Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем та програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

Магістр
(назва освітнього ступеня)

на тему: Проєктування інформаційної системи ідентифікації параметрів анормальний неврологічних рухів людини під дією техногених навантажень на Javascript технології

Виконав(ла): студент(ка) 6 курсу, групи СПм-61
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

_____ Стефанишин І. М.
(підпис) (прізвище та ініціали)

Керівник _____ Петрик М. Р.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Стоянов Ю. М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Петрик М. Р.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль 2022

РЕФЕРАТ

Кваліфікаційна робота на тему «Проектування інформаційної системи ідентифікації параметрів анормальних неврологічних рухів людини під дією техногених навантажень на Javascript технології» написана Стефанишином Іваном Миколайовичем, студентом Тернопільського національного технічного університету імені Івана Пулюя, Факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СПм-61.

Відомості про обсяг: сторінок – __, рисунків – 20, таблиць – 1, частин – 4, додатків – 3, посилань – 21, формул – 10.

Розроблено методику оцінки стану пацієнтів у клініках за допомогою методу розпізнавання тремору малюванням спіралі. На основі теорії розповсюдження хвильового сигналу створено математичну модель для аналізу анормальних неврологічних рухів та визначено сегментований опис 3D елементів досліджуваної траєкторії цих рухів.

В основі реалізації лежить метод визначення положення кінцівок рук пацієнта за допомогою електронного пера, яке відтворює малюнок у вигляді спіралі Архімеда на екрані інтерактивного планшета. Просторове 3D відхилення траєкторії пера від шаблону має складну форму і є цифровою інформацією, яка використовується для визначення стану пацієнта. Модель була використана для отримання коефіцієнтів ідентифікації та подальшого їх оцінювання.

Ключові слова: КОМП'ЮТЕРНЕ МОДЕЛЮВАННЯ, ДІАГНОСТИКА, ТРЕМОР, АВТОМАТИЗАЦІЯ, МАТЕМАТИЧНА МОДЕЛЬ, АЛГОРИТМИ, КОГНІТИВНІ ЗВ’ЯЗКИ, КОЕФІЦІЄНТИ ІДЕНТИФІКАЦІЇ, СПЕЦІАЛІЗОВАНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

ABSTRACT

The qualification work on the topic "Design of the information system for the identification of the parameters of abnormal neurological human movements under the effect of technological loads" was written by Stefanyshyn Ivan Mykolayovych, a student of Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering , group SPm-61.

Information about the volume: pages – __, figures – 20, tables – 1, parts – 4, appendices – 3, references – 21, formulas – 10.

A technique for assessing the condition of patients in clinics using the method of recognizing tremors by drawing a spiral has been developed. Based on the theory of wave signal propagation, a mathematical model was created for the analysis of abnormal neurological movements and a segmented description of the 3D elements of the investigated trajectory of these movements was determined.

The implementation is based on the method of determining the position of the patient's limbs using an electronic pen, which reproduces a drawing in the form of an Archimedean spiral on the screen of an interactive tablet. The spatial 3D deviation of the pen trajectory from the template has a complex shape and is digital information used to determine the patient's condition. The model was used to obtain identification coefficients and further evaluate them.

Keywords: COMPUTER SIMULATION, DIAGNOSTICS, TREMOR, AUTOMATION, MATHEMATICAL MODEL, ALGORITHMS, COGNITIVE CONNECTIONS, IDENTIFICATION COEFFICIENTS, SPECIALIZED SOFTWARE.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА ТЕРМІНІВ 6
ВСТУП 7
1 ПРОБЛЕМАТИКА ДОСЛІДЖЕННЯ СПОСОБІВ ОБСТЕЖЕННЯ СТАНУ ЗДОРОВ'Я ЛЮДИНИ З ПЕРІОДИЧНИМИ ПРОЯВАМИ ТРЕМОРУ 9
1.1 Вплив ІТ на медичну сферу 9
1.2 Стан медичних закладів для виявлення і діагностування тремору у людей з його ознаками в Україні 10
1.3 Медичне тлумачення хвороби "Тремор" та її класифікація 12
1.4 Складність точного виявлення ступеня захворювання тремору спіраллю Архімеда 15
1.5 Постановка завдання 16
2 АНАЛІЗ ТА ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ 17
2.1 Вибір методології розробки 17
2.2 Математичне представлення задачі 19
2.3 Пошук акторів та варіантів використання 23
2.3.1 Детальний опис варіантів використання 26
2.4 Застосування шаблону розробки застосунку 27
2.5 Розділення інформаційної системи на рівні 29
2.6 Представлення системи в абстрактному вигляді 31
2.7 Проєктування бази даних 51
2.8 Діаграма розгортання застосунку 54
3 РОЗРОБКА ТА ТЕСТУВАННЯ 57
3.1 Вибір мови програмування та технологій розробки 57
3.2 Розробка застосунку 62
3.3 Ілюстрування роботи розробленої інформаційної системи 66
4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ 72

4.1 Охорона праці 72
4.2 Підвищення стійкості роботи об’єктів господарської діяльності у воєнний час 74
ВИСНОВКИ 78
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 79
ДОДАТКИ 81
Додаток А – Технічне завдання 82
Додаток Б – Публікація у науковому виданні 92
Додаток В – Лістинг коду інформаційної системи 95
Додаток Г – Диск із кваліфікаційною роботою магістра 101

ПЕРЕЛІК СКОРОЧЕНЬ ТА ТЕРМІНІВ

ЦНС – центральна нервова система.

ДПЖ – діяльність у повсякденному житті.

ІДПЖ – інструментальну діяльність у повсякденному житті.

ЕМГ – Електроміографія.

MDS – Міжнародне товариство рухових розладів.

UPDRS – Уніфікована шкала оцінки хвороби Паркінсона.

MDS–UPDRS – Міжнародне товариство Паркінсона та рухових розладів

ВБС – віртуальною базовою станцією.

ВДТ – візуальний дисплейний термінал.

ООП – (Об’єктно-орієнтоване програмування) парадигма програмування, в якій основою є класи та об’єкти, які між собою взаємодіють.

UML – (Unified Modeling Language) уніфікована мова графічного представлення та об’єктного моделювання в області розробки програмного забезпечення парадигми об’єктно-орієнтованого програмування.

ВВ – варіанти використання.

ХП – хвороба Паркінсона.

JavaScript – динамічна, об'єктно-орієнтована прототипна мова програмування.

Node.js – платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript.

ReactJS – це відкрита JavaScript бібліотека для створення інтерфейсів користувача.

ВСТУП

Актуальність теми. У сучасному світі тремор кінцівок людини є найпоширенішим руховим порушенням. Тремор за природою має різні види походження та різну частоту, що ускладнює своєчасну діагностика всіх характеристик тремору. З цих причин швидке діагностування хвороби є одним з ключових проблем, що вирішуються в сфері сучасної неврології.

Зв'язок роботи з програмами наукових досліджень кафедри ПІ. Тема кваліфікаційної роботи співвідноситься з науковим спрямування кафедри програмної інженерії, а саме: аналізування проблемної області, проектування, імплементація, тестування та впровадження інформаційної системи, автоматизація діагностування за допомогою програмних інструментів, математичне моделювання, вдосконалення процесів та перевірка параметрів.

Мета і задачі дослідження. Метою даної роботи є проєктування та розробка інформаційної системи для автоматизації діагностування та оцінювання анормальних неврологічних рухів людини з використанням пристроїв оцифровування інформації, математичних методів моделювання складних систем та препроцесинг інформації, які характеризують зворотний зв’язок сигналів із кори головного мозку пацієнтів з анормальними неврологічними рухами.

Об’єктом дослідження є процедура проєктування та розробки інформаційної системи для автоматизації діагностування тремору шляхом обчислення коефіцієнтів ідентифікації руху кінцівок людини та подальший аналізу вихідних даних експертом для оцінювання ступеня тремору.

Предмет дослідження – були програмні методи автоматизації діагностування хвороб, способи побудови розроблених математичних моделей, які базуються на демонстрації та обробці інформації, яка була отримана під час діагностування

пацієнта або виконані методів перетворення інформації, методи швидкої обробки безперервного потоку даних, які приходять у вигляді сигналів.

Методи дослідження. Для того, щоб досягти поставлених цілей у кваліфікаційній роботі, були використані наступні методи: методи читання та парсингу текстових файлів, методи математичного моделювання, методи представлення математичної моделі за допомогою графіків, алгоритмізація процесів, виконання, програмні методи опису складних процесів для опису анормальних неврологічних рухів пацієнта з ознаками тремору, методи перетворення математичних виразів.

Елементи наукової новизни та/або здійснення інновацій в умовах невизначеності вимог в якості одержаних результатів. Було розроблено нову методологію обчислення коефіцієнтів ідентифікації для покращення виявлення ступеню тремору у пацієнта та спроєктовано та розроблено автоматизовану інформаційну систему на основі використання математичної моделі для оцінки наявності тремору у пацієнта, яке описує стан та поведінку 3D елементів траєкторії анормальних невролоогічних рухів та включає рішення проблематики нової некласичної багатопараметричної ідентифікації досліджуваної системи. Це потенційно може покращити швидкість, якість і простоту аналізу та оцінки даних, отриманих в результаті діагностування, а також підвищити роботу медичних установ у діагностиці стану тремору у пацієнтів з анормальними неврологічними рухами.

Практичне значення одержаних результатів. В наслідок розробки вдалих методик оцінювання ступеня тремору і правильно спроєктованій архітектур інформаційної системи, розроблені застосунки на основі перелічених переваг стають необхідним атрибутом будь-якого науково-технічного процесу. Процес автоматизація процесу оцінювання тремору з використанням тесту спіралі Архімеда оснований на обробці оцифрованих даних. Використовуючи математичну модель можна отримати високоякісний аналіз та числове представлення рухової поведінки пацієнта.

1 ПРОБЛЕМАТИКА ДОСЛІДЖЕННЯ СПОСОБІВ ОБСТЕЖЕННЯ СТАНУ ЗДОРОВ'Я ЛЮДИНИ З ПЕРІОДИЧНИМИ ПРОЯВАМИ ТРЕМОРУ

1.1 Вплив IT на медичну сферу

Технології змінюють те, як ми практикуємо медицину. Датчики та переносні пристрої стають меншими та дешевшими, а алгоритми стають достатньо потужними, щоб передбачати медичні результати. Проте, незважаючи на швидкий прогрес, охорона здоров'я відстає від інших галузей у справжньому застосуванні цих технологій. Основною перешкодою для входу є міждисциплінарний підхід, необхідний для створення таких інструментів, що вимагає знань багатьох людей у багатьох галузях. Тому в сьогоднішній час люди прагнуть рухати сферу вперед, розпаковуючи цей бар'єр, надаючи короткий вступ до основних понять і термінів, які визначають цифрову медицину. Зокрема, відбувається протиставлення «клінічні дослідження» та звичайну «клінічну допомогу», окреслюючи питання безпеки, етичні, нормативні та правові питання, які розробники повинні враховувати, коли продукти цифрової медицини виходять на ринок. Такоє відбувається класифікація типу цифрових вимірювань і способів використання та перевірки цих вимірювань у різних умовах. Виконання всіх вказаних пунктів повинно прискорити безпечне та ефективне просування сфери цифрової медицини.

Продукти цифрової медицини обіцяють покращити медичне вимірювання, діагностику та лікування. У той час як багато галузей прийняли цифровий зрив, галузь охорони здоров'я ще не відчула покращення результатів, доступу та економічної ефективності, які давно обіцяла цифрова революція. Охорона здоров'я відстає від інших галузей частково через нормативно-правове середовище, яке, як правило, уповільнює прогрес, оскільки органи охорони здоров'я прагнуть мінімізувати несприятливі наслідки.

Розробка ефективних інструментів цифрової медицини є інтенсивним і складним процесом, який вимагає міждисциплінарних зусиль широкого кола експертів, від інженерів і етиків до платників і постачальників. Багато проблем ускладнюються міждисциплінарним характером цієї галузі. Розвиток цифрової медицини зупиняється, коли складові експерти говорять різними мовами та мають різні стандарти, досвід і очікування.

1.2 Стан медичних закладів для виявлення і діагностування тремору у людей з його ознаками в Україні

Пандемія COVID-19 порушила клінічну практику, змусивши перейти на телемедицину для надання клінічної допомоги. Пандемія також порушила клінічні дослідження та викликала відповідні занепокоєння щодо безпеки учасників. Принаймні в найближчій перспективі клінічні та дослідницькі операції повинні будуть адаптуватися до цього нового середовища. Однією з критичних адаптацій буде включення альтернативних засобів оцінки та лікування поза межами клініки.

В наш час так склалось, що все частіше почалися появлятись медичні заклади, які спеціалізуються на виявлені, діагностуванні та лікуванні тремору. В загальному тенденція зростання кількості таких закладів спостерігається в великих обласних центрах України, таких як Львів, Київ, Харків та інші. Мова йде не лише про державні установи, але і про приватні заклади, які в більшості спеціалізуються лише над пов’язаним переліком проблем. Із виникненням нових структур для медичного обслуговування, як правило медична техніка закуповується сучасних зразків, що покращує процеси діагностування пацієнта. Відповідно, получається, що у людей з ознаками тремору появилось більше можливостей провести всі необхідні процедури

для діагностування людини, що спричинить виявлення тремору на ранніх стадіях захворювання.

Не слід забувати про невиліковні хвороби, а саме про хворобу Паркінсона. Основним завданням сучасної медицини при цій недузі є сповільнення прогресу хвороби і покращенні рівня життєдіяльності пацієнта. Ось чому у випадку діагностування хвороби Паркінсона на ранніх етапах і призначення відповідного лікування, хворий може тривалий час продовжувати своє активне життя, незважаючи на захворювання.

У клініці тремор діагностують під час процесу, обмеженого часом, під час якого спостерігають за пацієнтами та візуально оцінити характеристики тремору. Для деяких розладів тремору потрібен більш детальний аналіз цих характеристик. Для кращого уявлення про тремор можна використовувати акселерометрію та електроміографію.

Як правило, рутинна клінічна оцінка даних акселерометрії та електроміографії передбачає візуальний огляд лікарем та інколи обчислювальний аналіз для отримання об’єктивних характеристик тремору. Однак для деяких розладів тремору ці характеристики можуть відрізнятися під час повсякденної активності. Ця мінливість у представленні між клінікою та повсякденним життям ускладнює диференціальну діагностику.

Хоча створюють нові медичні заклади, однак частка вітчизняних засобів реєстрування результатів діагностування все ще залишаються та інколи можуть давати неточні результати. Для застосування сучасних методів для діагностування пацієнта, які базуватимуться на автоматизованому виявленні тремору, потрібні високоточні прилади для якісного аналізу даних.

При аналізі медичної сфери у напрямку діагностування тремору, було виділено декілька головних перешкод у провадженні автоматизованих процесів для виявлення хвороби у пацієнтів [6]:

- Використання застарілої техніки для діагностування

- Невелика статистична база даних пацієнтів із тремором
- Дороговартісність інтегрування сучасних високотехнологічних приладів діагностування у медичних заклади
- Складність визначення однозначності в дослідженнях
- Низька точність визначення степення тремору
- Вузькоспеціалізований аналіз інформації у спектрі великої кількості характеристичних ознак

Ось чому на порядку денному є питання про необхідність покращення методів та способів до сфери діагностування тремору в умовах реального часу. Основна проблема, яка перешкоджає втіленню реформи старих підходів, це недосконаліть методів аналізу і визначення захворювання пацієнта на тремор.

1.3 Медичне тлумачення хвороби “Тремор” та її класифікація

Тремор є найпоширенішим руховим розладом, що характеризується повторюваними і стереотипними рухами. Тремор людини – це клінічний прояв, що характеризується мимовільним, ритмічним, коливальним рухом частини тіла, який можна класифікувати різними способами залежно від його етіології, феноменології, частоти, локалізації та фармакологічної реакціями [1].

Ритмічні характеристики тремору навколо збалансованого положення являють собою регулярний ритм. Амплітуда та частота цього ритму полегшують ідентифікацію, а також диференціальну діагностику тремору, відрізняючи його від інших мимовільних рухів.

Рухи, спричинені тремором, можуть бути пов’язані з багатьма факторами, такими як неврологічні розлади та природні процеси. Останній часто називають фізіологічним тремором і присутній більшою чи меншою мірою у всіх людей.

Наявність важких треморних розладів викликає багато труднощів, а також може свідчити про наявність захворювань, пов'язаних з ЦНС.

Однак межа між фізіологічним тремором і результатом дисфункцій є незначною і точно не встановлена, оскільки зміни в контролі ЦНС, які викликають його, можуть бути пов’язані з багатьма факторами. Деякими прикладами патологічного тремору, тобто пов’язаного з факторами неврологічного розладу, є мозочковий, есенціальний і паркінсонічний тремор, а також інші, такі як психогенний, ортостатичний і невропатичний тремор, які вважаються відносно рідкісними в медичній літературі.

Згідно з феноменологією, або краще, відповідно до обставин, за яких тремор проявляється, його можна класифікувати на два основних типи: тремор спокою та тремор дії. Тремор у стані спокою можна спостерігати, коли частина тіла, в якій він з’являється, не страждає від дії сили тяжіння і м’язи не скорочуються. Зазвичай тремтіння в стані спокою має ознаки приведення-відведення або згинання-розгинання. Основним прикладом тремору спокою є паркінсонічний тремор. Тремор дії виникається під час довільного скорочення м'язів. Він охоплює постуральний, кінетичний, специфічний для завдання та ізометричний тремор.

Зазвичай есенціальний тремор і посилений фізіологічний тремор класифікують як постуральний тремор. Паркінсонічний тремор зазвичай є тремором у стані спокою і зменшує свою амплітуду під час руху. Натомість тремтіння мозочка виникає під час виконання руху, тому вважається тремором дії, переважно кінетичним. У мезенцефальному треморі та треморі Холмса можна виявити суміш спокою, постурального та кінетичного тремору з високою амплітудою та інтенсивністю.

За частотою тремтіння, а точніше, за кількістю коливань ураженого сегмента в одиницю часу, тремтіння можна класифікувати на три основні типи [2]:

- тремор низької частоти (менше чотирьох циклів на секунду або герц);
- середньочастотний тремор (між 4 і 7 Гц)

- високочастотний тремор (більше 7 Гц).

Для кращого розуміння нижче наведено таблицю класифікації тремору в залежності від частоти (табл. 1).

Таблиця 1 – Частота тремтіння при різних видах захворювання

Неврологічний розлад	Клінічний огляд	Діапазон частот	Уражені частини тіла
Есенціальний тремор	Концепція тесту «палець до носа» та частота S-діапазону	4-12Гц	Рука, голова та гортань
Тремор Паркінсона	ЕМГ-сигнали та поняття аналізу сингулярного спектру	4-6Гц	Все тіло, але переважно кінцівки
Фізіологічний тремор	Клінічний огляд, фізичне та фізіологічне обстеження.	3-30Гц	Тільки кінцівки під час виконання діяльності

Що стосується локалізації, то можна відзначити, що тремор може виникнути в будь-якій частині тіла; однак найбільше уражаються сегменти кінцівок і голова. Можуть бути залучені інші частини тіла, наприклад тулуб, але така ситуація не є поширеною.

Тисячі людей щороку починають проявляти певний тип моторної дисфункції, яка заважає їхній повсякденній діяльності та значно знижує якість життя цих людей. Низка досліджень і державна статистика показали, що люди похилого віку найбільше страждають від тремору та його наслідків, які є причиною фізичних обмежень цих осіб.

Прояв тремору може спричинити значну функціональну неспроможність, що призводить до соціальної ізоляції через втручання діяльність у повсякденному житті

і інструментальну діяльність у повсякденному житті, таку як їжа, письмо, одягання та догляд за собою.

1.4 Складність точного виявлення ступеня захворювання тремору спіраллю Архімеда

Якість досліджень тремору в аномальних станах знижується на 60-80% через застосування класичних моделей цифрової підрахунку, оскільки ці методи базуються на відкиданні шумів, а частка відкинутих важливих даних становить від 60 до 80%, які можуть відповісти на питання про справжні когнітивні системи впливу на різні сегменти кривої анормальних неврологічних рухів.

Через недосконаліть діагностичних методів медичних установ, тому інколи буває важко виявити та оцінити захворювання тремору кінцівок. Під час розгляду проблеми не було знайдено жодного сучасного засобу автоматичної діагностики пацієнтів. Визначено деякі проблеми: використання застарілої техніки для діагностування тремору, невелика статистична база даних пацієнтів із тремором, дороговартісність інтегрування сучасних високотехнологічних приладів діагностування у медичних заклади, складність визначення однозначності в дослідженнях, низька точність визначення степення тремору, вузькоспеціалізований аналіз інформації у спектрі великої кількості характеристичних ознак, нечіткі результати оцінки [4]. Крім того, деталі використання цього комплексу виявлення та ідентифікації можна віднести до недоліків, які не роблять його підходящим на всі випадки захворювання тремору. У цьому випадку найчастіше ставлять діагноз есенціальний тремор.

Максимальна частота дискретизації даних більшості датчиків і чутливих елементів наближається до 133 разів на секунду, що вповні вистачає для виявлення

всіх фізіологічних і патологічних рухів. На практиці, однак, цієї точності та частоти дискретизації недостатньо для діагностування тремтіння, яке є непомітним для людського ока [5]. Незважаючи на недоліки та обмежену продуктивність, нижньої межі чутливості планшета достатньо для вимірювання есенціального тремору у більшої частини пацієнтів із ознаками тремору.

