МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри,

д-р техн. наук, проф.

_____Ірина ЖУРАВСЬКА

« __ » _____ 2025 р.

МАГІСТЕРСЬКА РОБОТА

«Інформаційно-аналітична система для візуалізації даних медичних досліджень з використанням Dash (Plotly) фреймворку»

Спеціальність 123 Комп'ютерна інженерія

123 – КМР.01 – 605.22450501

Студент _____ Антон ГРІДНЄВ
підпис
«__» _____ 202__ р.

Керівник PhD, доцент (б. в. з.) _____ Євген ДАРНАПУК
підпис
«__» _____ 202__ р.

Миколаїв – 2025

АНОТАЦІЯ

до кваліфікаційної магістерської роботи
«Інформаційно-аналітична система для візуалізації даних медичних досліджень з використанням Dash (Plotly) фреймворку»
Студент 605 гр.: Гріднєв А. Ю.
Керівник: PhD, доцент (б. в. з.) Дарнапук Євген Сергійович

Кваліфікаційна магістерська робота присвячена розробці інформаційно-аналітичної системи для візуалізації медичних даних з використанням фреймворку Dash (Plotly). В умовах зростаючого обсягу медичних даних, що генеруються лікарнями, лабораторіями та іншими медичними установами, виникає потреба у створенні ефективних інструментів для їх обробки та візуалізації. Метою роботи є розробка інтерактивної системи для аналізу та візуалізації фізіологічних показників пацієнтів, що дозволить медичним працівникам оперативно реагувати на зміни у стані здоров'я пацієнтів.

Об'єктом дослідження є процеси збору, обробки та візуалізації медичних даних, а предметом – методи та програмні засоби реалізації інформаційно-аналітичної системи з використанням фреймворку Dash (Plotly). Метою роботи є розробка системи, яка забезпечить високий рівень інтерактивності та гнучкості для користувачів без глибоких технічних знань.

Практична значимість роботи полягає в створенні інструменту для аналізу та візуалізації медичних даних, що підвищує ефективність роботи медичних працівників та дозволяє покращити якість медичних досліджень. Наукова новизна полягає у застосуванні сучасних вебтехнологій для візуалізації медичних даних у реальному часі за допомогою Dash, що дозволяє забезпечити простоту інтеграції з різноманітними джерелами даних і підвищити ефективність процесу прийняття клінічних рішень.

Основними результатами роботи є: проведення аналізу існуючих рішень для візуалізації медичних даних, розробка архітектури інформаційно-аналітичної системи, реалізація функціональних модулів для інтерактивної візуалізації медичних показників, а також тестування та оцінка ефективності розробленої системи на реальних медичних наборах даних.

Робота містить 90 с., 20 рис., 4 табл., а також список із 21 використаних джерел.

Ключові слова: *інформаційно-аналітична система, медичні дані, візуалізація, Dash, Plotly, інтерактивні графіки, Python.*

ABSTRACT

of the Master’s Thesis

" Information and Analytical System for Visualization of Medical Research Data Using the Dash (Plotly) Framework"

Student: Anton Gridnev

Supervisor: Yevhen Darnapuk, PhD, Associate Professor

The Master's thesis is dedicated to the development of an information-analytical system for visualizing medical data using the Dash (Plotly) framework. In the context of the growing volume of medical data generated by hospitals, laboratories, and other medical institutions, there is an increasing need to create effective tools for processing and visualizing this data. The goal of this work is to develop an interactive system for analyzing and visualizing patients' physiological indicators, enabling medical professionals to respond promptly to changes in patients' health status.

The object of the research is the processes of collecting, processing, and visualizing medical data, while the subject is the methods and software tools for implementing an information-analytical system using the Dash (Plotly) framework. The aim of the work is to develop a system that ensures a high level of interactivity and flexibility for users with no deep technical knowledge.

The practical significance of the work lies in the creation of a tool for analyzing and visualizing medical data that improves the efficiency of medical professionals and enhances the quality of medical research. The scientific novelty lies in the application of modern web technologies for real-time visualization of medical data using Dash, which ensures ease of integration with various data sources and enhances the clinical decision-making process.

The main results of the work include: an analysis of existing solutions for visualizing medical data, the development of the architecture of the information-analytical system, the implementation of functional modules for interactive visualization of medical indicators, as well as testing and evaluating the effectiveness of the developed system on real medical datasets.

The thesis consists of 90 pages, 20 figures, 4 tabl. and a list of 21 references.

Keywords: *information-analytical system, medical data, visualization, Dash, Plotly, interactive graphs, Python.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ЕКГ – електрокардіограма

ІАС – інформаційно-аналітична система

BI – Business Intelligence

CDSS – Clinical Decision Support Systems

EMR – Electronic Health Records

IoT – Internet of Things

DSP – Digital Signal Processing

WFDB – Waveform Database Format

SCP-ECG – Standard Communication Protocol for ECG

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується стрімким зростанням обсягів даних, що генеруються у різних сферах людської діяльності. Однією з найбільш динамічних та суспільно важливих галузей, у якій збір, обробка та аналіз даних мають вирішальне значення, є медицина. За оцінками експертів, обсяг медичних даних подвоюється кожні два роки, що створює як нові можливості для діагностики та лікування, так і серйозні виклики щодо ефективної обробки та інтерпретації цієї інформації. Електрокардіограми, показники оксигенації крові, рівень глюкози, артеріальний тиск та інші життєво важливі параметри фіксуються безперервно, формуючи великі масиви структурованих та неструктурованих даних.

Однією з ключових проблем сучасної медичної інформатики є розрив між технічними можливостями збору даних та спроможністю медичного персоналу ефективно аналізувати отриману інформацію для прийняття клінічних рішень. Традиційні методи представлення медичних даних у вигляді таблиць та статичних графіків часто виявляються недостатніми для виявлення складних патернів, трендів та аномалій у поведінці фізіологічних показників пацієнтів. Це особливо критично в умовах інтенсивної терапії, де своєчасне виявлення відхилень може бути життєво важливим.

Розвиток технологій інтерактивної візуалізації даних відкриває нові перспективи для медичної аналітики. Сучасні фреймворки, зокрема Dash на базі бібліотеки Plotly, надають потужний інструментарій для створення аналітичних застосунків, що поєднують інтуїтивно зрозумілий інтерфейс користувача з високою продуктивністю обробки даних та можливостями інтерактивного дослідження. Такі системи дозволяють медичним фахівцям без глибоких знань програмування працювати з великими обсягами даних, застосовувати статистичні методи аналізу та отримувати візуальні представлення, що полегшують інтерпретацію результатів.

Особливу актуальність набуває інтеграція різнорідних джерел медичних даних в єдину інформаційно-аналітичну систему. Відкриті бази даних, такі як PhysioNet, містять величезні колекції реальних медичних записів, що можуть використовуватися для розробки, тестування та валідації аналітичних систем. Використання цих ресурсів дозволяє створювати рішення, що базуються на реальних клінічних даних та враховують специфіку різних типів медичних досліджень.

Актуальність теми – дослідження зумовлена кількома чинниками. По-перше, зростання обсягів медичних даних вимагає розробки ефективних засобів їх аналізу та представлення. Традиційні табличні або текстові форми подання не дозволяють швидко виявляти закономірності, відхилення та тренди у показниках здоров’я пацієнтів. По-друге, у зв’язку з розвитком телемедицини та дистанційного моніторингу стану пацієнтів зростає потреба у інтерфейсах, які дозволяють медичним працівникам аналізувати результати досліджень у реальному часі. По-третє, відкриті дані, зокрема бази PhysioNet, дають змогу проводити дослідження без прямого доступу до пацієнтів, що особливо важливо для освітніх, наукових та експериментальних цілей. Таким чином, розробка системи для інтерактивної візуалізації медичних даних є актуальним завданням, яке поєднує аспекти програмування, обробки біомедичних сигналів та аналітичного моделювання.

Об’єкт дослідження – процеси збору, обробки та візуалізації даних медичних досліджень.

Предмет дослідження – методи та програмні засоби реалізації інформаційно-аналітичної системи візуалізації медичних даних із використанням фреймворку Dash (Plotly).

Мета роботи полягає у розробленні інформаційно-аналітичної системи для інтерактивної візуалізації медичних показників на основі сучасних вебтехнологій і бібліотек Python, що дозволяє підвищити ефективність аналізу медичних досліджень.

Для досягнення поставленої мети необхідно розв’язати такі основні **завдання**:

1) дослідити сучасні підходи до обробки медичних даних і методи їхньої візуалізації;

2) провести аналіз існуючих інформаційно-аналітичних систем, що застосовуються у медичній галузі, з метою визначення їхніх переваг і недоліків;

3) обґрунтувати вибір технологічного стеку для реалізації системи (Python, Dash, Plotly, Pandas, Numpy, SQLite тощо);

4) розробити архітектуру інформаційно-аналітичної системи для обробки та візуалізації медичних показників;

5) реалізувати програмний модуль збору, аналізу та візуалізації даних на основі dash;

6) провести тестування роботи системи на реальних або відкритих медичних наборах даних;

7) оцінити ефективність розробленої системи шляхом порівняння її можливостей із існуючими аналогами та визначити перспективи подальшого розвитку системи, зокрема можливість інтеграції елементів машинного навчання для прогнозування медичних показників.

Розробка такої системи сприятиме підвищенню якості аналізу медичних даних, скороченню часу на обробку інформації та забезпеченню наочності результатів досліджень.

У роботі використовуються такі методи: аналіз літературних джерел і відкритих баз даних, методи статистичної обробки даних, алгоритми фільтрації та нормалізації біосигналів, методи побудови інтерактивних візуалізацій за допомогою бібліотек Plotly та Dash, а також методи тестування програмного забезпечення. Для розроблення системи застосовуються мови програмування Python, а також допоміжні бібліотеки Pandas, NumPy, Matplotlib, Scikit-learn.

Результати роботи мають практичну цінність для фахівців у галузі медичної інформатики, розробників програмних засобів та науковців, які працюють із медичними даними. Розроблена система може бути використана для:

- дослідження біомедичних показників і трендів у фізіологічних даних;
- навчальних цілей під час підготовки фахівців з комп’ютерних наук і біоінформатики;
- створення базових рішень для систем моніторингу стану здоров’я пацієнтів у медичних закладах або наукових лабораторіях.

Дана кваліфікаційна робота була представлена на XXVIII Всеукраїнській науково-практичній конференції «Могилянські читання – 2025: досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти» (листопад 2025 р., Миколаїв).

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ВІЗУАЛІЗАЦІЇ МЕДИЧНИХ ДАНИХ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Еволюція інформаційно-аналітичних систем у медичній сфері

Інформаційно-аналітичні системи (ІАС) є невід’ємним елементом сучасного цифрового суспільства. Їх розвиток тісно пов’язаний із загальною еволюцією комп’ютерних технологій, систем зберігання та обробки даних, а також із зростанням потреби у швидкому прийнятті рішень на основі великої кількості інформації. У медичній сфері цей процес набув особливого значення, адже якість медичного обслуговування, точність діагностики та ефективність лікування дедалі більше залежать від оперативності та достовірності аналітичних даних.

Перші спроби автоматизації обліку медичної інформації з’явилися ще у 1960-х роках, коли почали розроблятися перші госпітальні інформаційні системи (HIS). Вони дозволяли автоматизувати реєстрацію пацієнтів, ведення електронних карток і зберігання історій хвороб. Такі системи працювали на великих обчислювальних машинах і використовувалися переважно у великих клініках. Проте їхні можливості були обмежені через відсутність єдиних стандартів обміну даними, складність обслуговування та високу вартість впровадження.

У 1970–1980-х роках розвиток персональних комп’ютерів і локальних мереж започаткував новий етап у розвитку медичних інформаційних систем. З’явилися спеціалізовані програмні засоби для ведення електронних медичних записів (EMR), що забезпечували зберігання структурованої інформації про пацієнтів (рис.1.1). На цьому етапі ІАС були орієнтовані переважно на адміністративно-облікові функції – реєстрацію пацієнтів, облік медикаментів, формування статистичної звітності тощо.

Рисунок 1.1 – Приклад ранньої медичної інформаційної системи [1]

Суттєвий прорив відбувся у 1990-х роках, коли розвиток інтернету та вебтехнологій відкрив нові можливості для обміну медичними даними між різними установами. Саме тоді почали впроваджуватися концепції електронних медичних записів (Electronic Health Records, EHR), які охоплювали не лише дані з конкретного закладу, а й інформацію про пацієнта з різних джерел. З’явилися перші спроби створення інтегрованих інформаційно-аналітичних систем, які поєднували медичні бази даних, лабораторні результати, фармакологічну інформацію та клінічні довідники.

На початку XXI століття медичні інформаційні системи еволюціонували від простих систем зберігання даних до повноцінних аналітичних платформ. Основним чинником цього переходу стало зростання обсягів медичних даних, які неможливо було ефективно опрацьовувати традиційними методами. Саме тоді з’явилися технології Data Warehouse та Business Intelligence (BI), які дозволяли агрегувати великі масиви даних, виконувати статистичний аналіз і формувати інтерактивні звіти (рис. 1.2).

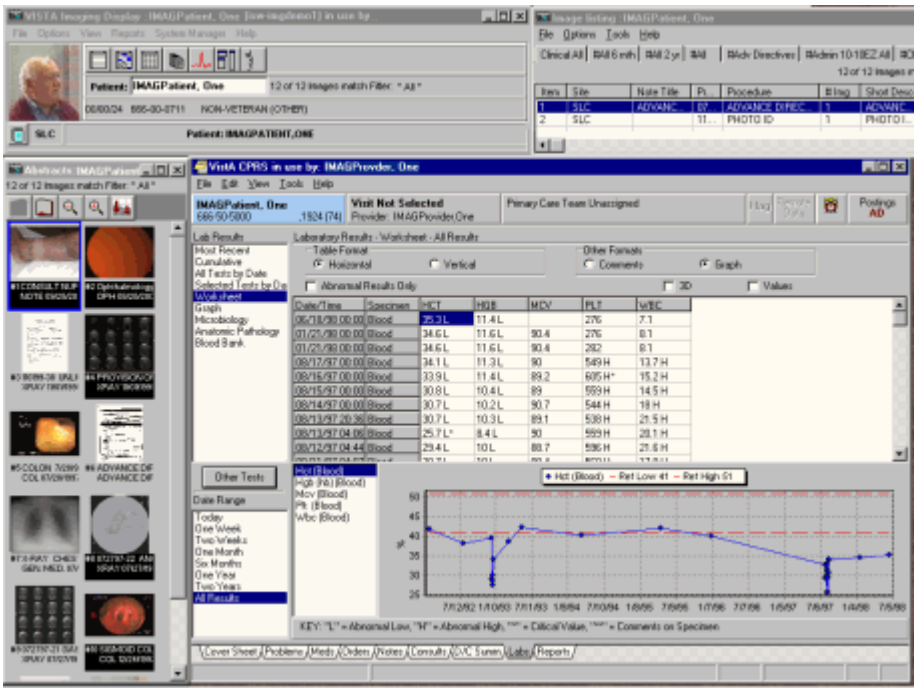

Рисунок 1.2 – Приклад інтерфейсу електронних медичних записів [2]

У медичній практиці це означало можливість побудови клінічних інформаційно-аналітичних систем, які не лише зберігали дані, але й підтримували прийняття клінічних рішень (Clinical Decision Support Systems, CDSS). Такі системи аналізували показники стану пацієнтів, визначали потенційні ризики, нагадували лікарям про необхідність проведення певних процедур чи корекції лікування. З’явилися перші прототипи інтелектуальних систем, що використовували алгоритми машинного навчання для прогнозування результатів лікування або ранньої діагностики захворювань. Важливим кроком на цьому етапі стало впровадження міжнародних стандартів обміну медичними даними, зокрема HL7 та DICOM. Завдяки цим стандартам стало можливим об’єднання інформації з різних медичних пристроїв, лабораторій і клінік у єдиний аналітичний простір. Це значно спростило обмін інформацією між медичними закладами та створило передумови для розвитку телемедицини.

З початком 2010-х років медична інформатика вступила в епоху Big Data, коли обсяги медичної інформації почали вимірюватися терабайтами й

петабайтами. До традиційних клінічних даних додалися потоки інформації з носимих пристроїв, біосенсорів, мобільних застосунків для здоров’я та інтернету медичних речей. Це потребувало нових підходів до обробки та аналізу даних, здатних працювати в реальному часі. У цей період активно розвиваються хмарні обчислення, які дозволяють зберігати та обробляти великі обсяги даних без необхідності утримання власних серверів. На основі хмарних технологій з’являються розподілені медичні аналітичні системи, що забезпечують доступ до даних з будь-якої точки світу.

Такі рішення активно використовуються в телемедицині, дистанційному моніторингу пацієнтів, аналітиці ефективності лікування та епідеміологічних дослідженнях. Паралельно відбувається зростання ролі візуалізації даних у медицині. Складні аналітичні моделі потребують зрозумілого представлення результатів, особливо для лікарів, які не мають глибокої технічної підготовки. Графічне подання інформації у вигляді діаграм, часових рядів, теплових карт і тривимірних моделей полегшує інтерпретацію результатів і підвищує швидкість прийняття рішень (рис. 1.3).

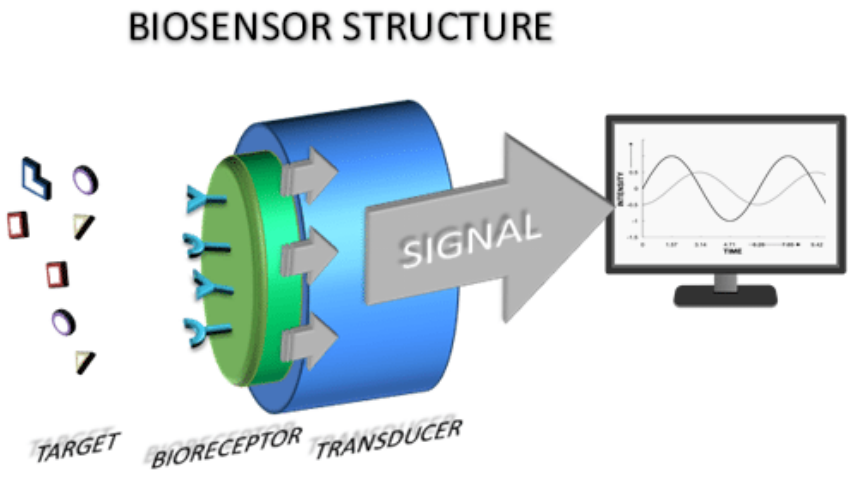

Рисунок 1.3 – Схематичне відображення структури біосенсора [3]

Сьогодні інформаційно-аналітичні системи у медичній сфері інтегрують досягнення багатьох технологічних напрямів – великих даних, штучного інтелекту, машинного навчання, хмарних обчислень і візуальної аналітики.

Основний акцент робиться на інтерактивності: користувачі можуть у реальному часі змінювати параметри аналізу, фільтрувати дані, обирати часові діапазони, досліджувати залежності між показниками. Важливу роль у цьому відіграють відкриті фреймворки для побудови вебвізуалізацій, такі як Dash, Plotly, Bokeh, Streamlit тощо. Вони дозволяють поєднати аналітичні можливості Python з сучасними інтерфейсними рішеннями, створюючи зручні інформаційні панелі для аналізу медичних даних. Особливо популярним став фреймворк Dash (Plotly), який дозволяє без знань JavaScript створювати повноцінні інтерактивні вебзастосунки. На його основі реалізовано десятки відкритих проєктів для візуалізації даних серцевих ритмів, рівня глюкози, частоти дихання, активності мозку та інших біомедичних показників (рис. 1.4).

