

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Факультет інформаційних технологій

**ПОГОДЖЕНО
Декан факультету**

Ігор БОЛБОТ

(підпис)

«__» _____ 2026 р.

**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри комп'ютерних наук**

Белла ГОЛУБ

(підпис)

«__» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Система аналізу кардіосигналів на основі згорткових нейронних мереж»

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми

д.е.н., професор

**Керівник бакалаврської
кваліфікаційної роботи**

асистент

Консультант:

д.т.н., професор

Виконав

(підпис) **Роман РУДЕНСЬКИЙ**

(підпис) **Денис ВІННІЧУК**

(підпис) **Віктор СЕМКО**

(підпис) **Василь
ГОРДАШЕВСЬКИЙ**

Київ-2026

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____

(підпис)

Белла ГОЛУБ

«___» _____ 2026 р.

**ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ**

_____ **Гордашевському Василю Васильовичу** _____

(прізвище, ім'я, по батькові)

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи

«Система аналізу кардіосигналів на основі згорткових нейронних мереж»
затверджена наказом від «6» січня 2026 р. № 46 «С»

Термін подання завершеної роботи на кафедру «25» травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту):

ЕКГ-сигнали з набору даних PTB-XL, навчена модель CNN-LSTM, тестові файли
X_test.npy та y_test.npy, засоби Python, TensorFlow/Keras, SQLite та вимоги до
програмного забезпечення аналізу кардіосигналів

Перелік питань, що підлягають дослідженню:

- Аналіз взаємодії користувача з системою.
- Проєктування і розробка інформаційного забезпечення системи.
- Проєктування і розробка програмного забезпечення системи.
- Впровадження та експлуатація системи.

Перелік графічних документів (за необхідності).

Дата видачі завдання «6» січня 2026 р.

**Керівник бакалаврської
кваліфікаційної роботи**

(підпис)

Денис ВІННІЧУК

Консультант:

д.т.н., професор

(підпис)

Віктор СЕМКО

**Завдання взяв до
виконання**

(підпис)

Василь ГОРДАШЕВСЬКИЙ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	4
ВСТУП	6
1 АНАЛІЗ ВЗАЄМОДІЇ КОРИСТУВАЧА З СИСТЕМОЮ	9
1.1 Постановка завдання	9
1.2 Огляд інформаційних джерел та існуючих рішень	12
1.3 Моделювання предметної області	15
2 ПРОЄКТУВАННЯ І РОЗРОБКА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ	21
2.1 Логічна модель даних	21
2.2 Вибір системи управління інформаційною базою	25
2.3 Створення інформаційної бази	29
3 ПРОЄКТУВАННЯ І РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ	37
3.1 Організаційна структура програмного забезпечення	37
3.2 Вибір інструментарію для створення ППЗ	42
3.3. Алгоритмізація та програмування програмних модулів	47
4 ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ СИСТЕМИ	57
4.1 Тестування системи	57
4.2 Вимоги до апаратного та програмного забезпечення	64
4.3 Склад інсталяційного пакету	68
ВИСНОВКИ	72
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	75
ДОДАТОК А	77

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ЕКГ - електрокардіограма.

Кардіосигнал - електричний сигнал, що відображає роботу серця та використовується для аналізу його функціонального стану.

CNN - Convolutional Neural Network, згорткова нейронна мережа.

LSTM - Long Short-Term Memory, різновид рекурентної нейронної мережі, що використовується для обробки послідовностей даних.

CNN-LSTM - комбінована нейронна мережа, яка поєднує згорткові шари для виділення ознак сигналу та LSTM-шари для аналізу часових залежностей.

PTB-XL - відкритий набір даних електрокардіографічних сигналів, який використовується для навчання та тестування моделей аналізу ЕКГ.

NORM - клас нормального стану кардіосигналу.

MI - Myocardial Infarction, інфаркт міокарда.

STTC - ST/T Change, зміни сегмента ST/T.

CD - Conduction Disturbance, порушення провідності.

HYP - Hypertrophy, гіпертрофія.

БД - база даних.

СУБД - система управління базами даних.

SQLite - вбудована реляційна система управління базами даних, яка зберігає дані в одному локальному файлі.

GUI - Graphical User Interface, графічний інтерфейс користувача.

Tkinter - бібліотека Python для створення графічного інтерфейсу користувача.

TensorFlow/Keras - інструментарій для створення, навчання, збереження та використання моделей нейронних мереж.

NumPy - бібліотека Python для роботи з числовими масивами даних.

Matplotlib - бібліотека Python для побудови графіків і візуалізації даних.

CSV - Comma-Separated Values, текстовий формат збереження табличних даних.

.keras - формат файлу збереженої моделі Keras.

.npy - формат файлу NumPy для збереження масивів даних.

ecg_analysis.db - файл локальної бази даних SQLite, у якому зберігаються результати аналізу кардіосигналів.

analysis_results - таблиця бази даних, у якій зберігаються результати проведених аналізів.

UML - Unified Modeling Language, уніфікована мова моделювання програмних систем.

ER-діаграма - діаграма «сутність-зв'язок», що використовується для моделювання структури бази даних.

CSV-звіт - файл звітно-статистичної інформації, сформований у форматі CSV.

ВСТУП

Серцево-судинні захворювання є однією з найпоширеніших причин погіршення стану здоров'я людини, тому своєчасне виявлення патологічних змін у роботі серця має важливе значення. Одним із найбільш доступних і поширених методів дослідження роботи серця є електрокардіографія. Електрокардіограма відображає електричну активність серця та дозволяє оцінити його функціональний стан, виявити ознаки порушення провідності, інфаркту міокарда, гіпертрофії та інших патологічних змін.

Аналіз ЕКГ-сигналів потребує уважності, досвіду та спеціальних знань. При великій кількості даних ручна обробка кардіосигналів може займати значний час. Крім того, електрокардіографічний сигнал має складну часову структуру, тому його аналіз потребує врахування як локальних особливостей форми сигналу, так і загальної послідовності змін у часі. У зв'язку з цим актуальним є використання методів машинного навчання та нейронних мереж для автоматизованого аналізу кардіосигналів.

Сучасні методи глибинного навчання дають змогу автоматично виділяти інформативні ознаки з медичних сигналів без необхідності ручного опису кожного параметра. Зокрема, згорткові нейронні мережі ефективно працюють із сигналами, оскільки можуть виявляти характерні локальні фрагменти. Для врахування часових залежностей у послідовностях доцільно використовувати рекурентні елементи, зокрема LSTM-шари. Поєднання CNN та LSTM дозволяє створити модель, здатну аналізувати кардіосигнал як послідовність значень і визначати його належність до нормального або патологічного стану.

Актуальність теми бакалаврської кваліфікаційної роботи полягає у необхідності розроблення програмного забезпечення, яке може автоматизувати процес попереднього аналізу ЕКГ-сигналів. Така система може бути корисною для навчальних, демонстраційних і дослідницьких цілей, оскільки дозволяє показати повний цикл роботи з кардіосигналом: від завантаження даних і побудови графіка до класифікації, збереження результату в базу даних та формування звітно-статистичної інформації.

Метою роботи є розроблення програмного забезпечення для аналізу кардіосигналів на основі згорткових нейронних мереж. Система повинна забезпечувати завантаження навченої моделі, вибір тестового ЕКГ-запису, побудову графіка кардіосигналу, класифікацію сигналу, визначення норми або патології, виведення ймовірностей, збереження результатів у базу даних SQLite та формування звітно-статистичної інформації.

У процесі розробки використано методи системного аналізу, UML-моделювання, проєктування баз даних, алгоритмізації, функціонального тестування та програмної реалізації інформаційної системи. Для створення програмного забезпечення застосовано мову програмування Python, бібліотеки TensorFlow/Keras, NumPy, Matplotlib, Tkinter, SQLite3 та CSV. Для збереження результатів використано локальну базу даних SQLite, а для впровадження системи передбачено інсталяційний пакет.

Практичним результатом роботи є настільний програмний застосунок для аналізу кардіосигналів. Розроблена система дозволяє завантажувати модель нейронної мережі, вибирати ЕКГ-запис із тестового набору, переглядати графік сигналу, отримувати результат класифікації, бачити ймовірність норми або патології, зберігати результати в базу даних і формувати звітно-статистичну інформацію.

Апробація програмного застосунку здійснювалася шляхом тестування основних функцій системи: запуску програми, завантаження моделі, вибору тестового ЕКГ-запису, побудови графіка, виконання класифікації, збереження результату в базу даних, перегляду збережених записів, формування звітів та експорту звітно-статистичної інформації у CSV-файл.

Структура бакалаврської кваліфікаційної роботи зі вступу, чотирьох розділів, висновків, списку використаних джерел і додатків. У першому розділі виконано аналіз взаємодії користувача з системою, розглянуто предметну область та побудовано UML-моделі. У другому розділі описано проєктування і розробку інформаційного забезпечення системи. У третьому розділі розглянуто організаційну структуру програмного забезпечення, вибір інструментарію та алгоритмізацію програмних модулів. У четвертому розділі наведено тестування

системи, вимоги до апаратного та програмного забезпечення і склад інсталяційного пакету.

1 АНАЛІЗ ВЗАЄМОДІЇ КОРИСТУВАЧА З СИСТЕМОЮ

1.1 Постановка завдання

Метою бакалаврської кваліфікаційної роботи є розроблення програмного забезпечення для аналізу кардіосигналів із використанням згорткових нейронних мереж. Розроблювана система призначена для автоматизованої обробки електрокардіографічних сигналів, визначення загального стану кардіосигналу та формування результату аналізу у зручному для користувача вигляді.