1.5 Постановка завдання

Дослідження, що проводяться з метою внеску в науку шляхом систематичного збору, інтерпретації та оцінки даних, а також планомірно називаються науковими дослідженнями. Тому, як правило, наукові дослідження не ставлять на перший план опцію матеріальної вигоди. Результати, отримані невеликою групою в ході наукових досліджень, оприлюднюються, і відкривається нова інформація щодо діагностики, лікування та надійності застосування. Метою цього огляду є надання інформації про визначення, класифікацію та методологію наукових досліджень.

Перед початком наукового дослідження дослідник повинен визначити предмет, спланувати та визначити методологію. У Гельсінській декларації зазначено, що «основна мета медичних досліджень на добровольцях полягає в тому, щоб зрозуміти причини, розвиток і наслідки захворювань і розробити захисні, діагностичні та терапевтичні заходи. Навіть найкращі перевірені втручання слід постійно оцінювати за допомогою досліджень щодо надійності, дієвості, ефективності, доступності та якості».

2 АНАЛІЗ ТА ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Вибір методології розробки

За останні кілька років сформувалося нове покоління розробки програмного забезпечення. Веб-застосунки мають надійну, гнучку та масштабовану архітектуру, розроблену для роботи з продуктивністю та навантаженням, необхідними для реальних великих і непередбачуваних обсягів. Як правило, ці архітектури є багаторівневими, забезпечуючи інкапсуляцію бізнес-логіки та інтеграцію кількох розрізнених «застарілих» систем. Особливо важко керувати процесом розробки програмного забезпечення і передбачати кінцевий результату системи.

Виходячи із складності та термінів проєкту було здійснено рішення використовувати методологію під назвою раціональний уніфікований процес розробки програмного забезпечення. Нижче наведено схематичний рисунок роботи раціонального уніфікованого процесу (рис. 2.1).

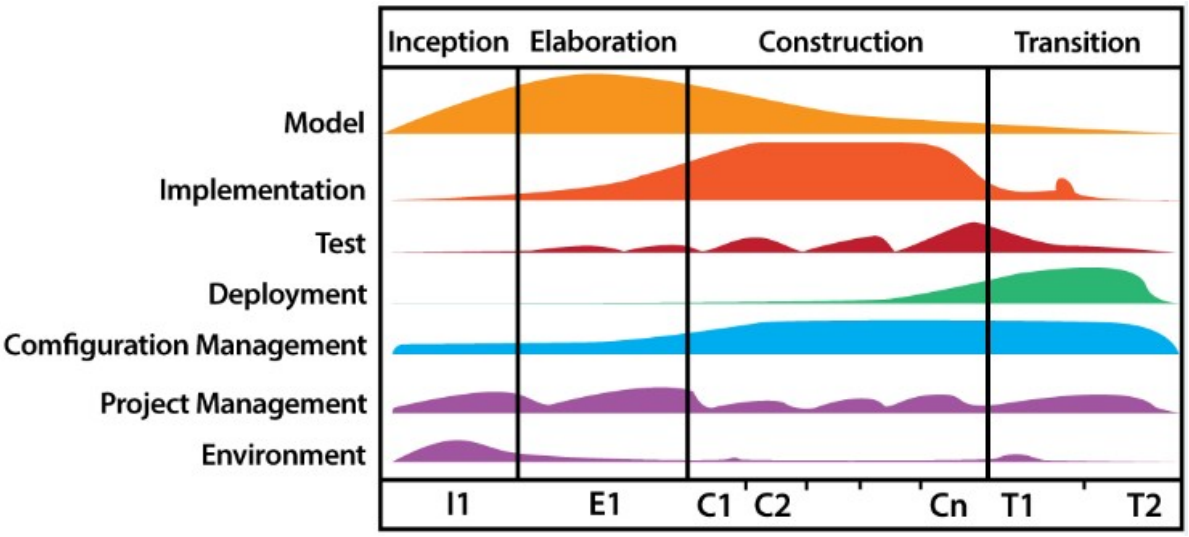

Рисунок 2.1 – Етапи раціонального уніфікованого процесу

Основною перевагою використання прийнятої методології розробки є принцип ітеративної та поетапної розробки. Так як зміни додаються невеликими порціями, то результат можна побачити за короткий проміжок часу і відповідно до отриманого результату коректувати або вносити нові зміни в існуючий функціонал. Також варто пам’ятати про принципи, які надає раціональний уніфікований процес розробки і не потрібно намагатися «виконати» все за один раз.

Слід завжди тримати в голові під час реалізації, вивчення та використання те, що цей підхід сам по собі є ітеративним і поступовим. Ось чому для кожного існуючого процесу потрібно брати декілька ключових областей та виконувати їх вдосконалення. Для покращення кожної з вибраних областей потрібно використати раціональний уніфікований процес, а потім, у наступних ітераціях або циклах розробки, поступово покращували інші сфери, які були менш головними на початкових етапах розробки.

В основі успішної роботи раціонального уніфікованого процесу розробки програмного забезпечення стоять три центральні елементи, які визначають його головні сторони [21]:

- Базовий набір філософій і принципів успішної розробки програмного забезпечення. Ці філософії та принципи є основою, на якій було розроблено раціональний уніфікований процес.
- Система перевикористання методів та процес будування блоків. Сам процес удосконалюється та покращується через сімейство плагін–методів раціонального уніфікованого процесу, на основі яких можна створювати власні конфігурації для методів і адаптовані процеси під власні потреби.
- Базовий метод і мова визначення процесу. В основі кожного елемента цього процесу лежить уніфіковано архітектурний метод мета–моделі. Ця модель забезпечує мову для опису вмісту методу та процесів. Ця нова мова є об’єднанням різних методів і мов розробки процесів, таких як уніфікованої мови моделювання для розробки процесів програмного забезпечення.

2.2 Математичне представлення задачі

Задача, яка постала перед нами, повинна бути подана у вигляді реалізованої інформаційної системи. Тому з необхідних урахувань розв'язку цієї проблеми, було представлено її імплементацію у вигляді функціональної процедери, яка буде обчислювати коефіцієнти ідентифікації амплітудних елементів фазової швидкості під час поширення хвилі анормальних неврологічних рухів b_k, де $k = \overline{1, n_1 + 1}$ є функцією часу і умов. На основі обрахованих результатів обчислень для кожного малого у розмірі k-го сегменту, де $k = \overline{1, n_1 + 1}$, переформатовується у пряму крайову задачу, як систему – однорідних початково-крайових задач для послідовних малих у розмірі сегментів анормальних неврологічних рухів з умовами.

$$\frac{\partial^2}{\partial t^2} u_k(t, z) = b_k^2 \frac{\partial^2}{\partial z^2} u_k + S_k^*(t, z), \tag{1}$$

де початкові умовами:

$$u_k(t, z)|_{t=0} = 0, \frac{\partial u_k}{\partial t}|_{t=0} = 0, k = \overline{1, n_1 + 1} \tag{2}$$

Крайові умови на кожному з малих у розмірі сегментів анормальних неврологічних рухів за віссю z.

$$u_{k-1}(t, z)|_{z=l_{k-1}} = U_{L_{l_{k-1}}}, u_k(t, z)|_{z=l_k} = U_{l_k}, k = \overline{1, n_1 + 1} \tag{3}$$

Після задання початкових рівнянь, потрібно вибрати функціоналу-нев'язки. Для початку потрібно припустити, що фазові швидкості хвилі анормальних неврологічних рухів b_k, $k = \overline{1, n_1 + 1}$ крайової задачі, де (1)-(3) функції є невідомими від часу. У відомих координатних величинах місцезнаходження пера $u_{,_k}(t,\ z)$ в координатах обстеження сегментів анормально неврологічних рухів $\gamma_k \subset \Omega_k$, $k = \overline{1, n_1 + 1}$.

$$u_k(t, z)|_{\gamma_k} = U_{l_k}(t, z)|_{\gamma_k} \tag{4}$$

Звідси початково-крайова задача (1)-(3) має можливість бути представленою для кожної точки з осі z для кожного малого у розмірі k_1-го сегмента шляху анорально неврологічних рухів. Воно матиме сенс у визначенні функцій $b_k \in D$, де $D = \{v(t, z)\colon\ v|_{\Omega_{k_1 T}} \in C(\Omega_{k_1 T}),\ v > 0,\ k = \overline{1, n_1 + 1}\}$.

Для знаходження розв'язку відхилення функціоналу-нев'язки від своїх слідів на $\gamma_k \in \Omega_{k_1}$ буде використовуватися рівняння в наступному вигляді.

$$J_k(b_k) = \frac{1}{2}\int_0^T \left(\left\|u_k(t, z, b_k) - U_k^*\right\|\right)^2 dt \tag{5}$$

Після проробленої роботи потрібно виконати математичне підтвердження способу розв'язання задачі. Щоб це здійснити, потрібно виконати скінченні інтегральні перетворення Фур'є. Тому після перетворень отримуємо єдине рівняння вихідної крайової задачі (1)-(3) у класичній формі.

$$u_k(t,z) = \int_0^t \int_{l_{k-1}}^{l} H_k^1(t-\tau, z, \xi) S_k^*(\tau, \xi) d\xi d\tau + \\ + \int_0^t (H_k^{21}(t-\tau, z, l_{k-1}) U_{l_{l_{k-1}}} - H_k^{22}(t-\tau, z, l_k) U_{l_{l_k}}) d\tau \quad (6)$$

В цьому рівняння матриці впливу матимуть наступний вигляд:

$$H_k^1(t-\tau, z, \xi) = \frac{2}{\Delta h} \sum_{m=0}^{\infty} \frac{\sin b_k B_m (t-\tau)}{b_k B_m} \sin B_m(\xi - l_{k-1}) \sin B_m(z - l_{k-1}),$$

$$H_k^{21}(t, z, l_{k-1}) = \frac{2b_k}{\Delta h} \sum_{m=0}^{\infty} b_k B_m t \sin B_m(z - l_{k-1}), \quad (7)$$

$$H_k^{22}(t, z, l_k) = \frac{2b_k}{\Delta h} \sum_{m=0}^{\infty} \sin(b_k B_m t)(-1)^2 \sin B_m(z - l_{k-1})$$

Розв’язок (6) задачі (1)-(3) після багатьох етапів перетворень і перенизки трансформувань можна перетворити отримане рівняня у зручний та зрозумілий формат ітераційних обчислень, щоб можна було використовувати його в процедурних функціях визначення коефіцієнтів ідентифікації. Також можна виконати попередню підстановку виразів і їх інтеграцію у формулі (6), після чого спростити отримане рівняння. В результаті рівняння (6) зводиться до класичних алгебраїчних виразів, які можна легко використати при реалізації процедури обчислення коефіцієнтів ідентифікації.

$$u_k(t,z) = \frac{2}{\Delta h}\sum_{m=0}^{\infty}\frac{1-cos(b_k B_m t)}{B_m} sin\, B_m(z-l_{k-1})(S_k^* \frac{1}{(b_k B_m)^2} * \\ * ((-1)^m - 1) + U_{l_{l_{k-1}}}(1-(-1)^m \frac{U_{l_{l_k}}}{U_{l_{l_{k-1}}}})) \quad (8)$$

Пройшовши чи малу кількість етапів, перейдемо до формування рівняння аналітичних виразів градієнтів функціоналу-нев'язки. Виконавши перетворення над приростами функціоналу-нев'язки, одержуємо формулу, що встановлює зв'язок між прямою крайової задачею (1)-(3) і спряженою крайовою задачею ідентифікації, появляється можливість отримати явні вирази компонентів ґрадієнта функціонала-нев'язки. Для задачі функціонального обчислення коефіцієнтів ідентифікації, маємо наступні формули для компонентів градієнтів функціоналу нев'язки, як функції від часу:

$$\nabla J_{b_k}(t) = \int_{l_{k-1}}^{l_k} \phi_k(t,z)\frac{\partial^2}{\partial z^2}u_k(t,z)dz \quad (9)$$

Ітераційний вирази для $n+1$-го кроку для визначення коефіцієнтів ідентифікації та функціональної залежності між ними. Слідуючи використанню методів мінімальних відхилень для визначення функціональної залежності між коефіцієнтами ідентифікації коливального руху сигналу від часу t для кожного k-го сегмента анормальних неврологічних рухів, получається наступне рівняння:

$$b_k^{n+1}(t) = b_k^n(t) - \nabla J_{b_k}(t) \frac{\left\| u_k(t, \gamma_k, b_k) - U_k^* \right\|^2}{\left\| \nabla J_{b_k}^n(t) \right\|_{\gamma_k}^2}, t \in (0, T), k = \overline{1, n_1 + 1} \quad (10)$$

2.3 Пошук акторів та варіантів використання

Після аналізу предметної області було виділено ряд питань до системи, які вона повинна вирішувати в межах даної роботи. Використовуючи ці питання, потрібно здійснити пошук всіх основних акторів застосунку та варіантів використання для кожного із них.

Розроблювана система передбачатиме наступних акторів:

- Неавторизований користувач
- Медичний працівник

В системі є лише один головний актор – Медичний працівник. Лише він матиме доступ у використанні даної програми. Також було виділено іншого користувача системи – Неавторизований користувач. Цей актор являє собою таких користувачів як: пацієнт та науковий працівник лабораторії чи клініки. Останнє перераховані користувачі не матимуть права використовувати програму та не можуть бути задіяними під час імплементації основної логики системи, тому що встановлення ступеня тремору є відповідальною та важливою справою, тому таке можуть виконувати лише висококваліфіковані доктора або інші працівники медичного закладу. Тому виходячи із цих причин право користування системою мають спеціалісти в області діагностування тремору.

Головними користувачами системи виявлення ступеня тремору є зацікавлені в доцільній, правильній та результативній технічній оцінці усіх ознак, отриманих під

час діагностування, для виявленні ступеня тремору або стану пацієнта. Ось чому головний користувач буде являти собою адміністратора системи і буде основним експертом при встановленні діагнозу.

Нижче наведено діаграму основних варіантів використання, які користувач буде виконувати найчастіше під час діагностування пацієнта (рис. 2.2).

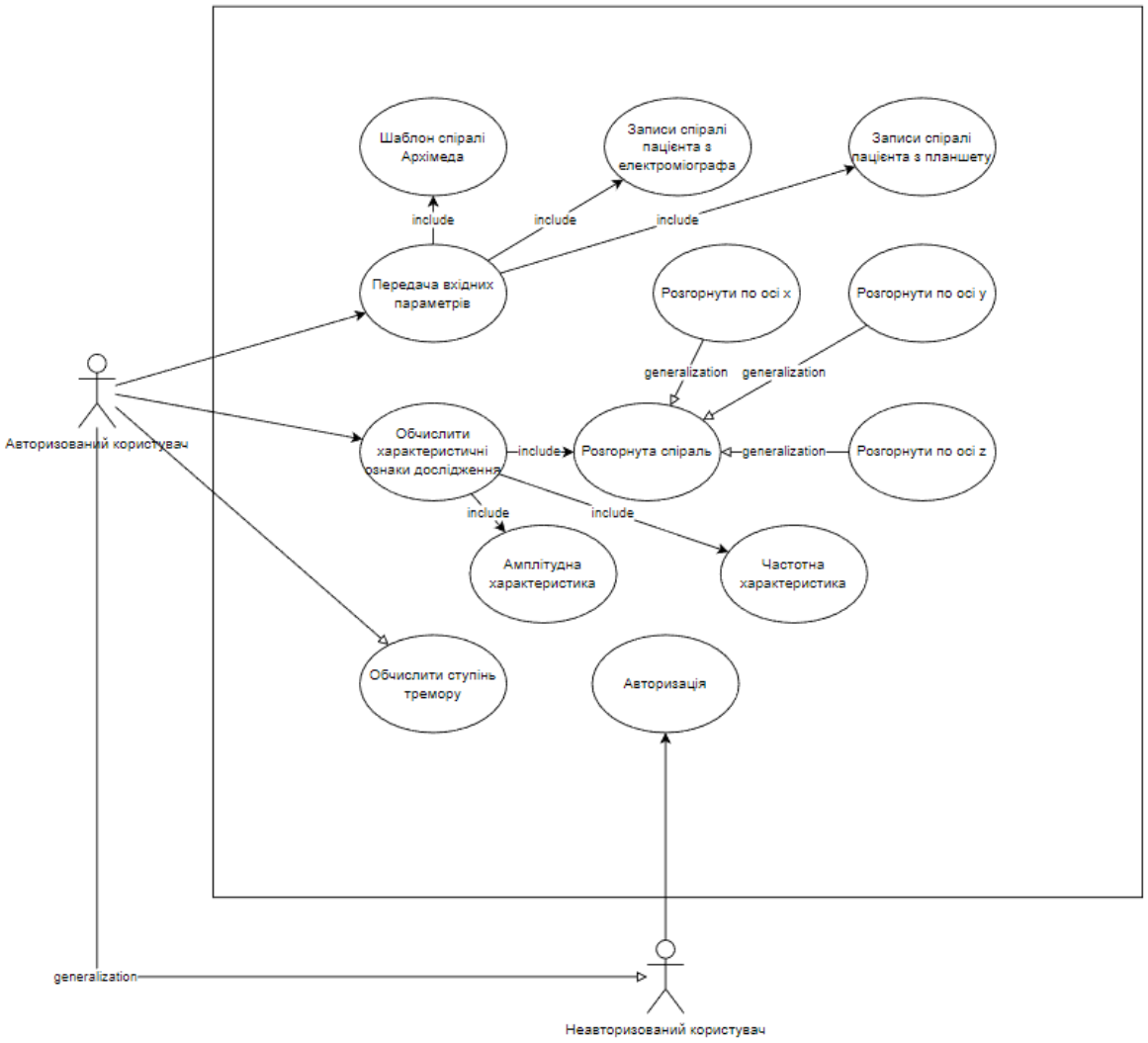

Рисунок 2.2 – Приклад діаграми варіантів використання

Щоб знати, що за користувач знаходиться у системі, йому треба пройти процедуру аутентифікації, яка базується на введенні логіну та паролю від аккаунта. Тож якщо він ще не виконував вказану процедуру, то даний користувач буде

Неавторизованим користувачем. У системі для такого актора вказаний лише один варіант використання і це авторизація.

Якщо взяти користувача з іншою ролю, то йому потрібно розлогінитись із системи і тоді він стане Неавторизованим користувачем.

Для актора Медичний працівник кількість варіантів використання є більшою, ніж у Неавторизованого користувача. Ось перелік варіантів використання, які передбачені для Медичного працівника:

- Авторизуватись
- Передати вхідні параметри (шаблон спіралі Архімеда, записи спіралі пацієнта з електроміографа та планшету)
- Змінити передані вхідні параметри
- Встановити дефолтні вхідні параметри
- Обчислити характеристичні ознаки дослідження
- Переобчислити характеристичні ознаки дослідження
- Розгорнути спіраль
- Отримати частотну та амплітудну характеристики
- Обчислити ступінь тремору
- Розлогінитись

Як можна побачити, що майже всі можливі функції у системі може виконувати лише один актор, тобто Медичний працівник, який є авторизованим у системі. Також додатковою особливістю для авторизованого користувача буде можливість попередню встановлювати дефолтні значення для всіх параметрів моделювання.

Розглянутий застосунок буде розроблений для використання в науково-дослідних лабораторіях. Відповідно до вимог, програма буде реалізована веб-застосунком для отримання результатів розрахунків на основі прогнозних моделей, а деякі з її модулів будуть реалізованими, як бібліотеки. Система розрахована на багатокористувацькі можливості, тому потрібно продумати або використати існуючі архітектурні рішень, які є часті при розробці

багатокористувацьких систем. Хоча користувачів не повинно бути багато, так як застосунок націлений на вузькоспеціалізований сегмент всієї аудиторії. Тож далі буде перелічено основні складності під час проектування та реалізації системи:

- Процес авторизації
- Хешування паролів для покращення безпеки
- Зберігання файлів
- Збереження результатів розрахунків

2.3.1 Детальний опис варіантів використання

Всі варіанти використання, які представлені на рисунку 3.1, затвердились під час фази проєктування рішень для обробки інформації та розрахунку оцінки ступеня тремору.

Тож розглянемо основні варіанти використання в більш детальному вигляді:

- “Авторизуватись”. Варіант використання стоїть на шляху у користувачів, які хочуть увійти у систему та хочуть використовувати реалізований функціонал. Під час виконання цього варіанту використання, користувач вводить дані для входу у систему, а система перевіряє чи надіслані дані введені коректно та чи має право користувач увійти у систему.
- “Передати вхідні параметри”. Щоб виконати розрахунки, потрібно передати всі вхідні дані, які необхідні для виконання розрахунків оцінювання ступеня тремору. Можна виділити, що головними даними для проведення розрахунків є результати отримані під час діагностування пацієнта. Результати представляють собою файл із записами виконаними цифровим планшетом та електроміографом. Після вибору результатів дослідження вказують додаткові параметри для виконання розрахунків. Обов’язкові параметри є порт сигналу

електроміографа та кількість сегментів. Для використання записів файлів в методах для розрахунків, потрібно кожен з них представити у форматі об’єкта. Важливою можливістю є змінення вхідних параметрів для перерахунку розрахунків обчислень.

- “Обчислити характеристичні ознаки”. Після подання всіх необхідних вхідних параметрів, можна виконати даний варіант використання. Він обробляє передані вхідні параметри відповідно до створених математичних методів та виконує побудови різноманітних графіків та обчислення проміжних значень для визначення ступеня тремору. Ось доступні можливості цього варіанту використання: отримання розгорнутих модей по вказаній змінній у часі, обчислення частотної та амплітудної характеристичних ознак.
- “Обчислити ступінь тремору”. Даний варіант використання виконується після виконання проміжних обчислень та дає можливість за допомогою розроблених математичних методів перетворення і статистично вибраних змінних обчислити значення для формування оцінки ступеня тремору.