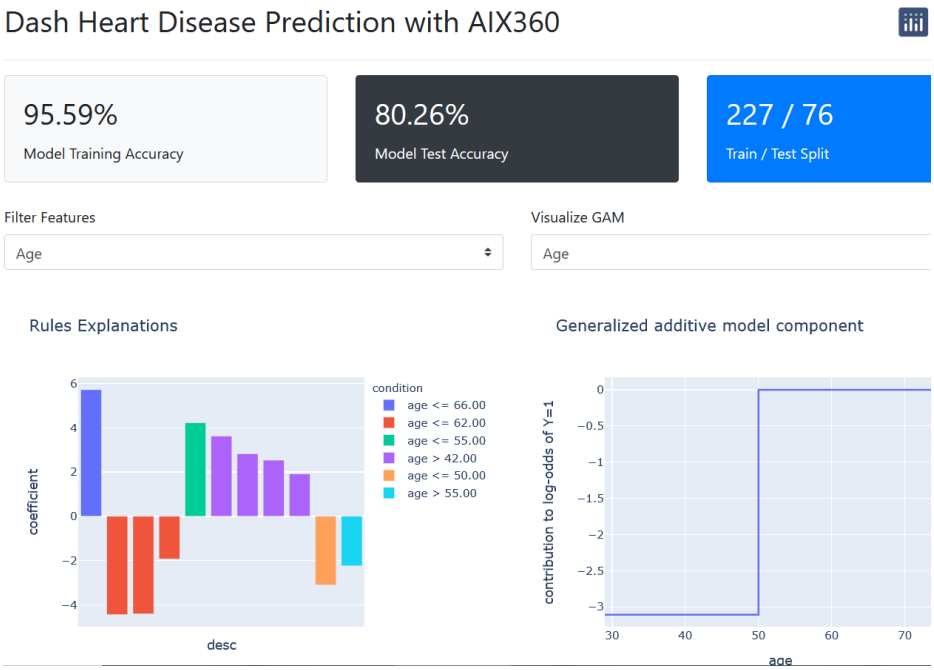

Рисунок 1.4 – Приклад візуалізації моделі прогнозування хвороби серця [6]

Іншим трендом сучасного етапу є інтелектуалізація аналітичних процесів. У системи інтегруються алгоритми машинного навчання для класифікації медичних зображень, виявлення аномалій у фізіологічних даних, прогнозування ризиків захворювань. Усе це робить інформаційно-аналітичні системи не лише інструментом для візуалізації, а й повноцінним засобом підтримки клінічних рішень.

Сучасні тенденції розвитку ІАС у медицині спрямовані на створення персоналізованих медичних платформ, що враховують індивідуальні особливості організму пацієнта. Застосування штучного інтелекту та предиктивної аналітики дозволяє здійснювати раннє виявлення патологічних змін і рекомендувати оптимальні схеми лікування. У цьому контексті особливого значення набуває візуальна аналітика – поєднання традиційних методів статистики з інструментами візуалізації, які дозволяють глибше розуміти структуру даних і взаємозв’язки між ними.

Подальший розвиток ІАС передбачає інтеграцію даних з різнорідних джерел – електронних карток пацієнтів, результатів біохімічних аналізів, показників носимих пристроїв, генетичних досліджень тощо. У майбутньому такі системи зможуть автоматично формувати комплексні профілі здоров’я людини, що використовуватимуться для діагностики, профілактики та лікування захворювань.

Таким чином, еволюція інформаційно-аналітичних систем у медичній сфері – це шлях від простих облікових програм до інтелектуальних аналітичних комплексів, здатних працювати з великими обсягами даних у реальному часі. Розвиток фреймворків, таких як Dash (Plotly), відкриває нові можливості для створення зручних, інтерактивних і візуально привабливих засобів аналізу медичної інформації. Це створює технологічне підґрунтя для підвищення ефективності систем охорони здоров’я та розвитку цифрової медицини загалом.

1.2 Сучасні підходи до обробки та візуалізації медичних даних

Швидке зростання обсягів медичних даних, що генеруються внаслідок роботи лікарень, лабораторій, клінічних досліджень, мобільних пристроїв та сенсорних систем, обумовлює необхідність розробки ефективних методів їх обробки та представлення. Медичні дані є різнорідними за структурою, форматом і джерелом походження: це числові значення, текстові звіти,

зображення, сигнали біосенсорів, дані про генетичні маркери тощо. Тому сучасні підходи до їх аналізу поєднують методи статистики, машинного навчання, біоінформатики та інформаційної візуалізації.

Медичні дані мають низку специфічних характеристик, які ускладнюють процес їх опрацювання:

- великий обсяг та різнорідність. У сучасних медичних системах накопичуються дані з десятків джерел – від лабораторних аналізів до носимих пристроїв. Вони можуть бути як структурованими (таблиці, бази даних), так і неструктурованими (зображення, текстові записи, звукові сигнали);

- чутливість інформації. Дані про стан здоров’я є конфіденційними, тому обробка повинна відповідати вимогам захисту персональних даних (зокрема, HIPAA, GDPR);

- висока динамічність. Показники життєдіяльності пацієнтів змінюються в реальному часі, що вимагає оперативної обробки потоків даних;

- неоднорідна якість. Дані можуть містити пропуски, шум, артефакти, похибки вимірювання, що знижує точність аналізу без попереднього очищення та нормалізації.

Відповідно до цих особливостей сучасна обробка медичних даних включає кілька ключових етапів: збір, фільтрацію, нормалізацію, інтеграцію, аналіз та візуалізацію.

На першому етапі здійснюється агрегація даних з різних джерел: лабораторних інформаційних систем (LIS), електронних медичних карток (EHR/EMR), сенсорних пристроїв, мобільних застосунків або відкритих наукових баз (наприклад, PhysioNet, MIMIC-III, Kaggle Health Data). Попередня обробка охоплює кілька процедур:

- очищення даних – видалення пропущених або некоректних значень, усунення дублювань, виправлення форматів;

- фільтрація шумів – особливо важлива для біомедичних сигналів (електрокардіограми, пульсограми, фотоплетизмограми), де застосовують цифрові фільтри (наприклад, фільтри Баттерворта, фільтри Калмана);

- нормалізація та стандартизація – приведення показників до єдиної шкали для подальшого порівняння;

- кодування та анонімізація – заміна персональних даних унікальними ідентифікаторами для забезпечення конфіденційності;

- інтеграція даних – об’єднання інформації з різних систем за допомогою стандартів HL7, FHIR (Fast Healthcare Interoperability Resources), DICOM.

Для реалізації цих етапів широко застосовуються бібліотеки мови Python – Pandas, NumPy, SciPy, OpenCV, BioPython. Вони забезпечують ефективну обробку числових і текстових наборів даних, а також базову аналітику. Візуалізація є ключовим етапом аналітичного процесу, оскільки саме наочне представлення результатів дозволяє лікарю або досліднику швидко зрозуміти тенденції, закономірності й відхилення. Сучасні підходи до візуалізації базуються на таких принципах:

- інтерактивність. Користувач має змогу змінювати параметри графіків, обирати часові діапазони, фільтрувати дані, масштабувати діаграми;

- динамічність. Візуалізації оновлюються в режимі реального часу (наприклад, моніторинг ЕКГ або SpO_2);

- мультимодальність. Комбінування різних типів даних – графіків, зображень, таблиць, теплових карт, 3D-моделей;

- веборієнтованість. Використання технологій HTML5, CSS3, JavaScript, React, що забезпечують доступність візуалізацій у браузері.

До найбільш популярних бібліотек для візуалізації даних у Python належать Matplotlib, Seaborn, Plotly, Bokeh та Dash. Бібліотека Plotly дозволяє

створювати інтерактивні графіки з підтримкою панорамування, масштабування, наведення курсору та вибору діапазонів. На її основі побудований фреймворк Dash, який поєднує можливості Python і вебтехнологій, надаючи інструменти для створення повноцінних вебзастосунків без використання JavaScript. Це відкриває широкі можливості для інтерактивної медичної аналітики, наприклад:

- відображення динаміки серцевого ритму у вигляді часових рядів;
- порівняння показників пацієнтів за різними параметрами;
- побудову теплових карт для візуалізації активності органів;
- інтерактивну фільтрацію даних за демографічними або клінічними ознаками.

Такі рішення підвищують інформативність результатів і дають змогу медичним працівникам швидко приймати обґрунтовані рішення. Попри значний прогрес, сучасні системи обробки медичних даних стикаються з низкою викликів:

- інтеграція даних із різних джерел та форматів;
- необхідність дотримання вимог безпеки й етичних норм;
- забезпечення масштабованості системи при зростанні обсягів даних;
- зручність інтерфейсу для користувачів без технічної підготовки.

Перспективними напрямами розвитку є:

- впровадження алгоритмів глибинного навчання для аналізу зображень і сигналів;
- використання штучного інтелекту для автоматичного виявлення патологій;
- інтеграція з IoT-пристроями для безперервного збору даних;

- розвиток візуальної аналітики, яка поєднує візуалізацію з аналітичними методами у єдиному інтерфейсі.

Таким чином, сучасні підходи до обробки та візуалізації медичних даних спрямовані на створення інтерактивних, гнучких та інтелектуальних систем, здатних працювати в режимі реального часу. Використання фреймворку Dash (Plotly) у таких системах дозволяє поєднати високий рівень аналітичної потужності Python із зручним вебінтерфейсом, що робить ці технології одним із найперспективніших напрямів розвитку цифрової медицини.

1.3 Аналіз існуючих рішень

Розвиток інформаційних технологій у сфері охорони здоров’я спричинив появу великої кількості програмних рішень, призначених для збору, аналізу та візуалізації медичних даних. Ці системи різняться за архітектурою, функціональністю, масштабом та цільовим призначенням: від внутрішньолікарняних електронних медичних карток до глобальних аналітичних платформ, що обробляють дані про мільйони пацієнтів. У даному підрозділі розглянемо найпоширеніші категорії рішень і здійснимо їх порівняльний аналіз, визначивши переваги, недоліки та можливості використання у проєкті.

Epic Systems є одним із провідних світових постачальників електронних медичних записів. Її рішення орієнтовані на великі лікарні та клінічні мережі. Система підтримує інтеграцію з лабораторними модулями, фармацевтичними базами, має аналітичний модуль Epic Cogito для побудови звітів і візуалізацій (рис. 1.5).

Переваги:

- повна інтеграція з EHR;
- надійний рівень безпеки;
- модульна архітектура.

Недоліки:

- закритість коду;
- висока вартість ліцензії;
- обмежена гнучкість для користувацької аналітики.

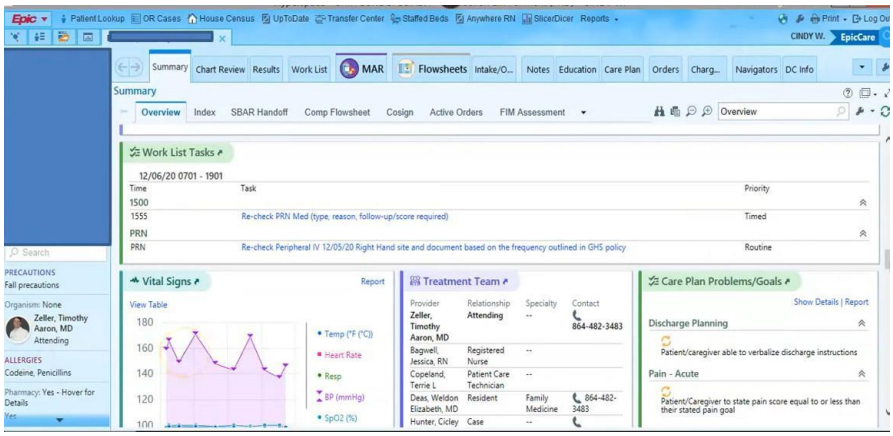

Рисунок 1.5 – Приклад медичного запису в Epic Systems [8]

IBM Watson Health

Ця платформа поєднує технології штучного інтелекту, машинного навчання та обробки природної мови для аналізу клінічних даних. Watson Health застосовується для прогнозування ризиків захворювань, персоналізованої медицини та аналітики великих масивів даних (рис. 1.6).

Переваги:

- потужні алгоритми AI;
- можливість інтеграції з різними джерелами даних;
- зручні аналітичні панелі.

Недоліки:

- складність налаштування;
- потреба у значних обчислювальних ресурсах;
- відсутність безкоштовної версії.

Рисунок 1.6 – Використання ШІ дозволяє IBM Watson Health попереджати виникнення раку та складати індивідуальний план лікування для кожного пацієнта [9]

Microsoft Power BI

BI-система, яка активно використовується у медичних організаціях для створення інтерактивних звітів (рис. 1.7). Підтримує підключення до баз даних, Excel, API медичних сервісів, має широкі можливості візуалізації.

Переваги:

- легка інтеграція з Microsoft 365;
- гнучкі графічні панелі;
- налаштовувані дашборди.

Недоліки:

- необхідність хмарного підключення для деяких функцій;
- обмежена масштабованість при великих обсягах медичних даних;
- складність кастомізації у медичних специфікаціях.

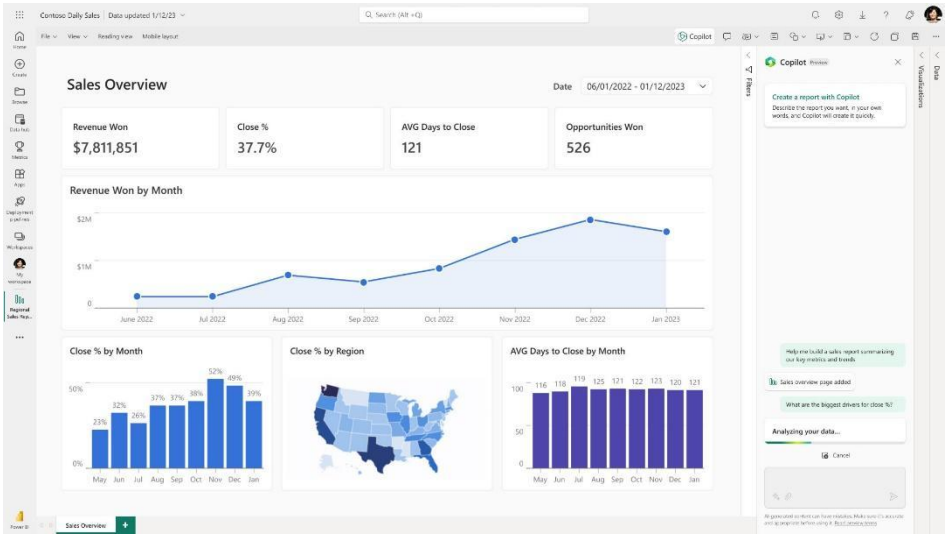

Рисунок 1.7 – Приклад візуалізації даних у Microsoft Power BI [10]

OpenMRS – відкрита платформа для зберігання та аналізу медичних даних, що використовується у країнах, які розвиваються (рис. 1.8). Система дозволяє лікарням впроваджувати власні модулі, адаптуючи функціонал до локальних потреб.

Переваги:

- open-source;
- гнучкість у налаштуванні;
- підтримка міжнародних стандартів HL7 і FHIR;

Недоліки:

- складна установка;
- обмежена функціональність для візуалізації;
- потребує значних ресурсів на технічне обслуговування.

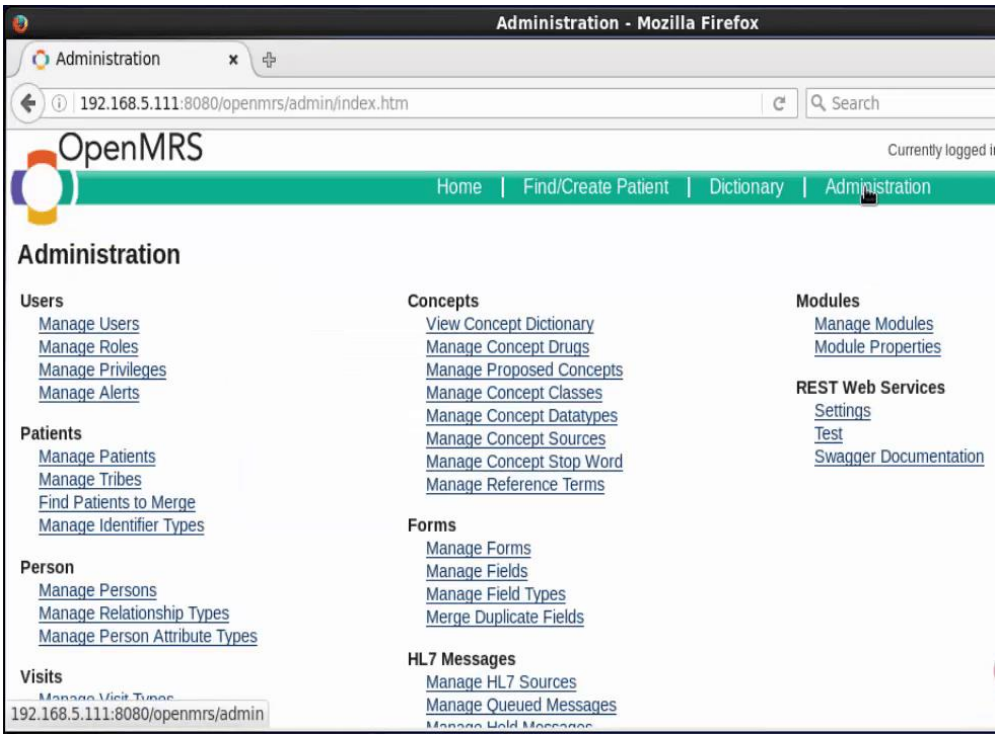

Рисунок 1.8 – Приклад інтерфейсу OpenMRS [11]

OHDSI – міжнародна ініціатива з відкритим кодом, яка забезпечує аналітику медичних даних із використанням єдиної структури OMOP Common Data Model. Використовується у великих наукових дослідженнях для уніфікації даних з різних джерел.

Переваги:

- стандартизація структури даних;
- підтримка складних статистичних методів;
- активна спільнота.

Недоліки:

- висока складність розгортання;
- потреба у фахівцях з R і SQL;
- неінтерактивний інтерфейс візуалізації.

VitalDB – це відкрита платформа, розроблена Сеульським університетом, для збору та візуалізації даних життєдіяльності пацієнтів під час операцій. Підтримує реальний час і багатоканальні біомедичні сигнали.

Переваги:

- робота з потоковими даними;
- можливість аналізу багатоканальних сигналів;
- інтеграція з Python.

Недоліки:

- відсутність гнучких засобів для побудови власних аналітичних панелей;
- обмежені можливості кастомізації.

Проведений аналіз показує, що більшість комерційних систем (Epic, IBM Watson, Power BI) орієнтовані на великі клінічні структури, мають обмежену відкритість та високу вартість. Відкриті системи (OpenMRS, OHDSI, VitalDB) більш гнучкі, проте складні у розгортанні та мають обмежену інтерактивність візуалізацій. Інструменти на основі Python, зокрема Dash (Plotly), надають можливість створити легку, інтерактивну, гнучку та масштабовану систему, яка:

- дозволяє інтегрувати дані з різних джерел (CSV, SQL, API);
- забезпечує динамічну візуалізацію результатів у реальному часі;
- підтримує розгортання у вебсередовищі;
- може бути адаптована під специфічні потреби медичних досліджень.

Таким чином, використання Dash (Plotly) є доцільним для реалізації сучасної інформаційно-аналітичної системи, орієнтованої на візуалізацію медичних даних. Цей фреймворк поєднує відкритість, інтерактивність та технологічну сумісність із Python-екосистемою, що робить його оптимальним вибором для дослідницьких і клінічних проєктів.

1.4 Постановка завдання дослідження

У попередніх підрозділах було розглянуто еволюцію інформаційно-аналітичних систем у медичній сфері, сучасні методи обробки та візуалізації даних, а також проведено аналіз існуючих рішень. Результати аналізу показали, що, незважаючи на значну кількість наявних систем для збору, аналізу та візуалізації медичних даних, більшість із них мають певні обмеження: високі вимоги до апаратних ресурсів, закритість вихідного коду, обмежену можливість кастомізації та складність інтеграції з сучасними відкритими бібліотеками Python. Сучасні тенденції у розвитку медичної аналітики потребують створення інструментів, які забезпечують високу гнучкість, інтерактивність і доступність. З огляду на це, актуальним завданням є розробка інформаційно-аналітичної системи нового типу, що дозволить ефективно обробляти та візуалізувати медичні дані у зручному для дослідників і лікарів форматі, використовуючи відкриті технології.

1.5 Формулювання специфікації вимог до системи

Розробка інформаційно-аналітичної системи для візуалізації медичних даних ставить перед собою ряд ключових функціональних вимог, що визначають її ефективність та зручність для користувачів. Система повинна забезпечити збирання медичних даних з різноманітних джерел, включаючи бази даних, сенсорні пристрої, зовнішні API та інші медичні системи. Це дозволить зібрати всю необхідну інформацію для подальшого аналізу.

Один із основних аспектів системи – це здатність проводити складний аналіз медичних даних. Використання передових статистичних методів і алгоритмів машинного навчання дозволить виявляти закономірності та тренди, які можуть бути корисні для прийняття клінічних рішень. Для ефективної інтерпретації результатів система повинна надавати інтуїтивно зрозумілий інтерфейс для візуалізації цих даних у вигляді графіків, діаграм та

часових рядів, що забезпечить лікарям і дослідникам швидке сприйняття складної медичної інформації.