Актуальність поставленого завдання зумовлена тим, що електрокардіографія є одним із найпоширеніших методів дослідження роботи серця. ЕКГ-сигнали містять важливу діагностичну інформацію, яка дозволяє виявляти ознаки нормального функціонування серця або наявності патологічних змін. Однак ручний аналіз кардіосигналів потребує спеціальних знань, часу та уважності. Тому використання методів штучного інтелекту, зокрема згорткових нейронних мереж, є доцільним для автоматизації попереднього аналізу ЕКГ.

Розроблена система має визначати, чи відповідає кардіосигнал нормі, чи свідчить про наявність патології. У разі виявлення патологічного стану програмне забезпечення повинно визначати тип патології. У межах роботи розглядаються такі діагностичні класи: NORM — норма, MI — інфаркт міокарда, STTC — зміни сегмента ST/T, CD — порушення провідності, HYP — гіпертрофія. Оскільки один ЕКГ-запис може мати кілька діагностичних ознак одночасно, система повинна підтримувати багатоміткову класифікацію.

Для реалізації поставленої мети необхідно виконати такі завдання:

- провести аналіз предметної області та визначити особливості обробки кардіосигналів;
- розглянути існуючі програмні рішення та підходи до автоматизованого аналізу ЕКГ;
- визначити функціональні та нефункціональні вимоги до програмного забезпечення;

- підготувати набір даних кардіосигналів для навчання, валідації та тестування моделі;
- виконати попередню обробку ЕКГ-сигналів, включаючи нормалізацію та формування вхідних масивів;
- розробити та навчити модель нейронної мережі для класифікації кардіосигналів;
- реалізувати графічний інтерфейс користувача для взаємодії з програмою;
- забезпечити відображення графіка ЕКГ-сигналу;
- реалізувати виведення результату аналізу із зазначенням норми або патології, типу патології та ймовірностей;
- реалізувати збереження результатів аналізу в локальну базу даних SQLite;
- забезпечити перегляд збережених результатів у базі даних;
- реалізувати формування звітно-статистичної інформації за результатами аналізів;
- підготувати інсталяційний пакет для впровадження програмної системи.

Об'єктом дослідження є процес автоматизованого аналізу електрокардіографічних сигналів.

Предметом дослідження є методи та програмні засоби класифікації кардіосигналів на основі згорткових нейронних мереж.

Вхідними даними для системи є підготовлені записи ЕКГ-сигналів, збережені у вигляді масивів даних, а також навчена модель нейронної мережі. Додатково користувач вводить дані пацієнта, зокрема ПІБ та вік, які використовуються для збереження результату аналізу в базі даних.

Вихідними даними системи є результат класифікації кардіосигналу, який містить загальний висновок про стан пацієнта, ймовірність норми, ймовірність патології, визначений тип патології, а також графічне відображення ЕКГ-сигналу. Крім того, система формує записи в базі даних та звітно-статистичну інформацію.

Розроблювана програмна система належить до класу інтелектуальних інформаційних систем, оскільки використовує модель машинного навчання для аналізу даних і формування результатів. За масштабом система є локальним настільним застосунком, призначеним для роботи на одному персональному комп'ютері. За характером обробки інформації система є інформаційно-аналітичною, оскільки виконує обробку вхідних сигналів, класифікацію, збереження результатів і формування статистичних відомостей.

Основні функціональні вимоги до системи:

- система повинна забезпечувати завантаження навченої моделі нейронної мережі;
- система повинна забезпечувати завантаження тестових ЕКГ-записів;
- система повинна відображати графік вибраного кардіосигналу;
- система повинна виконувати класифікацію ЕКГ-сигналу;
- система повинна визначати, чи є результат нормою або патологією;
- система повинна показувати ймовірності для діагностичних класів;
- система повинна дозволяти вводити дані пацієнта;
- система повинна зберігати результат аналізу в базу даних;
- система повинна забезпечувати перегляд збережених записів;
- система повинна формувати звітно-статистичну інформацію;
- система повинна підтримувати експорт звітів у формат CSV.

Нефункціональні вимоги до системи:

- інтерфейс користувача має бути простим і зрозумілим;
- результати аналізу повинні виводитися у наочному вигляді;
- програма повинна працювати на операційній системі Windows;
- база даних повинна зберігатися локально та не потребувати окремого сервера;
- система повинна забезпечувати коректне збереження даних після виконання аналізу;
- програмне забезпечення повинно мати можливість запуску як із середовища Python, так і у вигляді встановленого настільного застосунку.

Результатом виконання роботи є програмне забезпечення для аналізу кардіосигналів на основі згорткових нейронних мереж. Система забезпечує автоматичне визначення стану кардіосигналу, класифікацію патологій, візуалізацію ЕКГ, збереження результатів у базу даних SQLite та формування звітно-статистичної інформації. Отримане програмне рішення може бути використане як допоміжний інструмент для попереднього аналізу кардіосигналів та як основа для подальшого розвитку систем підтримки медичних рішень.

1.2 Огляд інформаційних джерел та існуючих рішень

Автоматизований аналіз електрокардіографічних сигналів є актуальним напрямом розвитку медичних інформаційних систем. ЕКГ є одним із базових методів діагностики функціонального стану серця, тому створення програмних засобів для автоматичного виявлення патологічних ознак у кардіосигналах має важливе практичне значення. Сучасні підходи до розв'язання цієї задачі базуються на використанні цифрової обробки сигналів, методів машинного навчання та глибоких нейронних мереж.

Під час аналізу предметної області було розглянуто наукові джерела, відкриті набори даних та програмні інструменти, які застосовуються для обробки електрокардіографічних сигналів. Одним із найважливіших джерел для досліджень у цій галузі є набір даних РТВ-XL, який містить 21837 записів 12-відведених ЕКГ тривалістю 10 секунд від 18885 пацієнтів[1]. Опис набору даних РТВ-XL і файли для його використання також доступні у відкритому репозиторії PhysioNet[2]. Важливою особливістю цього набору є наявність багатоміткових діагностичних анотацій, що дозволяє використовувати його для задач класифікації кількох патологічних станів одночасно. Саме ця властивість робить РТВ-XL доцільним для використання в межах даної кваліфікаційної роботи, оскільки розроблювана система повинна визначати не лише норму або патологію, а й конкретні класи патологій.

Також у галузі автоматизованого аналізу ЕКГ широко відомим є набір даних MIT-BIH Arrhythmia Database[3]. Він містить 48 півгодинних

двоканальних записів амбулаторних ЕКГ від 47 пацієнтів і часто використовується для оцінювання алгоритмів виявлення аритмій. Перевагою цього набору є його історична значущість і широке застосування в дослідженнях, однак для даного проєкту він є менш придатним, оскільки містить двоканальні записи та орієнтований переважно на аритмії, тоді як у роботі розглядаються 12-відведені ЕКГ та діагностичні супер-класи PTB-XL.

Для роботи з фізіологічними сигналами часто використовується формат WFDB який підтримується відповідними програмними засобами для Python [4]. Він застосовується для зберігання сигналів та анотацій, а також підтримується спеціалізованими програмними засобами. Для Python існує пакет wfdb, який дозволяє зчитувати, записувати, обробляти та візуалізувати фізіологічні сигнали й анотації. У межах розроблюваного програмного забезпечення використання цього пакета є доцільним на етапі підготовки набору даних, оскільки записи PTB-XL зберігаються у форматі, сумісному з WFDB.

Окрему групу рішень становлять готові медичні системи та комерційні програмні продукти для аналізу ЕКГ. Такі системи можуть входити до складу електрокардіографів, медичних інформаційних систем або хмарних платформ для обробки даних пацієнтів. Їх перевагами є висока інтеграція з медичним обладнанням, підтримка клінічних робочих процесів і наявність сертифікації. Водночас вони часто мають закриту архітектуру, обмежені можливості для модифікації алгоритмів, високу вартість і складність використання в навчальних або дослідницьких цілях.

У наукових дослідженнях для аналізу кардіосигналів широко застосовуються методи машинного навчання та глибокого навчання. Традиційні підходи часто передбачають попереднє виділення ознак, таких як інтервали, амплітуди та морфологічні характеристики зубців ЕКГ. Однак такі методи значною мірою залежать від якості ручного виділення ознак. На відміну від них, згорткові нейронні мережі можуть автоматично виділяти локальні ознаки сигналу без необхідності ручного опису кожного параметра. Це робить їх ефективними для задач класифікації часових медичних сигналів.

Для реалізації нейронної мережі в роботі доцільно використовувати TensorFlow/Keras, оскільки цей інструментарій надає готові засоби для створення, навчання, збереження та завантаження моделей глибокого навчання[5]. Keras є високорівневим API для TensorFlow, що спрощує опис архітектури нейронної мережі та дозволяє швидко створювати моделі з використанням згорткових, рекурентних і повнозв'язних шарів. Це відповідає архітектурі розроблюваної системи, у якій застосовується модель CNN-LSTM для класифікації кардіосигналів.