Усі варіанти використання впроваджуються у розроблювальну програму, яка буде інтегрованою для комплексну діагностичної систему ступеня тремору з подальшим утворенням екземплярів реалізованих класів і викликів їх методів, що містять інтегровані функції в класи.

2.4 Застосування шаблону розробки застосунку

Процес вибору архітектури та процесу розробки є складною справою та залежить від багатьох параметрів від набору команди до змін середовища.

В наш час шаленими темпами росте необхідність використання інтернету та використанню його у всіх справах, тому все більше розробників створюють веб-застосунки. Веб-застосунок – це клієнт-серверна програма, яка (як правило)

використовує веб-браузер, як клієнт. Браузери надсилають запити на сервер, а сервер генеруватиме відповіді та повертатиме їх браузерам. Він відрізняються від старих клієнт-серверних програм тим, що використовують загальну клієнтську програму, а саме веб-браузер (рис. 2.3).

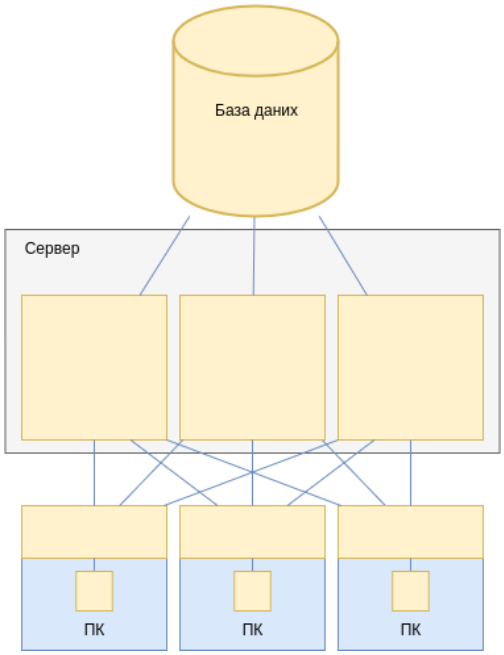

Рисунок 2.3 – Приклад схеми роботи веб-застосунку

Існують важливі переваги використання веб-браузерів як клієнтів:

1. Веб-браузери є всюдисущими. Вони присутні практично на кожному робочому столі і можуть використовуватися для взаємодії з багатьма різними веб-додатками. Немає необхідності встановлювати кілька спеціалізованих програм-клієнтів на робочий стіл, що значно зменшує головний біль з обслуговування

2. Браузери забезпечують механізми безпечного завантаження та виконання більш складних клієнтів (наприклад, аплетів, компонентів ActiveX і програвачів флеш-фільмів), коли потрібні додаткові функції, які одні браузери не можуть забезпечити

Хоча веб-застосунки вписуються в парадигму клієнт-сервер, вони виходять за її межі. Взаємодія між браузерами та серверами – це ще не все, що є в веб-програмі. Самі веб-сервери працюють як «клієнти», коли вони взаємодіють з іншими внутрішніми серверами, включаючи бази даних і застарілі системи. Таким чином, архітектуру більшості веб-додатків краще описати як багаторівневу архітектуру. У той час як простіші клієнт-серверні програми мають лише два рівні з різними ролями, такими як клієнт і сервер, багаторівневі програми мають, як випливає з назви, кілька рівнів, кожен з яких може діяти як клієнт, так і сервер під час обміну даними зі своїми сусідами.

У багаторівневих програмах кожен рівень представляє прикладний рівень: браузер, веб-сервер, сервер прикладних програм, база даних/застаріла система тощо. Нижній рівень – це веб-клієнт, як правило це місце займає зазвичай браузер або інтелектуальний агент), який ініціює обробку, надсилаючи запит на веб-сервер.

Кожен набір суміжних рівнів представляє пару клієнта та сервера, і кожен проміжний рівень може діяти як клієнт або сервер, залежно від того, з ким із сусідів він взаємодіє. Наприклад, так само, як браузер підключається до веб-сервера, щоб зробити запит, частина програми, що виконується на веб-сервері, може підключатися до рівня бізнес-логіки або моделі даних, діючи як клієнт для служб цього рівня.

Спираючись на вище перераховане було обрано проєктування архітектури системи за таким принципом.

2.5 Розділення інформаційної системи на рівні

Для забезпечення безпечного доступу до даних, розроблену систему було поділено на декілька рівні. Кожен із рівнень покриває свою область проблематики. В

такому випадку кожен рівень відповідає лише за свої обов’язки і при потребі звертається до інших рівнів.

Провівши аналіз предметної області, було виділено те, що система буде мати 5 рівні (рис. 2.4). Серед рівнів можна знайти рівні інтерфейси. Такі рівні використовуються для того, щоб вищі рівні могли спілкуватися із наступним рівнем лише за вказаними сценаріями.

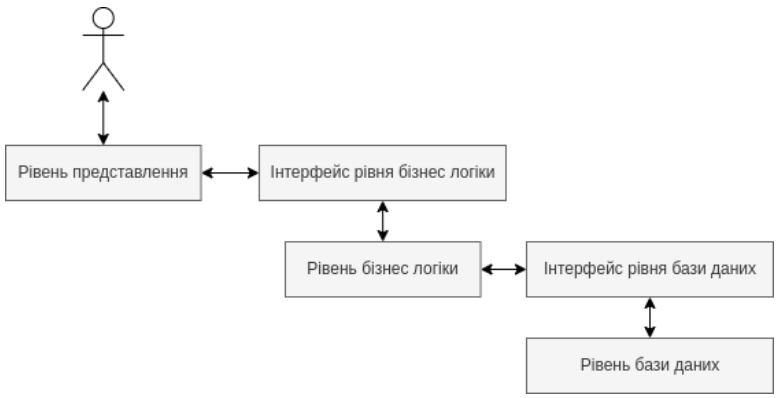

Рисунок 2.4 – Рівні інформаційної системи

Тепер виконаємо детальний опис кожного рівня. Рівень представлення відображає інтерфейс користувача та робить взаємодію користувача більш простою. Цей типовий рівень діаграми архітектури веб-додатку містить компоненти інтерфейсу користувача, які обробляють і відображають дані для користувачів. Існують також компоненти процесу користувача, які встановлюють взаємодію користувача. Цей рівень надає всі необхідні дані на стороні клієнта.

Наступним рівнем є інтерфейс рівня бізнес логіки. Цей рівень використовується для покращення спілкування користувацького рівня із рівнем бізнес логіки. Він задає яким чином може використовувати бізнес логіку.

Рівень бізнес логіки відповідає за належний обмін даними. Цей рівень визначає логіку бізнес-операцій і правил. Цей рівень завершує обробку запитів

клієнтів із браузера та керує шляхами доступу до цих даних. Отже, вказівки, за якими серверна частина отримує дані та запити клієнтів, закодовані на бізнес-рівні.

Наступним рівнем є інтерфейс рівня бази даних. Цей рівень використовується для покращення спілкування рівня бізнес логіки із рівнем бази даних. Він обробляє всі запити бізнес логіки, встановлює з'єднання із базою даних, обробляє помилки та повертає дані у вказаному форматі.

Рівень бази даних є централізованим розташуванням, яке отримує всі виклики на маніпулювання збереженими даними і забезпечує доступ до бази даних програми. Рівень тісно пов'язаний із бізнес-рівнем, тому бізнес логіка знає, з якою базою даних вона має спілкуватись, а процес отримання даних є більш оптимізованим. Інфраструктура зберігання даних включає сервер і систему керування базою даних.

2.6 Представлення системи в абстрактному вигляді

Після проведення аналізу всієї області системи, можна приступати до абстрактного представлення веб-застосунку. Процес розробки системи базувався на об'єктно-орієнтованому підході. Основною перевагою використання даного підходу є те, що він відображає компоненти реального світу у формі класів та їх полів, а їх взаємодію – у формі методів.

Об'єктно-орієнтоване програмування наразі є гнучким шаблоном програмування, який використовує кожен розробник програмного забезпечення чи програміст. Ця концепція ООП застосовується як до розробки веб-додатків, так і до розробки автономного програмного забезпечення. Багато мов програмування та фреймворків, зокрема для веб-додатків, тепер структуровані разом із технікою ООП. Зрозуміло, що всі останні фреймворки для розробки веб-додатків зберігають міцну структуру об'єктно-орієнтованих методів і шаблонів проектування.

Причинами використання ООП для розробки веб-застосунку були ряд позитивних сторін:

- Застосування повторного використання функції дизайну та коду
- Покращення ремонтопридатність веб-додатку. Оскільки об'єктно-орієнтований підхід розбиває складні програми на окремі модулі. Функції можна легко видалити або оновити на веб-сторінці, не впливаючи на інші функції
- Зменшення витрат та часу на розробку веб-додатку. Завдяки наявності функціональних кодів (наприклад, можна отримати код JavaScript, який відображає час у додатку та інтегрувати його в програму) та багатої бібліотеки об’єктно-орієнтованого програмування, є скорочення часу та витрат на розробку веб-додатків
- Покращення обмеження доступу до програми від неавторизованих осіб. Концепція інкапсуляції в об’єктно-орієнтованому підході дозволяє веб-програмі вказувати права доступу до функції або вмісту веб-програми
- Правильне обчислення та організація складних завдань. Об'єктно-орієнтований підхід допомагає визначити обов'язки та розподіл цих обов'язків між взаємодіючими об'єктами. Це допомагає вирішувати складні завдання та створює можливість для багатозадачності.

Базуючись на раніше прийнятих рішеннях, виконаному аналізі та створеній архітектурі, було розроблено програмне забезпечення із застосуванням усіх можливостей, які надає об’єктно-орієнтоване програмування, тобто використання класів, інтерфейсів, екземплярів класів, імплементації інтерфейсів та успадкування класів. Використовуваним у системі об’єктно-орієнтованим підхід передбачає процес реалізування всіх основних компонентів (класів, інтерфейсів, їх взаємодій) та усіх необхідних діаграм, побудованих на основі реалізованого UML. Такий підхід було обрано через специфіку предметної області, необхідність удосконалення системи та складність оцінки всього обсягу та структури системи на початкових етапах розробки.

UML – це мова візуального моделювання загального призначення, яка використовується для визначення, візуалізації, конструювання та адміністрування артефактів програмної системи [16]. Вона фіксує рішення та розуміння систем, які необхідно створити. Ця мова використовується для розуміння, проектування, перегляду, налаштування, підтримки та контролю інформації про таку системи. Вона призначена для використання з усіма методами розробки, етапами життєвого циклу, областями застосування та носіями. Мова моделювання призначена для уніфікації минулого досвіду щодо методів моделювання та включення поточних найкращих практик програмного забезпечення в стандартний підхід. UML включає семантичні концепції, нотацію та вказівки. Вона має статичну, динамічну, екологічну та організаційну частини. Вона призначена для підтримки інтерактивних інструментів візуального моделювання, які мають генератори коду та авторів звітів. Специфікація UML не визначає стандартний процес, але призначена для використання в ітераційному процесі розробки. Вона призначена для підтримки більшості існуючих процесів об'єктно-орієнтованої розробки. Отже ця мова є нероздільною частиною раціонального уніфікованого процесу розробки програмного забезпечення.

Розібравшись із вибором підходу розробки, потрібно виконати побудову діаграм спираючись на опрацьовану інформацію до цього моменту часу. Отже для початку потрібно оприділити всі класи, інтерфейси та взаємодію між ними, іншими словами спроектувати діаграму класів системи.

Використовуючи діаграми класів, можна візуалізувати класи, які утворюють більшість об’єктів у нашій задачі. Крім класів, ми також можемо представляти їхні властивості, методи та спосіб взаємодії інших класів з ними. Ці діаграми належать до мови, відомої як уніфікована мова моделювання.

Діаграма класів складається з набору класів та їхніх зв’язків. Крім класів, ми також можемо представити відносини між ними за допомогою нотації UML. Найпоширеніші типи зв’язків: композиція, агрегація та успадкування [18].

Діаграми класів у завданні аналізу вимог намальовані абстрактним способом і відображають лише об’єкти у веб-сервісах та те, як вони пов’язані один з одним за допомогою різних типів зв’язків. На рисунку нижче вказана початкова архітектура зразка веб-застосунку оцінювання ступеня тремору (рис. 2.5).

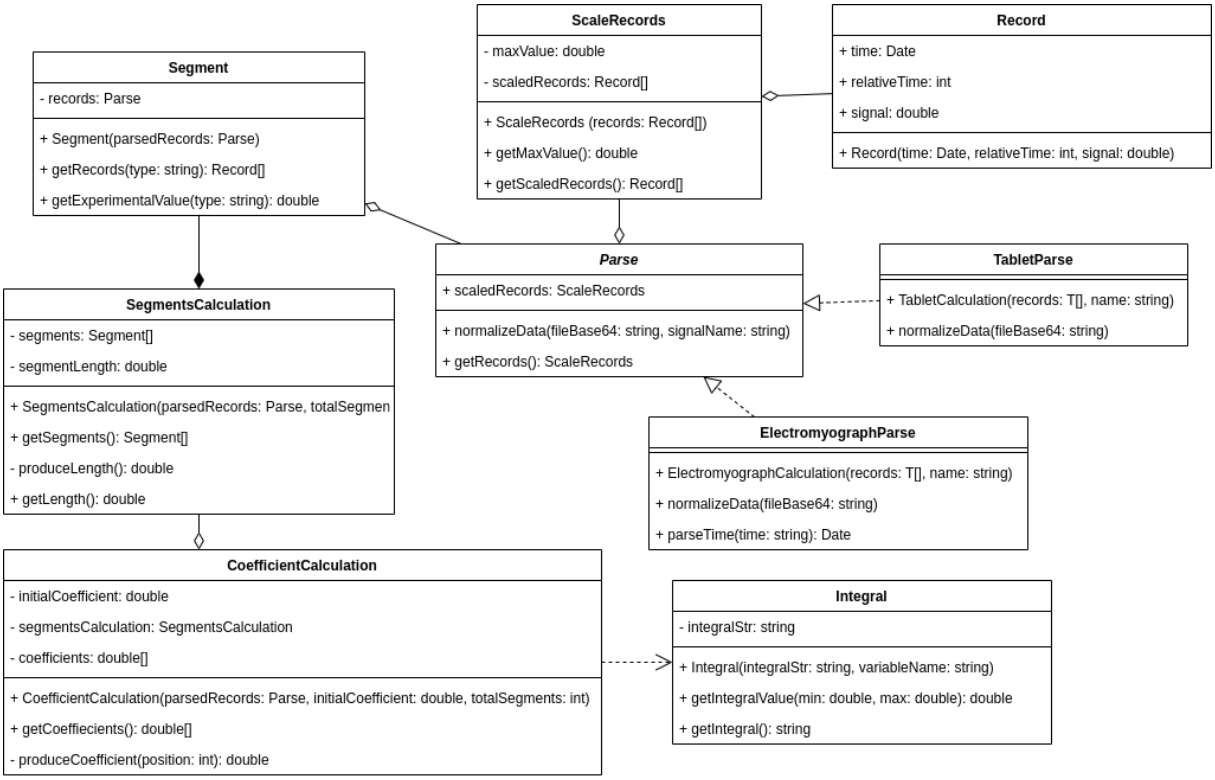

Рисунок 2.5 – Приклад діаграми класів системи оцінювання ступеня тремору

З рисунку вище продемонстровано класи, які відповідають за обчислення ступеня тремору. В основі стоїть Parse інтерфейс для класів, які будуть обробляти дані, які міститимуться у прокинутих вхідних файлах. Даний елемента в діаграмі класів являє собою загальне представлення способу взаємодії системи з вхідними текстовими файлами. Користувачам, які стикаються із таким представленням, буде легко зрозуміти, як можна взаємодіяти із цим інтерфейсом і як інтерфейс взаємодіє з навколишніми компонентами. В цьому класі міститься один основний метод, який потрібно буде реалізувати у всіх класам, які його наслідують. Цей метод виконується

для нормалізування даних та для можливої фільтрації некоректних даних. Під час процедури нормалізації записи з текстового файлика повинні обробитись і податися у вигляді об’єктів відповідних класів. Класи, які будуть реалізовувати метод нормалізації, можуть використовувати допоміжні методи, щоб не нагружати великою кількістю логіки та перевірок метод нормалізації.

У системі є ElectromyographParse, TabletParse класи, які реалізують абстрактний Parse клас. Кожен з цих класів містить реалізований метод normalizeDate та додаткові методи, які допомагають у процесі нормалізації та роблять код більш читабельним для людини. В загальному ці класи є службовими і використовуються для підготовчих дій. Вони Перетворюють записи текстового файлу у більш зручний формат для працювання з ними у системі під час виконання розрахунків та алгоритмів. В додачу кожен з цих класів буде дотримуватись Singleton патерну. Паттерн Singleton є статистичним патерном, тому для якої змінної існує лише один екземпляр, незалежно від того, скільки екземплярів класу існує. Звідси виходить, що створенні класи будуть являти собою глобальні змінні середовища, доступ до яких буде дуже зручним і ефективним.

Клас ScaleRecords виконуватиме масштабування вхідних даних. Він повинен створювати екземпляр класу, який міститиме поле масштабування, максимальне значення поля із всіх значеннях, а для створення йому потрібно буде передати записи представлені у вигляді об’єктів та поле за яким відбуватиметься масштабування. Цей клас надаватиме можливість отримати максимальне за модулем значення та масштабовані значення. Це клас є ключовим у сумісності даних отриманих від планшету та електроміографа під час діагностування пацієнта.

Для реалізації математичних алгоритмів використовуються SegmentsCalculation, Integral класи. Ці класи містять методи, які перетворюють, опрацьовують передані їм вхідні параметри. Кожен з класів є допоміжними і являють собою реалізацію певних загальних алгоритмів та математичних функцій. Перед використанням даних класів, потрібно створити їх екземпляри та передати

аргументи для ініціювання параметрів, які використовуються у розроблених алгоритмах.

Клас CoefficientCalculation реалізовуватиме математичні методи, які є розроблені для оцінювання ступеня тремору. Цей клас використовую всередині попередньо описані класи: SegmentsCalculation, Integral. Цей клас матиме конструктор, в який потрібно буде прокинути параметри для нього і для інших класів, які він містить у собі.

Тепер приступимо до детального опису діаграми класів. Перший на черзі стоїть Parse клас. Це є абстрактний клас, в якому потрібно буде реалізувати метод normalizeData. Цей клас використовується для обробки текстових файлів із вхідними даними, які були отримані від електроміографа та цифрового планшету.

У абстрактного Parse класу присутні наступні атрибути:

- scaledRecords – записи з файлу, представлені у вигляді об’єктів та заскейлені до одиничного масштабу

У абстрактного Parse класу присутні наступні методи:

- Parse(fileBase64: string) – конструктор з параметрами, який зберігає дані в базу даних.
- normalizeData(fileBase64: string) – абстрактний метод, який використовуватиметься для приведення записів файлу у представлення їх у вигляді екземплярів класів, які реалізують Parse клас. Для створення об’єктів записів метод використовує Record клас
- getRecords() – публічний метод, який повертає масив масштабованих об‘єктів

Абстрактний Parse клас використовує для зберігання масштабованих об’єктів ScaledRecords клас. Усі класи, які реалізовуватимуть його використовуватимуться на парсингу файлів електроміографа та цифрового планшету. Атрибути цього класу створені із public специфікатором доступу.

Перший із класів, які реалізують Parse клас, є TabletParse клас. Цей клас реалізує абстрактний метод normalizeData і створює масив об’єктів на основі записів файлу. В TabletParse класі присутні лише методі, а атрибути наслідуються від класу батька.

У TabletParse класу є наступні методи:

- TabletParse(fileBase64: string, name: string) – конструктор з параметрами, в якому відбувається нормалізація файлу. Також вхідний файл прокидається до батьківського класу.
- normalizeData(fileBase64: string, name: string) – публічний метод, нормалізує дані отримані від текстового файлу і приводить їх до об’єктів ScaledRecord класу. Під час роботи цього методу відбуваються перевірки читання файлу, коректності даних, їх фільтрації. Для створення об’єктів записів метод використовує Record клас

В TabletParse класі зібрана логіка обробки текстових файлів, яка відповідає стандарту результатів отриманих від цифрового планшету.

Наступний клас, який реалізує Parse клас, є ElectromyographParse клас. Цей клас реалізує абстрактний метод normalizeData і створює масив об’єктів на основі записів файлу. В TabletParse класі присутні лише методі, а атрибути наслідуються від класу батька. Також у цьому класі є додаткові методи для легшої нормалізації даних.

У TabletParse класу є наступні методи:

- ElectromyographParse(fileBase64: string, signalName: string) – констурктор з параметрами, виконується нормалізація даних і ініціюються атрибути. Вхідні параметри передаються в батьківський клас для подальшої їх нормалізації. Також при нормалізації вибирається один із сигналів від електроміограф.
- normalizeData(fileBase64: string, signalName: string) – публічний метод, нормалізує дані отримані від текстового файлу і приводить їх до об’єктів ScaledRecord класу. Під час роботи цього методу відбуваються перевірки читання файлу, коректності даних, їх фільтрації.

- parseTime(time: string) – публічний метод, парсить переданий рядок та створює на його основі об’єкт дати. Метод викидає помилку, якщо рядок невалідний.

В ElectromyographParse класі зібрана логіка обробки текстових файлів, яка відповідає стандарту результатів отриманих від електроміографа.