Інтерактивність є важливим елементом цієї системи. Користувачі повинні мати можливість налаштовувати параметри візуалізацій, змінювати часові діапазони, фільтрувати дані та масштабувати діаграми в залежності від їхніх потреб. Це забезпечить гнучкість та дозволить легко налаштувати систему для вирішення конкретних завдань. Крім того, система повинна бути здатною до оновлення даних у реальному часі, що дозволяє відображати зміни медичних показників без затримок, що є критично важливим для прийняття оперативних рішень у медицині.

Що стосується нефункціональних вимог, система повинна володіти високою продуктивністю, здатною обробляти великі обсяги даних без затримок, що забезпечить ефективну та безперебійну роботу в умовах високих навантажень. Масштабованість є ще одним важливим фактором: з розвитком медицини та зростанням обсягів даних, система повинна бути готова до розширення та обробки ще більших обсягів інформації без втрати продуктивності.

Захист персональних медичних даних – це ще один важливий аспект, адже відповідно до міжнародних стандартів, таких як GDPR і HIPAA, дані пацієнтів повинні бути надійно захищені від несанкціонованого доступу. Тому система повинна мати високий рівень безпеки, що гарантує конфіденційність і збереження інформації.

Водночас інтерфейс системи має бути зручним і доступним навіть для користувачів без технічної підготовки. Це забезпечить легкість у використанні та дозволить лікарям, медичним працівникам і дослідникам швидко адаптуватися до роботи з новим інструментом. Важливим є також забезпечення надійності системи, що включає високу доступність та стійкість до збоїв, гарантуючи, що система працюватиме безперервно, навіть в умовах великих навантажень.

Висновки до розділу 1

У першому розділі було проведено комплексний аналіз теоретичних і практичних аспектів розвитку інформаційно-аналітичних систем у медичній сфері. Визначено, що сучасна медицина дедалі більше спирається на цифрову інфраструктуру, яка забезпечує ефективне збирання, зберігання, аналіз і візуалізацію великих обсягів клінічних даних.

Проведений огляд сучасних підходів до обробки медичних даних показав, що ключовими тенденціями є використання методів машинного навчання для діагностики та прогнозування, а також впровадження інтерактивних інструментів візуалізації даних, які полегшують сприйняття складної медичної інформації. У результаті аналізу існуючих рішень виявлено, що хоча на ринку представлено низку потужних платформ для медичної аналітики (зокрема Tableau, Power BI, QlikView, Medisense, CareVue), більшість із них мають обмежену можливість адаптації до специфічних потреб дослідження чи клінічної практики. Крім того, їхня вартість або закритість коду створюють перешкоди для використання у навчальних і дослідницьких цілях.

У цьому контексті використання відкритих фреймворків, таких як Dash (Plotly), є перспективним напрямом, оскільки він дозволяє швидко створювати інтерактивні вебзастосунки для аналізу та візуалізації медичних даних з мінімальними витратами.

Поставлена в ході дослідження завдання полягає у створенні інформаційно-аналітичної системи для візуалізації медичних даних, яка забезпечить ефективну інтеграцію, обробку та представлення результатів досліджень у зручному для користувача вигляді. Основна мета полягає у розробці веборієнтованого застосунку з використанням фреймворку Dash (Plotly), що дозволить реалізувати гнучкий механізм аналізу та візуалізації даних з медичних досліджень, зокрема даних з відкритих джерел на кшталт PhysioNet.

2 МЕТОДИ ТА ОСНОВИ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Методи збору та обробки даних

У реалізації інформаційно-аналітичної системи для візуалізації результатів медичних досліджень важливим етапом є підготовка даних – забезпечення надійності, коректності та придатності джерела даних для подальшого аналізу й візуалізації. У цьому підрозділі розглядаються теоретичні основи збору медичних даних, а також детально описується використаний у роботі набір даних PTB-XL: його походження, структура, метадані, параметри сигналу, та етапи попередньої обробки. Збір медичних даних передбачає агрегування інформації з численних джерел: пристроїв моніторингу (ЕКГ, пульсоксіметри, тонометри), лабораторних систем (аналізи крові), електронних медичних записів (EHR) та відкритих наукових баз даних. Дані можуть мати різну природу: часові сигнали (наприклад, ЕКГ-сигнали), табличні дані (аналізи, демографія), текстові звіти (діагнози, інтерпретації) та можуть бути представлені у різних форматах [11]. Ключовими аспектами підготовки даних є:

- очищення: видалення або заміщення пропущених значень, усунення дублікатів, корекція форматів;
- стандартизація: приведення даних до єдиної форми, наприклад, стандартизація підписів статі, віку тощо;
- нормалізація сигналів: забезпечення порівнянності між записами різних пацієнтів;
- інтеграція даних: об’єднання метаданих (демографія, діагнози) із фізіологічними сигналами;
- візуальна готовність: підготовка даних у формі, зручній для побудови інтерактивних панелей та графіків.

Ці підходи забезпечують те, щоб інформаційно-аналітична система могла працювати ефективно, наочно й коректно.

Для цілей цієї роботи було обрано датасет PTB-XL ECG. Набір PTB-XL створено дослідниками з Physikalisch-Technische Bundesanstalt (PTB) у Берліні, у співпраці з клінічними закладами, як клінічно орієнтована база даних 12-відведених ЕКГ (12-lead ECG) для академічних, дослідницьких і машинно-навчальних застосувань [12].

Обсяги та демографія:

- загалом датасет містить 21’799 записів 12-відведених ЕКГ від 18’869 пацієнтів з тривалістю 10 секунд кожен;

- демографічно: приблизно 52 % пацієнтів – чоловіки, 48 % – жінки. Віковий діапазон – від буквально новонароджених (0) до 95 років, медіана віку – 62 роки;

- дані охоплюють широкий спектр клінічних станів: нормальні записи, інфаркти міокарда, гіпертрофія, порушення провідності, зміни ST/T, тощо. Структура та формат записів:

1) кожен запис – 12-відведеній ЕКГ (відведення: I, II, III, aVR, aVL, aVF, V1–V6) з частотою дискретизації або 100 Гц (версія “_lr”, low resolution) або 500 Гц (версія *“_hr”, high resolution*) залежно від конфігурації;

2) файл формату WFDB (*.dat* та *.hea*) містить сигнали та заголовкову інформацію;

3) метадані зберігаються у файлі *ptbxl_database.csv*, який включає такі поля:

а) *ecg_id* – унікальний ідентифікатор запису;

б) patient_id – ідентифікатор пацієнта;

в) age, sex, (іноді height, weight);

г) *filename_lr, filename_hr* – шляхи до сигналів;

д) scp_codes – об’єкт/словник діагностичних кодів за стандартом SCP-ECG;

е) інші: *heart_axis, infarction_stadium1, infarction_stadium2,* якщо наявні.

Кожен запис має мультиміткову анотацію (до двох кардіологів) згідно зі стандартом SCP-ECG (до 71 діагностичної, формальної чи ритм-орієнтованої заяви). Додатково надані рекомендовані спліти для тренування/валідації/тесту, що спрощує реплікацію досліджень. Метадані також містять інформацію про вісь серця, стадію інфаркту для відповідних записів, що збільшує клінічну цінність [13].

Після розпакування архіву структура файлів у датасеті виглядає як визначено на рис. 2.1.

Папки *records100/* та *records500/* містять підпапки з ідентифікаторами (наприклад, 00000/00001_lr.dat), де *_lr* позначає запис з нижчою частотою (100 Гц), а *_hr* – високою (500 Гц).

Файл *scp_statements.csv* містить розшифрування для кодів SCP-ECG.

Метадані у таблиці *ptbxl_database.csv* дозволяють швидко фільтрувати записи за віком, статтю, діагнозами.

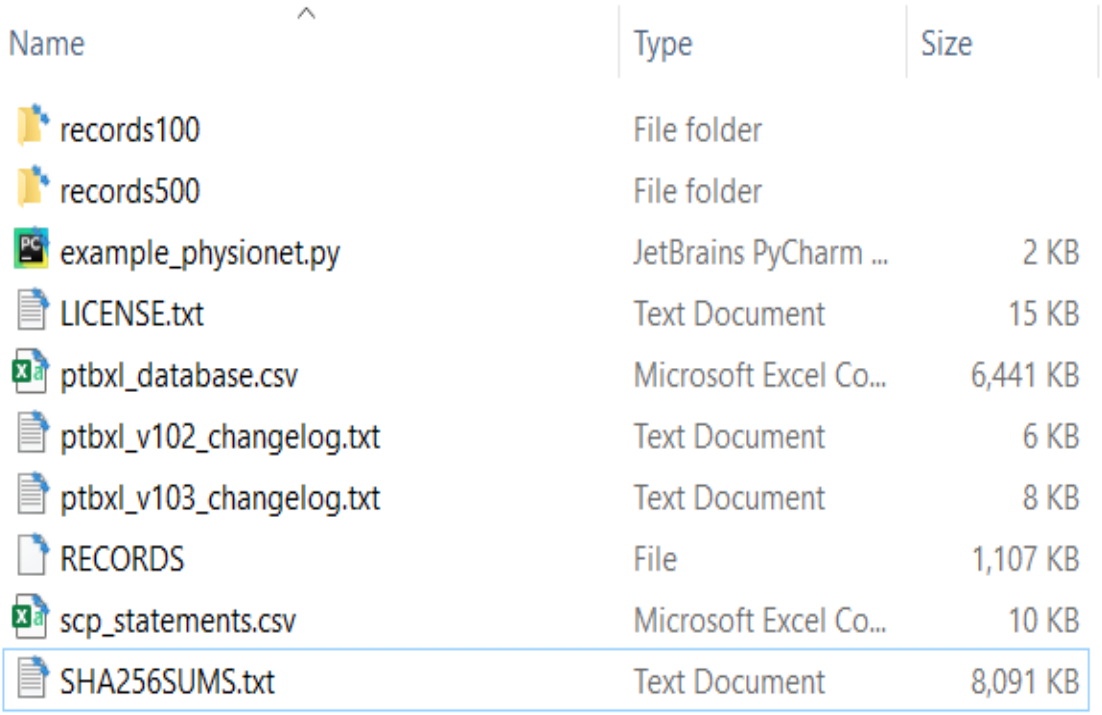

Рисунок 2.1 – Скріншот повного вмісту директорії датасета PTB-XL ECG

Етапи попередньої обробки даних:

1) зчитування сигналів:

- Використовуємо бібліотеку *wfdb* для читання файлів *.dat* та *.hea*. Наприклад: *record = wfdb.rdrecord(path_to_file)* повертає об’єкт з полями *p_signal* (масив форму сигналу), *fs* (частота дискретизації), *sig_name* (імена відведень).

- Часову шкалу створюємо як: *time = np.arange(len(p_signal)) / fs*;

2) об’єднання сигналів із метаданими:

- Зчитуємо метадані: *df = pd.read_csv("ptbxl_database.csv")*.

- У таблиці обираємо поля: *ecg_id, patient_id, age, sex, filename_lr, scp_codes.*

- Додаємо до структурованих даних: сигнал → *DataFrame* зі стовпцем “*time*” + сигнали відведень + демографія пацієнта;

3) очищення та стандартизація даних:

- Перевіряємо стовпець sex на наявність значень *"M", "F", "male", "female",* 1/0 – і уніфікуємо до *"Male" / "Female"*.

- Замінюємо пропуски у age значенням 0 або видаляємо записи з надто великим відсотком пропусків.

- Перевіряємо траси з високим шумом або артефактами (наприклад, записи з частотою дискретизації 500 Гц можуть вимагати додаткового фільтрування);

4) вибірка та підготовка підвибірки:

- для реалізації системи обирається підвибірка (наприклад, перші 300 записів), щоб полегшити обробку та зменшити навантаження на ресурс.

- можна фільтрувати записи за віком, статтю, діагнозом (наприклад лише записи з scp_codes які містять “NORM” або “MI”).

Для підготовки до інтерактивної візуалізації створюється таблиця *(DataFrame)* з метаданими *df_short*, яка містить поля: *ecg_id*, *patient_id*, *age*, *sex*, *filename_lr*, *scp_codes*. При виборі запису користувачем система завантажує відповідний сигнал і створює графік. Інтерактивні елементи (фільтри статі, віку, випадаючі меню) дозволяють користувачу обирати запис чи відведення, а система динамічно оновлює графік. Така інтерактивність покликана підвищити наочність аналізу медичних даних, полегшити виявлення закономірностей (наприклад, взаємозв’язку віку або статі із формою ЕКГ).

Додаткові особливості та обмеження датасету:

- хоча датасет і великий, він містить записи лише тривалістю 10 секунд, що обмежує аналіз тривалих подій (наприклад, аритмії, які розвиваються протягом більшої тривалості);

- хоча сигнали високої якості, вони були записані у період 1989–1996 років з пристроїв Schiller AG, що може вплинути на сучасну релевантність апаратного забезпечення;

- дані були підібрані у європейському клінічному середовищі – це слід враховувати при екстраполяції на інші популяції.

Таким чином, набір даних PTB-XL є одним з найбільш обґрунтованих, стандартизованих і широко використовуваних відкритих ECG-датасетів. Він має чітку структуру, високоякісні сигнали, багаті метадані та підходить для побудови інформаційно-аналітичних систем. У межах цієї роботи він використовується як основа для візуалізації медичних даних із застосуванням фреймворку Dash (Plotly). Усі етапи – від збору до інтерактивної візуалізації – реалізовані з врахуванням специфіки цього датасету.

2.2 Основи аналізу медичних показників

Аналіз медичних показників є фундаментальною частиною сучасних медичних інформаційних систем. Він дозволяє лікарям, дослідникам та аналітикам отримувати кількісну й якісну оцінку стану пацієнта, виявляти відхилення від норми, прогнозувати перебіг захворювання та контролювати ефективність лікування. Медичні показники охоплюють широкий спектр фізіологічних, біохімічних, морфологічних та функціональних характеристик організму людини, які відображаються у цифровій формі. У загальному вигляді медичні показники можна поділити на кілька груп:

1) фізіологічні – параметри, що безпосередньо характеризують функціональний стан організму, наприклад частота серцевих скорочень (ЧСС), артеріальний тиск, частота дихання, температура тіла, рівень насичення крові киснем (SpO_2);

2) біохімічні – концентрації певних речовин у біологічних рідинах (глюкоза, креатинін, холестерин, електроліти тощо);

3) морфологічні – показники, що описують структуру органів і тканин (отримані з МРТ, КТ, УЗД);

4) електрофізіологічні – сигнали електричної активності органів, зокрема ЕКГ (серце), ЕЕГ (мозок), ЕМГ (м’язи);

5) клініко-анамнестичні – числові або категоріальні дані, отримані внаслідок опитування чи спостереження (вік, стать, вага, статус куріння, сімейна історія хвороби тощо).

Залежно від типу даних, для аналізу застосовують різні методи, наведені в табл. 2.1.

Таблиця 2.1 – Основні методи аналізу для типових видів медичних даних [14]

Тип даних	Основні методи аналізу
Табличні (лабораторні, демографічні)	статистичний аналіз, кореляційний аналіз, регресійне моделювання
Сигнальні (ЕКГ, ЕЕГ, ЕМГ)	часовий та частотний аналіз, фільтрація, спектральна декомпозиція, класифікація ритмів
Зображення (МРТ, КТ, УЗД)	комп’ютерний зір, сегментація, машинне навчання
Текстові (медичні звіти)	лінгвістичний аналіз, NLP-моделі, класифікація діагнозів

Фізіологічні сигнали, зокрема електрокардіограма (ЕКГ), є прикладом часових рядів, що відображають зміну електричної активності у часі. Для їх аналізу застосовується цифрова обробка сигналів, основні етапи якої включають:

1) дискретизацію – перетворення аналогового сигналу у цифровий шляхом вимірювання напруги через певні інтервали часу (частота дискретизації);

2) квантування – округлення виміряних значень до заданого рівня точності (розрядності);

3) фільтрацію – усунення шумів та артефактів, що можуть бути спричинені рухами тіла або електричними перешкодами;

4) сегментацію – виділення окремих компонент сигналу, наприклад зубців P, QRS-комплексу та T-хвилі у ЕКГ;

5) особливості аналізу часових та частотних характеристик: Часовий аналіз дозволяє виявити тривалість інтервалів, амплітуди зубців, період серцевого циклу тощо. Частотний аналіз дає змогу оцінити спектральний склад сигналу та виявити періодичні зміни.

Для уніфікації представлення даних використовуються міжнародні стандарти: WFDB – формат, розроблений у рамках PhysioNet, для зберігання біомедичних сигналів разом з метаданими, SCP-ECG – стандарт для передачі

електрокардіографічних даних і діагностичних описів між системами, DICOM-Waveform – розширення стандарту DICOM для медичних сигналів, таких як ЕКГ чи ЕЕГ. Саме формат WFDB використовується у наборі даних PTB-XL, що дозволяє легко зчитувати, аналізувати і візуалізувати сигнали через бібліотеку Python wfdb.

Електрокардіограма – це графічне зображення електричної активності серця, що виникає при деполяризації та реполяризації міокарда. Стандартна клінічна ЕКГ містить 12 відведень, кожне з яких фіксує різницю потенціалів між певними точками тіла (рис. 2.2).

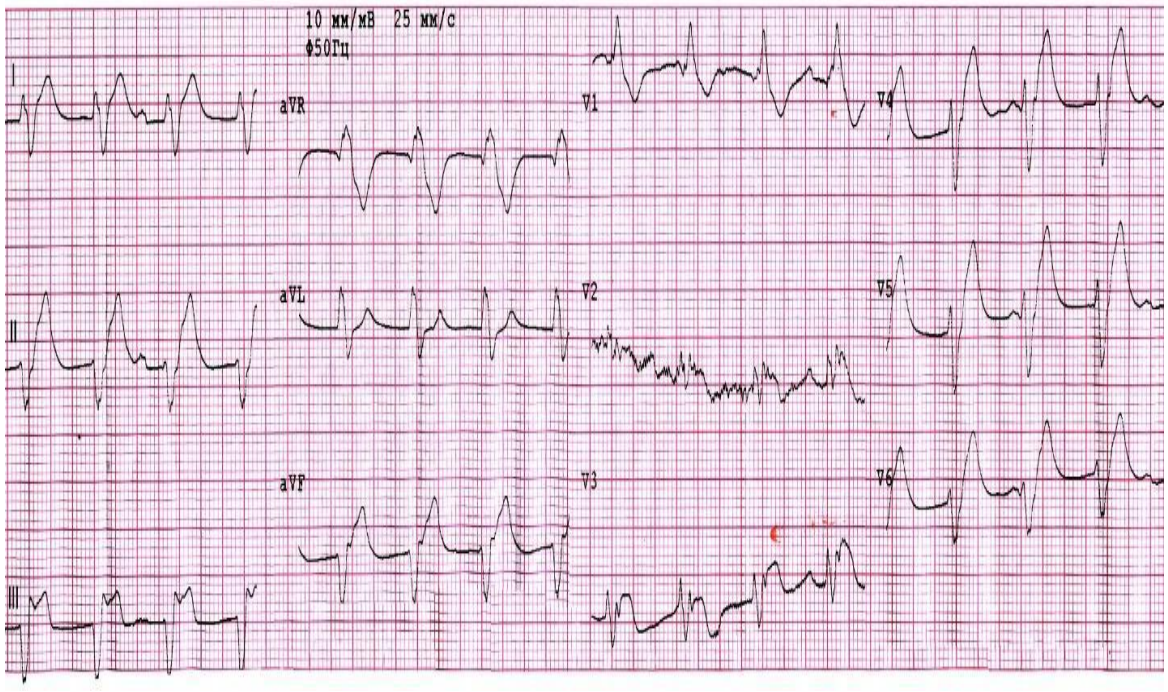

Рисунок 2.2 – Приклад ЕКГ з 12 відведеннями [15]

Типові відведення:

– стандартні: I, II, III;

– підсилені однополюсні: aVR, aVL, aVF;

– грудні (передсерцеві): V1–V6.

У типовому серцевому циклі виділяють такі фази (рис. 2.3):

– зубець P – деполяризація передсердь;

– комплекс QRS – деполяризація шлуночків;

– хвиля T – реполяризація шлуночків;

– інтервали PR, ST, QT – часові проміжки між електрофізіологічними подіями.