Для порівняння існуючих рішень і підходів доцільно виділити такі варіанти:

Таблиця 1.1 – Порівняння існуючих підходів до аналізу кардіосигналів

Підхід / рішення	Переваги	Недоліки
Ручний аналіз ЕКГ лікарем	Висока клінічна обґрунтованість, урахування стану пацієнта	Потребує часу та спеціальної кваліфікації
Класичні алгоритми цифрової обробки сигналів	Зрозуміла логіка роботи, невисокі вимоги до ресурсів	Потребують ручного виділення ознак
Комерційні системи аналізу ЕКГ	Готовий інтерфейс, інтеграція з медичним обладнанням	Висока вартість, закритість алгоритмів
Системи на основі глибокого навчання	Автоматичне виділення ознак, робота з великими наборами даних	Потребують навчальних даних і ресурсів
Розроблювана система	Поєднує CNN-LSTM, інтерфейс, SQLite та звіти	Орієнтована на навчальне й дослідницьке використання

На основі проведеного огляду було прийнято рішення розробити власне програмне забезпечення для аналізу кардіосигналів, яке поєднує модель глибокого навчання, графічний інтерфейс користувача, локальну базу даних і модуль звітно-статистичної інформації. На відміну від комерційних рішень,

розроблювана система є відкритою для модифікації, орієнтована на навчальне та дослідницьке використання, а також дозволяє демонструвати всі основні етапи роботи: від завантаження сигналу до отримання результату класифікації та збереження його в базу даних.

Таким чином, аналіз інформаційних джерел та існуючих рішень показав, що використання набору даних PTB-XL, Python-інструментів для обробки сигналів і моделі CNN-LSTM є обґрунтованим для реалізації програмної системи автоматизованого аналізу кардіосигналів. Обрана архітектура дозволяє поєднати можливості глибокого навчання з практичним інтерфейсом користувача, базою даних результатів і засобами формування звітів.

1.3 Моделювання предметної області

Моделювання предметної області є важливим етапом розробки програмного забезпечення, оскільки дозволяє описати основні процеси, функції системи та взаємодію користувача з програмним додатком. Для системи аналізу кардіосигналів моделювання дає змогу визначити, які дії виконує користувач, які програмні модулі беруть участь в обробці даних, а також у якій послідовності відбувається аналіз ЕКГ-сигналу.

У межах роботи було побудовано комплекс UML-діаграм, які описують різні аспекти функціонування системи. До таких моделей належать діаграма прецедентів, діаграма послідовності та діаграма діяльності. Вони дозволяють розглянути програмну систему з різних точок зору: функціональної, поведінкової та процесної.

Діаграма прецедентів відображає функціональні можливості програмної системи з точки зору користувача. Основним актором є користувач, який взаємодіє із системою через графічний інтерфейс. У межах системи користувач може завантажити модель нейронної мережі, вибрати тестовий запис ЕКГ, ввести дані пацієнта, візуалізувати кардіосигнал, отримати результат класифікації, переглянути ймовірність норми та патології, визначити тип патології, зберегти результат у базу даних і переглянути збережені записи.

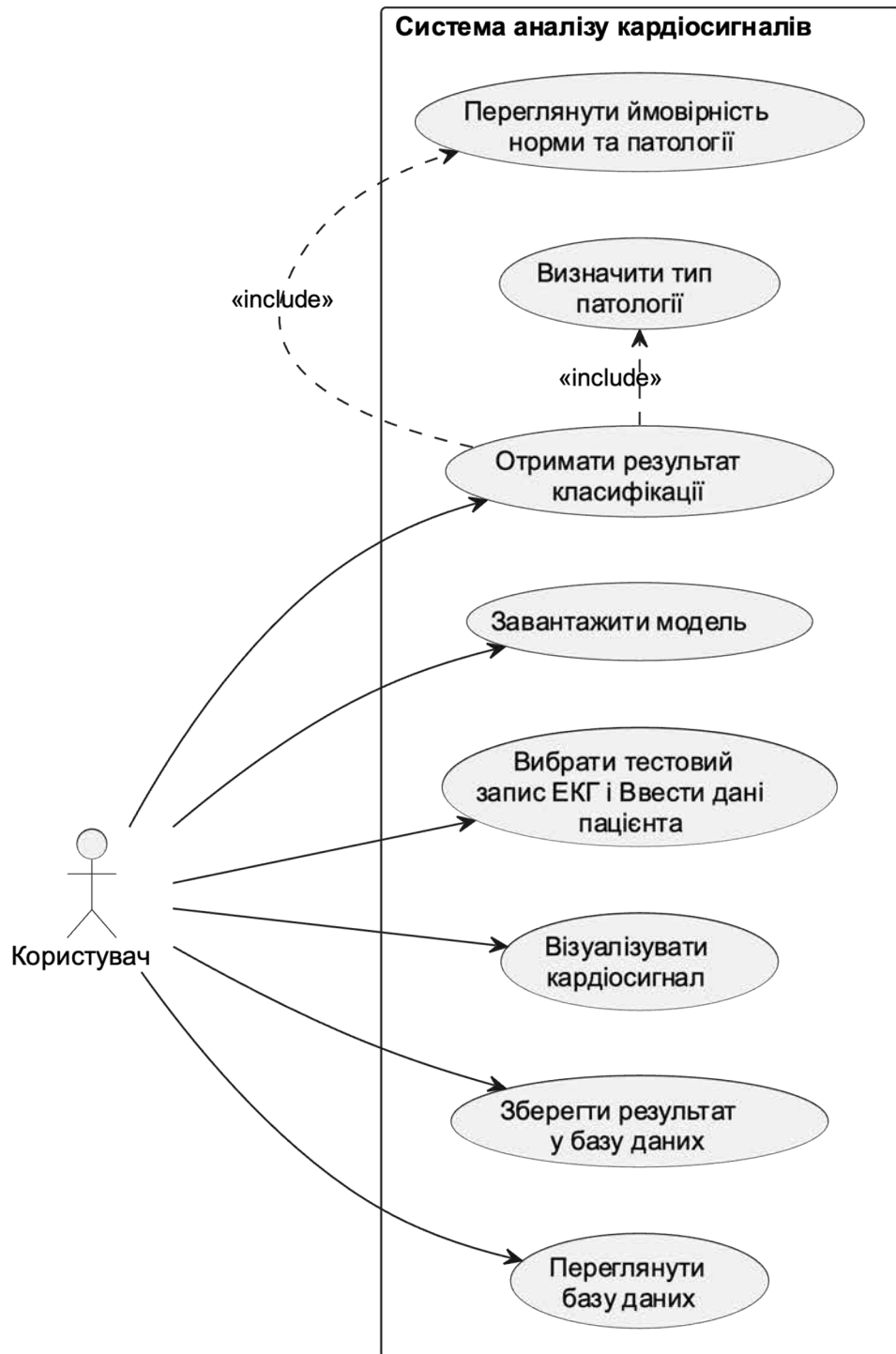


Рисунок 1.1 – Діаграма прецедентів системи аналізу кардіосигналів

На діаграмі прецедентів показано, що центральною частиною системи є процес аналізу кардіосигналу. Для його виконання користувач спочатку завантажує модель, після чого вибирає тестовий запис ЕКГ і вводить дані пацієнта. Далі система виконує візуалізацію сигналу та формує результат класифікації. Результат включає загальний висновок щодо стану пацієнта,

ймовірність норми або патології, а також тип патології у разі її виявлення. Окремо передбачено можливість збереження результату в базу даних і подальший перегляд записів.

Діаграма послідовності деталізує порядок взаємодії між користувачем, графічним інтерфейсом, моделлю CNN-LSTM та базою даних SQLite. Вона показує, як саме відбувається процес обробки ЕКГ-запису від моменту завантаження моделі до збереження результату аналізу.