Тепер розглянемо Record клас, який використовується для представлення записів текстових файлів у вигляді об’єктів.

Record клас містить наступний набір атрибутів:

- time – абсолютна величина часу, коли було здійснено запис даних з пристрою
- relativeTime – відносна величина часу, коли було здійснено запис даних з пристрою
- signal – величина сигналу записаного з пристрою

Record клас містить наступний набір методів:

- Record(time: Date, relativeTime: int, signal: double) – конструктор з параметрами, який ініціює значення об’єкта.
- getTime() – публічний метод, який повертає абсолютний час запису
- getRelativeTime() – публічний метод, який повертає відносний час запису
- getSignal() – публічний метод, який повертає величину сигналу

Всі атрибути Record класу мають private специфікатор доступу, а всі інші його методи public. Використовуючи такий підхід у користувачів не буде змоги змінити значення об’єкта запису після створення, що робить систему більш надійною.

Навколо Record класу є обгортка у вигляді ScaledRecords класу. Цей клас використовує масив об’єктів Record класу та змінює усі значення до масштабу 1. Клас зберігає найбільше значення до масштабування, щоб можна було повернути всі значення в оригінальні їм розміри. Масштабовані значення потрібні для розрахунку коефіцієнтів ідентифікації, тому що дані від електроміографа та цифрового планшету приходять в різних масштабах.

До атрибув ScaledRecords класу належать:

- scaledRecords – масив об’єктів Record класу з масштабованими значеннями
- maxValue – максимальне значення

До методів ScaledRecords класу належать:

- ScaledRecords(records: Record[]) – конструктор з параметрами, який шукає найбільше значення за модулем та ділить усі значення на максимальне і створює нові масштабовані значення
- getMaxValue() – публічний метод, який повертає найбільше значення за модулем
- getScaledRecords() – публічний метод, який масив масштабованих об’єктів записів

Як видно, що всі атрибути ScaledRecords класу мають private специфікатор доступу, а всі інші його методи public. Використовуючи такий підхід у користувачів не буде змоги змінити значення об’єкта запису після створення, що робить систему більш надійною.

Тепер приступимо до розбору класів, які використовуються для обчислення коефіцієнтів ідентифікації. Тому для початку розглянемо Segment клас. Алгоритм ідентифікації коефіцієнтів працює таким чином, що всі записи розбиваються об’єкти записи на вказану кількість сегментів. Отже цей клас міститиме лише певний послідовний кусок об'єктів і виконувати різні дії над ними.

У Segment класі передбачено атрибути:

- records – масив об’єктів Record класу

У Segment класі передбачено ряд методів:

- Segment(records:Record[]) – конструктор з параметрами, який ініціює record атрибут
- getRecords(type: string) – публічний метод, який повертає масив об’єктів Record класу

- getExperimentalValue() – публічний метод, який обробляє масив об’єктів Record класу та повертає скалярне значення. Метод повертає середнє значення сигналу записів

Segment клас має один атрибут з private специфікатором доступу та методи взаємодії із цим атрибутом, які в свою чергу маю public специфікатор доступу.

Далі в системі зроблено обгортку SegmentsCalculation клас. Тут відбувається додаткові обрахунки на основі всіх сегментів. Також використовуються класи, які наслідуються від Parse класу, для отримання записів.

У SegmentsCalculation класі присутні наступні атрибути:

- segments – масив сегментів
- segmentLength – довжина одного сегменту

У SegmentsCalculation класі присутні наступні методи:

- SegmentsCalculation(parsedRecords: Parse, totalSegments: int) – конструктор з параметрами, який шукає час виконання параметру, розбиває передані йому вхідні записи на сегменти і визначає довжину одного сегменту.
- getSegments() – публічний метод, який використовується для отримання всіх сегментів
- produceLength() – приватний метод, використовується для визначення довжини записів
- getLength() – публічний метод, який повертає довжину записів

Як і в попередніх класах в SegmentsCalculation класі атрибути маю private специфікатор доступу. Що до методів, то всі крім одного методу є публічними і лише один із private специфікатором доступу, який використовується для внутрішньої логіки класу.

Ще одним допоміжним класом, який використовуватиметься при обчислені коефіцієнтів ідентифікації, буде Integral клас. Цей клас використовується для отримання невизначеного інтегралу від переданої функції, а також отримання

скалярного значення, якщо потрібно обчислити визначений інтеграл. Цей клас використовує бібліотеки для отримання інтегралів.

До атрибув Integral класу належать:

- integralStr – невизначений інтеграл представлений у вигляді стрічки

До методів Integral класу належать:

- Integral(integralStr: string, variableName: string) – конструктор з параметрами, який бере інтеграл від переданого значення за переданою змінною. Система кидає помилку, якщо вона не може визначити інтеграл від вхідної стрічки
- getIntegralValue(min: double, max: double) – публічний клас, який обчислює визначений інтеграл в переданих межах. Система кидає помилку, якщо мінімальних вхідний параметр більший, ніж максимальний
- getIntegral() – публічний метод, який повертає розрахований невизначений інтеграл

Intergal клас має один атрибут з private специфікатором доступу та публічні методи для обрахунку інтеграла.

На останок переглянемо CoefficientCalculation клас, в якому відбувається підрахунок коефіцієнтів ідентифікації. Цей клас має зв’язок агрегації із SegmentsCalculation класом, тому що багато операцій виконується над сегментами. Також використовується Integral клас для визначення скалярного значення визначеного інтегралу, так як для побудови математичної моделі використовуються складне рівняння з великим спектром математичних прийомів.

До атрибутів CoefficientCalculation класу належать:

- initialCoefficient – початковий коефіцієнт
- segmentsCalculation – об’єкт SegmentsCalculation класу
- coefficients – обчисленні коефіцієнти

До методів CoefficientCalculation класу належать:

- CoefficientCalculation(parsedRecords: Parse, initialCoefficient: double, totalSegments: int) – конструктор з параментрами, він приймає об’єкт класу, який

наслідується від Parse класу, та переробляє його в SegmentsCalculation клас для розбиття записів на сегменти. Також в конструкторі обчислюються коефіцієнти ідентифікації.

- getCoeffiecients() – публічний метод, який повертає обчислені коефіцієнти ідентифікації.
- produceCoeffiecient(position: int) – приватний метод, який обчислює коефіцієнт ідентифікації для певного сегменту, індекс якого прокидається параметром.

Атрибути CoefficientCalculation клас маю private специфікатор доступу. Цей клас має два методи, один з них має public специфікатор доступу, а інший private. Приватний метод використовується для підрахунку коефіцієнта ідентифікації базуючись на попередніх значеннях або на початкових. Початкове значення береться із діапазону від 0 до 1.

Якісна реалізація архітектури для обчислень в майбутньому дозволить легко та швидко вносити зміни в існуючі складні алгоритми та знищить шанс припущення помилки. Тому на цьому кроці потрібно приділити максимум зусиль для досягнення цих цілей.

Крім процесу обчислення, система виконуватиме процес авторизації та перевірятиме права у користувачі на можливість користування даною програмою або деяким спектром можливостей. Нижче на рисунку продемонстровано діаграму класів авторизації, в якій використовуються багато технік, про які буде розказано згодом, та технологій підвищення безпеки користувацьких даних (рис. 2.6).

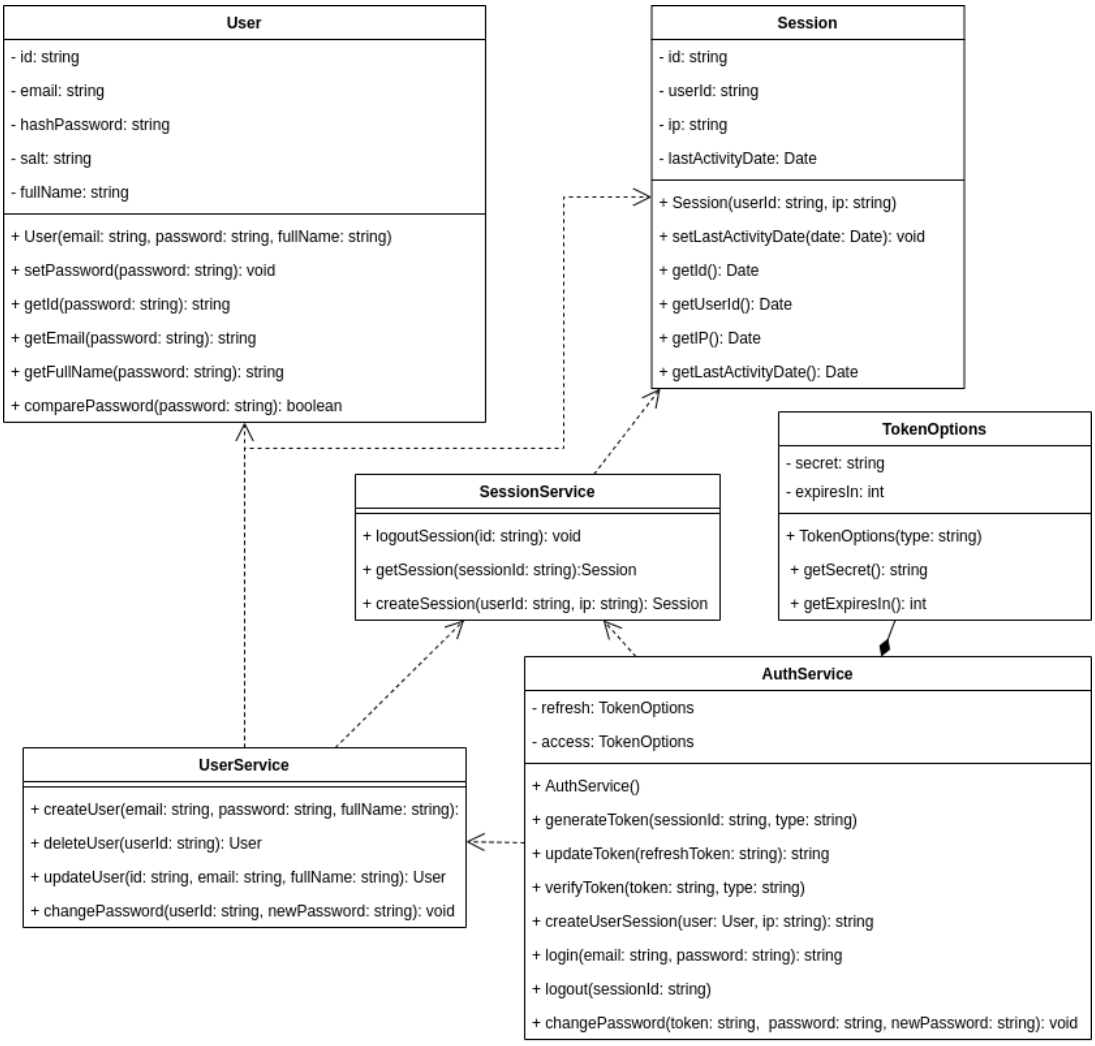

Рисунок 2.6 – Приклад діаграми класів процесу авторизації

Клас User відповідатиме за представлення користувача в розроблюванні системі. Для користувача буде надана можливість створення екземплярів цього класу та набір функцій для отримання інформації про створеного користувача системи. Клас використовується для спілкування з базою даних та бізнес логікою, що дозволяє покращити та виконувати більш безпечні звернення до бази даних.

Клас UserService являє собою проміжний рівень, що забезпечує підвищений рівень безпеки. В цьому класі відбуваються додаткові перевірки розмежування доступу між користувачами. За допомогою методів цього класу користувач може створювати, редагувати, видаляти і читати дані користувача зважаючи на права

доступу. Також використання методів класу UserService використовуватимуться при виконані логіки авторизації користувача.

В системі присутній клас Session, який в свою чергу відповідатиме за представлення сесії в системі. Сесія створюється під час виконання входу користувача у систему, а видаляється при виході користувача із системи. Під час створення сесії створюється набір приватних користувацьких значень, а при видаленні ці дані видаляються або стають некоректними. Цей клас, як і User клас, розміщуватиметься між рівнями бази даних та бізнес логіки та забезпечує коректне читання даних та додаткову обробку помилок.

За аналогією до UserService класу, SessionService відповідатиме за поведінку введення сесій. Клас може створювати, редагувати та видаляти сесії із системи. В даному класі будуть додаткові перевірки під створення нової сесії для користувача. Разі наявності сесії у користувача та коректності і валідності її даних, така сесія буде повторно використовуватись. Для цього потрібно буде просто оновити застарілі поля.

Клас Auth відповідатиме за всі операції, як є пов’язаними із процесами авторизації і аутентифікації. Клас використовуватиме UserService і SessionService класи для реєстрування, входу і виходу користувача з системи. Важливим методом для будь якої системи із обмеженим доступом є зміна паролю користувача. Ситуація коли користувачу забув пароль від свого акаунту в системі є досить частим. Однак в сучасних умовах тенденція відбування такої ситуації знижується за рахунок встроєних механізмів керування паролями та допоміжними програмами для нотування або зберігання паролів.

Після короткого опису кожного з класів області авторизації і аутентифікації потрібно провести детальний розбір кожного з класів та описати всі елементи, які вони містять.

Розпочнемо огляд класу User. Цей клас використовується для відображення реального користувача в системи. В даному класі містяться обробки помилок під час

створення, редагування та видалення користувача. Він містить набір наступних атрибутів:

- id – ідентифікатор користувача
- email – електронна пошта користувача, яка використовується при авторизації
- hashPassword – захешований пароль користувача
- salt – випадковий рядок, який використовується при хешуванні паролю
- fullName – повне ім’я користувача

User класу надає користувачу наступні методи:

- User(email: string, password: string, fullName: string) – конструктор з параметрами, зберігає дані в базу даних та створює екземпляр класу
- setPassword(password: string) – публічний метод, що дає можливість змінити пароль користувача. Хешування відбувається з використанням алгоритму bcrypt. Значення salt вибирається випадковим чином і використовується в алгоритмі хешування нового паролю. Після процесу хешування навіть при однакових паролях результат хеш–алгоритму стає непередбачуваним. Після отримання результатів хешування метод встановлює нові значення для salt і hashPassword атрибутів класу
- getId() – публічний метод, що дає можливість отримати ідентифікатор користувача
- getEmail() – публічний метод, що дає можливість отримати електронну пошту користувача
- getFullName(fullName: string) – публічний метод, що дає можливість отримати повне ім’я користувача
- comparePassword(password: string): boolean – публічний метод, що перевіряє переданий пароль на відповідність тому, що зберігається в базі даних. Захешированый пароль користувача розшифровується і порівнюється із вхідним паролем. В разі співпадіння метод поверне істину

Кожен атрибут клас User оголошений із private специфікатором доступу, а всі його методи – public. Використання такого підходу заберає можливості користуванню атрибутами класу напряму та заборонить їх поширення для класів, які наслідуватимуть User клас. Для доступу до них користувач матиме змогу використовувати публічні методи та отримувати вміст атрибутів або змінювати їх.

Наступний UserService клас являє собою рівень, де зберігатиметься вся бізнес логіка, яка підноситься до сутності користувача. Цей клас використовуватиме User клас для спілкування з базою даних. UserService клас не матиме власних атрибутів та конструкторів, не рахуючи дефолтного, а лише методи для взаємодії з ним. UserService має ряд наступних методів:

- createUser(email: string, password: string, fullName: string) – публічний метод, який створює нового користувача у системі. Метод виконується перевірка прав на створення, а також виконуються перевірки перед створенням користувача. Такою основною перевіркою є унікальність користувача за електронною поштою
- deleteUser(userId: string) – публічний метод, що видаляє користувача із системи. Після видалення користувача, система додатково видаляє усі сесії користувача. Перед видаленням користувача виконуються перевірки на наявність користувача за переданим ідентифікатором та прав доступу.
- updateUser(email: string, fullName: string) – публічний метод, який дозволяє оновити дані користувача. За допомогою цього метода користувач може оновити свою електронну пошту і повне ім’я. Перед оновленням користувача виконуються перевірки на наявністю користувача за переданим ідентифікатором та прав доступу
- changePassword(userId: string, newPassword: string) – публічний метод, який дозволяє оновити пароль користувача. За допомогою цього метода користувач може оновити свій пароль. Перед оновленням користувача виконуються перевірки на наявністю користувача за переданим ідентифікатором, співпадіння паролю та прав доступу

В кожному методі клас UserService використовувався User клас для кращого та надійнішого спілкування з базою даних. Всі методи цього класу маю специфікатор доступу public.

Інший клас, який представляє сесію є Session клас. Цей клас, як і User клас створений для спілкування системи із базою даних. Цей клас містить наступний ряд атрибутів:

- id – ідентифікатор сесії
- userId– ідентифікатор користувача
- ip – IP адреса користувача, з якої було здійснено останній візит у систему
- lastActivityDate – дата останнього візиту користувача

Session класу надає користувачу перелік методів для взаємодії:

- Session(userId: string, ip: string) – конструктор з параметрами, зберігає дані в базу даних та створює екземпляр класу
- setLastActivityDate(date: Date) – публічний метод, що оновлює дату останнього візиту користувача. Перед оновленням користувача виконується перевірка на те, що новий час візиту є більшим, ніж існуючий
- getId() – публічний метод, повертає ідентифікатор сесії
- getUserId() – публічний метод, що дає можливість отримати ідентифікатор користувача сесії
- getIP() – публічний метод, що дає можливість IP адресу під час останнього входу користувача у систему
- getLastActivityDate() – публічний метод, який повертає дату останього входу користувача у систему

Можна помітити, що всі атрибути Session класу мають private специфікатор доступу. Для доступу до атрибутів використовуються методи, які в свою чергу є з публічним специфікатором доступу. В цьому класі є методи відіграють роль обробників повернення та зміни приватних атрибутів.

Тепер розлянемо TokenOptions клас. Цей клас відповідає за комфортного зберігання конфігураційних даних, які використовуватимуться при генерації токенів. Тому це місце має бути добре захищеним. Переглянемо атрибути TokenOptions класу:

- secret – секретний ключ, який використовуватиметься при генерації токена
- expiresIn – час дії токена, після завершення його дії, він стає невалідним і для подальшого використання повинен створитися новий

У TokenOptions класі є лише невеликий набір методів для взаємодії з системою:

- TokenOptions(type: string) – конструктор з параметрами, за вхідним типом визначає, як конфігурувати клас. Поки діються лише два типи: access і refresh. Секретний ключ і термін дії токена у кожного з них є різним.
- getSecret() – публічний метод, який повертає секретний ключ
- getExpiresIn() – публічний метод, який повертає час дії токена

Для кращого захисту секретних даних, доступ до них буде обмеженим, тобто атрибути будуть приватними, і спроба їх отримання буде доступна лише за допомогою публічних методів класу. Користувач не може передати свої дані у конструктор класу, тому що ці дані беруться із глобальних змінних програми, які прокидаються при старті програми.

За аналогією до UserService класу, SessionService клас відображатиме рівень бізнес логіки для сутності сесії. Для комунікації із базою даних використовуватиметься Session клас. У цьому класі відсутні будь–які атрибути, але присутні лише методи для взаємодії із сесіями. У SessionService класі присутній наступні методи:

- logoutSession(id: string) – публічний метод, який дозволяє користувачу вийти із сесії. Перед виходом із сесії виконуються перевірки на наявність сесії у користувача і прав доступу.

- getSession(id: string) – публічний метод, який повертає сесію відповідно до переданого параметра. Перед поверненням сесії система виконує перевірки на наявність сесії за переданим ідентифікатором та прав доступу.
- createSession(userId: string, ip: string) – публічний метод, який створює сесію для користувача. Метод приймає ідентифікатор користувача і IP адресу створює нову сесію на основі переданих даних. Перед створенням сесії виконується перевірки наявності користувача за переданим ідентифікатором і прав доступу. Якщо в системі вже сесія із такими ж вхідними даними, то нова сесія не створюється, а використовується існуюча.

Всі методи SessionService класу маю public специфікатор доступу. Також кожен із цих методів використовує клас Session для кращої та безпечної взаємодії. Тому що ця ділянка системи є критичною і повинна бути максимально захищеною від стороннього втручання.

Останнім неописаним класом є AuthService клас. В ньому зберігається вся логіка процесу авторизації. Клас використовує TokenOptions клас для зберігання секретних даних, які використовуватимуться при генерації токена. AuthService клас матиме зв’язок композиції з TokenOptions класом. Також цей клас використовуватиме UserService і SessionService класи для управління користувачами і сесіями.