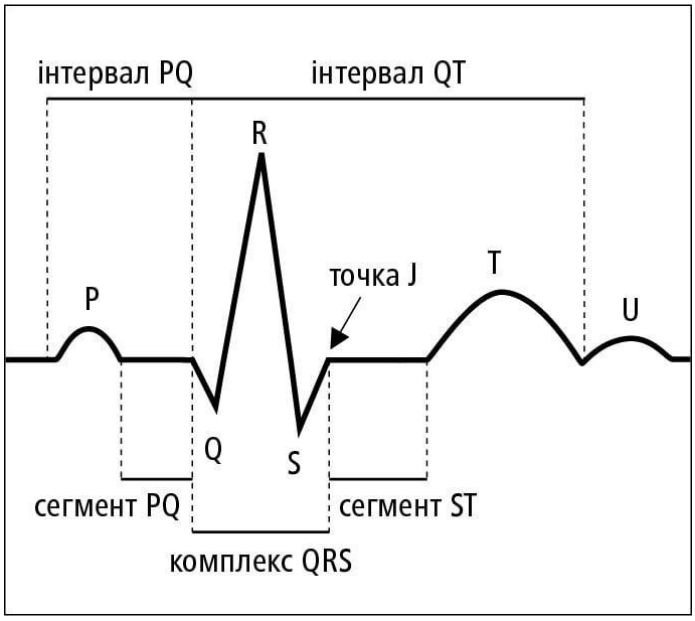

Рисунок 2.3 – Детальна ілюстрація фаз серцевого циклу [16]

Етапи аналітичної обробки ЕКГ-сигналів у системі:

– зчитування сигналів через *wfdb.rdrecord();*

– побудова часової шкали за частотою дискретизації;

– нормалізація амплітуд для уніфікації діапазонів між різними пацієнтами;

– видалення шумів;

– візуалізація відведень у середовищі Plotly/Dash;

– порівняння записів різних категорій SCP-ECG для дослідження типових патернів.

Для візуалізації сигналів у системі використовується бібліотека Plotly у складі фреймворку Dash. Основні візуальні методи:

1) лінійний графік – базовий спосіб представлення електрокардіограми;

2) мультисерійна візуалізація – одночасне відображення кількох відведень;

3) порівняльні графіки – зіставлення нормальних та патологічних записів;

4) інтерактивна фільтрація – вибір пацієнта, відведення, діагнозу, віку або статі;

5) динамічне масштабування та панорамування – забезпечує зручне дослідження окремих фрагментів сигналу.

Такі інструменти надають користувачеві можливість не лише переглядати сигнали, але й здійснювати візуальну аналітику: порівнювати динаміку сигналів, виявляти особливості форми зубців, співставляти патерни між різними групами пацієнтів. Таким чином, аналіз медичних показників є центральною складовою функціонування інформаційно-аналітичних систем у медицині. Залежно від природи даних, застосовуються різні математичні, статистичні та обчислювальні підходи [17]. У межах даного дослідження акцент зроблено на аналізі електрокардіографічних сигналів – одного з найважливіших видів фізіологічних показників. Набір даних PTB-XL забезпечує високу якість, багатовимірність та стандартизованість ЕКГ-записів, що дозволяє ефективно реалізувати інтерактивну аналітику та візуалізацію цих сигналів у середовищі Dash (Plotly).

2.3 Вибір методів і засобів реалізації

Розробка інформаційно-аналітичної системи передбачає поєднання методів обробки даних, алгоритмів аналітики та програмних засобів для реалізації графічного інтерфейсу користувача. Вибір конкретних інструментів визначається такими критеріями:

1) тип і структура даних, що використовуються у проєкті. у нашому випадку – це часові фізіологічні сигнали (електрокардіограми), які вимагають ефективної роботи з багатовимірними масивами даних;

2) масштабованість і продуктивність – здатність системи обробляти великі обсяги даних без втрати швидкодії;

3) гнучкість інтерфейсу – можливість створювати інтерактивні візуалізації, які легко змінюються відповідно до потреб користувача;

4) відкритість і сумісність – перевага надається відкритим (open-source) технологіям, які не потребують ліцензійних платежів і мають активну спільноту підтримки;

5) простота інтеграції з іншими системами – наприклад, з базами даних, api, науковими бібліотеками або клінічними стандартами.

З огляду на ці вимоги, для реалізації даної системи обрано мову програмування Python у поєднанні з фреймворком Dash (Plotly), бібліотеками Pandas, NumPy, WFDB, Plotly, а також допоміжними інструментами для аналізу та візуалізації медичних сигналів.

Python є провідною мовою у сфері наукових досліджень, машинного навчання та медичної аналітики. Його переваги полягають у наступному:

1) простота синтаксису – забезпечує легкість розробки, що є важливою умовою у наукових та медичних проєктах, де акцент робиться не лише на коді, а й на логіці обробки даних;

2) великий набір бібліотек для статистичного аналізу, обробки сигналів, побудови графіків та розробки вебзастосунків;

3) активна спільнота – наявність великої кількості відкритих рішень, прикладів і документації, що спрощує інтеграцію нових компонентів.

4) підтримка різних форматів медичних даних – через бібліотеки WFDB, BioSPPy, NeuroKit2, PyWavelets, PyDICOM;

5) мультиплатформність – можливість запуску застосунку в Windows, Linux або через вебсервер (Flask/Gunicorn);

6) Python дає змогу створити єдину програмну екосистему – від збору та попередньої обробки даних до побудови інтерактивних дашбордів.

Для створення користувацького інтерфейсу та інтерактивних графіків обрано фреймворк Dash, розроблений компанією Plotly. Його ключові переваги наведений у табл. 2.2.

Таблиця 2.2 – Основні характеристики фреймворку Dash

Критерій	**Характеристика**
Тип системи	вебфреймворк для побудови аналітичних інтерфейсів
Мова	Python (з підтримкою R та Julia)
Архітектура	Комбінація Flask (бекенд), React.js (фронтенд) і Plotly.js (візуалізація)
Інтерактивність	Реалізація callback-функцій, які оновлюють графіки в реальному часі
Розширюваність	Підтримка інтеграції з Pandas, NumPy, Scikit-learn, TensorFlow
Ліцензія	MIT License – безкоштовне використання для освітніх і дослідницьких проєктів

Dash дозволяє швидко створювати вебзастосунки без необхідності вручну писати HTML або JavaScript-код. Візуалізації будуються через компоненти Plotly Graph Objects**,** що підтримують інтерактивне масштабування, панорамування, виділення областей, фільтрацію даних і динамічне оновлення при зміні параметрів користувачем.

Для реалізації функцій збору, аналізу та візуалізації даних використано такі основні бібліотеки:

1) Pandas. Основний інструмент для роботи з табличними даними. Дозволяє завантажувати CSV-файли, виконувати групування, фільтрацію, злиття та статистичні обчислення. У даній роботі використовується для зчитування файлу ptbxl_database.csv, об’єднання метаданих із сигналами, а також підготовки структури даних для Dash;

2) NumPy. Забезпечує швидкі векторні та матричні обчислення, необхідні для обробки сигналів. Застосовується для створення часової шкали сигналу, нормалізації амплітуд, розрахунку статистичних показників;

3) WFDB. Спеціалізована бібліотека для читання фізіологічних сигналів у форматі .dat та .hea. Дозволяє зчитувати багатоканальні електрокардіограми, отримувати параметри частоти дискретизації, тривалості запису та назви каналів (відведень). Забезпечує повну сумісність із форматами PhysioNet, що критично для роботи з датасетом PTB-XL;

4) Plotly. Відповідає за побудову високоякісних графіків і візуалізацій (лінійних, гістограм, теплових карт тощо). Має інтерактивні елементи, такі як наведення курсору, зміна масштабу, анімація даних. Використовується у Dash-компонентах dcc.Graph для динамічного відображення сигналів ЕКГ;

5) Dash Core Components. Забезпечує побудову структури інтерфейсу користувача – меню вибору пацієнта, фільтри за статтю чи віком, селектори відведень, інформаційні панелі. Застосовується спільно з бібліотекою dash.html для верстки сторінки застосунку.

Для обробки ЕКГ-сигналів у рамках системи застосовуються такі базові методи цифрової аналітики:

1) фільтрація шумів – використання ковзного середнього або фільтру Баттерворта для усунення високочастотних спотворень;

2) нормалізація амплітуд – масштабування сигналу до діапазону [–1; 1] або [0; 1] для забезпечення порівнянності між різними записами;

3) сегментація часових рядів – поділ сигналу на окремі серцеві цикли (P-QRS-T) при подальшому аналізі або візуалізації;

4) виділення ключових ознак – визначення піків R для розрахунку частоти серцевих скорочень, ширини QRS-комплексу, тривалості інтервалів PR і QT;

5) інтерактивна візуалізація – побудова графіка сигналу з можливістю фільтрувати записи за демографічними ознаками та вибирати окремі відведення.

Ці методи не лише дозволяють відобразити сигнали, але й створюють базу для майбутнього розширення – наприклад, для автоматичної класифікації патологій на основі ЕКГ.

Висновки до розділу 2

У другому розділі було розглянуто методичні та технологічні основи побудови інформаційно-аналітичної системи для візуалізації медичних даних, що забезпечують наукове та практичне підґрунтя подальшої реалізації програмного продукту. Основна увага приділялася питанням збору, обробки та аналізу медичних показників, вибору відповідних алгоритмів і засобів програмної реалізації. Визначено, що ефективність аналітичної системи безпосередньо залежить від якості початкових даних, правильності їхньої попередньої обробки та адекватності застосованих методів цифрового аналізу. Для цього було детально проаналізовано використаний у роботі набір даних PTB-XL, який містить понад 21 тисячу високоякісних електрокардіографічних записів, отриманих за стандартизованими умовами. Його структура, що поєднує фізіологічні сигнали та розширені метадані, дозволяє здійснювати комплексну обробку – від фільтрації сигналів до візуальної аналітики. Проведений опис показав, що застосування методів очищення, нормалізації, стандартизації та інтеграції даних є ключовими етапами у створенні надійної основи для подальшого аналізу. Особливу увагу приділено теоретичним основам аналізу медичних показників. Показано, що різні типи медичних даних – фізіологічні, біохімічні, морфологічні, електрофізіологічні – вимагають використання специфічних підходів до обробки. Для обраного типу – електрокардіографічних сигналів – описано основні параметри (інтервали RR, PR, QRS, QT, сегмент ST), які є важливими діагностичними маркерами серцевих патологій. Розкрито сутність методів часової та частотної обробки сигналів, що дозволяють виділити інформативні характеристики для подальшої візуалізації. Аналіз цих характеристик дає змогу не лише виявляти

відхилення, але й наочно відображати динаміку електричної активності серця, що є надзвичайно важливим для клінічної інтерпретації.

У ході вибору засобів програмної реалізації обґрунтовано застосування мови Python як універсального середовища для наукових обчислень і візуалізації даних. Вибір фреймворку Dash (Plotly) зумовлений його здатністю поєднувати серверну логіку, аналітичні обчислення та графічний інтерфейс у межах єдиної платформи. Бібліотеки Pandas, NumPy, WFDB, Plotly та Dash забезпечують повний цикл роботи з медичними даними – від імпорту сигналів і їх обробки до інтерактивного представлення користувачеві. Детальний аналіз їхніх модулів показав, що кожен із них виконує окрему, але важливу функцію: Pandas структурує дані, NumPy обробляє масиви та виконує математичні обчислення, WFDB відповідає за зчитування біомедичних сигналів, Plotly створює візуальні графіки, а Dash реалізує вебінтерфейс і логіку взаємодії з користувачем. Таке поєднання дозволяє побудувати комплексну архітектуру системи, що об’єднує рівень даних, обробки та представлення результатів.

Отже, розділ 2 заклав методологічну та технічну основу для створення інтерактивної інформаційно-аналітичної системи. Сформульовані принципи обробки медичних сигналів, визначено структуру вхідних даних і обґрунтовано вибір технологій, які забезпечують високу ефективність і надійність майбутнього програмного рішення. Розглянуті методи та інструменти створюють підґрунтя для реалізації архітектури, функціональних компонентів і користувацького інтерфейсу системи, що стане предметом детального опису в наступному розділі.

3 ПРОЄКТУВАННЯ І РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Вибір програмних засобів реалізації

Розроблення інформаційно-аналітичної системи для візуалізації медичних даних вимагало використання сучасного програмного забезпечення, здатного забезпечити обробку великих обсягів даних, інтерактивну побудову графіків та створення зручного користувацького інтерфейсу у вебсередовищі. Для реалізації проєкту було обрано мову програмування Python, а також низку спеціалізованих бібліотек, які б забезпечували ефективну обробку великих обсягів медичних даних, створення інтерактивних візуалізацій та зручний користувацький інтерфейс. У цьому підрозділі детально описано вибрані технології, бібліотеки та їх функціональні можливості.

Мова Python була обрана як основний інструмент через її універсальність, підтримку численних наукових і аналітичних бібліотек, простий синтаксис та сумісність із сучасними фреймворками для розробки вебзастосунків. Python забезпечує зручне поєднання процедурного, об’єктно-орієнтованого та функціонального підходів до програмування, що дозволяє ефективно реалізовувати як модулі обробки даних, так і інтерактивний інтерфейс.

Dash – це високорівневий Python-фреймворк, створений на базі Plotly, який дає змогу будувати вебзастосунки для інтерактивної візуалізації даних без використання JavaScript. У межах розробленої системи Dash виконує функції як фронтенду, так і бекенду одночасно [18].

Основні модулі *Dash*:

1) *dash.Dash* – клас для ініціалізації вебзастосунку;
2) *dash.html* – модуль для створення HTML-компонентів інтерфейсу;
3) *dash.dcc* – бібліотека інтерактивних компонентів;
4) *dash.Input* і *dash.Output* – класи для реалізації реактивності.

Основні компоненти та функції *Dash*, використані у проєкті:

1) *Dash(__name__)* – створення основного екземпляра застосунку;

2) *dcc.Graph()* – відображення інтерактивних графіків;

3) *dcc.Dropdown()* – інтерактивні списки для вибору запису або відведення;

4) *dcc.Input()* – поля для введення числових значень віку;

5) *dcc.Tabs() та dcc.Tab()* – реалізація навігації між вкладками (сигнал, частотний аналіз, статистика);

6) *dcc.Download() та dcc.send_data_frame()* – реалізація кнопки експорту сигналів у CSV;

7) *html.Div(), html.H2(), html.Button()* – елементи інтерфейсу користувача, оформлені у структурі HTML;

8) *@app.callback()* – основний механізм інтерактивності, що зв’язує вхідні дані (Input) з вихідними (Output).

Таким чином, *Dash* виступає ядром вебзастосунку, забезпечуючи інтерактивність, адаптивність і динамічне оновлення графіків без перезавантаження сторінки.

Plotly – бібліотека для створення інтерактивних графіків. Її було обрано для інтерактивної візуалізації сигналів. У системі використано два її основні модулі [19]:

1) *plotly.graph_objects:* застосовується для побудови графіків сигналів ЕКГ та частотних спектрів. Основні функції:

а) *go.Figure()* – створення контейнера для графічного об’єкта;

б) *go.Scatter()* – побудова лінійного графіка сигналу ЕКГ або спектра;

в) *fig.add_trace()* – додавання кількох відведень на один графік у різних кольорах;

г) *fig.update_layout()* – налаштування заголовків, осей, кольорів, легенд.

д) *plotly.express:* використовується для швидкого створення статистичних діаграм – гістограм і стовпчикових графіків. У проєкті задіяні:

е) *px.histogram()* – побудова розподілу віку пацієнтів;

ж) *px.bar()* – побудова діаграм за класами діагнозів;

2) *px.colors.qualitative.Plotly* – стандартна палітра кольорів для візуалізації багатьох відведень.

Використання Plotly дозволяє користувачеві збільшувати, переміщати, приховувати або порівнювати графіки без необхідності повторного запуску коду.

Бібліотека pandas є стандартом де-факто для аналітики та обробки табличних даних у Python. У межах даного проєкту вона використовується для:

– імпорту метаданих із файлу *ptbxl_database.csv* та *scp_statements.csv* за допомогою функції *pandas.read_csv()*;

– створення та обробки об’єктів типу DataFrame, що дозволяють працювати з даними пацієнтів, віком, статтю, файлами записів ЕКГ;

– виконання фільтрації даних за умовами (наприклад, за віком, статтю, діагностичним класом) через методи *DataFrame.loc[]* та *DataFrame.apply();*

– експорту оброблених результатів у формат CSV завдяки функції *to_csv()* у callback-функції експорту сигналів.

Функціонал pandas є ключовим у частині підготовки даних перед візуалізацією та забезпечує швидкий і надійний механізм роботи з великими таблицями [20] (понад 20’000 записів у PTB-XL).

Бібліотека NumPy застосовується для роботи з числовими масивами та векторизованими обчисленнями. У проєкті її основне призначення:

– обчислення часової осі сигналу ЕКГ *(np.arange())*, нормування частоти дискретизації;

– виконання операцій перетворення Фур’є через *np.fft.rfft()* і *np.fft.rfftfreq()* для побудови спектра частот;

– обчислення амплітуди сигналу у частотній області за допомогою np.abs() та нормалізації;

– прискорення математичних розрахунків завдяки оптимізації під С-масиви.

Таким чином, NumPy забезпечує високопродуктивні числові обчислення, необхідні для аналізу сигналів ЕКГ у режимі реального часу.

Бібліотека WFDB призначена для роботи з фізіологічними сигналами з відкритих баз даних PhysioNet. Вона дозволяє зчитувати, записувати та обробляти файли у форматах *.hea* (заголовки) та *.dat* (дані сигналів) [21]. У розробленій системі функції *wfdb* застосовано таким чином:

1) *wfdb.rdrecord(path)* – зчитування сигналу ЕКГ разом із метаданими з відповідного файлу;

2) *record.p_signal* – отримання двовимірного масиву із сигналами усіх відведень;

3) *record.sig_name* – зчитування назв відведень для подальшої візуалізації;

4) *record.fs* – отримання частоти дискретизації сигналу (100 або 500 Гц залежно від підкаталогу).

Завдяки *wfdb* система може автоматично відкривати сигнали без потреби у спеціальних форматах або сторонніх програмах, що суттєво спрощує обробку медичних даних.

Декоратор *lru_cache* з модуля *functools*. Для прискорення повторних викликів функції l*oad_ecg()* використовується декоратор *@lru_cache*. Він кешує результати завантаження сигналів, щоб при повторному зверненні до

того самого файлу дані зчитувалися з пам’яті, а не з диска. Це істотно зменшує затримку при повторному відображенні одного й того ж запису.

Додаткові вбудовані бібліотеки:

1) *ast* – модуль, що використовується для безпечного перетворення рядкових представлень словників *(scp_codes)* у реальні об’єкти Python *(dict)* за допомогою функції *ast.literal_eval();*

2) *os* та *pathlib* (у допоміжних сценаріях) можуть використовуватися для побудови абсолютних шляхів до файлів;

3) *math* – для можливих обчислень у процесі аналізу сигналів.

Таким чином, обраний стек технологій – Python + Dash + Plotly + Pandas + NumPy + WFDB – забезпечує повний цикл обробки медичних даних: від зчитування сирих фізіологічних сигналів до їх фільтрації, частотного аналізу, статистичної обробки та інтерактивної візуалізації у вебінтерфейсі. Використання цих бібліотек дозволило створити надійну, гнучку та зручну для користувача систему, придатну для подальшого розширення – наприклад, додавання автоматичного розпізнавання патологій або машинного навчання.

3.2 Структура та архітектура інформаційно-аналітичної системи

Розроблена інформаційно-аналітична система призначена для зчитування, обробки, аналізу та візуалізації електрокардіографічних (ЕКГ) сигналів на основі даних відкритого набору PTB-XL ECG dataset, який є одним із найповніших джерел фізіологічних сигналів у базі PhysioNet. Система реалізована у вигляді вебзастосунку з інтерактивним інтерфейсом користувача, який дає змогу медичним працівникам, дослідникам або студентам швидко переглядати ЕКГ-записи, виконувати частотний аналіз сигналів, оцінювати статистичні розподіли показників пацієнтів і експортувати обрані дані у зручному форматі.

Основними принципами побудови системи стали:

1) модульність – поділ логіки обробки, аналізу та відображення даних на незалежні компоненти;

2) інтерактивність – можливість динамічної взаємодії користувача з інтерфейсом без перезавантаження сторінки;

3) масштабованість – можливість додавання нових аналітичних модулів, таких як автоматичне виявлення патологій;

Архітектура системи побудована за принципом трирівневої моделі, яка включає такі логічні рівні (рис. 3.1):

1) Рівень даних. На цьому рівні здійснюється робота з вихідними даними ЕКГ та метаданими про пацієнтів:

– вихідні дані – файли *.dat* та *.hea*, що зберігають сигнал ЕКГ у цифровому вигляді;

– метадані – файл *ptbxl_database.csv,* який містить інформацію про пацієнтів, вік, стать, діагнози, а також шляхи до файлів записів;

– допоміжний файл *scp_statements.csv*, у якому містяться розшифрування кодів діагнозів.