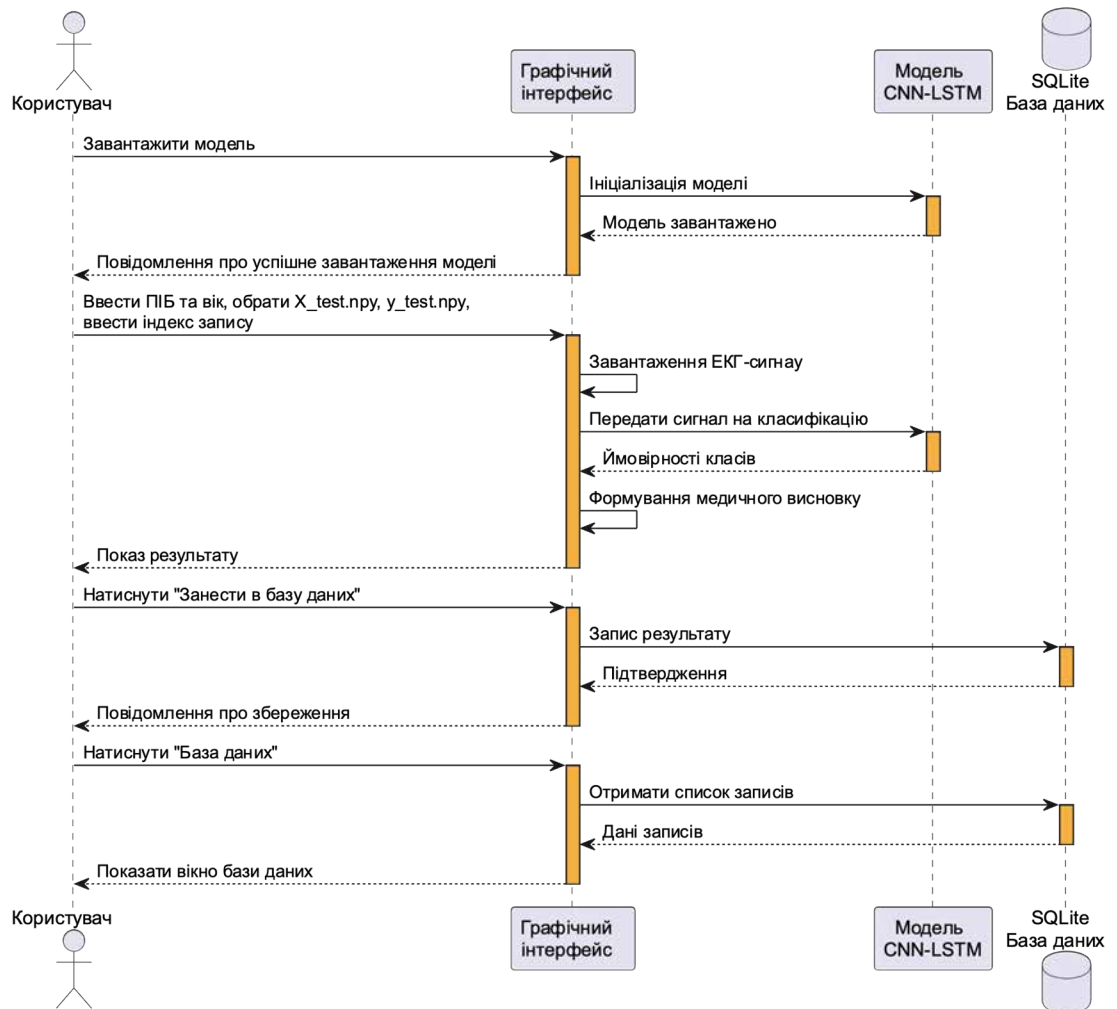


Рисунок 1.2 – Діаграма послідовності процесу аналізу кардіосигналу

Спочатку користувач завантажує модель через графічний інтерфейс. Після цього інтерфейс виконує ініціалізацію моделі CNN-LSTM і отримує повідомлення про успішне завантаження. Далі користувач вводить ПІБ та вік пацієнта, обирає файли X_test.npy і y_test.npy, а також вказує індекс тестового запису. Графічний інтерфейс завантажує відповідний ЕКГ-сигнал і передає його

на класифікацію до моделі CNN-LSTM. Модель повертає ймовірності класів, після чого інтерфейс формує медичний висновок і показує результат користувачу.

Якщо користувач натискає кнопку «Занести в базу даних», графічний інтерфейс передає результат до бази даних SQLite. Після запису даних база повертає підтвердження, а користувач отримує повідомлення про успішне збереження. Також на діаграмі показано сценарій перегляду бази даних: після натискання кнопки «База даних» система виконує запит до SQLite, отримує список записів і відкриває вікно перегляду збережених результатів.

Діаграма діяльності відображає загальний алгоритм роботи програмної системи. Вона показує послідовність дій, які виконуються під час аналізу кардіосигналу, а також умови переходу між окремими етапами.

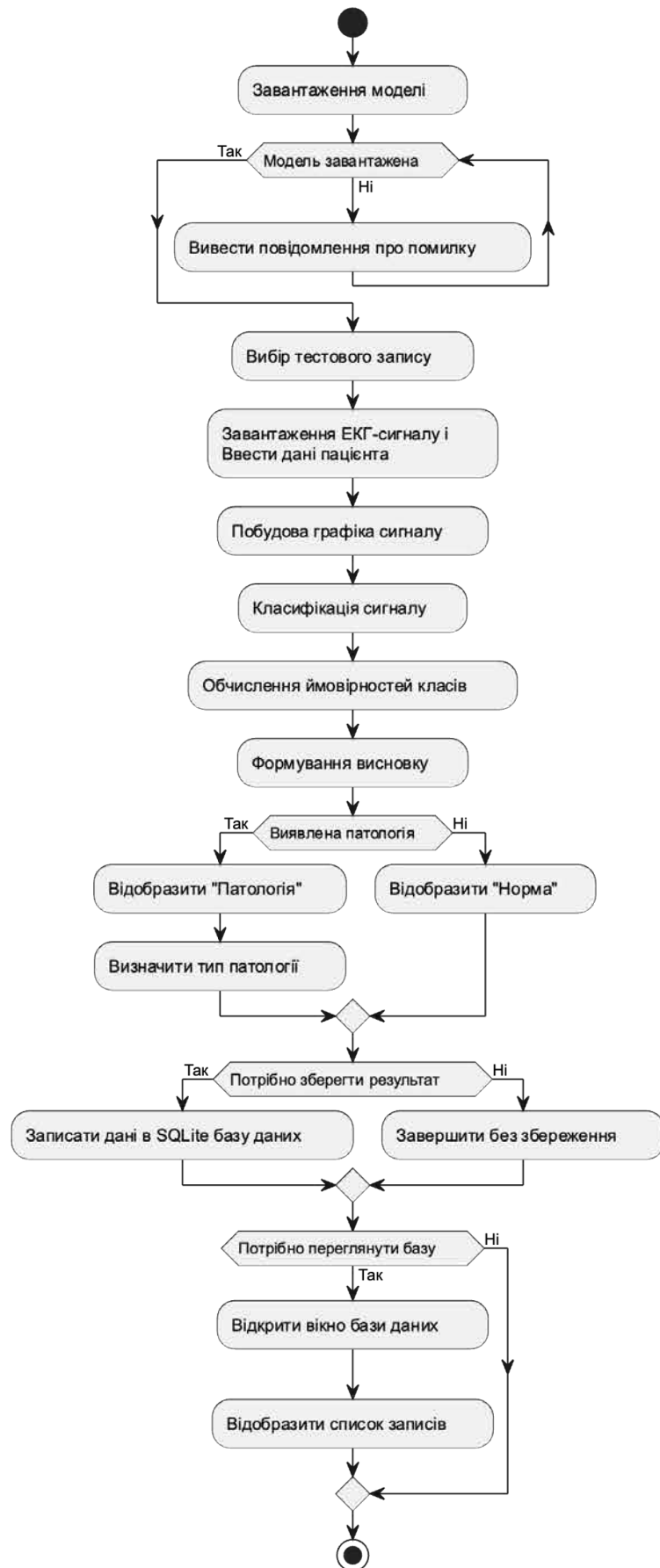


Рисунок 1.3 – Діаграма діяльності системи аналізу кардіосигналів

На початку роботи виконується завантаження моделі. Якщо модель не завантажена, система виводить повідомлення про помилку. Якщо модель завантажена успішно, користувач переходить до вибору тестового запису, завантаження ЕКГ-сигналу та введення даних пацієнта. Після цього програма будує графік сигналу, виконує класифікацію та обчислює ймовірності класів.

Наступним етапом є формування висновку. Якщо система виявляє патологію, у результаті відображається повідомлення «Патологія», а також визначається тип патології. Якщо патологію не виявлено, користувачу показується висновок «Норма». Після цього користувач може прийняти рішення щодо збереження результату. Якщо збереження потрібне, дані записуються в базу даних SQLite. Якщо збереження не потрібне, робота завершується без запису результату.

Окремо на діаграмі показано можливість перегляду бази даних. Якщо користувач хоче переглянути збережені записи, система відкриває вікно бази даних і відображає список результатів. Якщо перегляд бази не потрібен, виконання алгоритму завершується.

Таким чином, побудовані UML-діаграми описують основні аспекти роботи програмної системи: функції, доступні користувачу, послідовність взаємодії між модулями та загальний алгоритм обробки кардіосигналу. Це дає змогу перейти до подальшого проектування інформаційного забезпечення системи.

2 ПРОЄКТУВАННЯ І РОЗРОБКА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

2.1 Логічна модель даних

Логічна модель даних визначає структуру інформації, яка використовується та зберігається в програмній системі. Для програмного забезпечення аналізу кардіосигналів база даних необхідна для збереження інформації про пацієнтів, ЕКГ-записи, результати аналізу, виявлені патології та довідкові дані про типи патологій. Побудована логічна модель даних наведена на рис. 2.1.