До атрибутів AuthService класу відносяться:

- refresh – секретні дані для генерації токена оновлювача
- access – секретні дані для генерації токена доступу

Також AuthService клас має великий набір методів, які стосуються логіки авторизації системи:

- AuthService() – конструктор без параметрів, виконується ініціювання секретних даних для подальшого використання їх під час генерації токенів
- generateToken(sessionId: string, type: string) – публічний метод, який генерує токен для вказаного типу токена для вказаної в параметрах сесії. Перед

генерацією токена виконуються перевірки на наявністю сесії за переданим ідентифікатором та прав доступу

- updateToken(refreshToken: string) – публічний метод, який оновлює токен оновлювач і доступу. Система перевіряє валідність переданого токена і його дійсність, після генерації, система встановлює новий токен в cookie та повертає токен оновлювач. Перед генерацією нового токена виконуються перевірки на валідність токена та прав доступу
- verifyToken(token: string, type: string) – публічний метод, який виконує перевірку валідності переданого токену та прав доступу
- createUserSession(user: User, ip: string) – публічний метод, що дає можливість створити сесію для користувача та встановлює і повертає токени. Перед створенням сесії виконуються перевірки наявності переданого користувача та прав доступу. Також оновлюється інформація про IP адресу в користувача
- createUserSession(user: User, ip: string) – публічний метод, що дає можливість створити сесію для користувача та встановлює і повертає токени. Перед створенням сесії виконуються перевірки наявності переданого користувача та прав доступу. Також оновлюється інформація про IP адресу в користувача
- login(email: string, password: string) – публічний метод, виконує процес входу користувача у систему. Під час виконання методу створюється сесія та усіх необхідні токени. Під час виконання логінації виконується перевірка наявності користувача у системі, тобто лише неавторизований користувач може виконати цей метод, а також виконується перевірка співпадіння електронної пошти і пароля.
- logout(sessionId: string) – публічний метод, який дає можливість виходу із системи. Цей метод доступний лише для авторизованих користувачів. Під час виконання виходу система видаляє сесію та токени із cookie, а при наступній перевірці токени стають невалідними
- changePassword(token: string, password: string, newPassword: string) – публічний метод, який використовується для зміни паролю користувача. Лише

авторизований користувач зможе змінити пароль. Для цього йому потрібно вказати існуючий пароль і новий пароль. При зміні пароля перевірятиметься співпадіння поточного і нового паролів та прав доступу

Як можна побачити, що AuthService клас використовує UserService та SessionService класи, це дозволить розроблювані системі бути більш гнучкою для додавання змін і пошуку помилок. Клас має атрибути, але кожен з них є приватним і прямого доступу до них не має, їх використовують лише методи цього класу для внутрішньої взаємодії. В свою чергу всі методи цього класу використовують public специфікатор доступу.

2.7 Проєктування бази даних

На сьогодні майже кожна система використовує різні технології, щоб зберігати їх інформацію. Так само із цією системою. Для зберігання даних система використовуватиме документно–орієнтовану базу даних. Такий тип баз даних надають можливості швидкого читання при великій кількості даних та можливості легкого масштабування. Завдяки моделі документа, інформацію можна вставити в один документ, а не покладатися на дорогі операції з’єднання з традиційних реляційних баз даних. Це робить запити набагато швидшими та повертає всю необхідну інформацію за один виклик до бази даних.

Проаналізувавши всі вимоги системи та виділені її сутності, було окреслено діаграму бази даних системи для обчислення коефіцієнтів ідентифікації (рис. 2.7).

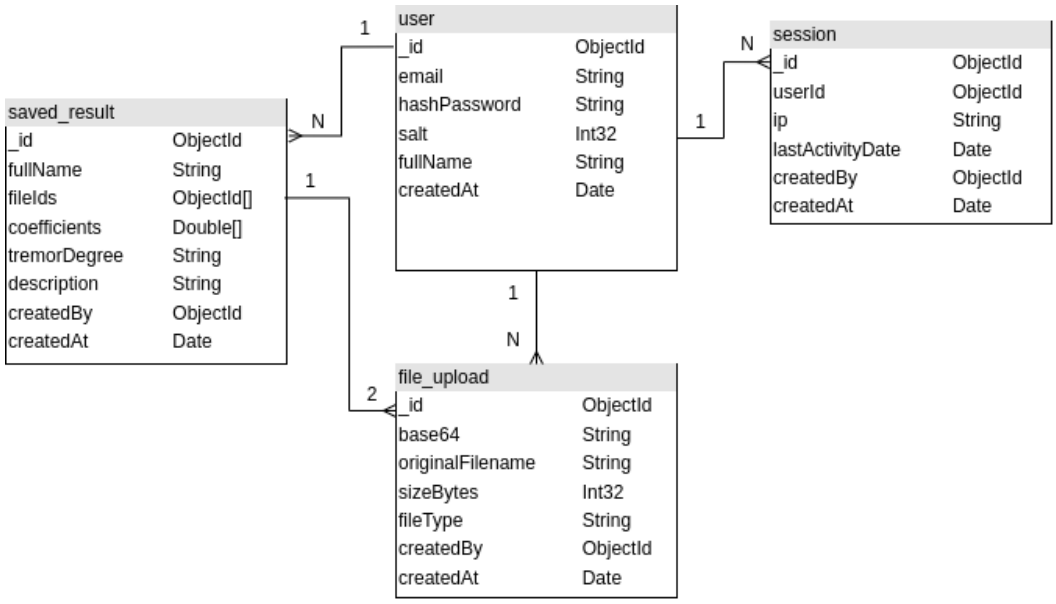

Рисунок 2.7 – Діаграма бази даних

На розробленій діаграмі відношення сутностей, відображено чотири колекції, які уособлюють конкретні сутність в досліджуваній галузі. Між сутностями присутні зв’язки, які вказують на відношення між ними.такий опис дає зрозуміти, скільки файлів користувач може мати або скільки разів виконати збереження розрахунків обчислення.

На діаграмі відношення сутностей присутні наступні відношення:

- У User сутності присутнє відношення один до багатьох з Session сутністю. Користувач може мати декілька сесії, тобто заходити з різних IP адрес. Однак сесія для користувача може бути лише одна.
- У User сутності присутнє відношення один до багатьох з Saved Result сутністю. Користувач може зберегти багато результатів оцінювання ступеня тремору за допомогою обчислених еокфіцієнтів ідентифікації. Однак збережений результам може належати лише одному користувачу.
- У User сутності присутнє відношення один до багатьох з File Upload сутністю. Користувач може завантажити велику кількість файлів. Однак загружений файл може належати лише одному користувачу.

- У Saved Result сутності присутнє відношення багато до багатьох з File Upload сутністю. Збережені результати посилаються лише на два завантажених файли. Однак завантажений файл може належати лише одному збереженому результатові.

Перейдемо до детального опису кожного документа бази даних. Почнемо з документа, який являє собою відображення користувача. User колекція зберігає документи про користувача системи.

У User документі міститься такий набір полів:

- _id – ідентифікатор користувача
- email – електронна пошта користувача
- hashPassword – захешований пароль користувача
- salt – випадковий рядок, який використовується при хешуванні та розшифруванні паролю користувача
- fullName – повне ім’я користувача
- createdBy – ідентифікатор користувача
- createdAt – дата створення користувача

Session колекція, яка зберігає документи про сесії користувачів. Сесія може відпоситися лише до одного користувача і лише одна.

У Session документі міститься такий набір полів:

- _id – ідентифікатор сесії
- userId – ідентифікатор користувача до якого відноситься сесія
- ip – IP адреса останнього входження користувача в систему
- lastActivityDate – дата останнього входження користувача в систему
- createdAt – дата створення користувача

File Upload колекція, яка зберігає документи про файли, які користувач завантажує в систему, щоб виконати оцінювання ступеня тремору. Файли зберігаються лише у випадку їх коректного парсингу.

У File Upload документі міститься такий набір полів:

- _id – ідентифікатор завантаженого файлу
- base64 – вміст завантаженого файлу у форматі base64
- originalFilename – ім’я завантаженого файлу
- sizeBytes – величина розміру завантаженого файлу в байтах
- fileType – тип завантаженого файлу
- createdBy – ідентифікатор користувача, який завантажив файл
- createdAt – час завантаження файлу

Saved Result колекція, яка зберігає документи про збережені результати обчислення коефіцієнтів ідентифікацій. Документи цієї колекції мають посилання на два завантажені файли. Завантажені файли являють відповідно результатами діагностування пацієнта.

У Saved Result колекції документи міститься такий набір полів:

- _id – ідентифікатор результатів обчислень оцінювання ступеня тремору
- fullName – повне ім’я пацієнта
- fileIds – ідентифікатори завантажених файлів пацієнта
- coefficients – обчислені коефіцієнти ідентифікації
- tremorDegree – ступінь тремору вказаний користувачем
- description – опис деталей отриманих розрахунків
- createdBy – ідентифікатор користувача
- createdAt – час проведення обчислень коефіцієнтів ідентифікації

2.8 Діаграма розгортання застосунку

Основна мета діаграм розгортання – описати, як програмне забезпечення доставляється в апаратну систему. Вона показує, як програмне забезпечення

взаємодіє з апаратним забезпеченням для виконання всіх функцій. Це термін, який описує, як програмне забезпечення взаємодіє з обладнанням і навпаки.

Веб-додаток використовує діаграму розгортання UML, щоб показати, як слід розгортати розроблене програмне забезпечення. Він уточнює зв’язок між вузлами системи, що допомагає проекту працювати відповідно до наданого йому дизайну. На діаграмах розгортання показано налаштування вузлів обробки під час виконання та компонентів, які на них розміщені.

На рисунку нижче зображена діаграма розгортання розроблюваної системи оцінювання ступеня тремору (рис 2.8).

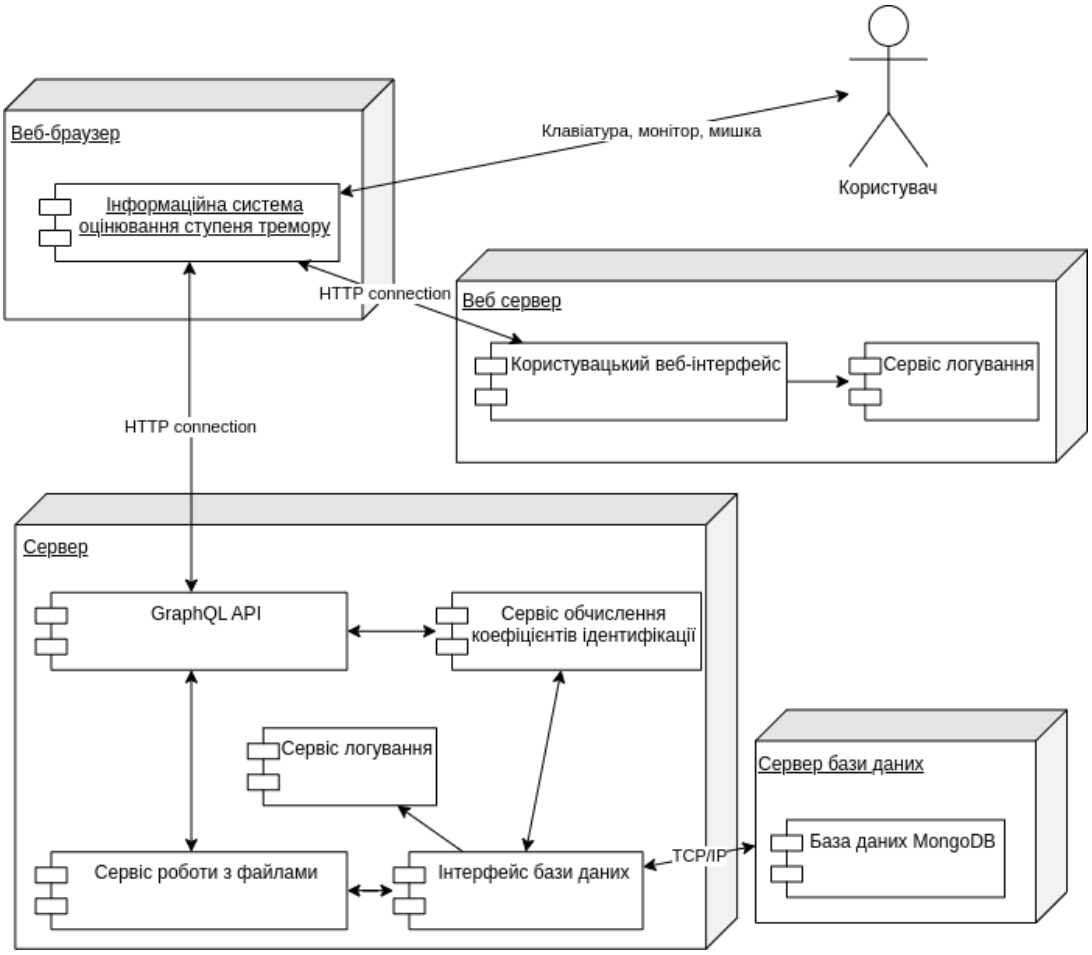

Рисунок 2.8 – Діаграма розгортання системи оцінювання ступеня тремору

Як видно з рисунку, що у розроблюваної системи має багато вузлів. Роботу системи можна відобразити таким чином. Користувач відкриває веб-браузер на своєму персональному комп'ютері та заходить на веб-застосунок оцінювання ступеня тремору за допомогою вхідних пристроїв таких, як клавіатура і мишка. Після переходу на інформаційну систему веб-браузер робить запит на веб сервер та отримує у відповідь розмітку і скомпільований JavaScript код. Веб сервер використовує сервіс логування для зберігання різних логів, які відбуваються під час використання переданого код. Це робиться через обгортку навколо розмітки. Отримавши код, веб браузер малює початковий контент.

Намалювавши початковий контент веб браузер може виконати запити на сервер, щоб отримати певні дані. Наприклад користувач зайшов у систему і хоче перейти до інтерфейсу із обчисленням тремору. Однак система робить запит на сервер, щоб перевірити чи користувач має права доступу на виконання такого запиту.

Для спілкування зі сервером використовується GraphQL API. Такий спосіб організації доступу до даних найкраще підходить для спілкування між веб браузером та сервером. Всередині сервера GraphQL API може використати будь-який внутрішній сервіс вузол, щоб виконати необхідні дії. Так в сервері є сервіси авторизації, роботи з файлами та обчислення коефіцієнтів ідентифікації. Якщо у цих сервісів виникає потреба у звернені до бази даних, то вони звертаються до інтерфейсу бази даних. Також для логування даних використовується відповідний сервіс. Він зберігає усі повідомлення про помилки.

В свою чергу інтерфейс бази даних відправляє запити до бази даних, щоб отримати, видалити або змінити дані.

3 РОЗРОБКА ТА ТЕСТУВАННЯ

3.1 Вибір мови програмування та технологій розробки

Під час вибору засобів реалізації застосунку, який буде давати оцінку ступеня тремору, потрібно звернути особливу на існуючі бібліотеки, які дозволяють ефективно та точно працювати і обробляти вхідні параметри. Так як під час фази проєктування було обрано багаторівневий веб-застосунок, то для початку варто встановити за допомогою яких засобів буде імплементовано клієнтський рівень. Клієнтський рівень у даній системі являє собою веб-браузер, який можна відкрити на великому різноманітні платформ від персонального комп'ютера до телефона. Як відомо, що ключовим способом відображення інформації у веб-браузері є використання мови розмітки HTML, а для кращого відображення використовується CSS3. Однак використання лише цих засобів надасть сторінці красивого та статичного вигляду. Саме в таких випадках використовують JavaScript для інтерактивного відображення елементів.

Також раніше зазначалось, що при розробці програмного забезпечення потрібно буде використовувати об'єктно-орієнтований підхід. Звідси випливає, що вибрана мова програмування матиме та дозволить нам виконувати представлення системи об'єктно-орієнтованому підході. Однак більшість розробників асоціюють об'єктно-орієнтоване програмування з мовами, які є загальноприйнятими у такому розумінні, як-от C++ і Java, які базують об'єктно-орієнтоване програмування навколо класів. Перш ніж користувач зможете щось робити цими мовами, йому потрібно створити клас, навіть якщо він просто пишете просту програму командного рядка. Саме тому цю роль отримує JavaScript [11].

Використання сирого JavaScript в веб-браузері є доволі проблематичним у відображенні складних компонентів, які повинні бути розробленими під час розробки графічного дизайну застосунку. Сучасний світ розробки графічних

інтерфейсів у браузері не стоїть на місці, ось чому на сьогодні є багато розроблених бібліотек для вирішення та полегшення розробки інтерфейсів. Спираючись на швидкість та якість розробки клієнтської сторони було прийнято рішення використовувати бібліотеку React, розроблену компанією Facebook.

Сама React бібліотека є невеликою бібліотекою і використовується лише для однієї частини роботи, тобто для швидкого і економного перемальовування графічного інтерфейсу. Компоненти інтерфейси у системі представляються класовими компонентами. Кожен такий компонент повинен наслідувати Component клас та реалізовувати метод render. Метод render відповідає за процес відмальовування компонента в інтерфейсі. Також класові компоненти надають велику кількість методів життя компонента. Життєвий цикл компонента складається з методів, які викликаються послідовно, коли компонент монтується або оновлюється. Ці методи викликаються до або після того, як компонент візуалізує інтерфейс користувача. Насправді сам метод рендерингу є частиною життєвого циклу компонента. Існує два основні життєві цикли: життєвий цикл встановлення та життєвий цикл оновлення.

Життєвий цикл монтування складається з методів, які викликаються, коли компонент монтується або демонтується. Іншими словами, ці методи дозволяють користувачу спочатку налаштувати стан, здійснювати виклики API, запускати та зупиняти таймери, маніпулювати відрендереним DOM, ініціалізувати сторонні бібліотеки тощо. Ці методи дозволяють користувачам включити JavaScript для допомоги в ініціалізації та знищенні компонента.

Життєвий цикл оновлення – це серія методів, які викликаються, коли змінюється стан компонента або коли від батьківського компонента надходять нові властивості. Цей життєвий цикл можна використовувати для включення JavaScript до оновлення компонента або для взаємодії з DOM після оновлення. Крім того, його можна використовувати для покращення продуктивності програми, оскільки він дає можливість скасувати непотрібні оновлення [12].

Так як уся логіка розрахунків оцінки ступеня тремору буде відбуватися на сервері, то слід подумати над технологією, яка забезпечуватиму високу продуктивність, швидку роботу з великими масивами даних та матиме інструменти реалізації об’єктно-орієнтованого підходу програмування. Тому на серверній частині застосунку буде використано Node.js. Node.js являє собою середовище виконання з відкритим вихідним кодом, яке основане на двигуні Chrome V8 JavaScript і написане на JavaScript і C/C++ [13]. Він використовується для операцій на стороні сервера, переносячи програмування JavaScript у серверну частину для запуску програм у Linux, Windows і OS X. Node.js запускає код JavaScript на стороні сервера за допомогою своєчасної компіляції. Хоча деякі називають Node.js фреймворком, це неточно. Мета фреймворку є надання програмісту набіру багаторазових інструментів для прискорення розробки. Проте програміст Node.js може використовувати низку спеціальних фреймворків, таких як Express, Meteor, Sails та багато інших. Нескінченна кількість фреймворків JavaScript також доступна для бекенда JS. Так як Node.js базується на JS, то вона має весь набір для реалізації об’єктно-орієнтованого підходу програмування.

Ось ключові переваги використання Node.js на сервері:

- Швидка обробка веб-завдань
- Багата екосистема: бібліотеки та інструменти Node.js
- Краща ефективність і загальна продуктивність розробника
- Наявність інструментів представлення системи за допомогою об’єктно-орієнтованого підходу програмування
- Безпечність
- Надійний стек технологій
- Поширення і повторне використання коду
- Сильна корпоративна підтримка
- Забезпечення гнучкості системі

- Доступність єдиної мови програмування під час імплементації клієнт-серверної архітектури
- Ефективне кешування
- Кросплатформна розробка

В обговоренні переваг Node.js гнучкість програмування є, мабуть, найбільшою перевагою. Коли програміст вносить зміни в Node.js систему, це стосується лише цього вузла. Там, де інші середовища виконання або фреймворки можуть вимагати від користувача внести зміни аж до основних концептів програмування системи, у випадку використання Node.js це не вимагає нічого, крім зміни вузла. І це чудово не лише для початкової збірки застосунку, але й для поточного обслуговування. І що найкраще, коли поєднувати у застосунку JSON з Node.js, то можна легко обмінюватися інформацією між клієнтськими серверами та веб-сервером. Також є багато можливостей використовувати API для створення TCP, HTTP, DNS та іншого на сервері для кращого захисту та безпечного доступу до критичних даних користувачів у застосунку.

Як зазначалося у вимогах до системи, застосунок повинен використовувати документо-орієнтовну базу даних. До такої класифікації відноситься MongoDB база даних. MongoDB – це NOSQL база даних з відкритим кодом. Вона була написана на мові програмування C++ [14]. Документо-орієнтована MongoDB зберігає дані у вигляді документів, а зберігаються вони у двійковій формі, що називається форматом BSON. BSON підтримує ряд типів: рядки, цілі числа, дати, логічні значення, числа з плаваючою комою і бінарні значення. MongoDB не є базою даних із фіксованою схемою бази даних, тому дає користувачеві свободу вставляти нові поля в документ або оновлювати попередню структуру документа. Вона не використовує об’єднання, як реляційні бази даних, оскільки має вбудовані документи, до яких можна швидко отримати доступ. Однак MongoDB має засоби для реалізації об’єднання документів із декількох таблиць, але такі операції є складні у виконанні. MongoDB також

підтримує реплікацію Master Slave, де підлеглі вузли містять репліки головних вузлів і використовуються для резервного копіювання та читання.

Основні можливості MongoDB, які задовольняють швидку та ефективну роботу із даним при виконанні обчислення коефіцієнтів ідентифікації:

- Гнучкість. MongoDB зберігає дані у форматі документа за допомогою JSON.
- Простота використання. Базу даних дуже легко встановити, використовувати, підтримувати та налаштовувати.
- Висока продуктивність. Забезпечує збереження даних. База даних зменшує активність вводу/виводу в системі бази даних завдяки підтримці вбудованих документів. Використання індексування підтримує швидші запити.
- Висока доступність. MongoDB підтримує засіб реплікації, який називається набором реплік. Набір реплік – це група серверів, який підтримує однаковий набір даних. Цей процес забезпечує автоматичне перемикання після відмови, резервування та підвищену доступність даних

На швидкість розробки програмного забезпечення впливає не лише вибір мови програмування або методів розробки, але й особливе місце займає середовище розробки системи. Тому для імплементування системи було взяти середовище розробки під назвою Visual Studio Code. З основних переваг даної системи є наявність підтримки мов програмування, які використовуватимуться під час розробки системи, і можливість додавання безлічі плагінів, які забезпечать роботу як із файлами так і допоможуть налаштувати ціле середовище розробки [15]. До останнього часу навряд чи існувало IDE або редактор коду, який був би настільки зручним для користувача, що навіть користувачі, які вперше знайомляться з ним, могли без жодних проблем використовувати кожну функцію. Зручна функція кодування та розпізнавання помилок у коді також допомагають користувачам зробити код більш ефективним і безпомилковим. До додаткових переваг даного

редактора коду можна віднести можливість безкоштовного встановлення та використання усіх його функцій.