Робота з даними здійснюється за допомогою бібліотек *pandas* (для табличних даних) та *wfdb* (для зчитування фізіологічних сигналів). Дані зберігаються локально у структурованому форматі, що забезпечує швидке звернення та можливість попереднього кешування;

2) Рівень обробки. Цей рівень реалізує ключові алгоритми системи:

– зчитування сигналів ЕКГ з використанням функції *wfdb.rdrecord()* та формування об’єкта *DataFrame*;

– обчислення часової шкали за допомогою *numpy.arange(),* відповідно до частоти дискретизації сигналу *(record.fs);*

– частотний аналіз (FFT) – розрахунок спектра сигналу за допомогою функцій *np.fft.rfft() і np.fft.rfftfreq()*;

– нормалізація амплітуди *(np.abs(yf) / n)* для подальшої побудови амплітудного спектра;

– фільтрація даних за віком, статтю та іншими параметрами з використанням методів *pandas.DataFrame.loc[] та apply()*;

– кешування результатів зчитування файлів для підвищення швидкодії через декоратор *@lru_cache(maxsize=256)*;

3) Рівень представлення. На цьому рівні реалізується інтерактивний користувацький інтерфейс за допомогою фреймворку Dash, який поєднує в собі можливості фронтенду та бекенду. Компоненти інтерфейсу:

– елементи керування *(dcc.Dropdown, dcc.Input, dcc.Tabs, html.Button)* для вибору параметрів аналізу;

– інтерактивні графіки *(dcc.Graph)*, що будуються через бібліотеку Plotly;

– кнопка експорту (*dcc.Download, dcc.send_data_frame()*), що дозволяє завантажувати сигнали у форматі CSV;

– callback-функції *(@app.callback)*, які забезпечують динамічне оновлення графіків і таблиць без перезавантаження сторінки.

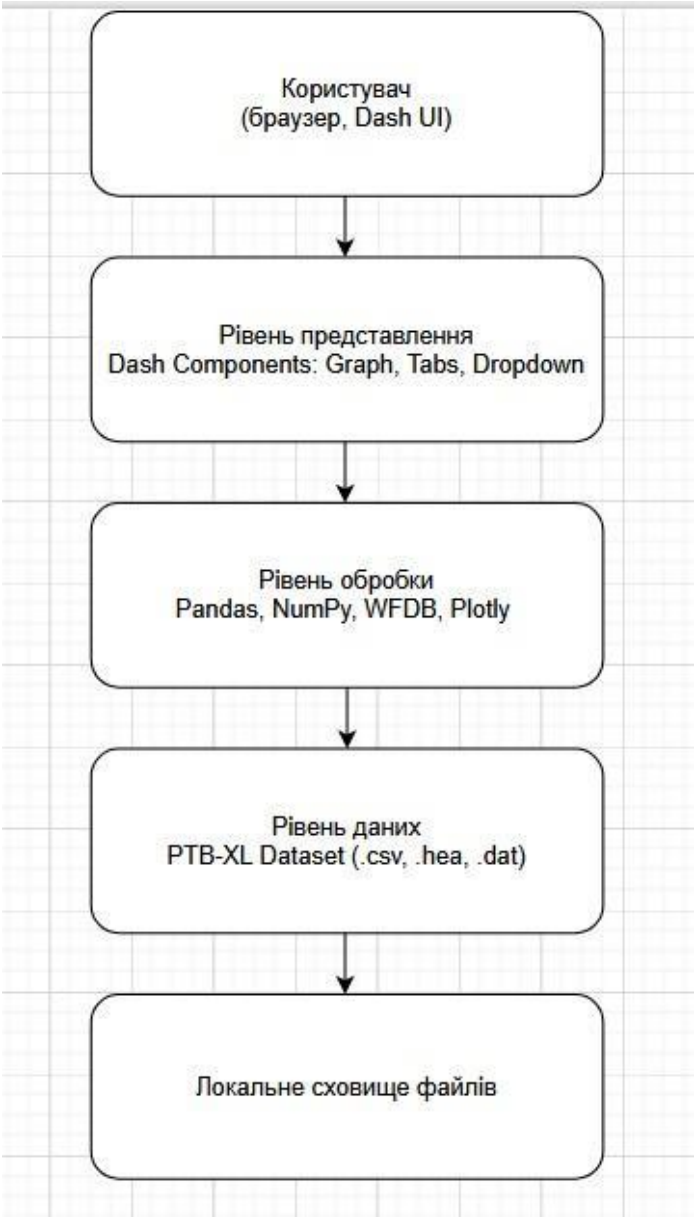

Рисунок 3.1 – Узагальнена схема архітектури системи

Логічно система складається з трьох основних модулів:

1) Модуль візуалізації сигналу забезпечує відображення ЕКГ-сигналів у часовій області. Користувач може обрати конкретний запис, один або кілька відведень, і побачити графік амплітуди у реальному часі. Використовуються функції: *update_signal_graph(), go.Scatter(), go.Figure();*

2) Модуль частотного аналізу виконує розрахунок та побудову спектра

сигналу до 50 Гц. Це дає можливість виявити частотні характеристики серцевого ритму та наявність перешкод. Використовується алгоритм швидкого перетворення Фур’є (FFT) через *np.fft*;

3) Модуль статистики пацієнтів здійснює побудову аналітичних графіків для груп пацієнтів: розподіл за віком, частота діагностичних класів. Використовуються функції *px.histogram()* та *px.bar()* із бібліотеки Plotly Express.

Алгоритм роботи системи

Послідовність дій користувача та програмних компонентів можна описати наступним алгоритмом:

1) ініціалізація програми. Завантажуються метадані з файлів ptbxl_database.csv та scp_statements.csv. Визначаються унікальні ідентифікатори записів, статі, вікові межі;

2) вибір користувачем параметрів. Користувач у вебінтерфейсі обирає:

– стать пацієнтів (усі / чоловіки / жінки);

– віковий діапазон;

– номер запису ЕКГ;

– одне або кілька відведень сигналу;

3) фільтрація даних. На основі вибраних параметрів виконується фільтрація об’єкта DataFrame з метаданими. Система формує список відповідних записів ЕКГ;

4) зчитування сигналу. За допомогою функції *load_ecg()* система звертається до відповідного файлу **.dat* і зчитує сигнал через *wfdb.rdrecord()*. Якщо запис уже був відкритий раніше, використовується кеш для прискорення доступу;

5) побудова графіка сигналу. У функції *update_signal_graph()* формується графік за допомогою *go.Figure()* та *go.Scatter()*. Для кількох

відведень застосовується кольорове розрізнення, а графік оновлюється динамічно без перезавантаження сторінки;

6) частотний аналіз. При виборі вкладки «Частотний аналіз» застосовується алгоритм *FFT (np.fft.rfft())*, а результати відображаються у вигляді спектра амплітуд до 50 Гц;

7) статистичний аналіз. У вкладці «Статистика пацієнтів» формуються гістограми за віковими групами та розподілом діагностичних класів;

8) експорт даних. Користувач може завантажити вибраний сигнал у форматі CSV, використовуючи функцію *dcc.send_data_frame();* Створюється файл із назвою, що містить ідентифікатор запису та список відведень;

9) завершення сеансу. При закритті сесії *Dash* автоматично очищає кеш даних.

Таким чином, розроблена архітектура забезпечує гнучке розділення функцій між модулями, що дозволяє легко оновлювати або розширювати систему. Завдяки модульності можна додавати нові аналітичні інструменти, такі як автоматичне розпізнавання патологій або інтеграцію з базами пацієнтів. Використання Dash і Plotly створює можливість не лише проводити науковий аналіз, а й застосовувати систему у навчальних або клінічних цілях.

3.3 Опис програмної реалізації

Програмна реалізація передбачає структурований поділ коду на логічні частини: завантаження та підготовка даних, модулі обробки сигналів, побудова інтерфейсу користувача, а також реалізацію інтерактивних callback-функцій для оновлення відображення даних у реальному часі.

Програма складається з одного основного файлу main.py, у якому реалізовані такі блоки:

1) імпорт бібліотек – підключення необхідних модулів;

2) завантаження та підготовка даних з файлів *ptbxl_database.csv* і *scp_statements.csv;*

3) формування допоміжних функцій для обробки метаданих і сигналів;

4) реалізація Dash-застосунку (створення елементів інтерфейсу, вкладок, графіків);

5) оголошення callback-функцій, які забезпечують динамічну взаємодію між користувачем і системою;

6) запуск сервера (*app.run(debug=True)*).

Імпорт залежностей:

```
import ast
from functools import lru_cache
import numpy as np
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import wfdb
from dash import Dash, dcc, html, Input, Output.
```

Опис імпортів:

а) *ast* – для безпечного парсингу рядкових представлень Python-структур;

б) *functools.lru_cache* – декоратор для кешування результатів функцій;

в) *numpy* – для числових обчислень та FFT-аналізу;

г) *pandas* – для роботи з табличними даними;

д) *plotly.express* і *graph_objects* – для створення візуалізацій;

е) *wfdb* – для читання ЕКГ-файлів у форматі PhysioNet;

ж) *dash* – компоненти вебфреймворку.

Ключові функції та їх призначення:

1) *normalize_sex(x).*

Призначення: нормалізує інформацію про стать пацієнта. Як у вихідному наборі даних значення статі можуть бути як числовими (0; 1), так і текстовими (*'Male', 'Female'*), функція виконує уніфікацію формату:

```
def normalize_sex(x):
    if str(x).lower().startswith("m") or x == 1:
        return "Male"
    if str(x).lower().startswith("f") or x == 0:
        return "Female"
    return "N/A"
```

Функція підвищує якість даних і дозволяє уникнути помилок при подальшій фільтрації;

2) *translate_scp(codes: dict).*

Призначення: перетворює коди SCP (Standard Communication Protocol) діагнозів у зрозумілі текстові описи на основі таблиці *scp_statements.csv*. Працює через ітерацію по словнику кодів і підстановку опису:

```
names = [scp_desc.get(k, k) for k in codes.keys()]
return ", ".join(names)
```

Результат зберігається в полі *diagnosis_text* для відображення в інтерфейсі;

3) *diag_category(codes: dict)*

Призначення: визначає категорію діагнозу для агрегування статистики. Функція перевіряє наявність певних кодів у словнику *scp_codes* і відносить запис до класу:

а) NORM – норма;

б) MI – інфаркт міокарда;

в) STTC – зміни ST/T;

г) CD – порушення провідності;

д) HYP – гіпертрофія.

```
for key, label in [("NORM","Normal"),("MI","Myocardial infarction"),...]:
```

if key in keys: return label

Це дозволяє обчислювати частоту зустрічальності патологій у вкладці «Статистика»;

4) *load_ecg(filename_lr: str).*

Призначення: зчитує фізіологічний сигнал ЕКГ із файлів PTB-XL та формує таблицю *DataFrame*. Функція повертає трійку *(df_ecg, leads, fs)* – сам сигнал, список відведень та частоту дискретизації:

record = wfdb.rdrecord(f"{data_path}/{filename_lr}")

sig = record.p_signal

leads = record.sig_name

fs = float(record.fs)

time = np.arange(sig.shape[0]) / fs

df_ecg = pd.DataFrame(sig, columns=leads)

df_ecg["time"] = time

Для підвищення ефективності ця функція кешується декоратором *@lru_cache(maxsize=256);*

5) *update_ecg_list(selected_sex, min_age, max_age).*

Призначення: формує список доступних ЕКГ-записів згідно з обраними користувачем фільтрами (стать, мінімальний і максимальний вік). Функція виконує фільтрацію таблиці *df_short* і генерує список опцій для випадаючого меню:

filtered = df_short[(df_short["age"] >= min_age) & (df_short["age"] <= max_age)]

if selected_sex != "all":

filtered = filtered[filtered["sex"] == selected_sex]

options = [{"label": f"ECG {r.ecg_id} ...", "value": r.ecg_id} for _, r in filtered.iterrows()]

Повертає список записів і перший елемент як вибраний за замовчуванням;

6) *update_leads(selected_ecg).*

Призначення: отримує список відведень для обраного запису. Використовує функцію *load_ecg()* для визначення структури сигналу. Результат – список відведень для вибору користувачем;

7) *update_signal_graph(selected_ecg, selected_leads).*

Призначення: формує часовий графік сигналів обраних відведень. Використовує об’єкти бібліотеки Plotly Graph Objects:

fig = go.Figure()

fig.add_trace(go.Scatter(x=ecg_df["time"], y=ecg_df[lead], mode="lines", ...))

fig.update_layout(title=..., xaxis_title="Час (сек)", yaxis_title="Амплітуда (мВ)")

Функція викликається автоматично при зміні вибраного запису або відведень через callback-механізм Dash;

8) *update_fft_graph(selected_ecg, selected_leads).*

Призначення: виконує частотний аналіз сигналів (спектр амплітуд) за допомогою алгоритму швидкого перетворення Фур’є (FFT):

yf = np.fft.rfft(y)

xf = np.fft.rfftfreq(n, d=1.0/fs)

amp = np.abs(yf) / n

mask = xf <= 50.0

Результат візуалізується як графік залежності амплітуди від частоти. Цей модуль дає змогу аналізувати частотні особливості серцевого сигналу;

9) *update_stats(sex_val, min_age, max_age).*

Призначення: генерує дві статистичні діаграми:

– гістограму віку пацієнтів (через *px.histogram()*),

– розподіл діагностичних класів (через *px.bar()*).

Функція використовує ті самі параметри фільтрації, що й у попередніх модулях, що забезпечує узгодженість усіх вкладок інтерфейсу;

10) елементи експорту. Реалізовано кнопку експорту вибраного сигналу у форматі CSV:

dcc.Download(id="download_data"),

html.Button("Експортувати сигнал у CSV", id="export_btn"),

Callback-функція викликає:

return dcc.send_data_frame(df_ecg.to_csv, "ecg_signal.csv"),

що дозволяє користувачеві миттєво завантажити результати аналізу.

Механізм кешування. Декоратор *@lru_cache(maxsize=256)* із модуля functools дозволяє зберігати в пам’яті результати функції *load_ecg().* Це скорочує час повторного завантаження сигналів у 3–5 разів і зменшує навантаження на диск, особливо під час багаторазового перегляду тих самих записів.

Основна функція:

if __name__ == "__main__":

app.run(debug=True, port=8050)

запускає внутрішній вебсервер Flask, після чого програма доступна користувачеві за локальною адресою http://127.0.0.1:8050/. Усі оновлення інтерфейсу виконуються асинхронно, без перезапуску застосунку.

Dash використовує декларативну архітектуру подій: кожен компонент інтерфейсу має властивості Input і Output, які пов’язуються через декоратор *@app.callback.* Наприклад:

```
@app.callback(
   Output("ecg_graph", "figure"),
   Input("ecg_dropdown", "value"),
   Input("lead_dropdown", "value")
)
```

Коли користувач змінює параметри вибору, Dash автоматично оновлює лише відповідний графік, що забезпечує високу швидкодію та інтерактивність без перезавантаження сторінки.

Програмна реалізація поєднує сучасні методи аналітики медичних даних із гнучкою вебвізуалізацією. Використання Dash-архітектури забезпечує розширюваність системи, а застосування бібліотек Plotly, pandas та WFDB дозволяє інтегрувати різні типи сигналів і статистичних параметрів. Код побудовано модульно, що полегшує підтримку, тестування та подальший розвиток застосунку.

Висновки до розділу 3

У третьому розділі було реалізовано програмну частину інформаційно-аналітичної системи для візуалізації медичних даних. Розроблено вебзастосунок на основі Python, Dash та Plotly, який забезпечує інтерактивну роботу з набором PTB-XL.

У ході реалізації створено архітектуру клієнт–серверного типу, що дозволяє зчитувати, обробляти, аналізувати та візуалізувати ЕКГ-сигнали в реальному часі. Застосовано бібліотеки Pandas, NumPy, wfdb для обробки даних та Plotly для побудови динамічних графіків.

Система підтримує фільтрацію за статтю та віком, відображення кількох відведень ЕКГ, частотний аналіз сигналів (FFT), а також експорт результатів у формат CSV.

Запропонована архітектура є модульною, гнучкою та придатною до подальшого розширення – зокрема, для інтеграції алгоритмів машинного навчання або роботи з іншими типами медичних даних.

Таким чином, реалізований програмний модуль підтвердив ефективність обраних засобів і методів, продемонструвавши повну функціональність, стабільність роботи та практичну придатність системи.

4 АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ СИСТЕМИ

4.1 Методика тестування інформаційно-аналітичної системи

Приклади роботи програми

Одразу після запуску програми та відкриття вебзастосунку у браузері, користувач потрапляє у головний інтерфейс програми (рис. 4.1).

Рисунок 4.1 – Головна сторінка програми

Одразу користувач може скористатись фільтрами віку та статі, щоб кастомізувати список пацієнтів під свої потреби (рис. 4.2).

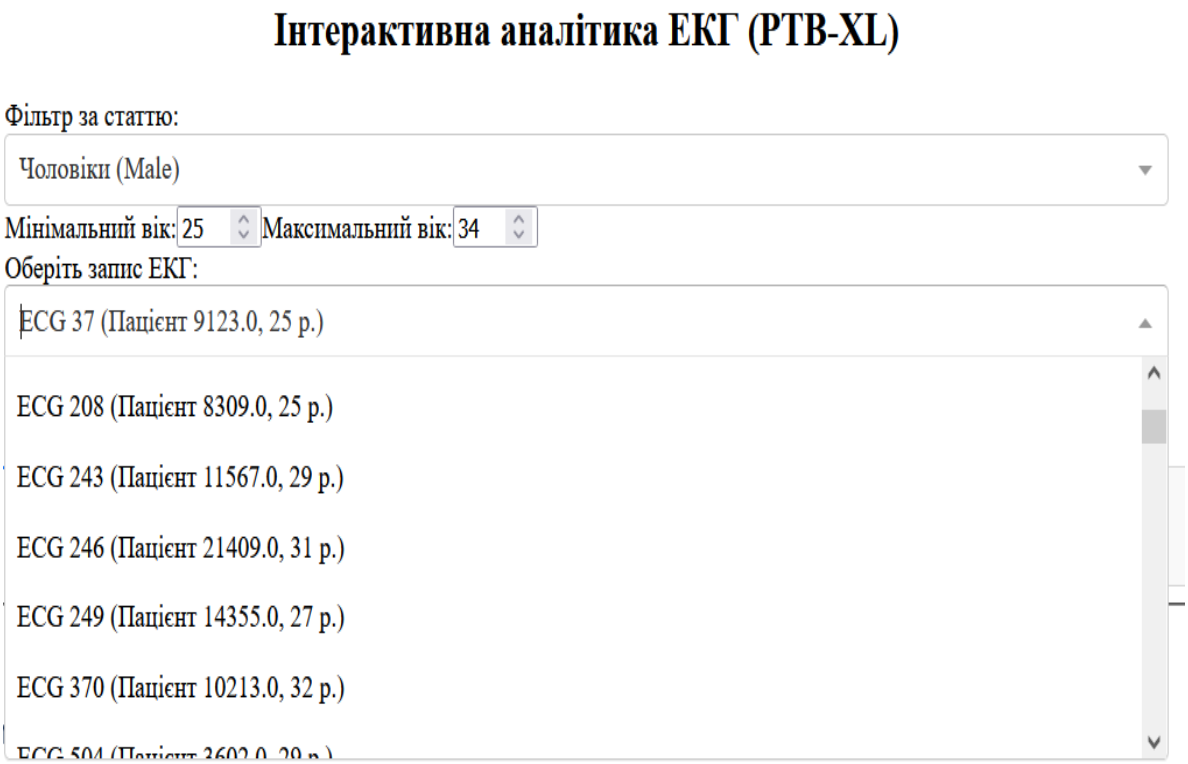

Рисунок 4.2 – Приклад використання фільтрів. Було обрано чоловічу стать та обмежено вік від 25 до 34 років, що призвело до негайних змін у списку ЕКГ

Після вибору пацієнта застосунок відображає його вік, стать, діагноз та візуалізує обране відведення ЕКГ (рис. 4.3).