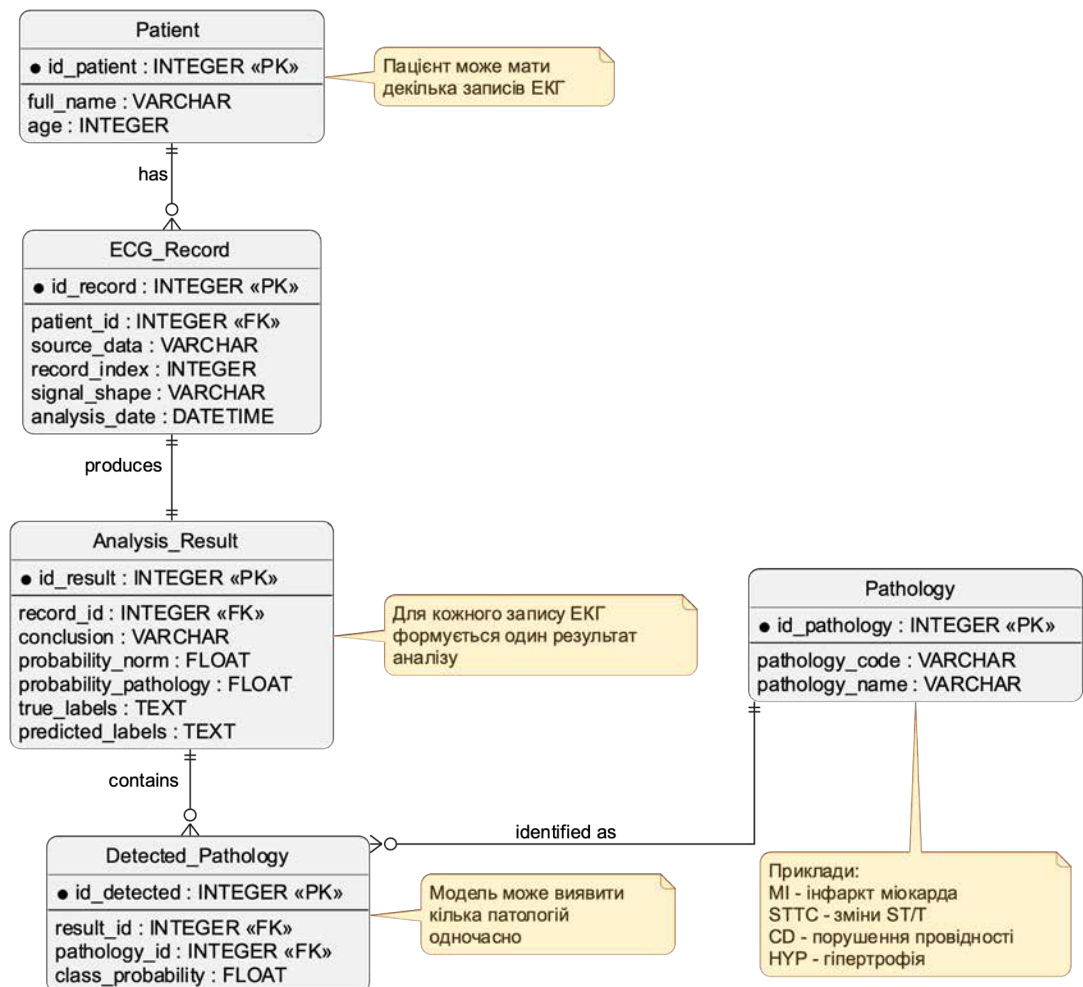


Рисунок 2.1 – Логічна модель даних програмної системи аналізу кардіосигналів

На діаграмі виділено п'ять основних сутностей: Patient, ECG_Record, Analysis_Result, Pathology та Detected_Pathology. Кожна з цих сутностей відповідає окремому інформаційному об'єкту системи та використовується для збереження даних, необхідних для аналізу кардіосигналів і подальшого перегляду результатів.

Сутність Patient призначена для збереження інформації про пацієнта. Вона містить такі атрибути: id_patient, full_name, age. Атрибут id_patient є первинним ключем і використовується для унікальної ідентифікації пацієнта в базі даних. Атрибут full_name зберігає ПІБ пацієнта, а age — його вік. Ці дані вводяться користувачем у графічному інтерфейсі перед виконанням аналізу або перед збереженням результату.

Сутність ECG_Record використовується для збереження інформації про електрокардіографічний запис, який аналізується системою. Вона містить атрибути id_record, patient_id, source_data, record_index, signal_shape, analysis_date. Атрибут id_record є первинним ключем. Атрибут patient_id є зовнішнім ключем і встановлює зв'язок із пацієнтом. Поле source_data зберігає джерело даних, наприклад файл або масив, з якого було завантажено сигнал. Поле record_index визначає індекс вибраного запису, signal_shape містить інформацію про розмірність сигналу, а analysis_date зберігає дату й час виконання аналізу.

Сутність Analysis_Result описує результат класифікації ЕКГ-сигналу. Вона містить атрибути id_result, record_id, conclusion, probability_norm, probability_pathology, true_labels, predicted_labels. Атрибут id_result є первинним ключем. Атрибут record_id є зовнішнім ключем і пов'язує результат із відповідним ЕКГ-записом. Поле conclusion зберігає загальний висновок системи, наприклад «Норма» або «Патологія». Атрибути probability_norm і probability_pathology містять імовірності нормального та патологічного стану. Поля true_labels і predicted_labels використовуються для збереження істинних та передбачених класів, що дає змогу аналізувати якість роботи моделі.

Сутність Pathology є довідником патологічних класів, які може визначати система. Вона містить атрибути id_pathology, pathology_code, pathology_name.

Атрибут `id_pathology` є первинним ключем. Поле `pathology_code` зберігає коротке позначення класу, наприклад MI, STTC, CD, НУР, а поле `pathology_name` містить повну назву патології. Наявність окремої сутності для патологій дозволяє зберігати довідкову інформацію структуровано та за потреби розширювати перелік діагностичних класів.

Сутність `Detected_Pathology` є проміжною сутністю, яка зв'язує результат аналізу з однією або кількома патологіями. Вона містить атрибути `id_detected`, `result_id`, `pathology_id`, `class_probability`. Атрибут `id_detected` є первинним ключем. Атрибути `result_id` і `pathology_id` є зовнішніми ключами, які пов'язують запис із результатом аналізу та конкретною патологією. Поле `class_probability` зберігає ймовірність належності сигналу до відповідного патологічного класу.

Для узагальнення сутностей логічної моделі даних наведено таблицю 2.1.

Таблиця 2.1 – Опис сутностей логічної моделі даних

Сутність	Основні атрибути	Призначення
Patient	<code>id_patient</code> , <code>full_name</code> , <code>age</code>	Збереження даних пацієнта
ECG_Record	<code>id_record</code> , <code>patient_id</code> , <code>source_data</code> , <code>record_index</code> , <code>signal_shape</code> , <code>analysis_date</code>	Збереження інформації про ЕКГ-запис
Analysis_Result	<code>id_result</code> , <code>record_id</code> , <code>conclusion</code> , <code>probability_norm</code> , <code>probability_pathology</code> , <code>true_labels</code> , <code>predicted_labels</code>	Збереження результату класифікації
Pathology	<code>id_pathology</code> , <code>pathology_code</code> , <code>pathology_name</code>	Довідник патологічних класів
Detected_Pathology	<code>id_detected</code> , <code>result_id</code> , <code>pathology_id</code> , <code>class_probability</code>	Збереження інформації про виявлені патології

Між сутностями моделі встановлено такі зв'язки. Сутність Patient пов'язана із сутністю ECG_Record зв'язком типу «один до багатьох». Це означає, що один пацієнт може мати декілька ЕКГ-записів, але кожен ЕКГ-запис належить лише одному пацієнту.

Сутність ECG_Record пов'язана із сутністю Analysis_Result зв'язком типу «один до одного». Для кожного ЕКГ-запису формується один результат аналізу, який містить загальний висновок і ймовірності класифікації.

Сутність Analysis_Result пов'язана із сутністю Detected_Pathology зв'язком типу «один до багатьох». Це пов'язано з тим, що модель може визначити одну або кілька патологій одночасно. У такому випадку для одного результату аналізу може бути створено кілька записів у таблиці виявлених патологій.

Сутність Pathology також пов'язана із сутністю Detected_Pathology зв'язком типу «один до багатьох». Одна патологія може бути виявлена в багатьох результатах аналізу, але кожен запис у Detected_Pathology посиляється на одну конкретну патологію.

Для кращого розуміння зв'язків між сутностями можна подати їх у вигляді таблиці 2.2.

Таблиця 2.2 – Зв'язки між сутностями логічної моделі даних

Зв'язок між сутностями	Тип зв'язку	Опис зв'язку
Patient — ECG_Record	1:M	Один пацієнт може мати декілька ЕКГ-записів, але кожен ЕКГ-запис належить одному пацієнту.
ECG_Record— Analysis_Result	1:1	Для одного ЕКГ-запису формується один результат аналізу, який містить висновок і ймовірності класифікації.
Analysis_Result—	1:M	Один результат аналізу може

Detected_Pathology		містити одну або кілька виявлених патологій.
Pathology— Detected_Pathology	1:M	Одна патологія може зустрічатися в багатьох результатах аналізу, але кожен запис у Detected_Pathology посилається на одну конкретну патологію.

Як видно з табл. 2.2, логічна модель даних передбачає зв'язки між пацієнтом, ЕКГ-записом, результатом аналізу та патологічними класами. Зв'язок між сутностями Analysis_Result і Pathology реалізується через проміжну сутність Detected_Pathology, оскільки один результат аналізу може містити декілька патологічних ознак одночасно. Такий підхід відповідає особливості задачі багатоміткової класифікації кардіосигналів.

Таким чином, побудована логічна модель даних відображає інформаційну структуру програмної системи аналізу кардіосигналів. Вона дозволяє зберігати дані пацієнтів, інформацію про ЕКГ-записи, результати роботи моделі CNN-LSTM, ймовірності класифікації та відомості про виявлені патології. Така структура є достатньою для реалізації функцій збереження результатів, перегляду бази даних і формування звітної-статистичної інформації.

2.2 Вибір системи управління інформаційною базою

Для реалізації інформаційної бази програмної системи аналізу кардіосигналів необхідно обрати систему управління базою даних, яка забезпечить надійне збереження результатів аналізу, даних пацієнтів, інформації про ЕКГ-записи та виявлені патології. Оскільки розроблювана система є настільним застосунком і працює локально на персональному комп'ютері користувача, для її реалізації було обрано SQLite.

