilService'
// eslint-disable-next-line import/no-duplicates
import News from '../services/New'
import moment from 'moment'
// eslint-disable-next-line import/no-duplicates
import TaskS from '../services/New'
axios.defaults.headers.common['Authorization'] = localStorage.getItem('jwtToken')
export default {
  name: 'DoctorMain',
  components: {
    Rec
  },
},
data () {
  return {
    list1: [ { id: 0, Title: 'Ризик' }, { id: 1, Title: 'Тиск' } ],
    graf: [],
    par: [],
    pacientData: [],
    date: '',
    arr1: [],
    arr2: [],
    title: [],
    adds: false,
    edit: false,
    color: '',
    rec: '',
    today: moment(),
    dateContext: moment(),
    selectedDate: moment(),
    days: ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'],
    options: {
      chart: {
        id: 'salesChart'
      },
      dataLabels: {
        enabled: false
      },
      xaxis: {
        categories: [
        ]
      },
    },
    colors: '#6e3780'
  },
  series: [{
    name: '',
    data: []
  }],
  pacient: '',
  info: null,

```

```

    tasks: [],
    id: '',
    name: '',
    surname: '',
    key: '',
    sex: '',
    zvych: '',
    IDpatient: [],
    selectedPatient: null,
    selectedVariant: null,
    ryz: [],
    show: false,
    selectedMonth: moment().format('MMM'), // За замовчуванням - поточний місяць
    selectedYear: moment().format('YYYY'), // За замовчуванням - поточний рік
    months: moment.months(), // Масив всіх дванадцяти місяців
    years: []
  }
},
mounted () {
  this.getMe()
  const startYear = moment().year() - 10 // Наприклад: 10 років тому
  const endYear = moment().year()
  for (let year = endYear; year >= startYear; year--) {
    this.years.push(year.toString())
  }
},
computed: {
  // eslint-disable-next-line vue/no-async-in-computed-properties
  async selectedDoctorObject () {
    console.log(this.selectedPatient)
    try {
      // eslint-disable-next-line vue/no-side-effects-in-computed-properties,vue/no-
      async-in-computed-properties
      this.info = await News.fetchPatient(
        this.selectedPatient
      )
    } catch (e) {}
    // eslint-disable-next-line vue/no-side-effects-in-computed-properties
    if (this.info.data.sex === '0') this.sex = 'Чоловіча'
    else if (this.info.data.sex === '1') this.sex = 'Жіноча'
    if (this.info.data.zvych === '0') this.zvych = 'Hi'
    else if (this.info.data.zvych === '1') this.zvych = 'Так'
    return this.info || {}
  },
  // eslint-disable-next-line vue/no-async-in-computed-properties,vue/return-in-
  computed-property
  async selectedVariantObject () {
    // eslint-disable-next-line vue/no-async-in-computed-properties
    await this.getPatientDataByDate()
  }
},
watch: {
  IDpatient: {
    handler () {
      this.getMe()
    }
  }
},
methods: {
  async Recomend () {
    if (this.selectedMonth !== moment().format('MMM')) {
      this.show = true
    } else this.show = false
  },
  async addRec (event) {
    try {
      await TaskS.addRec({
        rec: event.data.rec,

```

```

        year: this.selectedYear,
        month: this.selectedMonth,
        doctor: localStorage.id,
        pacient: this.selectedPacient
    })
} catch (err) {
    console.log(err.message)
}
},
async removeSelectedPacient () {
    await News.deletePacient({
        id: this.id,
        IDpacient: this.selectedPacient
    })
    this.selectedPacient = null
},
async getPacientDataByDate () {
    let b
    try {
        if (this.selectedVariant === 1) {
            const response = await News.getPacientDataByDate(
                this.selectedPacient,
                this.selectedYear,
                this.selectedMonth
            )

            this.tasks = response.data.tasks
            // Оновити дані для графіка
            var count = moment().month(this.selectedMonth).daysInMonth()
            var days = []
            this.graf = []
            this.par = []