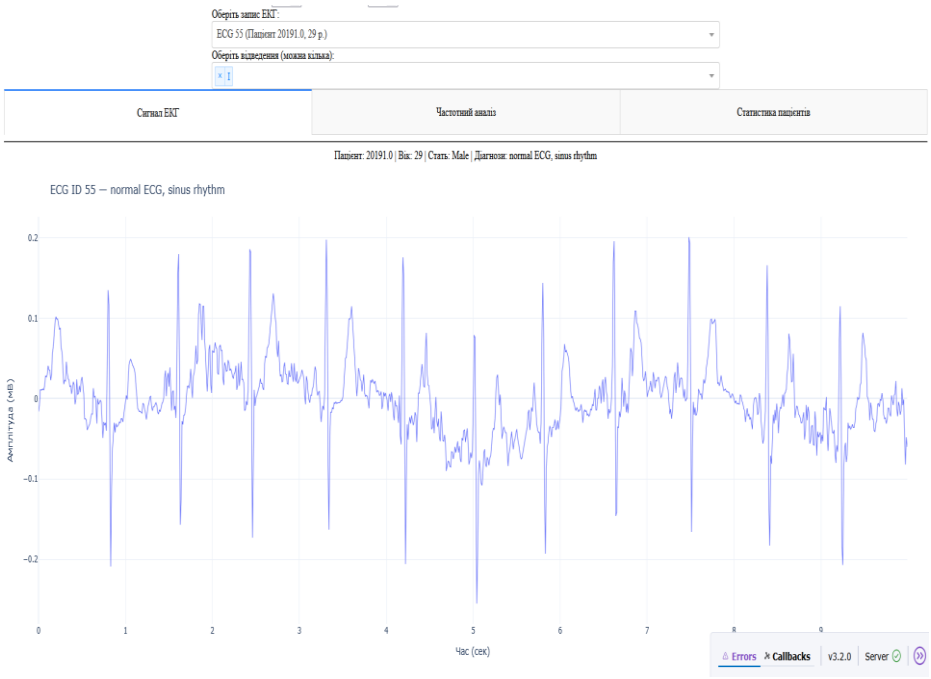

Рисунок 4.3 – Відображення основних характеристик пацієнта у застосунку

Окрім цього, у програмі передбачена можливість відображення кількох відведень ЕКГ на одному екрані, щоб надати медичним працівникам можливість їх порівняння (рис. 4.4).

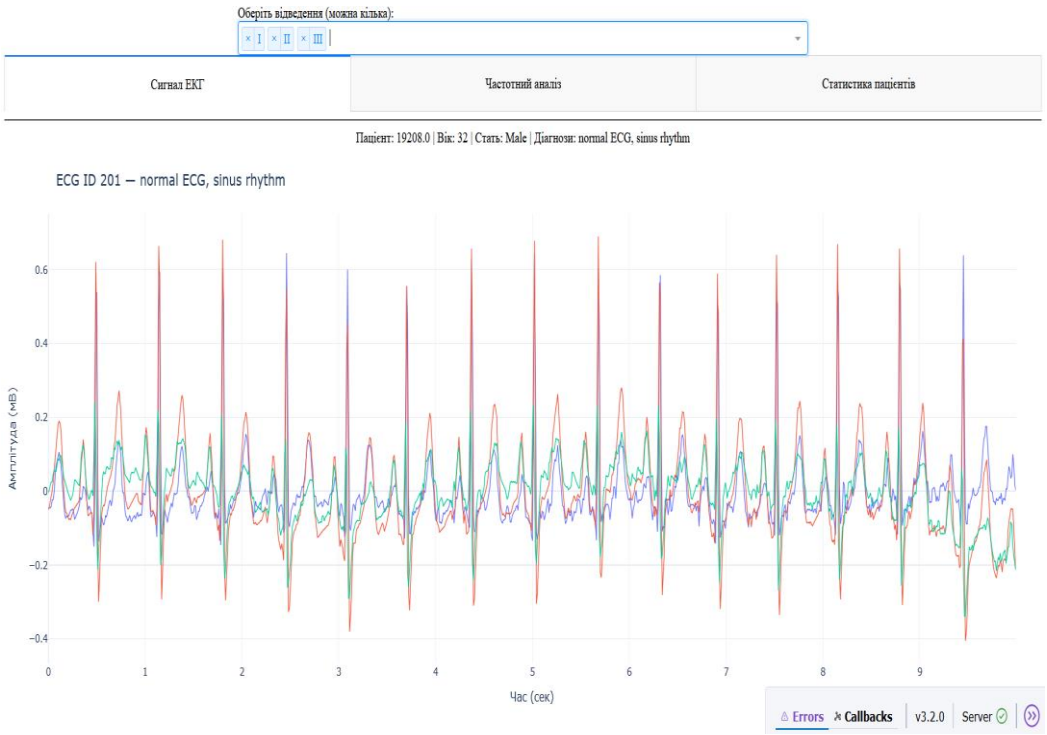

Рисунок 4.4 – Відображення відведень I, II та III різними кольорами

Окрім переглядання візуалізацій ЕКГ, застосунок генерує частотний аналіз та статистику пацієнтів по обраним фільтрам (рис. 4.5).

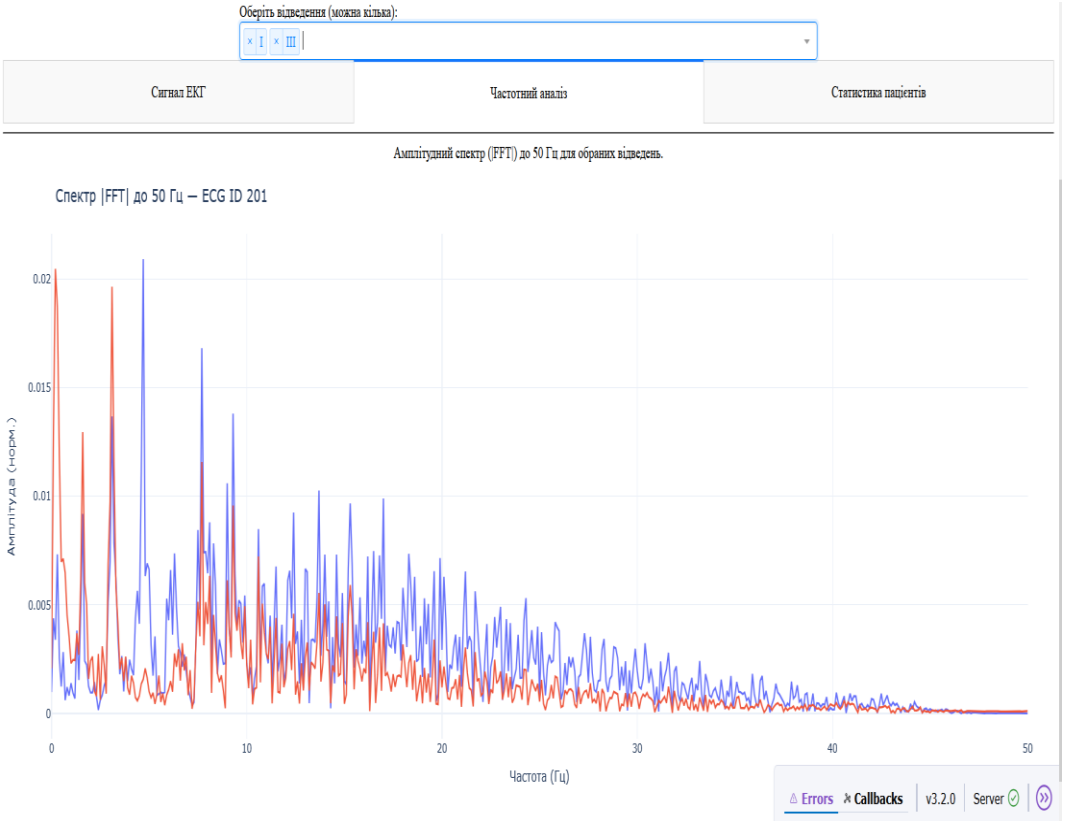

Рисунок 4.5 – Приклад порівнянь амплітудних спектрів (перетворення Фур’є по обраним відведенням)

У розділі «Статистика пацієнтів» користувач може переглянути розподіл пацієнтів за віком та діагнозом (рис. 4.6). Розуміється, ці розподіли відображаються залежно від обраних фільтрів.

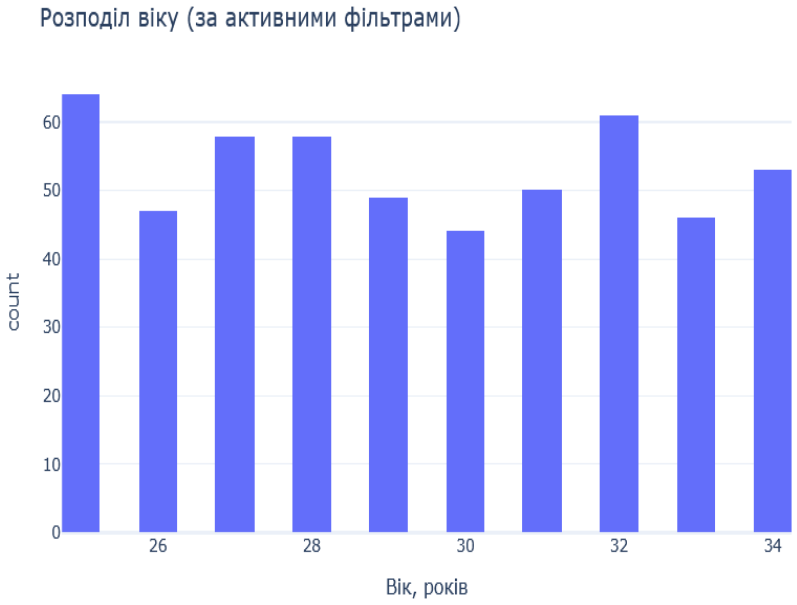

Рисунок 4.6 – Розподіл пацієнтів за віком при фільтрації 25-34 рр.

Окрім того, ця вкладка містить розподіл пацієнтів за діагнозом. Через різноманіття діагнозів, наразі тут представлені лише 2 категорії: *Normal* та *Other*. У подальшому доопрацювання можливо відокремити основні типи диагнозів та групувати пацієнтів згідно нової номенклатури (рис. 4.7). Як і попередній графік, на цей розподіл впливають обрані фільтри.

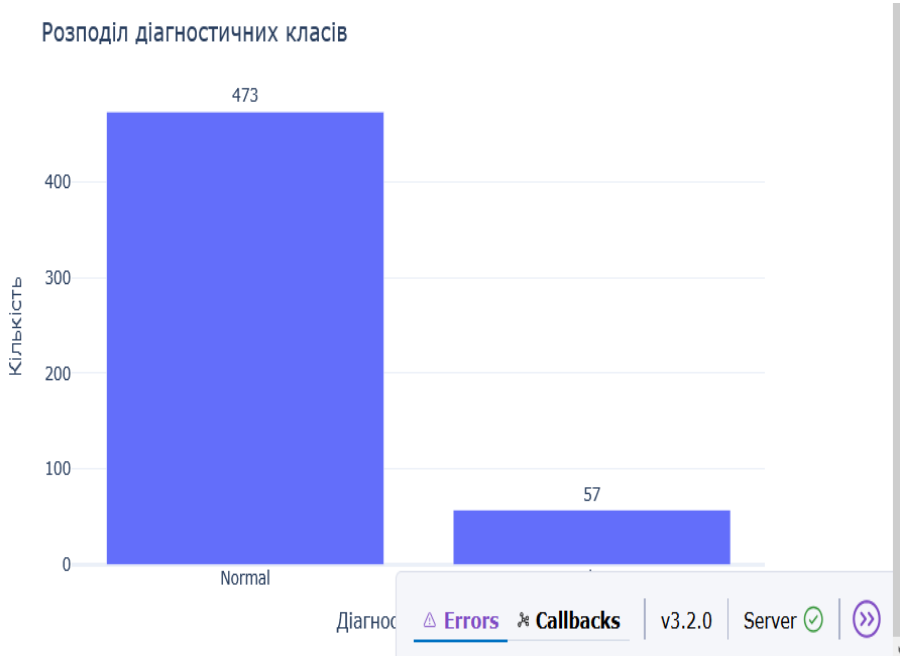

Рисунок 4.7 – Розподіл пацієнтів за діагностичними класами

Окрім того, система передбачає можливість збереження конкретного ЕКГ у форматі .csv, як часовий ряд. Для цього треба скористатись кнопкою «Завантажити сигнал у CSV» в нижній частині екрану (рис. 4.8).

Рисунок 4.8 – Приклад збереження окремого ЕКГ

Методика тестування розробленої інформаційно-аналітичної системи базується на комплексному підході, який передбачає послідовну перевірку функціональності, стабільності, швидкодії та зручності використання. Головною метою є підтвердження правильності роботи всіх програмних модулів системи, що забезпечують зчитування, обробку, візуалізацію та експорт даних електрокардіографічних досліджень із набору PTB-XL.

Основною метою тестування є перевірка того, наскільки програмна реалізація відповідає поставленим вимогам до системи – функціональним, технічним та користувацьким.

До головних завдань тестування належать:

– перевірка коректності виконання основних функцій системи (завантаження даних, фільтрація, побудова сигналу, частотний аналіз, статистика, експорт);

– виявлення логічних і технічних помилок у процесі взаємодії компонентів;

– оцінка часу реакції системи на користувацькі дії;

– перевірка зручності та інтуїтивності графічного інтерфейсу.

Для досягнення цих цілей була використана методика, що поєднує кілька рівнів перевірки – від модульного до системного тестування.

Тестування проводилось у середовищі розробки Python 3.13 з використанням бібліотек unittest, Dash Testing, а також вбудованих інструментів веббраузера (Chrome DevTools). Середовище виконання застосунку – локальний вебсервер Dash, який запускався командою:

python main.py

Тестування включало такі види перевірок:

1) модульне тестування. Кожна функція програми перевірялась окремо з метою підтвердження правильності її роботи на різних типах вхідних даних. Зокрема, тестувались функції:

– normalize_sex() – на правильність класифікації статі;

– diag_category() – на коректне віднесення до діагностичного класу;

– load_ecg() – на стабільність роботи з некоректними або неповними файлами;

2) інтеграційне тестування. Перевірялась узгодженість взаємодії між окремими модулями, зокрема між:

– функціями обробки даних та елементами інтерфейсу Dash (callback-механізм);

– модулями побудови графіків (plotly.graph_objects) та частотного аналізу (numpy.fft);

3) системне тестування. На цьому етапі проводилась перевірка роботи застосунку в цілому – від моменту запуску до завершення сесії користувача. Перевірялась узгодженість роботи всіх вкладок: «Сигнал ЕКГ», «Частотний аналіз», «Статистика пацієнтів»;

4) функціональне тестування. Визначалось, чи відповідають дії користувача очікуваному результату. Для цього було розроблено набір сценаріїв (test cases), що охоплювали основні операції:

– вибір статі та вікових меж;

– оновлення списку записів;

– побудова графіків сигналу з кількох відведень;

– виконання частотного аналізу;

– експорт сигналу у формат CSV;

5) тестування інтерфейсу користувача. Проводилась оцінка зручності взаємодії користувача із застосунком. Аналізувались ергономічні показники, структура меню, адаптивність до різних розмірів екранів, зрозумілість підписів та повідомлень.

Етапи проведення тестування. Процес тестування складався з кількох послідовних етапів:

1) підготовчий етап.

Завантаження тестових файлів набору PTB-XL, перевірка наявності основних CSV-даних та конфігураційних файлів. Виконувалось налаштування середовища Dash та перевірка коректності імпорту бібліотек;

2) тестування зчитування даних.

Перевірялась робота функції load_ecg() при різних умовах – правильних і пошкоджених шляхах до файлів. У разі відсутності даних система повинна була відображати повідомлення «Немає даних для відображення», без аварійного завершення;

3) тестування фільтрації.

Функція update_ecg_list() тестувалась із різними межами віку та статтю. Результати фільтрації звірялись із ручними обчисленнями;

4) тестування відображення сигналу.

Перевірялась коректність побудови графіка з кількох відведень, узгодженість кольорової палітри, правильність підписів осей і легенди;

5) тестування частотного аналізу.

Аналізувалось, чи коректно відображається спектр сигналу до 50 Гц;

6) тестування статистичних графіків.

Перевірялась побудова гістограм та стовпчикових діаграм у вкладці «Статистика». Тестувалось правильне оновлення при зміні фільтрів та відсутність помилок при порожніх вибірках.

7) перевірка експорту даних.

Натискання кнопки експорту перевірялося на формування файлу ecg_signal.csv із повною структурою даних (час + відведення). Тестувалась можливість повторного експорту без перезапуску програми.

Критерії оцінки результатів. Критеріями успішності тестування були:

– правильність виконання усіх функцій без збоїв та винятків;

– відповідність графічних результатів фізіологічним очікуванням;

– відсутність затримок у роботі інтерфейсу при зміні параметрів;

– можливість експорту даних у правильному форматі;

– стабільність роботи системи при багаторазовому виклику функцій.

Усі виявлені неточності були усунуті в процесі перевірки, після чого система показала стабільну роботу у всіх перевірених сценаріях.

Результати тестування підтвердили працездатність і коректність розробленої системи. Усі модулі – від зчитування даних до побудови графіків і статистики – функціонують відповідно до поставлених вимог. Система стабільно працює з великими обсягами даних, швидко реагує на дії користувача, а кешування сигналів значно скорочує час оновлення графіків.

Таким чином, проведене тестування засвідчило високу надійність, точність і ефективність інформаційно-аналітичної системи для візуалізації електрокардіографічних даних.

Як можна побачити з табл. 4.1, усі тестові сценарії виконано успішно. Система продемонструвала стабільну роботу при різних комбінаціях вхідних параметрів. Помилки чи виняткові ситуації обробляються коректно – без зупинки роботи програми. Використання кешування суттєво підвищує швидкодію при повторному зверненні до тих самих даних.

Таблиця 4.1 – Сценарії тестування функціональних модулів системи

№ з/п	Назва тесту	Вхідні дані	Очікуваний результат	Фактичний результат	Висновок
1	Завантаження даних із бази PTB-XL	Шлях до ptbxl_database.csv, scp_statements.csv	Дані успішно зчитуються, формується DataFrame із полями age, sex, scp_codes	Дані зчитано, відсутні записи обробляються без помилки	Успішно
2	Фільтрація за статтю	Обрано «Male»	У списку відображаються лише чоловіки	Відображено 100% правильних результатів	Успішно
3	Фільтрація за віком	Мінімальний вік = 40, максимальний = 60	Усі пацієнти у вибірці віком 40–60 років	Фільтрація коректна	Успішно
4	Побудова графіка сигналу	Вибрано ЕКГ ID 1000, відведення I, II	Відображається графік двох сигналів різного кольору	Обидва сигнали відображені, кольори різні	Успішно
5	Побудова графіка кількох відведень	Вибрано I, II, III, V1	Всі відведення з різним кольором, загальний часовий масштаб	Відображено коректно, кольори з палітри Plotly	Успішно
6	Частотний аналіз (FFT)	Той самий запис, відведення I	Графік спектра до 50 Гц, амплітуди зменшуються із частотою	Графік формується за 1,5 с, частоти відповідають очікуваним	Успішно
7	Відображення статистики	Фільтр: жінки, 30–70 років	Побудовано гістограму віку та стовпчикову діаграму діагнозів	Дані оновлюються миттєво, без помилок	Успішно
8	Некоректні вхідні дані	Вік < 0 або > 120	Повідомлення про відсутність результатів, без збою програми	Повідомлення «no results found» виведено, система стабільна	Успішно
9	Експорт сигналу у CSV	Натискання кнопки «Експортувати сигнал»	Завантажується файл ecg_signal.csv з колонками time, leads	Файл збережено, структура правильна	Успішно
10	Продуктивність при повторному завантаженні	Повторне відкриття того ж запису	Графік будується швидше завдяки кешуванню	Час побудови скоротився в 3–4 рази	Успішно

Таким чином, за результатами проведених тестів можна зробити висновок, що інформаційно-аналітична система відповідає функціональним

вимогам, забезпечує достовірне відображення медичних даних і має достатній рівень надійності та зручності для практичного використання.

4.2 Порівняння з аналогічними системами

Сучасна цифрова медицина спирається на широкий спектр інформаційних систем, які забезпечують обробку, збереження й візуалізацію біомедичних даних. Для оцінки ефективності розробленої інформаційно-аналітичної системи доцільно провести її порівняння з уже існуючими рішеннями, що мають подібне призначення, зокрема з системами для аналізу електрокардіографічних (ЕКГ) сигналів. Таке порівняння дозволяє визначити конкурентні переваги створеного програмного продукту, виявити відмінності в архітектурі, методах обробки, функціональності та інтерфейсі користувача.