SQLite є вбудованою реляційною системою управління базами даних, яка не потребує окремого серверного програмного забезпечення[6]. Уся база даних зберігається в одному файлі, що значно спрощує використання, перенесення та впровадження системи. Такий підхід є зручним для програмного забезпечення аналізу кардіосигналів, оскільки система не потребує багатокористувацької серверної архітектури та може повністю працювати на одному комп'ютері.

Вибір SQLite обґрунтований тим, що база даних у розроблюваній системі використовується для збереження відносно невеликого обсягу інформації: даних пацієнтів, результатів аналізу, ймовірностей класифікації, типів патологій та дати проведення аналізу. Для таких задач SQLite є достатньо продуктивною та простою у використанні. Вона дозволяє виконувати основні операції з даними: створення таблиць, додавання записів, вибірку, перегляд, оновлення та формування статистичної інформації.

Важливою перевагою SQLite є її інтеграція з мовою програмування Python. Для роботи з SQLite у Python використовується стандартний модуль `sqlite3`, який не потребує встановлення додаткових складних компонентів[7]. Це дозволяє реалізувати роботу з базою даних без використання окремого сервера, додаткових драйверів або складного налаштування середовища.

Для розроблюваної системи це є суттєвою перевагою, оскільки програмний застосунок повинен бути простим у встановленні та використанні. Користувачу достатньо запустити програму, після чого база даних створюється або відкривається автоматично. У разі відсутності файлу бази даних програма може створити його самостійно під час першого запуску.

Для обґрунтування вибору системи управління інформаційною базою було виконано порівняння кількох можливих СУБД. Результати порівняння наведено в табл. 2.3.

Таблиця 2.3 – Порівняння систем управління базами даних

СУБД	Переваги	Недоліки	Доцільність використання в
------	----------	----------	----------------------------

			проекті
SQLite	Не потребує сервера, проста інтеграція з Python, база зберігається в одному файлі	Не призначена для великих багатокористувацьких систем	Найбільш доцільна для локального настільного застосунку
MySQL	Підходить для веб-систем і багатокористувацької роботи	Потребує встановлення та налаштування сервера	Надлишкова для даного проєкту
PostgreSQL	Потужна СУБД, підтримує складні запити та масштабування	Потребує серверного середовища, складніша в налаштуванні	Не є необхідною для локальної системи
MS Access	Має графічне середовище для роботи з БД	Менш зручна для інтеграції з Python і машинним навчанням	Може використовуватись, але не є оптимальною
MS SQL Server	Потужна корпоративна СУБД	Вимагає значних ресурсів і складнішого адміністрування	Недоцільна для локального навчального застосунку

На основі порівняння можна зробити висновок, що SQLite найкраще відповідає вимогам розроблюваної системи. На відміну від серверних СУБД, вона не потребує окремого встановлення, адміністрування користувачів, налаштування мережевого доступу або підтримки серверної інфраструктури. Це дозволяє зосередитися на основній функціональності системи — аналізі кардіосигналів, збереженні результатів і формуванні звітів.

У межах проєкту інформаційна база є локальною. Це означає, що всі дані зберігаються на комп'ютері користувача, а доступ до них здійснюється безпосередньо програмним застосунком. Така організація є достатньою для

розробленої системи, оскільки вона не передбачає одночасної роботи багатьох користувачів з однією базою даних через мережу.

Основними операціями з інформаційною базою є створення таблиць бази даних, додавання нового результату аналізу, збереження даних пацієнта, збереження інформації про виявлені патології, перегляд усіх збережених записів, вибірка даних для формування звітно-статистичної інформації та експорт звітних даних у формат CSV.

У процесі роботи системи користувач не взаємодіє з базою даних безпосередньо. Усі операції виконуються через графічний інтерфейс. Наприклад, після отримання результату аналізу користувач натискає кнопку збереження, а програма автоматично виконує SQL-запит для додавання нового запису до бази даних. Аналогічно, під час відкриття вікна бази даних програма виконує запит на вибірку записів і відображає їх у зручному табличному вигляді.

SQLite також використовується для формування звітно-статистичної інформації. На основі збережених записів система може визначити загальну кількість проведених аналізів, кількість результатів із висновком «Норма», кількість результатів із висновком «Патологія», розподіл патологій за типами та кількість аналізів для окремих пацієнтів. Це дозволяє використовувати базу даних не лише як сховище результатів, а й як джерело для аналітичної обробки інформації.

У розробленій системі база даних представлена файлом `ecg_analysis.db`. Цей файл містить таблиці, необхідні для збереження пацієнтів, ЕКГ-записів, результатів аналізу та виявлених патологій.

Таким чином, вибір SQLite як системи управління інформаційною базою є обґрунтованим для даного проєкту. Вона забезпечує простоту встановлення, локальне збереження даних, достатню швидкодію, інтеграцію з Python та можливість виконання всіх необхідних операцій з даними. Обрана СУБД повністю відповідає характеру розроблюваної системи, яка є локальним настільним застосунком для аналізу кардіосигналів.

2.3 Створення інформаційної бази

Створення інформаційної бази є наступним етапом після побудови логічної моделі даних та вибору системи управління базою даних. У межах розроблюваної програмної системи аналізу кардіосигналів інформаційна база реалізована засобами SQLite. Вона використовується для збереження результатів аналізу ЕКГ-сигналів, даних пацієнта, імовірностей класифікації, загального висновку системи, інформації про джерело сигналу та шлях до використаної моделі.

У програмній реалізації база даних зберігається у файлі `ecg_analysis.db`. Назва файлу бази даних задається в коді через константу `DB_FILENAME`, а шлях до бази формується відносно директорії основного файлу застосунку. Це дозволяє зберігати базу даних поруч із програмою та спрощує перенесення системи на інший комп'ютер.

Фізична структура бази даних відповідає практичній реалізації програмного забезпечення. У базі використовується основна таблиця `analysis_results`, у якій кожен запис відповідає одному проведеному аналізу кардіосигналу. Такий підхід є доцільним для локального настільного застосунку, оскільки система не потребує складної серверної архітектури або багатокористувацького доступу.

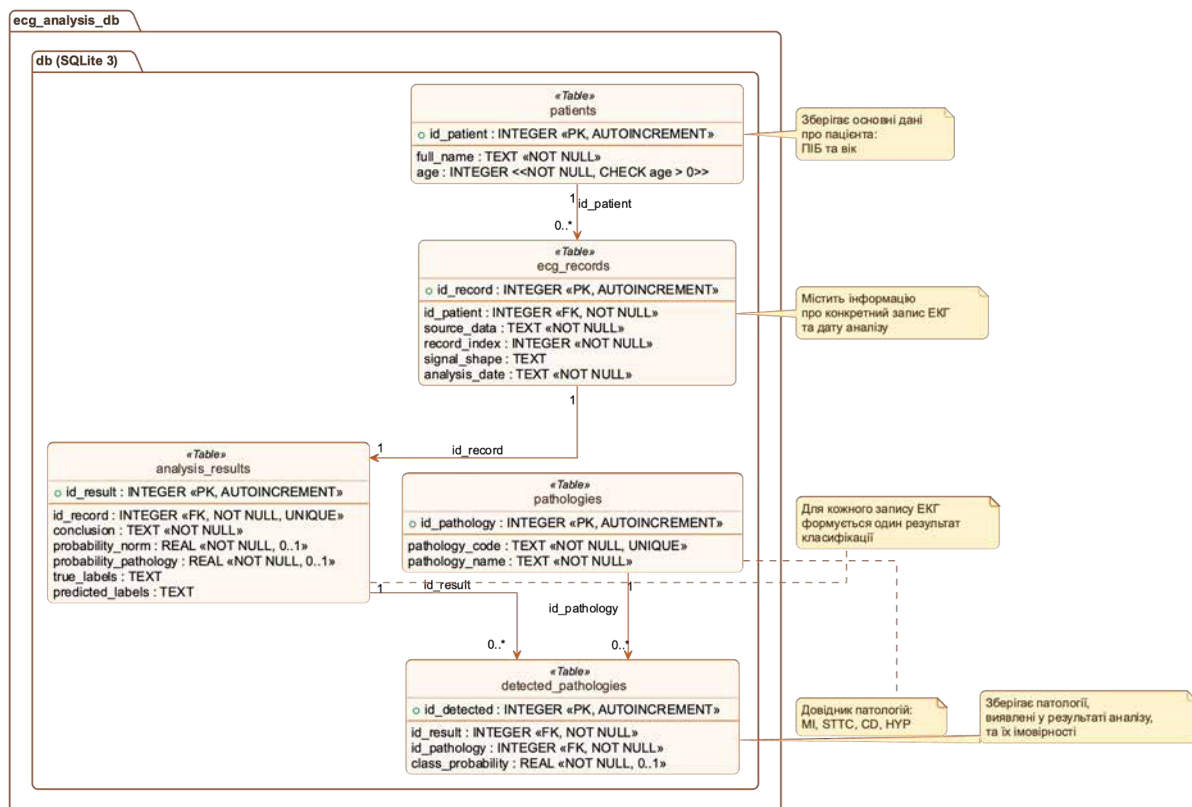


Рисунок 2.2 – Структура бази даних програмної системи на фізичному рівні

На рис. 2.2 наведено структуру бази даних на фізичному рівні. Центральним елементом є таблиця `analysis_results`, яка зберігає всі дані, необхідні для подальшого перегляду результатів, формування статистики та експорту звітної інформації.