            for (var i = 1; i < count + 1; i++) {
                if (i > 9) {
                    this.graf[i - 1] = 1
                    String(days.push(moment().month(this.selectedMonth).date(i)))
                } else {
                    this.graf[i - 1] = 0
                    String(days.push(moment().month(this.selectedMonth).date(i)))
                }
                let y = 0
                let result = this.tasks.reduce((a, c) =>
                    (a[c.date] = (a[c.date] || 0) + Number(c.title), a), {})
                var results = this.tasks.reduce(function (acc, el) {
                    acc[el.date] = (acc[el.date] || 0) + 1
                    return acc
                }, {})

                let l1 = []
                for (let [key, value] of Object.entries(result)) {
                    this.arr1 = value
                }
                var res = Object.keys(result).map(function (key) {
                    return result[key]
                })
                var res2 = Object.keys(results).map(function (key) {
                    return results[key]
                })
                var res2 = Object.keys(result).map(function (key) {
                    return key
                })

                for (let i = 0; i < res.length; i++) {
                    l1[i] = res[i] / res2[i]
                }
                var objs = res2.map(function (item, index) {
                    return {'date': item, 'title': l1[index]}
                })
            }

```



```

    },
    async Add () {
      await News.adds({
        id: localStorage.id,
        IDpatient: this.pacient
      })
      this.pacient = ''
    },
    async Logout () {
      await ProfilService.Logout()
      localStorage.clear()
      this.$router.push({name: 'DoctorLog'})
    }
  }
}
</script>

```

Головна сторінка користувача/пацієнта

```

<template>
  <b-container style="background-color: white; margin-top: 90px">
    <div class="header-rectangle"> <a style="color: black" href="#"
    @click="goToProfile">Профіль</a></div>
    <div class="header-rectangle" style="top: 70px"> <a style="color: black" href="#"
    @click="Log">Вийти</a></div>

    <main class="b-calendar">
      <Add v-if="adds"
      @close="adds = false"
      @done="addManually"/>
      <Edit v-if="edit"
      @close="edit = false"
      @done="editManually"
      />
      <b-row>
        <b-col md="4">
          <div class="b-calendar__information">
            <div class="today d-flex justify-content-center align-items-center">
              <div class="weekDay">
                {{selectedWeekDay | capitalize}}
              </div>
              <div class="day">
                {{selectedDayAndMonth.day}}
              </div>

              <div class="month">
                {{selectedDayAndMonth.month | capitalize}}
              </div>
              <a href="#" id="goTodayLink" @click="goToday" v-
              show="!todayInCurrentMonthAndYear || !todayIsEqualSelectDate">
                Today
              </a>

              <b-tooltip target="goTodayLink" v-show="!todayInCurrentMonthAndYear ||
              !todayIsEqualSelectDate">
                Back to today
              </b-tooltip>
              <div class="weekDay">
                <ul id="example-1">
                  <li v-for="item in tasks" :key="item.title" v-if="item.date===Select"
                  style="list-style-type: none;">
                    <a href="#" @click="edit=true; Edit(item._id)">sysBP: {{ item.title
                    }}</a>
                  </li>
                </ul>
                <p>{{this.ast}}</p>
                <p>{{this.norm}}</p>

```



```

        <div class="blockd" style="background-color:lightgreen;">1%</div>
        <div class="blockd" style="background-color:yellow;">2%</div>
        <div class="blockd" style="background-color:orange;">3-4%</div>
        <div class="blockd" style="background-color:orangered;">5-9%</div>
        <div class="blockd" style="background-color:red;">9-14%</div>
        <div class="blockd" style="background-color:darkred;">15%</div>
    </div>

</div>
<div class="col-md-12 " >
    <span class="day " >Рекомендації від лікаря: {{this.rec}}
    </span>
</div>
<div style="z-index: 99999999" >
    <span class="day" > </span>

</div>

<apexchart style="background-color: white;color:black;z-index:9999999; "
type="bar" :options="options" :series="series"></apexchart>

</div>
<div v-show="ap!=="true || but!=="true " v-if="Starts" class="date text-right"
: class="{
    'today': date.today,
    'blank': date.blank,

    'no-border-right': date.key % 7 === 0,
    }" v-for="date in dateList" :key="date.key" :data-
date="date.date" @click="setSelectedDate(date.moment)" >