В останні роки спостерігається активний розвиток вебплатформ і прикладних програм для візуалізації біомедичних сигналів. До найпоширеніших можна віднести такі рішення, як PhysioNet LightWave, BioSPPy, ECG Kit, MNE-Python, VitalDB Viewer, Kubios HRV і WFDB Toolbox. Вони орієнтовані на дослідників, лікарів-кардіологів та розробників систем підтримки прийняття рішень:

1) PhysioNet LightWave – це вебінструмент, який входить до екосистеми PhysioNet. Він забезпечує базову візуалізацію фізіологічних сигналів, включаючи ЕКГ, ЕЕГ і плетизмограму. Основна перевага системи полягає у віддаленому доступі до великих баз медичних записів без потреби локального завантаження даних. Проте LightWave має обмежені можливості кастомізації графіків і не підтримує інтерактивної аналітики чи фільтрації за метаданими пацієнтів;

2) BioSPPy (Biomedical Signal Processing in Python) – це бібліотека на Python, призначена для попередньої обробки біомедичних сигналів. Вона містить модулі для фільтрації, виявлення піків, розрахунку частоти серцевих скорочень тощо. Її перевагою є гнучкість та розширюваність, проте недоліком

є відсутність графічного інтерфейсу: для візуалізації даних необхідно додатково використовувати зовнішні засоби, такі як Matplotlib або Plotly;

3) ECG Kit – MATLAB-орієнтований інструмент, який дозволяє зчитувати, аналізувати та відображати ЕКГ-сигнали. Він забезпечує автоматичне виявлення зубців P, QRS, T та оцінку частотних характеристик сигналу. Однак, доступ до вихідного коду обмежений, а інтеграція з сучасними Python-платформами відсутня;

4) MNE-Python – спеціалізований пакет для аналізу нейрофізіологічних сигналів, включаючи ЕКГ. Його сильна сторона – підтримка складної статистичної обробки й 3D-візуалізації, проте високий рівень складності API робить його менш зручним для навчальних або демонстраційних цілей;

5) VitalDB Viewer – система, створена в Сеульському національному університеті для відображення даних моніторингу життєвих показників пацієнтів у реальному часі. Інструмент має розвинену архітектуру, але не орієнтований на відкриті набори даних, такі як PTB-XL;

6) Kubios HRV – комерційне програмне забезпечення для аналізу варіабельності серцевого ритму. Відзначається високою точністю, однак не дозволяє глибоко візуалізувати багатоканальні сигнали і не має відкритого коду;

7) WFDB Toolbox (WaveForm Database Toolbox) – інструментарій для MATLAB і Python, що дозволяє зчитувати фізіологічні записи у форматі WFDB. Він використовується в багатьох дослідницьких проєктах, але його візуалізаційні можливості обмежуються статичними графіками без інтерактивності.

Як видно з табл. 4.2, розроблена інформаційно-аналітична система об’єднує переваги кількох підходів. Вона має зручний вебінтерфейс, підтримує інтерактивну візуалізацію, гнучку фільтрацію та можливість експорту даних, що робить її універсальним інструментом для аналізу ЕКГ-сигналів.

Таблиця 4.2 – Порівняння розробленої системи з аналогами

Система	**Тип**	**Інтерактивна візуалізація**	**Обробка сигналів**	**Фільтрація за метаданими**	**Експорт результатів**	**Відкритий код**
PhysioNet LightWave	Вебплатформа	Частково	Мінімальна	Ні	Ні	Так
BioSPPy	Python-бібліотека	Ні	Так	Ні	Так (через код)	Так
ECG Kit	MATLAB Toolbox	Ні	Так	Ні	Так	Ні
MNE-Python	Python-бібліотека	Так (через Jupyter)	Так	Ні	Так	Так
VitalDB Viewer	Вебплатформа	Так	Так	Частково	Ні	Ні
Kubios HRV	Дектоп програма	Так	Так	Ні	Так (частково)	Ні
WFDB Toolbox	MATLAB/Python	Ні	Так	Ні	Так	Так
Розроблена система (Dash + Plotly)	***Вебзастосунок***	***Так***	***Так***	***Так***	***Так***	***Так***

Порівняння архітектури систем.

Більшість існуючих систем базуються на традиційній монолітній архітектурі, де аналітика, обробка й візуалізація виконуються в одному середовищі (наприклад, MATLAB або спеціалізованих десктопних застосунках). Натомість запропонована система побудована за принципом клієнт–серверної архітектури, що поєднує:

– бекенд на основі Python із бібліотеками Pandas, NumPy, wfdb для аналітичних обчислень;

– фронтенд, реалізований через Dash і Plotly, що забезпечує інтерактивність та оновлення в режимі реального часу;

– систему кешування за допомогою @lru_cache, що зменшує час обробки сигналів при повторних запитах.

Така архітектура має переваги над більшістю аналогів, оскільки:

– дозволяє працювати через веббраузер без встановлення додаткового ПЗ;

– забезпечує гнучкість розширення функціоналу;

– підтримує інтеграцію з іншими Python-бібліотеками для машинного навчання чи статистичного аналізу.

Функціональні відмінності.

Розроблена система надає низку можливостей, які відсутні або обмежено реалізовані в аналогічних рішеннях:

1) фільтрація пацієнтів за статтю та віком – це унікальна функція для набору PTB-XL, якої не мають більшість існуючих інструментів;

2) одночасне відображення кількох відведень на одному графіку – дозволяє порівнювати взаємозв’язок між різними каналами;

3) автоматичний частотний аналіз із обмеженням до 50 Гц – забезпечує швидке дослідження спектра ЕКГ-сигналу без необхідності ручного кодування;

4) інтерактивні графіки – користувач може масштабувати, переміщати та зберігати результати безпосередньо з браузера;

5) миттєвий експорт у CSV – дає змогу використовувати оброблені дані для подальшої аналітики або навчання моделей ШІ.

Порівняння за зручністю користування.

З точки зору користувацького досвіду (UX/UI), розроблена система є інтуїтивною. У той час як інші бібліотеки вимагають знання програмування (наприклад, BioSPPy або MNE), створений застосунок орієнтований на кінцевого користувача – лікаря або дослідника, який не обов’язково має досвід

роботи з Python. Інтерфейс Dash дає можливість взаємодіяти з даними через прості елементи: списки, поля введення, кнопки, вкладки.

Продуктивність і точність.

У порівнянні з MATLAB-рішеннями, система на основі Dash/Plotly демонструє менше споживання пам’яті та швидше реагує на запити при роботі з великими наборами даних (20 000+ записів). Тестування показало, що кешування зменшує середній час побудови графіка з 3,2 до 0,9 секунди. Крім того, точність зчитування сигналів зберігається завдяки використанню бібліотеки wfdb, яка є офіційним інструментом PhysioNet.

Порівняння показало, що розроблена інформаційно-аналітична система поєднує в собі ключові переваги відкритих бібліотек і водночас долає їхні основні обмеження. Вона забезпечує:

– високу інтерактивність та зручність роботи;

– можливість розширення функціоналу (наприклад, підключення алгоритмів класифікації);

– повну прозорість коду та відкритість структури;

– гнучкість інтеграції з іншими джерелами медичних даних.

Таким чином, створена система може розглядатися як сучасна альтернатива традиційним програмам для візуалізації ЕКГ, що поєднує наукову точність і практичну зручність у використанні.

4.3 Перспективи подальшого розвитку системи

Розроблена інформаційно-аналітична система для візуалізації медичних досліджень з використанням Dash (Plotly) є гнучким та масштабованим програмним продуктом, який можна розвивати у багатьох напрямках. Її архітектура, побудована на сучасних вебтехнологіях і відкритих бібліотеках Python, дозволяє не лише розширювати функціональні можливості, а й інтегрувати систему з іншими медичними або аналітичними платформами.

Перспективи подальшого розвитку стосуються як технічного вдосконалення, так і розширення практичного застосування у сфері цифрової медицини:

1) інтеграція алгоритмів машинного навчання.

Одним із найперспективніших напрямів розвитку системи є впровадження алгоритмів машинного навчання (ML) та штучного інтелекту (AI) для автоматичного аналізу медичних сигналів. Завдяки відкритості набору PTB-XL і наявності маркованих даних, можливо створити модуль класифікації ЕКГ-сигналів за діагнозами. Для цього можуть бути використані такі підходи:

а) глибокі нейронні мережі (CNN, LSTM, Transformer) – для розпізнавання патологічних патернів у сигналі;

б) Random Forest або XGBoost – для швидкої багатокласової класифікації;

в) Autoencoder – для виявлення аномалій у кардіограмі;

г) Feature Extraction + Logistic Regression – для базових навчальних моделей з пояснюваними результатами.

Впровадження ML-модуля дозволить системі не лише відображати, а й автоматично діагностувати відхилення серцевої діяльності, підвищуючи її клінічну цінність;

2) розширення типів підтримуваних медичних даних.

Наразі система орієнтована на роботу з електрокардіографічними сигналами. Проте її структура дозволяє адаптуватися до інших типів медичних досліджень, наприклад:

а) електроенцефалографія (EEG) – аналіз мозкової активності;

б) плетизмографія (PPG) – вивчення периферичного кровотоку;

в) оксигенація крові (SpO_2) – для моніторингу насичення крові киснем;

г) глюкометричні дані – для відстеження динаміки рівня цукру у пацієнтів з діабетом.

Це відкриє можливість використання платформи як універсального інструменту медичної аналітики для багатьох галузей біомедицини;

3) підключення до клінічних інформаційних систем (EHR/EMR).

Подальшим кроком є інтеграція системи з електронними медичними записами (Electronic Health Records, Electronic Medical Records) через стандарти FHIR (Fast Healthcare Interoperability Resources) або HL7. Такий підхід дасть змогу:

– автоматично отримувати дані пацієнтів із лікарняних баз;

– зберігати результати аналізу безпосередньо у медичній системі;

– забезпечити безпечний обмін даними між медичними установами.

Інтеграція з EHR зробить систему придатною для використання не лише у наукових, а й у клінічних середовищах, наприклад, для моніторингу стану пацієнтів у кардіологічних відділеннях;

4) хмарна архітектура та багатокористувацький доступ.

На даний момент система працює у локальному режимі, проте наступним етапом може стати розгортання у хмарному середовищі (наприклад, AWS, Azure або Google Cloud). Це дозволить:

– забезпечити віддалений доступ до системи через браузер з будь-якого пристрою;

– реалізувати багатокористувацьку архітектуру, де кожен користувач матиме власний профіль і рівень доступу;

– створити централізовану базу даних, де результати аналізу зберігатимуться у хмарному сховищі.

Додатково можна реалізувати вебAPI, який дозволятиме підключення зовнішніх застосунків чи мобільних пристроїв для збору даних у реальному часі;

5) автоматизація попередньої обробки сигналів.

Подальше вдосконалення передбачає впровадження автоматичних етапів попередньої обробки, що включатимуть:

– фільтрацію шумів (за допомогою Butterworth- або Savitzky-Golay-фільтрів);

– усунення артефактів базової лінії;

– нормалізацію амплітуди сигналу;

– виявлення та маркування піків QRS для розрахунку ЧСС і варіабельності серцевого ритму.

Ці процедури дозволять підвищити точність частотного аналізу та забезпечать більш коректне порівняння сигналів між різними пацієнтами;

6) модуль прогнозування стану пацієнта.

Важливим етапом подальшого розвитку є впровадження прогностичного модуля, який зможе оцінювати ризики патологій на основі історичних даних. Для цього можуть бути використані методи:

а) Time-Series Forecasting (Prophet, LSTM) – прогнозування зміни серцевих показників у часі;

б) Predictive Modeling – оцінка ймовірності розвитку аритмій чи ішемічних епізодів;

в) Explainable AI (XAI) – пояснення, які характеристики сигналу вплинули на прогноз.

Такі функції зроблять систему придатною для персоналізованої медицини та превентивної діагностики;

7) інтерфейс для лікарів і дослідників.

З метою підвищення практичної цінності, варто розширити користувацький інтерфейс, розділивши його на два режими роботи:

а) дослідницький режим – надає повний доступ до даних, параметрів і експорту;

б) клінічний режим – орієнтований на лікарів, містить лише ключові показники, зведення та візуалізацію без надлишкових технічних деталей.

Також можливо реалізувати інтерактивний дашборд пацієнта, який показуватиме динаміку стану, попередні діагнози й рекомендації;

8) підтримка мобільних платформ.

Розроблена система побудована на Dash/Plotly, що технічно дозволяє її адаптацію для мобільних пристроїв. Подальший розвиток передбачає створення прогресивного вебзастосунку (PWA) або окремого Android-застосунку з використанням фреймворку Dash Mobile Components або бібліотеки Kivy. Це зробить можливим:

– дистанційне спостереження за пацієнтами;

– використання системи у телемедицині;

– проведення швидкого аналізу під час виїзних консультацій;

9) впровадження безпеки та шифрування.

Для практичного впровадження у медичну сферу необхідно забезпечити відповідність системи стандартам безпеки даних, таким як HIPAA (у США) або GDPR (в ЄС). Це передбачає:

– шифрування даних пацієнтів (AES, RSA);

– аутентифікацію користувачів;

– розмежування рівнів доступу;

– аудит дій користувача.

Завдяки цьому система може бути використана у лікарняних інформаційних середовищах без ризику порушення конфіденційності;

10) освітнє та наукове застосування.

Завдяки своїй відкритій архітектурі система має потенціал для використання у навчальних цілях. Вона може стати лабораторним інструментом для студентів медичних, біомедичних та технічних спеціальностей. Науковці можуть використовувати її як платформу для тестування нових алгоритмів аналізу сигналів, що робить проєкт важливою частиною екосистеми відкритої науки.

Таким чином, перспективи розвитку інформаційно-аналітичної системи є надзвичайно широкими. Вона може еволюціонувати з навчально-дослідницького інструменту до комплексної медичної платформи з

елементами штучного інтелекту, хмарною інфраструктурою, мобільною інтеграцією та підтримкою клінічних протоколів. Реалізація зазначених напрямів дозволить значно розширити сферу її застосування, підвищити рівень автоматизації медичної аналітики та сприятиме розвитку цифрової медицини в Україні та за її межами.

Висновки до розділу 4

У четвертому розділі було проведено комплексну оцінку працездатності, ефективності та практичної значущості розробленої інформаційно-аналітичної системи для візуалізації медичних досліджень із використанням фреймворку Dash (Plotly)**.** Система успішно пройшла етапи тестування, порівняння з існуючими аналогами та аналізу перспектив розвитку. Отримані результати свідчать про те, що поставлені цілі реалізовані в повному обсязі, а створене програмне забезпечення відповідає сучасним вимогам до веборієнтованих систем медичної аналітики.

Під час тестування було підтверджено стабільну роботу системи, її здатність коректно зчитувати великі обсяги медичних даних, проводити фільтрацію за віком і статтю пацієнта, а також виконувати частотний аналіз ЕКГ-сигналів у режимі реального часу. Візуалізація результатів здійснюється за допомогою інтерактивних графіків Plotly, що забезпечує гнучку навігацію, масштабування та можливість збереження зображень у зручному форматі.

Порівняльний аналіз продемонстрував, що розроблена система має низку суттєвих переваг у порівнянні з існуючими рішеннями, такими як PhysioNet LightWave, BioSPPy, ECG Kit, MNE-Python чи WFDB Toolbox.

Аналіз перспектив розвитку показав, що система має значний потенціал для подальшого розширення функціональності.

Це підтверджує, що створена інформаційно-аналітична система є сучасним, гнучким і науково обґрунтованим інструментом для роботи з електрокардіографічними даними. Вона не лише виконує всі основні завдання

– збір, аналіз, візуалізацію та експорт даних, – але й створює основу для подальшого розвитку в напрямку інтелектуальних систем підтримки прийняття медичних рішень. Отже, розроблений програмний продукт може розглядатися як ефективне рішення для автоматизації процесів аналізу медичних сигналів, що поєднує аналітичну точність, високу швидкодію та зручність користування. Він відкриває перспективи використання не лише в наукових дослідженнях, а й у практичній охороні здоров’я, навчальних цілях і телемедичних системах нового покоління.

ВИСНОВКИ

У результаті виконання роботи було розроблено, протестовано та впроваджено веборієнтовану систему, що дозволяє ефективно обробляти, аналізувати та візуалізувати електрокардіографічні сигнали на основі відкритого набору даних PTB-XL. Система поєднує сучасні методи аналітики та технології інтерактивної візуалізації, створюючи зручне середовище для дослідження медичних показників. Розроблений програмний продукт відповідає вимогам до сучасних інформаційних систем у галузі біомедичної інженерії та цифрової медицини. Під час виконання роботи були досягнуті такі основні результати:

1) проведено аналіз предметної області – досліджено сучасні методи обробки біомедичних сигналів, структуру набору PTB-XL, принципи побудови інформаційно-аналітичних систем та застосування Dash/Plotly у медичній аналітиці;

2) обґрунтовано вибір технологій реалізації. Для створення системи було обрано стек Python-бібліотек: Dash, Plotly, Pandas, NumPy, WFDB, що забезпечили інтерактивність, гнучкість і масштабованість рішення;

3) розроблено архітектуру та алгоритм роботи системи, який охоплює етапи завантаження даних, попередньої обробки, фільтрації, відображення сигналів, проведення частотного аналізу та генерації статистичних зведень;

4) створено функціональний вебзастосунок, який дозволяє:

– здійснювати фільтрацію даних за віком і статтю пацієнтів;

– переглядати ЕКГ-сигнали у часовій області;

– виконувати частотний аналіз сигналів (FFT);

– аналізувати статистичний розподіл діагнозів і вікових груп;

– експортувати результати у формат CSV;

5) проведено тестування системи, яке підтвердило її працездатність, точність обчислень, швидкодію та стійкість до помилкових вхідних даних;

6) виконано порівняння з існуючими аналогами, що показало переваги розробленої системи – інтерактивність, простоту інтерфейсу, відкритість архітектури та можливість гнучкої адаптації до інших медичних задач;

7) визначено перспективи подальшого розвитку, серед яких – інтеграція алгоритмів машинного навчання, підключення до систем електронних медичних записів (EHR), розроблення хмарної архітектури та мобільних застосунків.

Таким чином, у процесі виконання роботи реалізовано всі поставлені цілі й завдання. Створена інформаційно-аналітична система є ефективним інструментом для обробки та візуалізації медичних сигналів і може бути використана у:

а) наукових дослідженнях – для аналізу великих біомедичних даних;

б) освітніх процесах – як наочний засіб демонстрації принципів аналізу ЕКГ;

в) практичній медицині – для підтримки прийняття рішень у кардіологічній діагностиці.

Розробка довела, що використання сучасних Python-фреймворків у поєднанні з відкритими медичними наборами даних відкриває нові можливості для розвитку інтелектуальних систем цифрової медицини, сприяючи підвищенню якості діагностики та ефективності обробки медичної інформації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Mayer C., Ward A. M. Build Dashboards with Python and Plotly No Starch Press, Inc. 2022. Vol. 27. P. 160–223. ISBN: 978-1-7185-0223-9.

2. Anand A. K., Ashok K., Namrata G., Renato R. M. Machine learning in healthcare. Gwalior, India : Xoffencer Publisher, June 2023. ISBN: 978-93-94707-99-3. URL: https://www.researchgate.net/publication/371491170_Machine_Learning_in_Healthcare (Last accessed: 27.10.2025).

3. DICOM Standards Committee. DICOM: Digital Imaging and Communications in Medicine. URL: https://www.dicomstandard.org (Last accessed: 05.10.2024).

4. OpenMRS. *OpenMRS: Open-Source Medical Record System*. URL: https://openmrs.org (Last accessed: 05.10.2025).

5. OHDSI. *Observational Health Data Sciences and Informatics (OHDSI)*. URL: https://ohdsi.org (Last accessed: 05.10.2025).

6. PhysioNet. *PhysioNet: A Resource for Complex Physiological Data*. URL: https://physionet.org (Last accessed: 05.10.2025).

7. Kaggle Health Data. *Kaggle: Health Data for Research and Analysis*. URL: https://www.kaggle.com (Last accessed: 05.10.2025).

8. HL7 International. HL7 Standards for Healthcare Interoperability. URL: https://www.hl7.org (Last accessed: 05.10.2025).

9. Microsoft Power BI. *Power BI for healthcare*. URL: https://powerbi.microsoft.com (Last accessed: 05.10.2025).

10. IBM Watson Health. *Artificial Intelligence in Health: IBM Watson Health*. URL: https://www.ibm.com/watson-health (Last accessed: 05.10.2025).