Структуру таблиці `analysis_results` наведено в табл. 2.4.

Таблиця 2.4 – Структура таблиці `analysis_results`

Поле	Тип даних	Призначення
id	INTEGER PRIMARY KEY AUTOINCREMENT	Унікальний ідентифікатор запису
created_at	TEXT NOT NULL	Дата і час збереження результату
full_name	TEXT NOT NULL	ПІБ пацієнта
age	INTEGER NOT NULL	Вік пацієнта
result_type	TEXT NOT NULL	Тип результату: «Норма» або

		«Патологія»
pathology_name	TEXT NOT NULL	Назва або код визначеної патології
general_status	TEXT NOT NULL	Загальний висновок системи
true_labels	TEXT	Істинні мітки з тестового набору
predicted_labels	TEXT	Передбачені моделлю мітки
norm_probability	REAL NOT NULL	Імовірність нормального стану
pathology_probability	REAL NOT NULL	Імовірність патологічного стану
threshold	REAL NOT NULL	Поріг бінаризації результату
source_info	TEXT	Інформація про джерело сигналу
model_path	TEXT	Шлях до використаної моделі

Створення бази даних виконується автоматично під час запуску програми. Для цього в класі ECGAnalyzerApp використовується метод `_init_database()`. У цьому методі програма відкриває з'єднання з файлом SQLite та виконує SQL-запит `CREATE TABLE IF NOT EXISTS`. Якщо таблиця вже існує, вона не створюється повторно. Якщо файл бази даних або таблиця відсутні, вони створюються автоматично.

Фрагмент коду створення таблиці наведено на рис. 2.3.

```
def _init_database(self):
    with sqlite3.connect(self.db_path) as conn:
        conn.execute(
            """
```

```

CREATE TABLE IF NOT EXISTS analysis_results (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    created_at TEXT NOT NULL,
    full_name TEXT NOT NULL,
    age INTEGER NOT NULL,
    result_type TEXT NOT NULL,
    pathology_name TEXT NOT NULL,
    general_status TEXT NOT NULL,
    true_labels TEXT,
    predicted_labels TEXT,
    norm_probability REAL NOT NULL,
    pathology_probability REAL NOT NULL,
    threshold REAL NOT NULL,
    source_info TEXT,
    model_path TEXT
)
""
)
conn.commit()

```

Рисунок 2.3 – Створення таблиці analysis_results у базі даних SQLite

Після виконання класифікації кардіосигналу користувач може зберегти отриманий результат у базу даних. Для цього у вікні результату передбачена кнопка «Занести в базу даних». Після її натискання викликається метод

`save_current_result_to_db()`. Цей метод перевіряє наявність результату аналізу, отримує ПІБ і вік пацієнта, після чого виконує SQL-запит на додавання нового запису до таблиці `analysis_results`.

До бази даних заносяться такі дані: дата й час аналізу, ПІБ пацієнта, вік, тип результату, назва патології, загальний висновок, істинні мітки, передбачені мітки, імовірність норми, імовірність патології, поріг класифікації, інформація про сигнал і шлях до файлу моделі.

Фрагмент коду збереження результату наведено на рис. 2.4.

```
conn.execute(
    """
    INSERT INTO analysis_results (
        created_at, full_name, age, result_type, pathology_name, general_status,
        true_labels, predicted_labels, norm_probability, pathology_probability,
        threshold, source_info, model_path
    ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
    """,
    (
        datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
        full_name,
        age,
        data["result_type"],
        data["pathology_name"],
        data["conclusion"],
        data["true_labels"],
        data["predicted_labels"],
```

```

data["norm_probability_raw"],

data["pathology_probability_raw"],

float(data["threshold"]),

data["signal_info"],

self.model_path_var.get().strip(),

),

)

conn.commit()

```

Рисунок 2.4 – Збереження результату аналізу в таблицю analysis_results

Після успішного збереження результату програма виводить повідомлення про те, що запис було занесено до бази даних. Якщо в цей момент відкрите вікно бази даних або вікно звітів, програма автоматично оновлює їхній вміст. Це дозволяє користувачу одразу побачити новий запис без перезапуску програми.

Для перегляду збережених результатів у програмі реалізовано окреме вікно «База даних». У цьому вікні записи відображаються у табличному вигляді. Під час відкриття вікна програма виконує запит SELECT, який отримує записи з таблиці analysis_results і сортує їх за спаданням ідентифікатора, тобто найновіші записи відображаються першими.

Фрагмент SQL-запиту для перегляду записів наведено на рис. 2.5.

```

SELECT id, created_at, full_name, age, result_type, pathology_name,

general_status, norm_probability, pathology_probability

FROM analysis_results

ORDER BY id DESC

```

Рисунок 2.5 – Вибірка результатів аналізу для відображення у вікні бази даних

У вікні бази даних користувач може переглянути основні відомості про кожен збережений аналіз: дату, ПІБ пацієнта, вік, тип результату, патологію, загальний висновок, імовірність норми та імовірність патології. Також у програмі реалізовано перегляд детальної інформації про окремий запис. Для цього користувач може двічі натиснути на потрібний рядок у таблиці, після чого відкривається окреме вікно з повною інформацією.

База даних також використовується для формування звітно-статистичної інформації. Для цього в програмі реалізовано метод `_get_report_stats()`, який виконує SQL-запити до таблиці `analysis_results`. За допомогою цих запитів система визначає загальну кількість проведених аналізів, кількість результатів типу «Норма», кількість результатів типу «Патологія», розподіл патологій за типами та кількість аналізів по кожному пацієнту.

Фрагмент коду формування статистики наведено на рис. 2.6.

```
def _get_report_stats(self):

    with sqlite3.connect(self.db_path) as conn:

        total = conn.execute("SELECT COUNT(*) FROM
analysis_results").fetchone()[0]

        normal_count = conn.execute(

            "SELECT COUNT(*) FROM analysis_results WHERE result_type =
'Норма'"

        ).fetchone()[0]

        pathology_count = conn.execute(

            "SELECT COUNT(*) FROM analysis_results WHERE result_type =
'Патологія'"

        ).fetchone()[0]

        pathology_rows = []
```

```

for code in PATHOLOGY_CLASSES:

    count = conn.execute(

        "SELECT COUNT(*) FROM analysis_results WHERE pathology_name
LIKE ?",

        (f"%{code}%",),

    ).fetchone()[0]

    pathology_rows.append((code, CLASS_LABELS[code], count))

```

Рисунок 2.6 – Формування звітно-статистичної інформації на основі бази даних

Крім відображення статистики у вікні програми, передбачено можливість експорту звіту у формат CSV[8]. Для цього використовується метод `export_reports_csv()`. Він отримує статистичні дані з бази, відкриває файл для запису та формує CSV-файл із загальною статистикою, розподілом патологій та кількістю аналізів по пацієнтах.

Фрагмент коду експорту звіту наведено на рис. 2.7.

```

with open(save_path, "w", encoding="utf-8-sig", newline="") as f:

    writer = csv.writer(f, delimiter=';')

    writer.writerow(["Показник", "Значення"])

    writer.writerow(["Усього аналізів", stats["total"]])

    writer.writerow(["Норма", stats["normal"]])

    writer.writerow(["Патологія", stats["pathology"]])

    writer.writerow([])

    writer.writerow(["Код патології", "Назва", "Кількість"])

    for row in stats["pathology_rows"]:

```

```
writer.writerow(row)

writer.writerow([])

writer.writerow(["ПІБ", "Кількість аналізів"])

for row in stats["patient_rows"]:

    writer.writerow(row)
```

Рисунок 2.7 – Експорт звітно-статистичної інформації у CSV-файл

Таким чином, створена інформаційна база забезпечує виконання основних функцій системи: збереження результатів аналізу кардіосигналів, перегляд записів, деталізацію окремого результату та формування звітно-статистичної інформації. Використання SQLite дозволило реалізувати базу даних у вигляді одного локального файлу `ecg_analysis.db`, що є зручним для настільного застосунку та не потребує встановлення окремого сервера баз даних.

3 ПРОЄКТУВАННЯ І РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Організаційна структура програмного забезпечення

Організаційна структура програмного забезпечення визначає склад основних модулів системи та зв'язки між ними. Для програмної системи аналізу кардіосигналів структура побудована таким чином, щоб забезпечити зручну взаємодію користувача з програмою, завантаження моделі нейронної мережі, обробку ЕКГ-сигналу, виведення результатів, збереження даних у базу SQLite та формування звітно-статистичної інформації.

Розроблена система реалізована у вигляді настільного застосунку мовою Python. Основним класом програми є ECGAnalyzerApp, у якому об'єднано логіку побудови графічного інтерфейсу, завантаження моделі, роботи з тестовими даними, виконання класифікації, збереження результатів у базу даних і формування звітів. У коді застосунку використовуються бібліотеки tkinter і ttk для створення інтерфейсу, matplotlib для побудови графіка ЕКГ, tensorflow для завантаження та використання навченої моделі, numpy для роботи з масивами сигналів, sqlite3 для роботи з базою даних і csv для експорту звітів.

Організаційно програмне забезпечення можна поділити на такі основні модулі:

- модуль графічного інтерфейсу користувача;
- модуль завантаження моделі;
- модуль завантаження тестового ЕКГ-запису;
- модуль візуалізації кардіосигналу;
- модуль класифікації ЕКГ-сигналу;
- модуль формування результату аналізу;
- модуль роботи з базою даних;
- модуль перегляду збережених результатів;
- модуль формування звітно-статистичної інформації;

- модуль експорту звітів у CSV.