        <span class="day momemnt" v-bind:style="{backgroundColor:
Colors(date.date)}">{{date.dayNumber}}</span>
        <span class="weekday">{{date.weekDay}}</span>
        <div class="additional" v-show="date.additional">
            <span class="year" v-
show="date.additional.year">{{date.additional.year}}</span>
            <span class="month" v-
show="date.additional.month">{{date.additional.month}}</span>
        </div>
    </div>
</div>
</div>
</b-col>
</b-row>
</main>
<footer class="b-footer">
    <b-row align-v="center" align-h="center">

        <b-col md="8" class="text-center">
            <button v-if="this.month===this.today.format('MMMM') " @click="adds=true"
class="block">+</button>

        </b-col>

    </b-row>
</footer>

</b-container>
</template>
<script>
import moment from 'moment'
import Add from './Add'
import Edit from './Edit'
import TaskS from '../services/New'
import '../assets/style.css'
import Slider from '@jeremyhamm/vue-slider'
import axios from 'axios'
import ApexCharts from 'apexcharts'

```

```

import ProfilService from '../services/ProfilService'
axios.defaults.headers.common['Authorization'] = localStorage.getItem('jwtToken')
class Memento {
  constructor (tasks, selectedDate) {
    this.tasks = tasks
    this.selectedDate = selectedDate
  }
  getTasks () {
    return this.tasks
  }

  getSelectedDate () {
    return this.selectedDate
  }
}
export default {
  name: 'Kalendar',
  components: {
    Add,
    Edit,
    'slider': Slider
  },
  data () {
    return {
      userData: {
        age: 65,
        sex: 1,
        smoker: 1,
        systolic: 120,
        cholesterol: 7
      },
      rec: [],
      risk: [],
      memento: null,
      but: false,
      Visibility: true,
      ap: false,
      age: '',
      norm: '',
      ast: '',
      you: '',
      tasks: [],
      ID: '',
      ryz: '',
      alg: '',
      sum: [],
      par: [],
      title: [],
      graf: [],
      arr1: [],
      arr2: [],
      user: [],
      titles: '',
      date: '',
      adds: false,
      edit: false,
      color: '',
      today: moment(),
      dateContext: moment(),
      selectedDate: moment(),
      days: ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'],
      options: {
        chart: {
          id: 'salesChart'
        },
        dataLabels: {
          enabled: false
        },
        xaxis: {

```

```

        categories: [
        ]

    },
    colors: '#6e3780'

},
series: [{
    name: 'АД',

    data: []
}]
}
},
mounted () {
    this.Find()
    this.calculateAlgForPreviousMonth()
},

methods: {
    goToProfile () {
        this.$router.push({ name: 'Profil' })
    },
    saveState () {
        this.memento = new Memento(this.tasks, this.selectedDate)
    },
    async updateChart () {
        let z = 0
        let leng = 0
        let k = 0
        var count = moment().month(this.month).daysInMonth()
        var days = []
        this.graf = []
        this.par = []
        for (var i = 1; i < count + 1; i++) {
            if (i > 9) {
                this.graf[i - 1] = String(days.push(moment().month(this.month).date(i)))
                console.log('this.graf[j]')
                console.log(this.graf[i - 1])
            } else {
                this.graf[i - 1] = '0' + String(days.push(moment().month(this.month).date(i)))
                console.log('this.graf[i]')
                console.log(this.graf[i - 1])
            }
        }
        console.log(this.graf.length)
        let y = 0
        let result = this.tasks.reduce((a, c) =>
            (a[c.date] = (a[c.date] || 0) + Number(c.title), a), {})

        var results = this.tasks.reduce(function (acc, el) {
            acc[el.date] = (acc[el.date] || 0) + 1
            return acc
        }, {})
        let l1 = []
        var obj = {}
        console.log(result)
        for (let [key, value] of Object.entries(result)) {
            this.arr1 = value
        }
        var res = Object.keys(result).map(function (key) {
            return result[key]
        })
        var ress = Object.keys(results).map(function (key) {
            return results[key]
        })
        var res2 = Object.keys(result).map(function (key) {
            return key
        })
    }
}

```