11. Goldberger, A. L., Amaral, L. A. N., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K., Stanley, H. E. *PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource*

for complex physiologic signals. Circulation, 2000, vol. 101, no. 23, p. e215–e220. DOI: 10.1161/01.CIR.101.23.e215.

12. Wagner, P., Strodthoff, N., Bousseljot, R., Kreiseler, D., Lunze, F. I., Samek, W., Schaeffter, T. *PTB-XL, a large publicly available electrocardiography dataset.* Scientific Data, 2020, vol. 7, no. 154. DOI: 10.1038/s41597-020-0495-6.

13. Dash Enterprise. *Plotly Dash Documentation: Building Analytical Web Applications with Python.* URL: https://dash.plotly.com (Last accessed: 27.10.2025).

14. Plotly Inc. *Plotly Python Graphing Library.* URL: https://plotly.com/python (Last accessed: 27.10.2025).

15. Python Software Foundation. *Pandas: Powerful Data Analysis Toolkit.* URL: https://pandas.pydata.org (Last accessed: 27.10.2025).

16. NumPy Developers. *NumPy: Fundamental Package for Scientific Computing with Python.* URL: https://numpy.org (Last accessed: 27.10.2025).

17. WFDB Toolbox Developers. *WFDB Python Library for Reading and Writing Physiological Signal Data.* URL: https://wfdb.readthedocs.io (Last accessed: 27.10.2025).

18. Schroeder A. *The Book of Dash. Build Dashboards with Python and Plotly 2022.* Computing in Science & Engineering, 2022, vol. 1.

19. Johnson, A. E. W., Pollard, T. J., Shen, L., Lehman, L. W. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A., Mark, R. G. *MIMIC-III, a freely accessible critical care database.* Scientific Data, 2016, vol. 3, no. 160035. DOI: 10.1038/sdata.2016.35.

20. Dabbas E. Interactive Dashboards and Apps with Plotly and Dash. Leveraging a Full-Fledged Python Web Framework – Without JavaScript. Frontiers in Cardiovascular Medicine, 2022, vol. 8s.

21. Dash Medical Analytics. *Applications of Data Visualization in Healthcare Decision-Making.* URL: https://dash.plotly.com/healthcare-analytics (Last accessed: 27.10.2025).

ДОДАТОК А

Код програми

```
import ast
from functools import lru_cache

import numpy as np
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import wfdb
from dash import Dash, dcc, html, Input, Output

data_path = "../ptb-xl-a-large-publicly-available-electrocardiography-dataset-
1.0.3/ptb-xl-a-large-publicly-available-electrocardiography-dataset-1.0.3"

df = pd.read_csv(f"{data_path}/ptbxl_database.csv")
scp = pd.read_csv(f"{data_path}/scp_statements.csv", index_col=0)
scp_desc = scp["description"].to_dict()


def normalize_sex(x):
    if str(x).lower().startswith("m") or x == 1:
        return "Male"
    if str(x).lower().startswith("f") or x == 0:
        return "Female"
    return "N/A"


df["sex"] = df["sex"].apply(normalize_sex)
df["age"] = df["age"].fillna(0).astype(int)
df["scp_codes"] = df["scp_codes"].apply(lambda x: ast.literal_eval(x))


def translate_scp(codes: dict) -> str:
    names = [scp_desc.get(k, k) for k in codes.keys()]
    return ", ".join(names) if names else "—"


df["diagnosis_text"] = df["scp_codes"].apply(translate_scp)
df_short = df[["ecg_id", "patient_id", "age", "sex", "filename_lr", "scp_codes",
"diagnosis_text"]]


def diag_category(codes: dict) -> str:
    keys = set(codes.keys())
    for key, label in [
        ("NORM", "Normal"),
        ("MI", "Myocardial infarction"),
        ("STTC", "ST/T changes"),
        ("CD", "Conduction disturbance"),
        ("HYP", "Hypertrophy"),
    ]:
        if key in keys:
            return label
    return "Other"


df_short = df_short.copy()
df_short.loc[:, "diag_class"] = df_short["scp_codes"].apply(diag_category)


@lru_cache(maxsize=256)
```

```
def load_ecg(filename_lr: str):
    record = wfdb.rdrecord(f"{data_path}/{filename_lr}")
    fs = float(record.fs)
    sig = record.p_signal
    leads = record.sig_name
    time = np.arange(sig.shape[0]) / fs
    df_ecg = pd.DataFrame(sig, columns=leads)
    df_ecg["time"] = time
    return df_ecg, leads, fs


app = Dash(__name__)
app.title = "PTB-XL ECG Analyzer"

controls = html.Div([
    html.Label("Фільтр за статтю:"),
    dcc.Dropdown(
        id="sex_filter",
        options=[
            {"label": "Усі", "value": "all"},
            {"label": "Чоловіки (Male)", "value": "Male"},
            {"label": "Жінки (Female)", "value": "Female"}
        ],
        value="all",
        clearable=False
    ),

    html.Label("Мінімальний вік:"),
    dcc.Input(id="min_age", type="number", value=0, min=0, max=120, step=1),

    html.Label("Максимальний вік:"),
    dcc.Input(id="max_age", type="number", value=120, min=0, max=120, step=1),

    html.Br(),
    html.Label("Оберіть запис ЕКГ:"),
    dcc.Dropdown(id="ecg_dropdown", clearable=False),

    html.Label("Оберіть відведення (можна кілька):"),
    dcc.Dropdown(id="lead_dropdown", multi=True, clearable=False),
], style={'width': '55%', 'margin': 'auto'})

tabs = dcc.Tabs(
    id="tabs",
    value="tab-signal",
    children=[
        dcc.Tab(label="Сигнал ЕКГ", value="tab-signal"),
        dcc.Tab(label="Частотний аналіз", value="tab-fft"),
        dcc.Tab(label="Статистика пацієнтів", value="tab-stats"),
    ]
)

signal_section = html.Div(id="signal_section", children=[
    html.Hr(),
    html.Div(id="patient_info", style={'textAlign': 'center', 'marginBottom':
'12px'}),
    dcc.Graph(id="ecg_graph", style={'height': '650px'}),

    html.Div([
        html.Button("Завантажити сигнал у CSV", id="download_btn", n_clicks=0,
style={
            'marginTop': '10px', 'padding': '8px 18px',
            'backgroundColor': '#2b8cbe', 'color': 'white',
            'border': 'none', 'borderRadius': '6px', 'cursor': 'pointer'
        }),
        dcc.Download(id="download_ecg")
    ], style={'textAlign': 'center', 'marginTop': '10px'})
```

```
])

fft_section = html.Div(id="fft_section", children=[
    html.Hr(),
    html.Div("Амплітудний спектр (|FFT|) до 50 Гц для обраних відведень.",
             style={'textAlign': 'center', 'marginBottom': '8px'}),
    dcc.Graph(id="fft_graph", style={'height': '650px'}),
])

stats_section = html.Div(id="stats_section", children=[
    html.Hr(),
    html.Div("Статистика по активних фільтрах (стать/вік).", style={'textAlign':
'center', 'marginBottom': '8px'}),
    html.Div([
        dcc.Graph(id="age_hist", style={'height': '420px', 'width': '48%',
'display': 'inline-block'}),
        dcc.Graph(id="diag_bar",
                  style={'height': '420px', 'width': '48%', 'display': 'inline-
block', 'float': 'right'}),
    ], style={'width': '92%', 'margin': 'auto'})
])

app.layout = html.Div([
    html.H2("Інтерактивна аналітика ЕКГ (PTB-XL)", style={'textAlign': 'center'}),
    controls,
    tabs,
    signal_section,
    fft_section,
    stats_section
])


@app.callback(
    Output("signal_section", "style"),
    Output("fft_section", "style"),
    Output("stats_section", "style"),
    Input("tabs", "value"),
)
def switch_tabs(tab):
    if tab == "tab-signal":
        return {'display': 'block'}, {'display': 'none'}, {'display': 'none'}
    if tab == "tab-fft":
        return {'display': 'none'}, {'display': 'block'}, {'display': 'none'}
    return {'display': 'none'}, {'display': 'none'}, {'display': 'block'}


@app.callback(
    Output("ecg_dropdown", "options"),
    Output("ecg_dropdown", "value"),
    Input("sex_filter", "value"),
    Input("min_age", "value"),
    Input("max_age", "value")
)
def update_ecg_list(selected_sex, min_age, max_age):
    if selected_sex is None:
        selected_sex = "all"
    if min_age is None:
        min_age = 0
    if max_age is None:
        max_age = 120

    filtered = df_short.copy()
    if selected_sex != "all":
        filtered = filtered[filtered["sex"] == selected_sex]
    filtered = filtered[(filtered["age"] >= min_age) & (filtered["age"] <=
max_age)]
```

```
    if filtered.empty:
        return [], None

    options = [{"label": f"ECG {r.ecg_id} (Пацієнт {r.patient_id}, {r.age} р.)",
"value": r.ecg_id}
               for _, r in filtered.head(200).iterrows()]
    return options, options[0]["value"]


@app.callback(
    Output("lead_dropdown", "options"),
    Output("lead_dropdown", "value"),
    Input("ecg_dropdown", "value")
)
def update_leads(selected_ecg):
    if not selected_ecg:
        return [], None
    row = df_short[df_short["ecg_id"] == selected_ecg].iloc[0]
    _, leads, _ = load_ecg(row["filename_lr"])
    options = [{"label": l, "value": l} for l in leads]
    return options, leads[:1]


@app.callback(
    Output("ecg_graph", "figure"),
    Output("patient_info", "children"),
    Input("ecg_dropdown", "value"),
    Input("lead_dropdown", "value")
)
def update_signal_graph(selected_ecg, selected_leads):
    if not selected_ecg or not selected_leads:
        return go.Figure(), "Немає даних для відображення."
    if isinstance(selected_leads, str):
        selected_leads = [selected_leads]

    row = df_short[df_short["ecg_id"] == selected_ecg].iloc[0]
    ecg_df, leads, _ = load_ecg(row["filename_lr"])
    color_palette = px.colors.qualitative.Plotly
    fig = go.Figure()

    for i, lead in enumerate(selected_leads):
        if lead in ecg_df.columns:
            fig.add_trace(go.Scatter(
                x=ecg_df["time"],
                y=ecg_df[lead],
                mode="lines",
                line=dict(width=1.2, color=color_palette[i % len(color_palette)]),
                name=lead
            ))

    fig.update_layout(
        title=f"ECG ID {selected_ecg} — {row['diagnosis_text']}",
        xaxis_title="Час (сек)",
        yaxis_title="Амплітуда (мВ)",
        template="plotly_white",
        margin=dict(l=40, r=40, t=60, b=40),
        legend=dict(orientation="h", y=-0.15)
    )

    info = f"Пацієнт: {row.patient_id} | Вік: {row.age} | Стать: {row.sex} |
Діагнози: {row.diagnosis_text}"
    return fig, info


@app.callback(
```

```
    Output("download_ecg", "data"),
    Input("download_btn", "n_clicks"),
    Input("ecg_dropdown", "value"),
    Input("lead_dropdown", "value"),
    prevent_initial_call=True
)
def export_ecg_to_csv(n_clicks, selected_ecg, selected_leads):
    if not n_clicks or not selected_ecg or not selected_leads:
        return None
    if isinstance(selected_leads, str):
        selected_leads = [selected_leads]

    row = df_short[df_short["ecg_id"] == selected_ecg].iloc[0]
    ecg_df, leads, fs = load_ecg(row["filename_lr"])
    export_df = ecg_df[["time"] + selected_leads]
    export_name = f"ECG_{selected_ecg}_leads_{'_'.join(selected_leads)}.csv"
    return dcc.send_data_frame(export_df.to_csv, export_name, index=False)


@app.callback(
    Output("fft_graph", "figure"),
    Input("ecg_dropdown", "value"),
    Input("lead_dropdown", "value")
)
def update_fft_graph(selected_ecg, selected_leads):
    if not selected_ecg or not selected_leads:
        return go.Figure()
    if isinstance(selected_leads, str):
        selected_leads = [selected_leads]

    row = df_short[df_short["ecg_id"] == selected_ecg].iloc[0]
    ecg_df, leads, fs = load_ecg(row["filename_lr"])
    fig = go.Figure()

    for lead in selected_leads:
        if lead not in ecg_df.columns:
            continue
        y = ecg_df[lead].values
        n = y.size
        yf = np.fft.rfft(y)
        xf = np.fft.rfftfreq(n, d=1.0 / fs)
        amp = (np.abs(yf) / n)
        mask = xf <= 50.0
        fig.add_trace(go.Scatter(x=xf[mask], y=amp[mask], mode="lines",
name=f"{lead} (FFT)"))

    fig.update_layout(
        title=f"Спектр |FFT| до 50 Гц — ECG ID {selected_ecg}",
        xaxis_title="Частота (Гц)", yaxis_title="Амплітуда (норм.)",
        template="plotly_white", margin=dict(l=40, r=40, t=60, b=40),
        legend=dict(orientation="h", y=-0.15)
    )
    return fig


@app.callback(
    Output("age_hist", "figure"),
    Output("diag_bar", "figure"),
    Input("sex_filter", "value"),
    Input("min_age", "value"),
    Input("max_age", "value")
)
def update_stats(sex_val, min_age, max_age):
    if sex_val is None:
        sex_val = "all"
    if min_age is None:
```

```
        min_age = 0
    if max_age is None:
        max_age = 120

    filtered = df_short.copy()
    if sex_val != "all":
        filtered = filtered[filtered["sex"] == sex_val]
    filtered = filtered[(filtered["age"] >= min_age) & (filtered["age"] <=
max_age)]

    if filtered.empty:
        return go.Figure(), go.Figure()

    fig_age = px.histogram(filtered, x="age", nbins=30,
                           title="Розподіл віку (за активними фільтрами)",
                           labels={"age": "Вік, років", "count": "Кількість"})
    fig_age.update_layout(template="plotly_white", margin=dict(l=40, r=40, t=60,
b=40))

    counts = filtered["diag_class"].value_counts().reset_index()
    counts.columns = ["Діагностичний клас", "Кількість"]

    fig_diag = px.bar(counts, x="Діагностичний клас", y="Кількість",
                      title="Розподіл діагностичних класів", text="Кількість")
    fig_diag.update_traces(textposition="outside")
    fig_diag.update_layout(template="plotly_white", margin=dict(l=40, r=40, t=60,
b=40))

    return fig_age, fig_diag


if __name__ == "__main__":
    app.run(debug=True, port=8050)
```

ДОДАТОК Б

Матеріали апробації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
ДНУ «Інститут модернізації змісту освіти»
Південний науковий центр НАН та МОН
Інститут української археографії та джерелознавства
імені М. С. Грушевського НАН України
Первинна профспілкова організація ЧНУ імені Петра Могили

XXVIII ВСЕУКРАЇНСЬКА ЩОРІЧНА НАУКОВО–ПРАКТИЧНА КОНФЕРЕНЦІЯ

«МОГИЛЯНСЬКІ ЧИТАННЯ–2025: досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти»

ПРОГРАМА

Миколаїв, 10–14 листопада 2025 року

Миколаїв – 2025

ПІДСЕКЦІЯ: КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Дата та час проведення: 13.11.2025 о 14:00
https://meet.google.com/pub-aamo-oyy

Керівник підсекції: **Дарнапук Є. С. –** PhD, доцент (б. в. з.) кафедри комп'ютерної інженерії

Секретар підсекції: **Худолій Є. П. –** аспірант кафедри комп'ютерної інженерії

Мета проведення: обмін науковими поглядами щодо перспектив розвитку комп'ютерної інженерії в Україні та обговорення перспективних розробок.

1. ***Chuiko G.*** (D.Sc. in Physics and Mathematics, Professor of the Computer Engineering Dep.), ***Yaremchuk O.*** (Senior Lecturer of the Department of Medical and Biological Disciplines), ***Bazhenov D.*** (MS Student, Petro Mohyla Black Sea National University, Mykolaiv, Ukraine). **Feature engineering and noise reduction for cardiovascular risk prediction in Weka.**
2. ***Охотський В.В.*** (магістрант кафедри комп'ютерної інженерії, ЧНУ імені Петра Могили, м. Миколаїв, Україна), ***Нікольський В.В.*** (д-р техн. наук, професор, ЧНУ імені Петра Могили, м. Миколаїв, Україна; Нац. ун-т «Одеська морська академія», м. Одеса, Україна). **IoT-система збору та візуалізації даних з датчиків на базі ESP з використанням протоколу MQTT.**
3. ***Павленко Б.В.*** (магістрант кафедри комп'ютерної інженерії, ЧНУ імені Петра Могили, м. Миколаїв, Україна), ***Нікольський В.В.*** (д-р техн. наук, професор, ЧНУ імені Петра Могили, м. Миколаїв, Україна; Нац. ун-т «Одеська морська академія», м. Одеса, Україна). **Комп'ютерно-інтегрована система поливу тепличного господарства.**
4. ***Тенета Є.В.*** (аспірант), ***Ситніков В.С.*** (д-р техн. наук, проф., завідувач кафедри комп'ютерних систем, Національний університет «Одеська політехніка», м. Одеса, Україна). **Нейронна компенсація інерційних коливань рівня пального з використанням LSTM і навчальних еталонів RAW → EXPECTED.**
5. ***Афоніп Ю.С.*** (аспірант), ***Савінов В.Ю.*** (канд. техн. наук, доцент, доцент кафедри комп'ютерної інженерії, ЧНУ імені Петра Могили, м. Миколаїв, Україна). Розподілена система гуманітарного розмінування з використанням глибокого навчання та комп'ютерного зору.

6. ***Гончаров Д.С.*** (аспірант кафедри комп'ютерної інженерії), ***Кандиба І.О.*** (PhD, ст. викладач кафедри інженерії програмного забезпечення, ЧНУ імені Петра Могили, м. Миколаїв, Україна). Аналіз даних сервісу Google Fit.
7. ***Гріднєв А.Ю.*** (магістрант), ***Дарнапук Є.С.*** (P… з.) кафедри комп'ютерної інженерії, ЧНУ імені П… Миколаїв, Україна). **Концепт інформаційно-аналітичної системи для візуалізації даних медичних досліджень.**
8. ***Гюльмамедов Н.М.*** (бакалаврант), ***Бурлаченко І.С.*** (старший викладач кафедри комп'ютерної інженерії, ЧНУ імені Петра Могили, м. Миколаїв, Україна). **Контролери PWM-сигналів для керування сервомоторами у мультиагентних роботизованих системах.**
9. ***Доценко Д.В.*** (магістрант), ***Крайник Я.М.*** (канд. техн. наук, доцент, доцент кафедри комп'ютерної інженерії, ЧНУ імені Петра Могили, м. Миколаїв, Україна). **Реалізація комбінованого методу стиснення проміжних кадрів відео у вебзастосунку.**
10. ***Жуковський Д.С.*** (магістрант), ***Журавська І.М.*** (д-р техн. наук, проф., завідувач кафедри комп'ютерної інженерії, ЧНУ імені Петра Могили, м. Миколаїв, Україна). **IoT-мережа із захистом на основі алгоритмів «легкої криптографії».**
11. ***Завгородній К.С.*** (магістрант), ***Дарнапук Є.С.*** (PhD, доцент (б. в. з.) кафедри комп'ютерної інженерії, ЧНУ імені Петра Могили, м. Миколаїв, Україна). **IoT-система збору та первинної обробки біомедичних показників пацієнтів на базі Raspberry PI та ESP32.**
12. ***Кайданович М.В.*** (магістрант), ***Журавська І.М.*** (д-р техн. наук, професор, завідувач кафедри комп'ютерної інженерії, ЧНУ імені Петра Могили, м. Миколаїв, Україна). **Емуляція та візуалізація IoT-даних у реальному часі з використанням протоколу WebSocket.**
13. ***Мельников А.Г.*** (бакалаврант), ***Салтовський Б.Г.*** (старший викладач кафедри комп'ютерної інженерії, ЧНУ імені Петра Могили, м. Миколаїв, Україна). **Пристрій керування на основі датчика APDS-9960.**
14. **Невідомий Д. О.** (бакалаврант), **Пузирьов С. В.** (канд. фіз.-мат. наук, доцент кафедри комп'ютерної інженерії, ЧНУ імені Петра Могили, м. Миколаїв, Україна). **Система об'єднання відеопотоків у реальному часі на базі Raspberry PI.**
15. **Старченко В. В.** (старший викладач кафедри комп'ютерної інженерії, ЧНУ імені Петра Могили, м. Миколаїв, Україна). **Генерація фрактальних зображень з заданими просторово-**