Усі вибрані мови програмування та технології задовольняють розробку вибраної архітектури. Також при цьому виборі враховувалися власний досвід та практичні навички, можливість швидко та якісно розробити систему за всіма вимогами. Також враховувалося те, що клієнтський інтерфейс буде відображатися у веб-браузері. Проблемою в цій ситуації є нереалізований функціонал браузерів, який потрібно було писати базуючись на базоних можливостях веб-браузерів. Тому потрібно забезпечити коректне відображення даних на користувацькому інтерфейсі у кожному із сучасних веб-браузерів, як десктопних так і мобільних.

3.2 Розробка застосунку

Проаналізувавши проблематичну область, спроєктувавши архітектуру застосунку та всіх його складових, настав час перетворити набуті діаграми в практичне реалізування їх у вигляді коду. Також в цій частині буде наведено частини реалізації програми за допомогою лістинг коду.

Для початку потрібно навести шаблон спіралі Архімеда, який використовувався при діагностуванні пацієнта. Побудова цієї спіралі задавалась рівнянням в декартовій системі координат. Метод для задання координат реалізується за допомогою ітеративної умови. Нижче наведено код створеного методу у вигляді рисунку лістингу коду (рис. 3.1).

```
const getArchimedesSpiralCoordinates = (TOTAL_POINTS = 300) => {
  const spiralCoordinates = [...new Array(TOTAL_POINTS)].map((e, i) => {
    const r = 9 / (2 * Math.PI);
    const angle = 0.1 * i;
    const x = r * angle * Math.cos(angle);
    const y = r * angle * Math.sin(angle);
    return { x, y };
  });
  return spiralCoordinates;
};
```

Рисунок 3.1 – Програмний код шаблону спіралі Архімеда

Для відображення даних в користувацькому інтерфейсі було використано користувацьку бібліотеку, яка добре вміє працювати із відображенням функцій і точок. Нижче наведено рисунок лістингу прикладу відображення шаблону спіралі Архімеда та результатів від цифрового планшету (рис 3.2).

```
<Chart
  name="tabletRecords"
  coordinates={tabletRecords}
  label={t('calculation.series.tablet')}
  options={XAndYOptions}
  datasets={[
    {
      label: t('calculation.series.spiral'),
      type: 'line',
      data: getArchimedesSpiralCoordinates() as any,
      backgroundColor: 'blue',
      borderColor: 'blue',
      borderWidth: 0.7,
      borderDash: [5, 5],
      pointRadius: 0.4,
    },
  ]}
/>
```

Рисунок 3.2 – Програмний код компоненту для відображення графіків

За аналогією малювання інших графіків буде відбуватись подібним чином.

Тепер перейдемо до серверної реалізації. В цій частині застосунку відбувається процес обчислення і збереження результатів обчислення коефіцієнтів ідентифікації. Під час виконання обчислень виконуватимуться різні перевірки верифікації, щоб забезпечити коректність виконання операцій та отримання результатів.

Очевидно, що перше, що може спасти на думку, при виконанні обчислень коефіцієнтів ідентифікації є завантаження результатів дослідження та парсинг їх вмісту. Завантаження та парсинг файлів цифрового планшету і електроміографа відбуваються схожим чином, тому для прикладу розуміння відобразимо лише одну з реалізацій завантаження та парсингу файлів. Візьмемо файл, де знаходяться результати діагностування електроміографом. Спочатку відбувається читання файлу, після успішного читання метод виконує парсинг записів текстового файлу та перетворює їх у об’єкт ScaledRecords класу, а після успішного виконання всіх етапів, текстовий файл зберігається у базу даних застосунку. Нижче наведено рисунок лістингу реалізованого методу для завантаження та парсингу текстового файлу із результатами діагностування (рис. 3.3).

```
      async normalizeData(base64: string, signalPort: ElectromyographPort):
Promise<[ElectromyographRecord[], Date]> {
        const ssv = base64ToString(base64);
        const { data, errors }: { data: unknown[][]; errors: unknown[] } = Papa.parse(ssv, {
delimiter: '\t' });
        const headers = data.filter((row) => row.length === 1).flat() as string[];
        const startDate = getStartDate(headers);
        const signalPortIndex = ElectromyographPortIndex[signalPort as keyof typeof
ElectromyographPortIndex];
        if (!signalPortIndex) throw new Error(BackendErrors.UNAVAILABLE_SIGNAL_PORT);
        if (errors.length) throw new Error(BackendErrors.TABLET_FILE_INCORRECT_FORMAT);
        const records = this.parseData(data, startDate, signalPortIndex)
        await this.uploadFile(base64);
        return [records, startDate];
      }
```

Рисунок 3.3 – Програмний код для завантаження та парсингу текстових файлів

Ось так виглядає завантаження результатів діагностування від електроміографа. Як видно, що парсинг текстового файлу відбувається в межах

іншого методу. Такий підхід допомагає не навантажувати одну функцією різним функціоналом, а також це є дотриманням принципів SOLID, що функція повинна виконувати лише одну задачу. Нижче наведено рисунок лістингу реалізації парсингу текстового файлу (рис. 3.4).

```
function parseData(data, startDate, signalPortIndex) {
  const records = data
    .map((row) => {
      const time = this.parseTime(row[0] as string, startDate);
      const signal = +(row[signalPortIndex] as number);
      if (Number.isFinite(time.getTime()) && Number.isFinite(signal)) {
        const record: ElectromyographRecord = {
          time,
          relativeTime: startDate.getTime() - time.getTime(),
          signal,
        };
        return record;
      }
      return null;
    })
    .filter(Boolean) as ElectromyographRecord[];
  return records;
}
```

Рисунок 3.4 – Програмний код для парсингу завантажених текстових файлів

Для обчислення коефіцієнтів ідентифікації, дані із файлу повинні бути масштабованими, що є обов’язковою вимогою для коректності результатів. Тому після парсингу даних, отримані дані сигнал масштабуються до діапазону від -1 до 1. Ця логіка закладена в конструкторі ScaledRecords класу, що дає можливість простій та ефективній роботі із даними. Нижче наведено рисунок лістингу масштабування даних до заданого діапазону (рис. 3.5).

```
       constructor(data: T[], name: keyof T) {
         const maxValue = Math.max(...data.map((row) => Math.abs(row[name] as number)));
         this.maxValue = maxValue;
          this.scaledRecords = data.map((row) => ((row[name] as number) / maxValue) as unknown
as typeof row);
       }
```

Рисунок 3.5 – Програмний код для масштабування записів

Тепер перейдемо до розгляду ключового методу, який виконує обчислення коефіцієнтів ідентифікації. Даний метод називається produceCoefficient із CoefficientCalculation класу. Всередині цього класу виконуються різні трансформування та перетворення даних. Результат роботи методу повертається після завершення його роботи. Метод приймає масштабовані значення із текстового файлу та виконує всі маніпуляції над ними, а не оригінальні. Реалізація методу обчислення коефіцієнтів ідентифікації буде представлена у додатку Д.

При розробці методів та механізмів для визначення ступеня тремору, в застосунку використовувалися технології, які дозволяють швидко, ефективно та коректно обчислити всі характеристичні величини, які відносяться до побудови математичної моделі.

3.3 Ілюстрування роботи розробленої інформаційної системи

Після розробки інформаційної системи в результаті було розроблено користувацький інтерфейс для взаємодії із логікою обчислення коефіцієнтів ідентифікації та відображення відповідних графіків для можливості подальшого оцінювання ступеня тремору.

Демонстрація програми почнеться із того, як користувач заходить на веб-застосунок з будь-якого пристрою. Система перевіряє те, чи користувач авторизований у ній. Якщо авторизований, то система перенаправляє користувача на

сторінку для обчислень коефіцієнтів ідентифікації. В іншому випадку, система заставляє користувача авторизуватись. На рисунку нижче відображена сторінка авторизації (рис. 3.6).

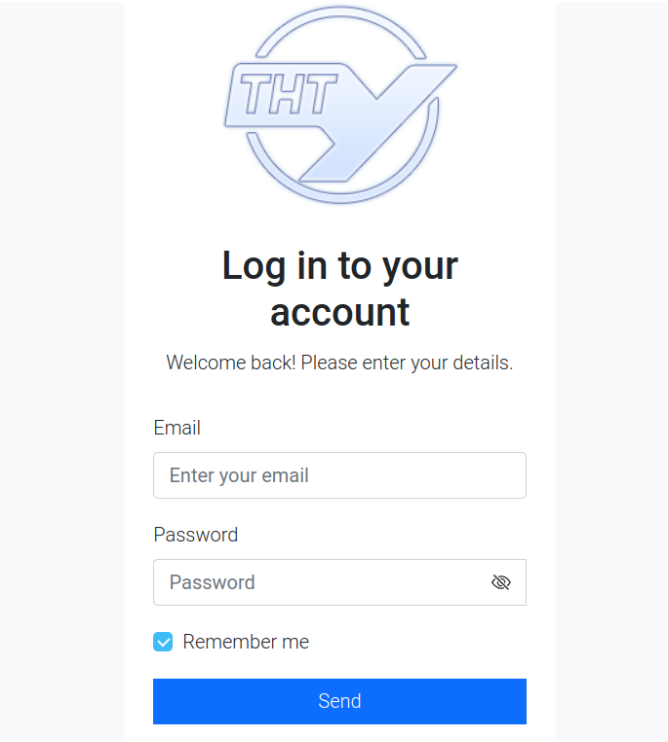

Рисунок 3.6 – Сторінка авторизації

Щоб виконати процедуру авторизації користувач обов’язково повинен ввести свою електронну пошту і пароль. Якщо користувач забув ввести якесь поле, то покажеться відповідна помилка (рис 3.7). Також є специфічне поле під назвою запам’ятати мене. Воно означає, що при виконанні процесу авторизації, отриманий токен обновлятися не буде. А після того, як він стане недійсним, користувач зному буде змушеним пройти процес авторизації.

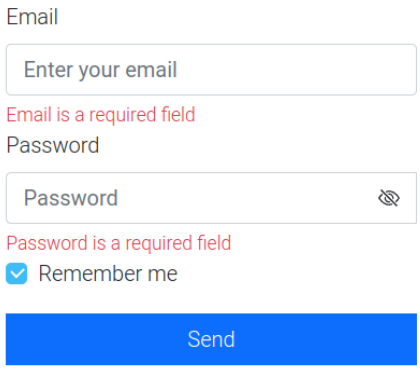

Рисунок 3.7 – Обробка помилок у формі авторизації

Після заповнення всіх обов’язкових полів, користувач може відправити запит на сервер, щоб авторизуватись. У випадку недійсності надісланих даних, користувачу відображається відповідне повідомлення і система просить користувача ввести коректні дані.

Після успішної авторизації система перенаправляє користувача на сторінку обчислення коефіцієнтів ідентифікації. На сторінці відображається форма для обчислення коефіцієнтів ідентифікації (рис. 3.8).

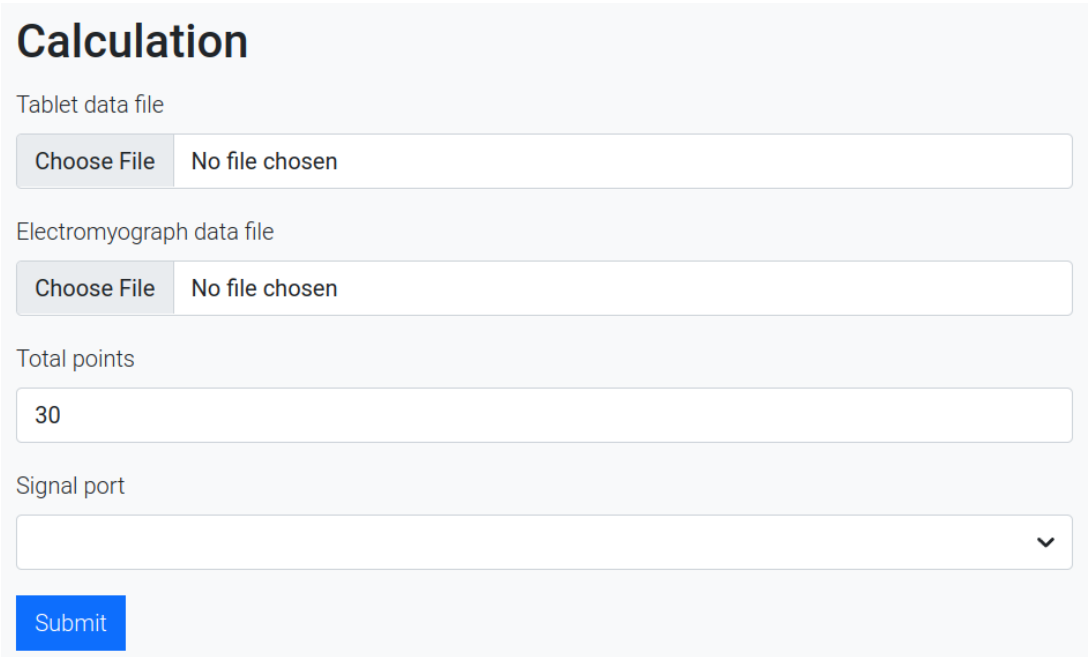

Рисунок 3.8 – Сторінка обчислення коефіцієнтів ідентифікації

Як і для форми авторизації, форма обчислень має свою схему валідації. Серед обов'язкових її полів є текстові файли із результатами діагностування електроміографа та цифрового планшету, кількість сегментів та сигнал над яким виконуватимуться процес обчислення коефіцієнтів ідентифікації. Сигнали беруться із текстового файлу електроміографа. Всього сигналів є 16. Нижче наведено приклад відображення помилок для форми обчислення (рис. 3.9).

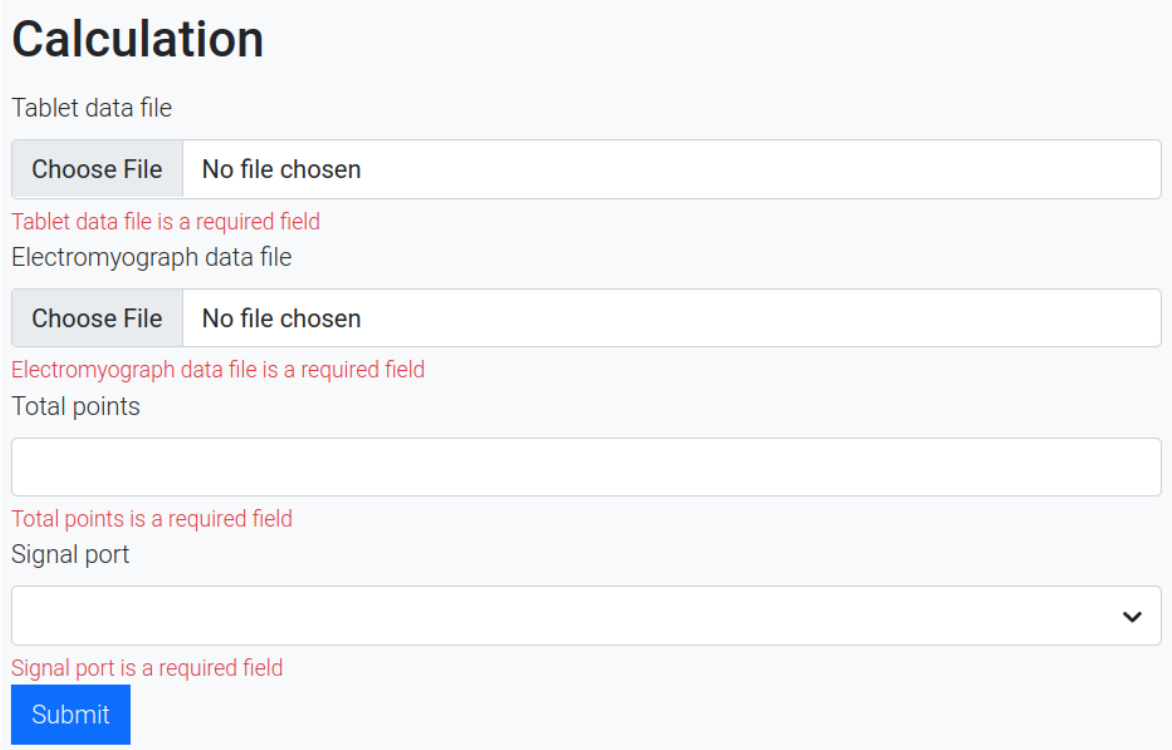

Рисунок 3.9 – Обробка помилок у формі обчислення коефіцієнтів ідентифікації

Вибравши всі обов'язкові поля, користувач може відправити запит, щоб система обчислила коефіцієнти ідентифікації. Отримавши результати обчислень, користувацький інтерфейс вимальовує отримані дані у вигляді графіків. Користувач має можливість побачити графіків порівняння шаблону спіралі Архімеда та даних користувача від цифрового планшету, результатів від електроміографі, розкладення результатів від цифрового планшету за зміною у часі, а також розроблену

математичну модель. Нижче наведено результати діагностування у вигляді графіків (рис 3.10).

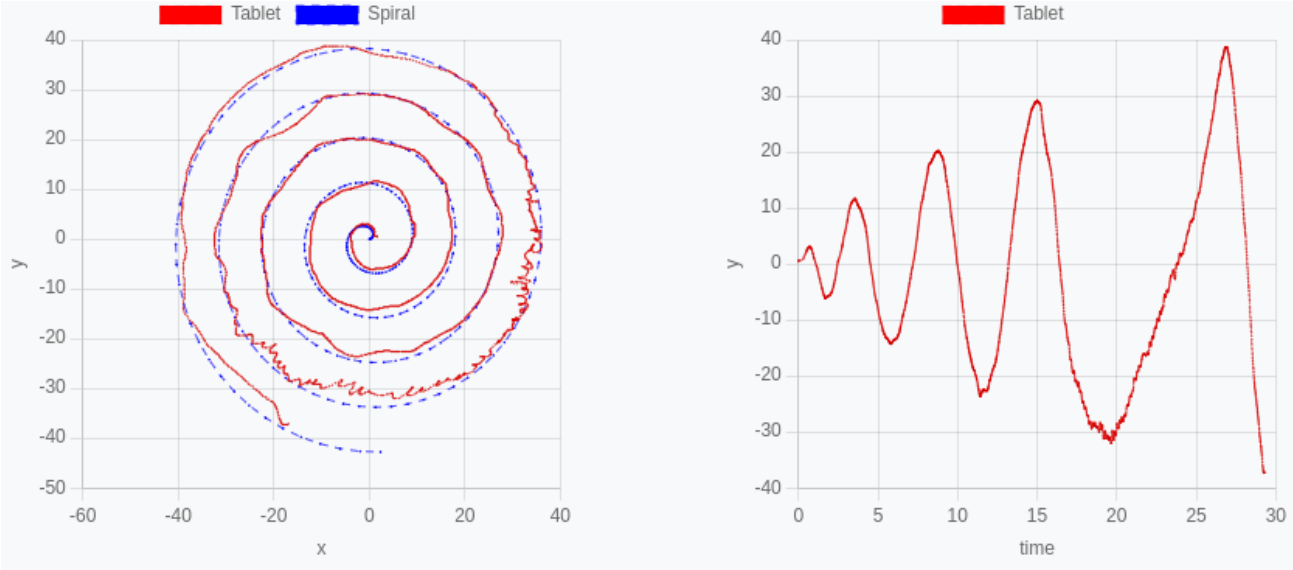

Рисунок 3.10 – Демонстрація результатів діагностування

Також відбувається розкладання траси пацієнта за координатою горизонталі та силою тиску цифрового пера на планшет у часі (рис. 3.11).

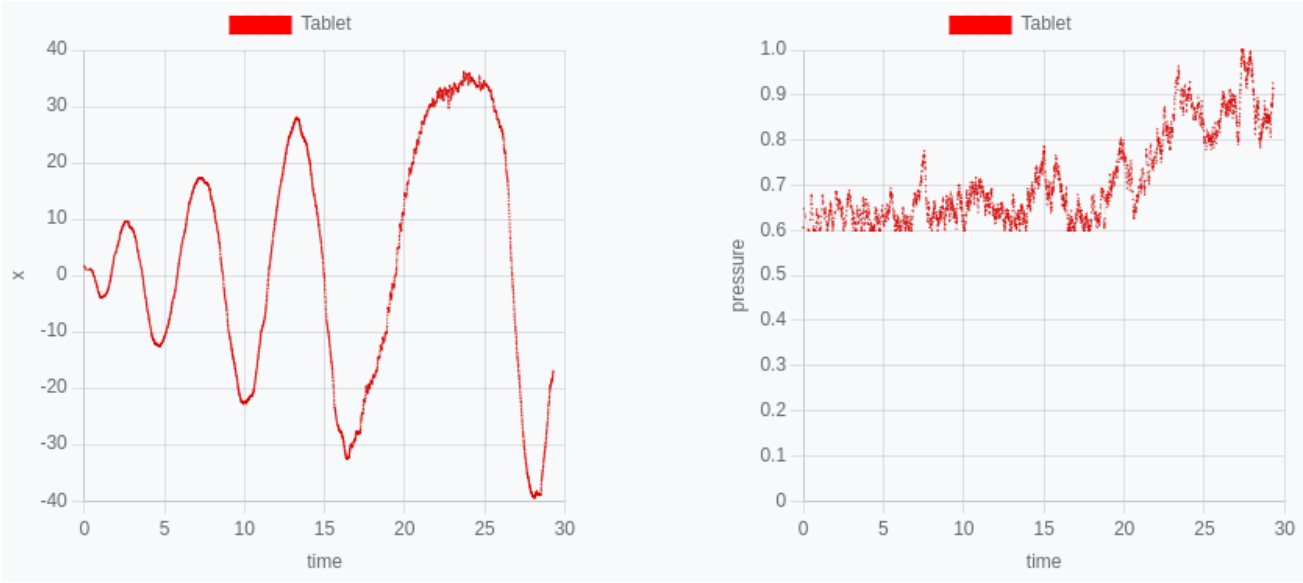

Рисунок 3.11 – Демонстрація результатів діагностування

Після відображення всіх необхідних початкових графіків, за ними користувач може побачити графік на якому продемонстровано результати тестування математичної моделі (рис. 3.12).