```

    })
    for (let i = 0; i < res.length; i++) {
        l1[i] = res[i] / ress[i]
    }
    console.log(Object.assign(...res2.map((n, i) => ({ [n]: l1[i] }))))
    var objs = res2.map(function (item, index) {
        return {'date': item, 'title': l1[index]}
    })
    for (let j = 0; j < this.graf.length; j++) {
        this.par[j] = ' '
        for (let i = 0; i < objs.length; i++) {
            var a = objs[i].date.split(' ')[1] // 50ml
            var b = objs[i].date.split(' ')[2] // $100
            if (b === this.year) {
                if (a === this.month) {
                    if (this.graf[j] === objs[i].date.split(' ')[0]) {
                        this.par[j] = objs[i].title.toFixed(1)
                    }
                }
            }
        }
    }
    this.risk = await ProfilService.fetchMonthandYear(
        {
            owner: localStorage.id,
            year: this.year,
            month: this.month
        }
    )
    const r = await TaskS.getRec({
        pacient: localStorage.id,
        year: this.year,
        month: this.month
    })
    if (r.data.length !== 0) { this.rec = r.data[0].rec } else this.rec = ' '
    ApexCharts.exec('salesChart', 'updateOptions', {
        xaxis: {
            categories: this.graf
        }
    })

    ApexCharts.exec('salesChart', 'updateSeries', [
        {
            data: this.par
        }
    ])
    if (this.risk.data[0].score !== null) { this.alg = this.risk.data[0].score }
},
    async calculateAlgForPreviousMonth () {
        const todayMinusOneMonth = moment(this.today).subtract(1, 'month')
        const previousMonth = todayMinusOneMonth.format('MMMM')
        const previousYear = todayMinusOneMonth.format('Y')
        var response = await TaskS.fetchTasks()
        const result = await ProfilService.fetchMonthandYear({
            owner: localStorage.id,
            year: previousYear,
            month: previousMonth
        })
        if (result.data === null) {
            this.sum = ''
            this.alg = ''
            this.ryz = ''
            let z = 0
            let leng = 0
            let k = 0
            let xx = ''
            for (let i = 0; i < response.data.tasks.length; i++) {
                const a = response.data.tasks[i].date.split(' ')[1] // 50ml
                const b = response.data.tasks[i].date.split(' ')[2] // $100
            }
        }
    }
}

```

```

    if (toString(b) === toString(previousYear)) {
      if (a.toString() === previousMonth.toString()) {
        z += Number(response.data.tasks[i].title)
        leng++
        k = 1
        xx = response.data.tasks[i]
      }
    }
  }
  z = z / leng
  if (z) {
    this.but = true
    const num = await ProfilService.predict_risk({
      smoke : Number(resp.data.zvych),
      sex : gen,
      age : Number(xx.age),
      gen : Number(resp.data.sex),
      ad : z,
      chol : Number(xx.hol)
    })
    this.alg = num.toFixed(2)
    if (num.toFixed(2) < 1) {
      this.ryz = 'Дуже низький'
    }
    if (num.toFixed(2) > 15) {
      this.ryz = 'Дуже високий'
    }
    if (num.toFixed(2) >= 2 && num.toFixed(2) < 3) {
      this.ryz = 'Помірний'
    }
    if (num.toFixed(2) >= 1 && num.toFixed(2) < 2) {
      this.ryz = 'Низький '
    }
    if (num.toFixed(2) >= 3 && num.toFixed(2) <= 4) {
      this.ryz = 'Середній'
    }
    if (num.toFixed(2) > 4 && num.toFixed(2) <= 9) {
      this.ryz = 'Високий'
    }
    if (num.toFixed(2) > 9 && num.toFixed(2) <= 14) {
      this.ryz = 'Дуже високий'
    }
    if (num.toFixed(2) > 15) {
      this.ryz = 'Надзвичайно високий'
    }
    console.log('SCORE')
    console.log(this.alg)
    console.log(this.ryz)
    await TaskS.addRisk({
      score: this.alg,
      ryz: this.ryz,
      month: previousMonth,
      year: previousYear,
      owner: localStorage.id
    })
  } else {
  }
} else {
  console.log('Reserved')
}
},
async DoIt () {
  this.sum = ''
  let z = 0
  let leng = 0
  for (let i = 0; i < this.tasks.length; i++) {
    if (this.tasks[i].date === this.selectedDate.format('DD MMMM YYYY')) {
      this.sum = this.tasks[i].title
      z += Number(this.tasks[i].title)
    }
  }
}

```

```

        leng++
    }
}
z = z / leng
this.you = ' '
this.ast = ' '
this.norm = ' '
if (z) {
    const response = await ProfilService.getAST({sysBP: z})
    const responseData = response.data
    this.ast = responseData
    let num = z
    var prof = await ProfilService.getMe()
    const res = await ProfilService.getNorm({sex: prof.data.sex,
        age: prof.data.age,})
    const resData = response.data
    this.norm = 'Ваша норма' + response.data
    this.you = 'Ваш тиск за цей день' + num.toFixed(1)
}
},
async Log () {
    await ProfilService.Log()
    localStorage.clear()
    this.$router.push({name: 'Log'})
},
async Find () {
    this.titles = ''
    var response = await TaskS.fetchTasks()
    this.tasks = response.data.tasks
    for (let i = 0; i < this.tasks.length; i++) {
        if (this.tasks[i].date === this.selectedDate.format('DD MMMM YYYY')) {
            this.titles = this.tasks[i].title
            this.sum = this.tasks[i].title
        } else this.titles = ''
    }
    this.you = ' '
    this.ast = ' '
    this.norm = ' '
    if (this.today.format('DD MMMM YYYY') !== this.selectedDate.format('DD MMMM YYYY'))
    {
        this.DoIt()
    }
},
}
</script>

```

Серверна частина (адміністратор)

```

router.post("/create", async (req, res) => {

    try {
        const doctor = new Doctor({
            name: req.body.name,
            surname: req.body.surname,
            key: req.body.key,
            password: req.body.password,
        });
        await doctor.save();
    } catch (e) {
        res.status(400).send()
    }

});

router.get("/alldoctor", auth, async (req, res) => {

```

```

    try {
      const doctor = await Doctor.find();
      res.status(200).send({doctor: doctor});
    } catch {
      res.status(500).send();
    }
  });
  router.post("/delete", auth, async (req, res) => {

    try {
      console.log(req.body._id)
      const doctor = await Doctor.findById(req.body.id);
      doctor.remove()
      res.status(200).send();
    } catch(e) {
      res.status(500).send(e.message);
    }
  });

  router.post("/login", jsonParser, async(req, res) => {

    try {
      const admin= await Admin.findOneByCredentials(req.body.word, req.body.password);
      const token=await admin.generateAuthToken();
      res.send({admin, token});

    } catch (e) {
      res.status(400).send()
    }
  });
  router.post("/logout", auth, async(req, res) => {

    try{
      console.log(req.admin.tokens)
      console.log(req.token)
      console.log('jnjjnjjn')

      req.admin.tokens=req.admin.tokens.filter((token)=>{
        return token.token!==req.token;
      });
      await req.admin.save();
      console.log(req.admin)
      res.send('exit');

    } catch (e) {
      res.status(500).send(e.message)
    }
  });
  module.exports = router;
  Серверна частина( лікар)
  router.get('/me', auth, async (req, res) => {
    res.send(req.doctor);
  });

  router.post("/login", jsonParser, async(req, res) => {
    console.log('jfjhbjhj')
    try {
      const doctor= await Doctors.findOneByCredentials(req.body.key, req.body.password);
      const token=await doctor.generateAuthToken();
      res.send({doctor, token});
    } catch (e) {
      console.log(e.message)
      res.status(500).send(e.message)
    }
  });

  router.get("/foundpatient/:patientId", auth, async (req, res) => {

```

```

    try{
      console.log('req.body._id')
      let user = await Users.findById(req.params.pacientId);
      console.log(user)
      res.status(200).send(user);
    } catch(error) {
      res.send(error.message);
    }
  });

  router.post("/logout", auth, async(req, res) => {

    try{
      req.doctor.tokens=req.doctor.tokens.filter((token)=>{
        return token.token!==req.token;
      });
      await req.doctor.save();
      console.log(req.doctor)
      console.log('EXIT')
      res.send('exit');