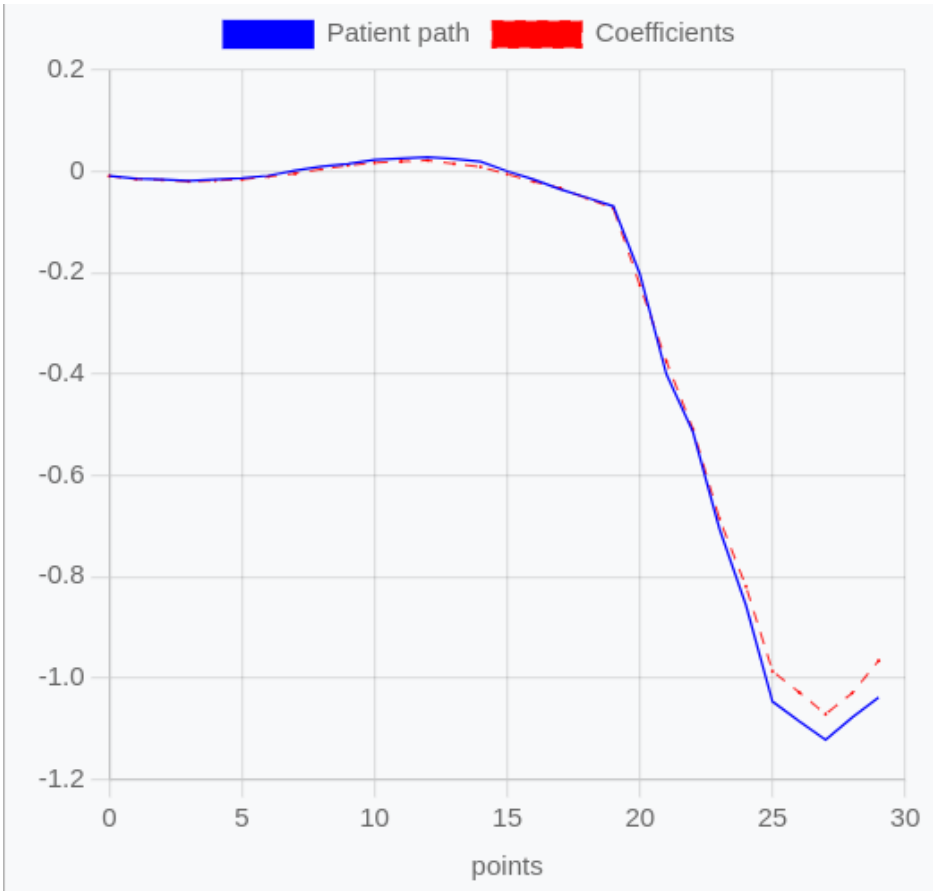

Рисунок 3.12 – Результати обчислення коефіцієнтів ідентифікації

Як видно з рисунку, користувач вибрав малу кількість розбиттів всієї траси на сегменти. При малій кількості, гррафік показуватиме відносно велику похибку відносно реальної траси пацієнта. Тому користувачу надана можливість регулювати кількість сегментів, щоб зменшити результуючу похибку.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ

4.1 Охорона праці

Проєктування та розробка інформаційної системи ідентифікації параметрів анормальних неврологічних рухів людини відбувалися з використанням персональних комп’ютерів. Тому важливим фактором безпеки праці є дотримання основних правил користування комп’ютерною технікою, основних норм та правил охорони праці. Для цього потрібно забезпечити користувачам максимально комфортні та безпечні умови для їх перебування в приміщенні для того, щоб вони якісно та ефективно виконували поставлені завдання.

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності [7].

Вимоги щодо охорони праці, зокрема охорони праці офісних працівників, містять Кодекс законів про працю, Закон України «Про охорону праці» та інші нормативно-правові акти. У відповідності до вимог ст. 153 Кодексу законів про працю України та ст. 6 Закону України «Про охорону праці» на всіх підприємствах, організаціях створюються безпечні і нешкідливі умови праці [7], [8]. Забезпечення відповідних умов праці повинні контролювати власник або уповноважений ним орган.

Робочі місця працівників, які обладнані комп’ютерами пристроями, повинні відповідати вимогам НПАОП 0.00-7.15-18 «Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» та ДСанПІН 3.3.2.007-98 «Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

Робочі місця працівників з екранними пристроями мають бути спроектовані так і мати такі розміри, щоб працівники мали простір для зміни робочого положення та рухів.

Для забезпечення безпеки та захисту здоров'я працівників усе випромінювання від екранних пристроїв має бути зведене до гранично допустимого рівня (вплив на людину факторів довкілля – шуму, вібрації, забруднювачів, температури тощо, який не спричиняє соматичних або психічних розладів, а також змін стану здоров'я, працездатності, поведінки, що виходять за межі пристосувальних реакцій) з погляду безпеки та охорони здоров'я працівників.

Організація робочого місця працівника з екранними пристроями має забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним, антропологічним, психофізіологічним вимогам, а також характеру виконуваних робіт [9].

Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги з урахуванням характеру та особливостей трудової діяльності. При розміщенні подібних робочих місць необхідно дотримуватись наступних вимог [10]:

- відстані між бічними поверхнями ВДТ 1,2 м
- відстань від тильної поверхні одного ВДТ до екрана іншого ВДТ – 2,5 м

Вимоги цих пунктів щодо відстаней беруться з урахуванням особливостей стін та перегородок.

Побудова робочого місця користувача комп'ютерними засобами має забезпечувати зберігання задовільної робочої пози за наступними рекомендованими правилами: підошва – на підлозі або на підставці для ніг стегна – в горизонтальному положенні; передпліччя – вертикально; лікті – під кутом 70°-90° до вертикальної поверхні; зап'ястя зігнуті під кутом не більше 20° відносно горизонтальної поверхні, нахил голови – 20° відносно вертикальної поверхні; монітор персонального комп'ютеру – не менше 60 см від обличчя користувача [10].

Оскільки інноваційні технології ідуть вперед, деякі галузі потребують роботу з широкоформатними моніторами. У цих випадках відстань у 60 см між користувачем та дисплеєм є дуже небезпечною та порушує медичні норми роботи з обчислювальними пристроями. Згідно медичним нормам, оптимальна відстань до монітору повинна складати більше мінімального – від півтора до двох діагоналей.

Клавіатуру слід розташовувати на поверхні столу на відстані 100-300 мм від краю, звернутого до працюючого. У конструкції клавіатури має передбачатися опорний пристрій (виготовлений із матеріалу з високим коефіцієнтом тертя, що перешкоджає мимовольному її зсуву), який дає змогу змінювати кут нахилу поверхні клавіатури [10].

Робочий стілець оператора обчислювальної машини повинен бути оснащений наступними елементами: сидінням, спинкою та стаціонарними або змінними підлокітниками.

Стілець може бути додатково обладнаний засобами, що поліпшують умови роботи, а саме: підголовник та підставка для ніг.

4.2 Підвищення стійкості роботи об’єктів господарської діяльності у воєнний час

Приділяючи велику увагу зміцненню обороноздатності нашої країни, урядом неодноразово підкреслювалося, що оборонна міць держави складається не тільки з високої готовності й оснащення Збройних сил, а й нерозривно позв’язана з високим рівнем економічного розвитку країни, підготовкою населення й об’єктів народного господарства до захисту від зброї масового ураження.

Під стійкістю об’єктів господарства країни в цілому розуміється його здатність в умовах надзвичайних ситуацій мирного і воєнного часу випускати продукцію в

запланованому обсязі й номенклатурі, а при одержанні слабких і середніх руйнувань, порушенні зв’язків по кооперації і постачанням відновлювати виробництво в мінімальний термін.

Здатність об’єкта господарства випускати продукцію залежить від захисту і нормального функціонування чотирьох основних елементів сучасного виробництва, якими є:

- виробничий персонал (робітники та службовці);
- будинки і споруди з технологічним устаткуванням;
- система постачання енергією, водою, паливом, устаткуванням і ремонтною базою;
- система виробничих і кооперативних зв’язків з іншими об’єктами.

Захист робітників та службовців в умовах надзвичайних ситуацій мирного і воєнного часу є найголовнішою задачею по підвищенню стійкості роботи об’єкта господарювання. Робітники й службовці – головна продуктивна сила і тому стійкість економіки визначається, насамперед, здатністю захистити і зберегти цю силу.

Військові конфлікти супроводжуються руйнуванням будинків, споруджень і знищенням основної продуктивної сили – працюючого населення. Тому серед усіх задач по підвищенню стійкості роботи об’єктів народного господарства основною є задача завчасного вживання заходів по забезпеченню захисту робітників та службовців і членів їхніх родин.

Захист робітників та службовців від зброї масової поразки в сучасних умовах здійснюється трьома основними способами:

- укриття людей у захисних спорудженнях (сховищах, протирадіаційних укриттях);
- проведення евакуації робітників, службовців і членів їхніх родин;
- використання засобів індивідуального захисту, а також проведенням заходів щодо протирадіаційного, протихімічного і протибактеріологічного захисту з урахуванням конкретних обставин.

Стійкість роботи об’єкта господарської діяльності – це здатність його в умовах НС випускати продукцію у запланованому обсязі та визначеної номенклатури, а у разі слабких та середніх руйнувань або порушення матеріального постачання – відновлювати виробництво власними силами у короткий термін.

Підвищення стійкості роботи об’єкта господарства у воєнний час і в умовах надзвичайних ситуацій досягається завчасним проведенням комплексу інженерно-технологічних, технологічних і організаційних заходів, спрямованих на максимальне зниження впливу вражаючих факторів зброї масового ураження і створення умов для швидкої ліквідації наслідків. Підготовка до відновлення порушеного виробництва здійснюється завчасно і передбачає планування відбудовних робіт по декількох варіантах: підготовку ремонтних бригад, створення необхідного запасу матеріалів і устаткування, надійний його захист.

Інженерно-технічні заходи, як правило, включають комплекс робіт, що забезпечують підвищення стійкості виробничих будинків і споруджень, верстатного і технологічного устаткування, комунально-енергетичних систем.

Технологічні заходи забезпечують підвищення стійкості роботи об’єкта шляхом зміни технологічного процесу, що сприяє прискоренню виробництва продукції і виключає можливість утворення вторинних вражаючих факторів.

Організаційні заходи передбачають розробку і планування дій керівного, командно-начальницького складу, штабу, служб і формувань ЦЗ при захисті робітників та службовців підприємства й інших невідкладних робіт, відновленні виробництва, а також по випуску продукції на збережених потужностях.

Для виробництва продукції необхідні: електроенергія, вода, паливо, сировина, матеріали й інші матеріально-технічні засоби. Забезпечення підприємств цими ресурсами багато в чому визначає можливість нормального їхнього функціонування в умовах воєнного часу. Це досягається проведенням таких заходів, що сприяють підвищенню не ураженості комунально-енергетичних мереж, транспортних

комунікацій і джерел постачання, надійному захисту необхідних запасів палива, сировини, напівфабрикатів, що комплектують, виробів тощо.

Під час розробки системи електропостачання є однією із ключових необхідностей для успішного та вчасного створення проєкту. Порушення нормальної подачі електроенергії на об'єкт чи окремі ділянки виробництва може призвести до повного припинення роботи об'єкта. Для забезпечення надійного електропостачання в умовах війни при його проектуванні та будівництві повинні бути враховані наступні основні вимоги, що випливають із задачі ЦЗ.

До основних вимог систем електропостачання відоносять:

- Електропостачання повинне здійснюватися від енергосистем, до складу яких входять електростанції, що працюють на різних видах палива. Великі електростанції варто розміщати одну від одної і від великих міст на відстані не менше двох радіусів зон можливих руйнувань.
- Постачання електроенергією великих міст і об'єктів, що не припиняють роботу у воєнний час, необхідно передбачати від двох незалежних джерел. При електропостачанні об'єкта від одного джерела повинне бути не менш двох уведень з різних напрямків.
- Трансформаторні підстанції необхідно надійно захищати, їхня стійкість повинна бути не нижчою ніж стійкість самого об'єкта. 5. Електроенергію до ділянок виробництва варто подавати по незалежних електрокабелях, прокладених у землі на глибині 0,8 – 1,2 м.
- Необхідно створювати автономні резервні джерела електропостачання. Для цього можна використовувати пересувні електростанції на залізничних платформах і суднах, малопотужні електростанції, не включені в енергосистеми,тощо.
- При проектуванні систем електропостачання варто зберігати в якості резервних дрібні стаціонарні електростанції об'єктів.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було спроєктовано та розроблено інформаційну систему автоматизації оцінювання ступеня тремору пацієнта. Розроблена система включає в себе інтерактивний користувацький інтерфейс з можливістю входу у систему, а також внесення вхідних даних для обчислення коефіцієнтів ідентифікації. Система відображає вихідні дані для користувача у зручному для аналізування даних форматі.

Під час здійснення фази аналізу було проаналізовано стан медичної сфери у світі та в Україні, виділено основні проблеми, які постають перед пацієнтами та медичними працівниками під час діагностування тремору в сучасних умовах.

Також було проведено аналіз існуючих методів, за якими діагностують та виконують оцінювання стану тремору у пацієнта. У виконаній роботі виконувалась автоматизація діагностування тремору у пацієнта за допомогою тесту спіралі Архімеда. На даний момент проведення діагностування за допомогою спіралі Архімеда не було автоматизованим і спиралося на суб’єктивну оцінку медичного працівника. Це давало великі неточності та можливе неправильне діагностування хвороби та практичну неможливість визначення тремору на ранніх етапах захворювання у пацієнта.

У системі є можливість зберігати результати діагностування. Така можливість з'являється після проведення обчислення коефіцієнтів діагностування. Медичний працівник заповнює форму та вносить дані про користувача і заключення діагнозу з коротким описом всіх особливостей вихідних даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Louis, E.D.; Ferreira, J.J. How common is the most common adult movement disorder? Update on the world-wide prevalence of essential tremor. Mov. Disord. 2010,25, 534-541.

2. E. D. Louis, A. Gillman, S. Boschung, C. W. Hess, Q. Yu, and S. L. Pullman, ‘‘High width variability during spiral drawing: Further evidence of cerebellar dysfunction in essential tremor,’’ Cerebellum, vol. 11, no. 4, pp. 872-879, Dec. 2012.

3. Tam F, Huang Y, Schwartz ML, Schweizer TA, Hynynen K, Graham SJ. A computerized tablet system for evaluating treatment of essential tremor by magnetic resonance guided focused ultrasound. BMC Neurol. 2017;17(1):74.

4. Pullman SL. Spiral analysis: a new technique for measuring tremor with a digitizing tablet. Mov Disord. 1998;13(S3):85-9.

5. Петрик М. Р., Мудрик І. Я., Михалик Д. М., Петрик О. Ю., Биць Т. П. Огляд математичних моделей анормальних неврологічних рухів з урахуванням когнітивних feedback-впливів нейровузлів кори головного мозку. Прикладні питання математичного моделювання. 2020. Т. 3, № 2.2. С. 221-234. ISSN 2618-0332.

6. Методи математичного моделювання та ідентифікації складних процесів і систем на основі високопродуктивних обчислень [Електронний ресурс] – Режим доступу: https://elartu.tntu.edu.ua/bitstream/lib/30745/17/Vysorpprov_Metody_Mono_IK_HANU_4new_2019_4_sydoruk_2_4.pdf.

7. Закон України «Про охорону праці» [Електронний ресурс] // № 2695 – XII від 14.10.1992 р. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2694-12.

8. Кодекс законів про працю України [Електронний ресурс] // № 322 – VIII від 10.12.1971 р. – Режим доступу: https://zakon.rada.gov.ua/laws/show/322-08.

9. Наказ України «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [Електронний ресурс]

// № 207 від 14.02.2018 р. – Режим доступу: https://zakon.rada.gov.ua/laws/show/z0508-18.

10. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин [Електронний ресурс] // № 7 від 10.12.1998 р. – Режим доступу: https://zakon.rada.gov.ua/laws/show/z0508-18.

11. Мова програмування JS [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript/.

12. Документація бібліотеки React.js [Електронний ресурс] – Режим доступу: https://reactjs.org/.

13. Серверна мова програмування Node.js [Електронний ресурс] – Режим доступу: https://nodejs.org/en/.

14. Документація бази даних MongoDB [Електронний ресурс] – Режим доступу: https://www.mongodb.com/home.

15. Редактор коду Visual Studio Code [Електронний ресурс] – Режим доступу: https://code.visualstudio.com/.

16. Документація UML [Електронний ресурс] – Режим доступу: http://www.uml.org/.

17. Діаграма варіантів використання [Електронний ресурс] – Режим доступу: https://www.lucidchart.com/pages/uml-use-case-diagram/.

18. Діаграма класів [Електронний ресурс] – Режим доступу: https://www.lucidchart.com/pages/uml-class-diagram/.

19. Діаграма розгортання [Електронний ресурс] – Режим доступу: https://www.lucidchart.com/pages/uml-deployment-diagram/.

20. Діаграма відношення сутностей [Електронний ресурс] – Режим доступу: https://www.lucidchart.com/pages/er-diagrams.

21. Методологія розробки РУП [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Rational_Unified_Process/.

ДОДАТКИ

Додаток А – Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ
Кафедра “Програмної інженерії”

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання кваліфікаційної роботи магістра
«Проєктування інформаційної системи ідентифікації параметрів анормальних неврологічних рухів людини під дією техногенних навантажень на Javascript технології»

Керівник роботи:
Петрик М. Р.
“___” _____ 2022р.
_____

Виконавець:
студент групи СПм-61
Стефанишин Іван Миколайович
“___” _____ 2022р.
_____

Тернопіль 2022

ЗМІСТ

1. ПІДСТАВИ ДО РОЗРОБКИ
2. ПРИЗНАЧЕННЯ РОЗРОБКИ
3. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ
    3.1 Функціональні вимоги
    3.2 Технічні вимоги
    3.3 Програмні вимоги
4. СТАДІЇ НАПИСАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ
5. СУПРОВІДНА ДОКУМЕНТАЦІЯ
6. ПОРЯДОК ЗДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ МАГІСТРА
7. ВІДМІТКИ ПРО ВИКОНАННЯ ЕТАПІВ В РОБОТІ

1 ПІДСТАВИ ДО РОЗРОБКИ

Розробка проводиться у відповідності до графіку навчального плану підготовки магістрів за спеціальністю «Інженерія програмного забезпечення» 2021-2022 н.р.

Тема кваліфікаційної роботи магістра: «Проєктування інформаційної системи ідентифікації параметрів анормальних неврологічних рухів людини під дією техногенних навантажень на Javascript технології».

Термін виконання: до __.___._____р.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

Для розробки веб-орієнтованої системи планування витрат у сфері будівництва на основі JavaScript технологій, було використано мову програмування JavaScript бібліотеку React, серверний фреймворк Express.js, об’єктно-орієнтовану мову програмування JavaScript. В якості СКБД для даної системи було обрано MongoDB.

У сучасному світі тремор кінцівок людини є найпоширенішим руховим порушенням. Тремор за природою має різні види походження та різну частоту, що ускладнює своєчасну діагностика всіх характеристик тремору. З цих причин швидке діагностування хвороби є одним з ключових проблем, що вирішуються в сфері сучасної неврології.

В наслідок розробки вдалих методик оцінювання ступеня тремору і правильно спроєктованій архітектур інформаційної системи, розроблені застосунки на основі перелічених переваг стають необхідним атрибутом будь-якого науково-технічного процесу. Процес автоматизація процесу оцінювання тремору з використанням тесту спіралі Архімеда оснований на обробці оцифрованих даних. Використовуючи математичну модель можна отримати високоякісний аналіз та числове представлення рухової поведінки пацієнта.

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Функціональні вимоги

Система повинна надавати користувачу такі можливості:

- здійснення входу;
- вихід із системи;
- представлення вхідних даних за допомогою графіків;
- порівняння шаблону спіралі із спіралю пацієнта;
- представлення розгорнутої спіралі пацієнта за осями;
- представлення обраного сигналу електроміографа;
- представлення результатів обчислення коефіцієнтів із вхідними даними пацієнта;
- зміна вхідних параметрів;
- повторне обчислення коефіцієнтів ідентифікації;
- збереження результатів оцінювання;

Вхідна інформація отримується шляхом введення інформації користувачами.