    }catch (e) {
      res.status(500).send(e.message)
    }
  });

  router.post("/delete", auth, async (req, res) => {

    try {
      const doctor = await Doctors.findById(req.body.id);
      doctor.IDpacient = doctor.IDpacient.filter(
        (IDpacient) => IDpacient !== req.body.IDpacient
      );
      await doctor.save();
      res.status(200).send();
    } catch(e) {
      res.status(500).send(e.message);
    }
  });

  router.post("/add", async (req, res) => {

    try {
      const doctor = await Doctors.findById(req.body.id)
      doctor.IDpacient.push(req.body.IDpacient);
      await doctor.save();
      res.status(200).send(doctor);
    } catch (e) {
      res.status(500).send(e.message);
    }
  });

  router.get('/pacientData/:pacientId/:year/:month', async (req, res) => {
    try {
      const tasks = await Tasks.find({owner: req.params.pacientId});
      res.status(200).send({tasks: tasks});
    } catch(e) {
      console.log(e.message)
      res.status(500).send();
    }
  });
  module.exports = router;

```

Серверна частина (пацієнт)

```

const express=require("express");
const router=new express.Router();
const Users = require("../models/user");
const jsonParser = express.json();

```

```

const auth = require("../middleware/auth");
const { PythonShell } = require('python-shell');
const tf = require('@tensorflow/tfjs-node');

router.post("/tyisk", async (req, res) => {
  const {sysBP} = req.body;
  const options = {
    mode: 'text',
    scriptPath: 'D:/home/magistr/Diplom/server/src/routers', // Замініть на власний
    шлях
    args: [sysBP],
  };
  try {
    const results = await PythonShell.run('predict.py', options);
    const prediction = results[0];
    res.status(200).send(prediction);
  } catch (err) {
    console.error(err);
  }
})

router.get('/me', auth, async (req, res) => {
  res.send(req.user);
});
router.post("/create", async (req, res) => {
  try {
    const user = new Users({
      name: req.body.name,
      surname: req.body.surname,
      tel: req.body.tel,
      password: req.body.password,
      vik: req.body.vik,
      height: req.body.height,
      weight: req.body.weight,
      sex: req.body.sex,
      zvyh: req.body.zvyh,
      hol: req.body.hol,
    });

    await user.save();
    const token = await user.generateAuthToken();
    res.status(201).send({user, token});
  } catch (e) {
    res.status(400).send()
  }
});

router.post("/login", jsonParser, async (req, res) => {
  try {
    const user = await Users.findOneByCredentials(req.body.tel, req.body.password);
    const token = await user.generateAuthToken();
    res.send({user, token});

  } catch (e) {
    res.status(400).send()
  }
});

router.get("/found/:id", async (req, res) => {
  try {
    let user = await Users.findById(req.params.id);
    res.status(200).send(user);
  } catch (error) {
    res.send(error.message);
  }
}

```

```

});

router.patch("/:id", async (req, res) => {

  try{
    const user = await Users.findById(req.params.id);
    const updates = ['name', 'surname', 'tel',
'zvych', 'hol', 'weight', 'height', 'vik', 'sex'];
    updates.forEach((update) => user[update] = req.body[update]);
    await user.save();
    res.status(200).send(user);

  } catch(e) {
    res.status(400).send(e.message);
  }
});

var upload = multer({storage: storage});
router.patch("/upload_files/:id", upload.single("file"), async ( req, res)=>{
  const user = await Users.findById(req.params.id);
  const updates = ['img'];
  updates.forEach((update) => user[update] = req.file.filename);
  await user.save();

});

router.post("/logout", auth, async(req, res) => {

  try{
    req.user.tokens=req.user.tokens.filter((token)=>{
      return token.token!==req.token;
    });
    await req.user.save();
    res.send('exit');

  }catch (e) {
    res.status(500).send(e.message)
  }
});

module.exports = router;

```