Вихідна інформація:

інформацію про обчислення коефіцієнтів ідентифікації відображається у вигляді графіків;

3.2 Технічні вимоги

Вимоги до серверної частини: ОС Linux, Windows не менше ніж 4Гб ОЗП;

Вимоги до клієнтської частини: Наявність браузера, пристрої вводу і виводу інформації;

Додаткові вимоги: наявне підключення до мережі Інтернет, автоматичне резервування, забезпечення одночасної роботи до 1000 клієнтів.

3.3 Програмні вимоги

Використання СУБД: MongoDB.

Розробка клієнтської частини: мова програмування JavaScript, бібліотека React.

Розробка серверної частини: платформа Node.js, фреймворк Express.js.

Середовище розробки: Microsoft VSCode.

4 СТАДІЇ НАПИСАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Написання кваліфікаційної роботи проводиться в наступному порядку:

- вибір та затвердження теми
- аналіз предметної області
- постановка задач розробки
- проєктування діаграми варіантів використання
- вибір СКБД та опис її фізичної моделі
- опис програмної реалізації
- опис схем використання програмного забезпечення
- тестування програмного забезпечення
- написання розділу «Охорона праці»
- написання розділу «Безпека в надзвичайних ситуаціях»
- оформлення записки кваліфікаційної роботи магістра
- попередній захист
- нормоконтроль
- захист кваліфікаційної роботи магістра

Результати виконання кожного етапу кваліфікаційної роботи магістра погоджуються з керівником роботи.

5 СУПРОВІДНА ДОКУМЕНТАЦІЯ

Для кваліфікаційної роботи магістра повинні бути розроблені наступні документи:

- записка кваліфікаційної роботи;
- презентація;
- рецензія на кваліфікаційну роботу магістра;
- відгук керівника на кваліфікаційну роботу магістра;
- авторська довідка;
- протокол аналізу звіту подібності керівником роботи;
- диск з кваліфікаційною роботою магістра.

Записка кваліфікаційної роботи магістра оформляється згідно діючих вимог до нормоконтролю.

6 ПОРЯДОК ЗДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ МАГІСТРА

Розроблена системи повинна відповідати вимогами, що складаються з перерахованих у 3 розділі цього документу характеристик.

Для здачі проекту необхідно підготувати весь перелік документів зазначений у розділі 5 цього документу.

Приймання проекту проводиться спеціально створеною комісією в термін зазначені в розділі 1 цього документу.

7 ВІДМІТКИ ПРО ВИКОНАННЯ ЕТАПІВ В РОБОТІ

Назва етапу	Відмітка*
Вибір та затвердження теми	
Аналіз предметної області	
Постановка задач розробки	
Проектування діаграми варіантів використання	
Вибір СКБД та опис її фізичної моделі	
Опис програмної реалізації	
Опис схем використання програмного забезпечення	
Тестування програмного забезпечення	
Написання розділу «Охорона праці»	
Написання розділу «Безпека в надзвичайних ситуаціях»	
Оформлення записки кваліфікаційної роботи магістра	
Попередній захист	
Нормоконтроль	
Захист кваліфікаційної роботи магістра	

* відмітки про виконання етапу ставляться керівником проекту

Додаток Б – Публікація у науковому виданні

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

X НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

7–8 грудня 2022 року

ТЕРНОПІЛЬ
2022

УДК 004.4
І. Стефанишин, М. Петрик
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІДЕНТИФІКАЦІЇ ПАРАМЕТРІВ АНОРМАЛЬНИХ НЕРВОЛОГІЧНИХ РУХІВ ЛЮДИНИ ПІД ДІЄЮ ТЕХНОГЕНИХ НАВАНТАЖЕНЬ

УДК 004.4
I. Stefanyshyn, M. Petryk

DESIGN OF THE INFORMATION SYSTEM FOR THE IDENTIFICATION OF THE PARAMETERS OF ABNORMAL NEUROLOGICAL HUMAN MOVEMENTS UNDER THE EFFECT OF TECHNOLOGICAL LOADS

На сьогодні існує багато захворювань на різні захворювання, які спричиняють дисфункцію моторно-рухових рухів людини. Особливу увагу приділяють захворюванням верхніх кінцівок людини. Відповідно до даних Всесвітньої організації охорони здоров'я в Україні в середньому 6% осіб у віці 65 років і більше, а також 3,8% осіб до 40 років скаржаться на різні типи тремору [1].

Область дігностування пацієнтів з ознаками тремору має ряд складностей, які заважають медичним працівникам поставити коректний діагноз. До основних перешкод оцінювання можна віднести біологічний, техногений і людський фактори: різна природа походження тремору, пізнє звернення в медичну установу для діагностування тремору, застаріле обладнання та програмне забезпечення. Застосування сучасних інформаційних технологій в медицині дозволяє підвищити якість діагностики захворювань за рахунок надання додаткової інформації про виникнення патологічних процесів.

Метою наукового дослідження є виявлення ознаків тремору під час проходження тест спіралі Архімада. Базуючись на отриманих даних від цифрового планшету та електроміографа розроблялася математична модель. Ця модель розроблена для оцінювання степеня захворювання пацієнта.

Проєктування та розробка такого роду системи є не із простих. Потрібно бути уважним до кожних дрібниць, бо від результатів у майбутньому залежитиме здоров'я людини. Велика кількість процесів обробки та аналізу даних, отриманих від пристроїв діагностування, додавали свої перешкоди для коректної розробки застосунку [2].

Література

1. Петрик М. Р., Михалик Д. М., Мудрик І. Я. Спосіб цифрового вимірювання параметрів анормальних неврологічних рухів верхніх кінцівок у пацієнтів із проявами тремору. Патент на корисну модель № 130247, Бюл. № 22 від 26.11.2018.
2. Haubenberger D, Kalowitz D, Nahab F B, Toro C, Ippolito D, Luckenbaugh DA, Wittevrongel L, Hallett M. Validation of Digital Spiral Analysis as Outcome Parameter for Clinical Trials in Essential Tremor. Movement Disorders 26 (11), 2073–2080, (2011).

Н. Свитак
ПІДВИЩЕНИЙ РІВЕНЬ ПРОКРАСТИНАЦІЇ В ЛЮДЕЙ ЧЕРЕЗ ЗБІЛЬШЕННЯ ІНФОРМАЦІЇ І ДОСТУПУ ДО НЕЇ
N. Svytak
INCREASED LEVEL OF PROCRASTINATION IN PEOPLE DUE TO INCREASING INFORMATION AND ACCESS TO IT 128
І. Сіжук, В. Бревус
РОЗРОБКА ЗВУКОВИХ ЕФЕКТІВ ДЛЯ ПЛАГІНІВ ВІРТУАЛЬНОЇ СТУДІЇ ЦИФРОВИХ ЗВУКОВИХ РОБОЧИХ СТАНЦІЙ
I. Sizhuk, V. Brevus
DEVELOPMENT OF SOUND EFFECTS FOR VIRTUAL STUDIO PLUG-INS` DIGITAL AUDIO WORKSTATIONS 128
Д. Сомін, А. Паламар, В. Волоський
ПЕРЕВАГИ WEBASSEMBLY ЯК ІНСТРУМЕНТА РЕАЛІЗАЦІЇ АЛГОРИТМІВ У РЕСУРСОМІСТКИХ ВЕБ-ДОДАТКАХ
D. Somin, A. Palamar, V. Voloskyi
WEBASSEMBLY ADVANTAGES AS A TOOL FOR ALGORITHMS IMPLEMENTATION IN HIGH-LOAD WEB APPLICATIONS 130
І. Стефанишин, М. Петрик
ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІДЕНТИФІКАЦІЇ ПАРАМЕТРІВ АНОРМАЛЬНИХ НЕРВОЛОГІЧНИХ РУХІВ ЛЮДИНИ ПІД ДІЄЮ ТЕХНОГЕНИХ НАВАНТАЖЕНЬ
I. Stefanyshyn, M. Petryk
DESIGN OF THE INFORMATION SYSTEM FOR THE IDENTIFICATION OF THE PARAMETERS OF ABNORMAL NEUROLOGICAL HUMAN MOVEMENTS UNDER THE EFFECT OF TECHNOLOGICAL LOADS 131
О. Сторожук
ТЕХНОЛОГІЯ БЛОКЧЕЙН В ОСНОВІ NFT
O. Storozhuk
NFT TECHNOLOGY BASED ON BLOCKCHAIN 132
Є. Тимченко, Г. Цуприк
РОЗРОБКА РОЗУМНОЇ СИСТЕМИ ОБЛІКУ РОБОТИ КОМП'ЮТЕРНОЇ МЕРЕЖІ ПІДПРИЄМСТВА В СЕРЕДОВИЩІ EMBARCADERO RAD STUDIO XE
Ye. Tymchenko, H. Tsupryk
DEVELOPMENT OF A ACCOUNTING SMART SYSTEM OF COMPUTER NETWORK SERVICE OF THE ENTERPRISE IN THE EMBARCADERO RAD STUDIO XE ENVIRONMENT 133

СЕКЦІЯ 5. **НОВІТНІ ФІЗИКО-ТЕХНІЧНІ ТА ОСВІТНІ ТЕХНОЛОГІЇ**

Я. Войтович, А. Лупенко
МЕТОДИ ПОБУДОВИ ШИРОКОСМУГОВИХ МЕРЕЖ ДОСТУПУ
Y. Voitovych, A. Lupenko
METHODS OF BUILDING BROADBAND ACCESS NETWORKS 134
Ю. Попович
РОЗРОБЛЕННЯ АВТОМАТИЗОВАНОГО МЕТОДУ КОНТРОЛЮ ПОВЕРХОНЬ МЕТАЛОКОНСТРУКЦІЙ, ВІДНОВЛЕНИХ РОБОТИЗОВАНИМ НАПЛАВЛЕННЯМ
Yu. Popovych
DEVELOPMENT OF AN AUTOMATED METHOD OF CONTROLLING THE SURFACES OF METAL STRUCTURES RESTORED BY ROBOTIC SURFACING 135

Лістинг коду 1 – Програмний код для відображення графіків

```
import React, { useLayoutEffect, useRef } from 'react';
import { Chart as ChartJS, ChartDataset, ChartOptions, ChartTypeRegistry, registerables
} from 'chart.js';
import { merge } from 'lodash';

export type Coordinate = {
  x: number;
  y: number;
};

ChartJS.register(...registerables);

export type ChartOptionProps = ChartOptions<keyof ChartTypeRegistry>;
export type ChartDatasetProps<T> = ChartDataset<keyof ChartTypeRegistry, T>;

const Chart = <T,>(props: {
  label?: string;
  name: string;
  coordinates?: Coordinate[];
  options?: ChartOptionProps;
  datasets?: ChartDatasetProps<T>[];
}) => {
  const { coordinates, name, options, label, datasets = [] } = props;
  const ref = useRef<HTMLDivElement | null>(null);

  useLayoutEffect(() => {
    if (!ref.current) return;
    while (ref.current.firstChild) {
      ref.current.removeChild(ref.current.firstChild);
    }
    const canvas = document.createElement('canvas');
    canvas.setAttribute('id', 'name_' + Math.random());
    canvas.setAttribute('width', '400');
    canvas.setAttribute('height', '400');
    ref.current.appendChild(canvas);
    new ChartJS(canvas, {
      type: 'scatter',
      data: {
        datasets: [
          coordinates
            ? {
                label,
                data: coordinates as any,
                backgroundColor: 'red',
                borderColor: 'red',
                pointRadius: 0.4,
                type: 'line',
                borderWidth: 0.2,
                borderDash: [5, 5],
              }
            : (undefined as any),
          ...datasets,
        ].filter(Boolean),
      },
      options: merge({ interaction: { mode: 'nearest' } }, options),
    });
  }, [coordinates, label, name, options, datasets]);

  return <div ref={ref} />;
```

```
};

export default Chart;
```

Лістинг коду 2 – Програмний код для розкладання вхідних даних

```
import React from 'react';
import { useTranslation } from 'react-i18next';
import Chart, { ChartOptionProps } from '../components/Chart';
import { TabletRecord } from 'graphql/types.generated';
import Width from '../components/Width';
import s from './Chart.module.sass';

const YAndTimeOptions: ChartOptionProps = {
  scales: {
    yAxisKey: { title: { text: 'y', display: true } },
    xAxisKey: { title: { text: 'time', display: true } },
  },
  parsing: { xAxisKey: 'time' },
};

const XAndTimeOptions: ChartOptionProps = {
  scales: {
    yAxisKey: { title: { text: 'x', display: true } },
    xAxisKey: { title: { text: 'time', display: true } },
  },
  parsing: { xAxisKey: 'time', yAxisKey: 'x' },
};

const PressureAndTimeOptions: ChartOptionProps = {
  scales: {
    yAxisKey: { title: { text: 'pressure', display: true }, beginAtZero: true },
    xAxisKey: { title: { text: 'time', display: true } },
  },
  parsing: { xAxisKey: 'time', yAxisKey: 'pressure' },
};

const DecomposeTabletChart = ({ tabletRecords }: { tabletRecords?: TabletRecord[] }) =>
{
  const { t } = useTranslation();

  if (!tabletRecords) return null;
  return (
    <>
      <div>
        <div className={s.half}>
          <Chart
            name="decomposeTabletByYAndTime"
            coordinates={tabletRecords}
            label={t('calculation.series.tablet')}
            options={YAndTimeOptions}
          />
        </div>
        <Width data={tabletRecords} parse={{ x: 'time', y: 'y' }} />
      </div>
      <div>
        <div className={s.half}>
          <Chart
            name="decomposeTabletByXAndTime"
            coordinates={tabletRecords}
            label={t('calculation.series.tablet')}
            options={XAndTimeOptions}
          />
        </div>
```

```
            <Width data={tabletRecords} parse={{ x: 'time', y: 'x' }} />
          </div>
          <div>
            <div className={s.half}>
              <Chart
                name="decomposeTabletByPressureAndTime"
                coordinates={tabletRecords}
                label={t('calculation.series.tablet')}
                options={PressureAndTimeOptions}
              />
            </div>
            <Width data={tabletRecords} parse={{ x: 'time', y: 'pressure' }} />
          </div>
        </>
      );
    };

    export default DecomposeTabletChart;
```

Лістинг коду 3 – Програмний код для парсингу вхідного файлу електроміографа

```
    import { parse } from 'date-fns';
    import Papa from 'papaparse';
    import { errors as BackendErrors } from '../../../constants/errors';
    import { ElectromyographRecord } from '../classes/ElectromyographRecord';
    import { ElectromyographPort, ElectromyographPortIndex } from '../constants';
    import { base64ToString, getStartDate } from './utils';

    class ElectromyographCalculationService {
      parseTime(dateStr: string, referenceDate: Date) {
        const parsedDate = parse(dateStr, 'H:m:s.SSS', referenceDate);
        return parsedDate;
      }

          async normalizeData(base64: string, signalPort: ElectromyographPort):
Promise<[ElectromyographRecord[], Date]> {
        const ssv = base64ToString(base64);
         const { data, errors }: { data: unknown[][]; errors: unknown[] } = Papa.parse(ssv,
{ delimiter: '\t' });

        const headers = data.filter((row) => row.length === 1).flat() as string[];
        const startDate = getStartDate(headers);
            const signalPortIndex = ElectromyographPortIndex[signalPort as keyof typeof
ElectromyographPortIndex];

        if (!signalPortIndex) throw new Error(BackendErrors.UNAVAILABLE_SIGNAL_PORT);
        if (errors.length) throw new Error(BackendErrors.TABLET_FILE_INCORRECT_FORMAT);

        const records = data
          .map((row) => {
            const time = this.parseTime(row[0] as string, startDate);
            const signal = +(row[signalPortIndex] as number);
            if (Number.isFinite(time.getTime()) && Number.isFinite(signal)) {
              const record: ElectromyographRecord = {
                time,
                relativeTime: startDate.getTime() - time.getTime(),
                signal,
              };
              return record;
            }

            return null;
```

```
            })
            .filter(Boolean) as ElectromyographRecord[];

          return [records, startDate];
        }
      }

      export default new ElectromyographCalculationService();
```

Лістинг коду 4 – Програмний код для обчислення коефіцієнтів ідентифікації

```
      import { groupBy, random } from 'lodash';
      import { ElectromyographRecord } from '../classes/ElectromyographRecord';
      import { produceJ, produceJn } from './utils';

      export class CoefficientCalculation {
        initialB: number;

        constructor(initialB: number) {
          this.initialB = initialB;
        }

        getSegmentLength(records: ElectromyographRecord[], totalSegments: number) {
                  const   absoluteTime   =   records[records.length   -   1].time.getTime()   -
records[0].time.getTime();
          return absoluteTime / totalSegments;
        }

        produceAveragePoint(records: ElectromyographRecord[]) {
          return records.reduce((acc, record) => acc + record.signal, 0) / records.length;
        }

         calculateBs({  records,  totalSegments  }:  {  records:  ElectromyographRecord[];  name:
keyof ElectromyographRecord; totalSegments: number }) {
          const segmentLength = this.getSegmentLength(records, totalSegments);
           const groupedSegments = groupBy(records, (record) => Math.ceil(record.relativeTime
/ segmentLength));

          return Object.values(groupedSegments).reduce((acc, segmentRecords) => {
            const prevValue = acc[acc.length - 1] ?? this.initialB;

                                    const     b     =     prevValue     +     (produceJ()     *
Math.pow(this.produceAveragePoint(segmentRecords), 2)) / Math.pow(produceJn(), 2);

            acc.push(b);

            return acc;
          }, []);
        }
      }

      export default new CoefficientCalculation(random(0, 1));
```

Лістинг коду 5 – Програмний код для сервісу авторизації

```
      import { DocumentType } from '@typegoose/typegoose';
      import  {  JWTTokenParams,  JWTTokenPayload,  LoginUserProps,  LoginWithRefreshTokenProps  }
from './AuthService.types';
      import jwt from 'jsonwebtoken';
      import { AuthenticationError, ForbiddenError } from 'apollo-server-express';
```

```
import SessionModel from '../models/SessionModel';
import UserModel from '../../user/models/UserModel';
import SessionsService from './SessionsService';
import { TokenType } from './AuthService.types';
import { jwtOptionsByTokenType, jwtSecretByTokenType } from '../constants';
import { User } from '../../user/classes/User';
import { errors } from '../../../constants/errors';
class AuthService {
   authenticateWithEmailAndPassword = async ({ email, password }: Pick<LoginUserProps,
'email' | 'password'>) => {
  const userWithOnlyServices = await UserModel.findByUsername(email).select('services');
    if (!userWithOnlyServices)
      throw new Error(errors.INCORRECT_CREDENTIALS);
    const auth = await userWithOnlyServices.authenticate(password);
    if (!auth.user) throw new Error(auth.error);
    const user = await UserModel.findByUsername(email);
    return user;
  };
   createUserSession = async ({ user, rememberMe, ip }: { user: DocumentType<User>;
rememberMe: boolean; ip: LoginUserProps['ip'] }) => {
    const userId = user.id;
    const session = await SessionModel.createSession({ userId, ip });
    const sessionId = session.id;
    const jwtPayload = { sub: userId, email: user.email, sessionId };
    const accessToken = this.generateToken(jwtPayload, TokenType.access);
     const refreshToken = rememberMe ? this.generateToken(jwtPayload, TokenType.refresh)
: undefined;
    return { accessToken, refreshToken };
  };
  login = async ({ email, rememberMe, password, ip }: LoginUserProps) => {
    const user = await this.authenticateWithEmailAndPassword({ email, password });
     if (user.deactivated) throw new ForbiddenError('You are not allowed to access this
application');
       const { accessToken, refreshToken } = await this.createUserSession({ user,
rememberMe, ip });
   return { accessToken, refreshToken, user };
  };
  loginWithRefreshToken = async ({ refreshToken, ip }: LoginWithRefreshTokenProps) => {
    const tokenData = await this.verifyToken(refreshToken, TokenType.refresh);
          await SessionModel.updateSession(tokenData.sessionId as string, { ip,
lastActivityDate: new Date() });
    const { sub, email, sessionId } = tokenData;
    return this.generateToken({ sub, email, sessionId }, TokenType.access);
  };
```

```
    logout = async (accessToken: string) => {
      await SessionsService.logout(accessToken);
    };
    getUnixExpirationOfToken(token: string) {
      const decoded = jwt.decode(token) as JWTTokenParams;
      if (!decoded || !decoded.exp) return null;
      return decoded.exp * 1000;
    }
    generateToken = (payload: JWTTokenPayload, type: TokenType) => {
      const secret = jwtSecretByTokenType[type];
      const options = jwtOptionsByTokenType[type];
      return jwt.sign(payload, secret, options);
    };
    verifyToken = async (token: string, type: TokenType = TokenType.access) => {
      const secret = jwtSecretByTokenType[type];
      const tokenData = jwt.verify(token, secret) as JWTTokenParams;
      const session = await SessionModel.findById(tokenData.sessionId);
      const skipSessionCheck = TokenType.resetPassword === type;
      if (!skipSessionCheck && !session)
        throw new AuthenticationError(errors.TOKEN_EXPIRED);
      return tokenData;
    };
     changePassword = async (token: string, currentPassword: string, newPassword: string):
Promise<boolean> => {
      const { email } = await this.verifyToken(token, TokenType.access);
      try {
           const user = await this.authenticateWithEmailAndPassword({ email, password:
currentPassword });
        await user.setPassword(newPassword);
        await user.save();
        return true;
      } catch (error: any) {
        if (error.message.includes(errors.INCORRECT_CREDENTIALS))
          throw new Error(errors.INCORRECT_PASSWORD);
        throw error;
      }
    };
  }
  export default new AuthService();
```

Додаток Г – Диск із кваліфікаційною роботою магістра