Графічний інтерфейс користувача є центральним модулем системи, через який користувач виконує всі основні дії. У головному вікні програми розміщено блок завантаження моделі, блок введення даних пацієнта, блок вибору тестового запису, блок керування та область візуалізації ЕКГ. У коді ці частини реалізовані окремими методами: `_build_model_block()`, `_build_patient_block()`, `_build_dataset_block()`, `_build_actions_block()` та `_build_plot_block()`. Такий поділ робить інтерфейс більш структурованим і спрощує подальшу підтримку програми.

Модуль завантаження моделі відповідає за вибір і відкриття файлу навченої нейронної мережі у форматі `.keras`. Після натискання кнопки «Завантажити модель» викликається метод `load_model()`, який перевіряє наявність TensorFlow, перевіряє існування файлу моделі та завантажує її за допомогою `tf.keras.models.load_model()`. Після успішного завантаження користувач отримує відповідне повідомлення.

Модуль завантаження тестового ЕКГ-запису забезпечує вибір файлу `X_test.npy`, зчитування масиву сигналів і вибір конкретного запису за індексом. У програмі це реалізовано методом `load_sample_from_test()`. Під час завантаження виконується перевірка форми масиву, перевірка коректності індексу, збереження вибраного сигналу у змінній `current_signal` та запуск подальшої класифікації.

Модуль візуалізації кардіосигналу відповідає за побудову графіка ЕКГ. Для цього використовується бібліотека `matplotlib`, яка інтегрована в інтерфейс Tkinter через `FigureCanvasTkAgg`. Користувач може вибрати одне з 12 стандартних відведень ЕКГ, після чого програма будує графік відповідного сигналу. У коді ця функціональність реалізована методом `plot_current_signal()`.

Модуль класифікації ЕКГ-сигналу виконує прогнозування на основі завантаженої моделі CNN-LSTM. Метод `predict_current_signal()` перевіряє, чи завантажена модель і чи вибраний сигнал, після чого формує вхідний масив для моделі, виконує прогнозування та отримує ймовірності для класів NORM, MI,

STTC, CD, НУР. Далі отримані значення порівнюються із заданим порогом бінаризації.

Модуль формування результату аналізу реалізовано методом `_build_result_data()`. Він обробляє ймовірності, отримані від моделі, визначає загальний тип результату — «Норма» або «Патологія», формує текстовий висновок, визначає найімовірнішу патологію та готує дані для відображення у вікні результату. У разі виявлення патологічних класів система показує їх перелік і відповідні ймовірності.

Окремим модулем є вікно результату класифікації. Воно відкривається після натискання кнопки або автоматично після завантаження та аналізу запису. У цьому вікні користувач бачить істинні мітки, передбачені мітки, загальний висновок, ймовірність норми, ймовірність патології, виявлені патології, найімовірнішу патологію та пояснення рішення. Також у цьому вікні розміщена кнопка «Занести в базу даних», яка передає результат до модуля роботи з SQLite.

Модуль роботи з базою даних забезпечує створення, заповнення та читання локальної SQLite-бази `ecg_analysis.db`. Створення таблиці виконується методом `_init_database()`, а збереження результату — методом `save_current_result_to_db()`. Після збереження дані можуть бути переглянуті у вікні «База даних», яке відкривається методом `open_database_window()`. Для оновлення таблиці використовується метод `refresh_database_view()`.

Модуль звітно-статистичної інформації використовується для узагальнення даних, які зберігаються в базі. Він підраховує загальну кількість аналізів, кількість результатів типу «Норма», кількість результатів типу «Патологія», розподіл патологій за типами та кількість аналізів по пацієнтах. У коді ця функціональність реалізована методами `_get_report_stats()`, `open_reports_window()` та `refresh_reports()`. Для збереження статистики у файл використовується метод `export_reports_csv()`.

Для узагальнення організаційної структури програмного забезпечення побудовано діаграму пакетів. Вона показує основні логічні частини системи та взаємодію між ними.

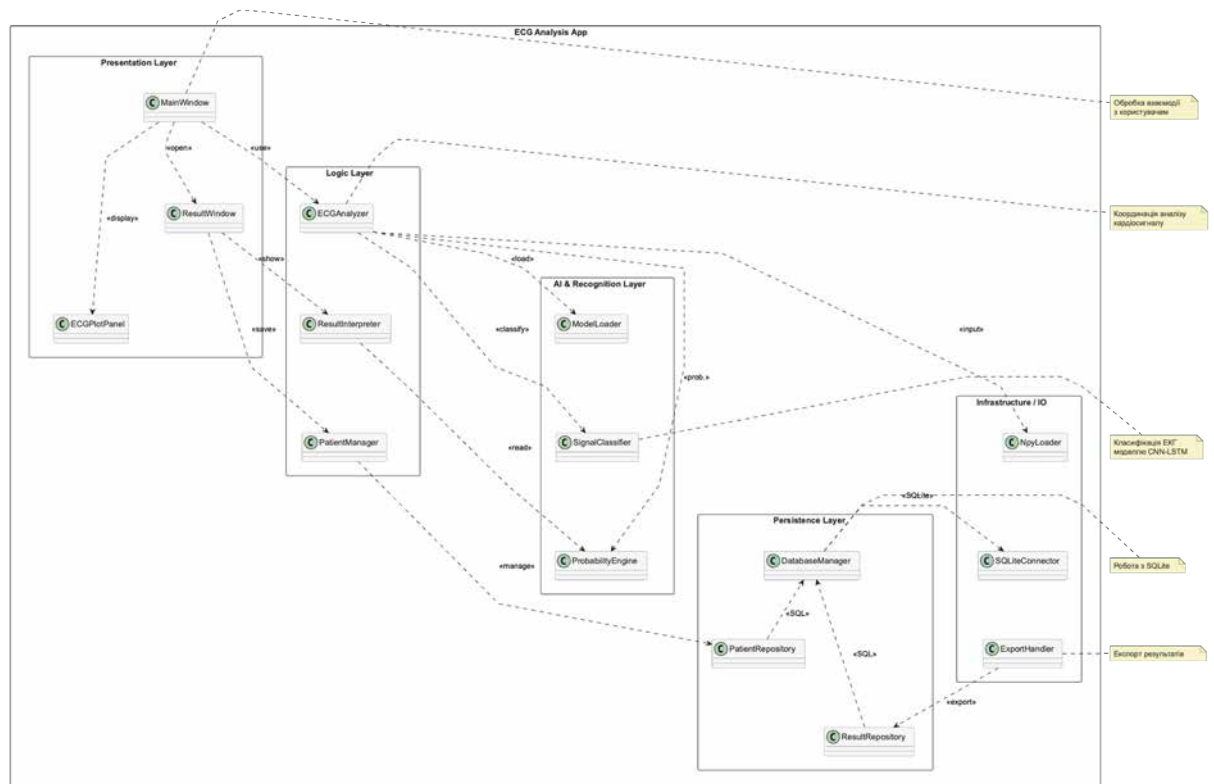


Рисунок 3.1 – Діаграма пакетів програмної системи аналізу кардіосигналів

На діаграмі пакетів доцільно виділити такі основні пакети програмної системи:

- GUI Module — відповідає за графічний інтерфейс користувача;
- Model Module — відповідає за завантаження та використання моделі CNN-LSTM;
- ECG Data Module — забезпечує завантаження тестових ЕКГ-записів;
- Visualization Module — відповідає за побудову графіка кардіосигналу;
- Analysis Module — виконує класифікацію сигналу та формування висновку;
- Database Module — забезпечує роботу з SQLite;
- Reports Module — формує звітно-статистичну інформацію;
- Export Module — забезпечує експорт звітів у CSV.

Для кращого розуміння призначення основних модулів системи наведено таблицю 3.1.

Таблиця 3.1 – Основні модулі програмного забезпечення

Модуль	Реалізація в коді	Призначення
Графічний інтерфейс	<code>_build_ui()</code> , <code>_build_model_block()</code> , <code>_build_patient_block()</code> , <code>_build_dataset_block()</code> , <code>_build_actions_block()</code>	Формування головного вікна програми та елементів керування
Завантаження моделі	<code>load_model()</code>	Завантаження файлу моделі .keras
Завантаження ЕКГ-запису	<code>load_sample_from_test()</code>	Зчитування тестового сигналу з X_test.npy
Візуалізація ЕКГ	<code>plot_current_signal()</code>	Побудова графіка вибраного відведення ЕКГ
Класифікація сигналу	<code>predict_current_signal()</code>	Отримання прогнозу моделі CNN-LSTM
Формування результату	<code>_build_result_data()</code> , <code>show_result_window()</code>	Підготовка та відображення висновку системи
Робота з БД	<code>_init_database()</code> , <code>save_current_result_to_db()</code> , <code>open_database_window()</code> , <code>refresh_database_view()</code>	Створення бази, збереження та перегляд результатів
Формування звітів	<code>_get_report_stats()</code> , <code>open_reports_window()</code> , <code>refresh_reports()</code>	Підрахунок статистичних показників
Експорт звіту	<code>export_reports_csv()</code>	Збереження звітної статистичної інформації у CSV-файл

Взаємодія між модулями відбувається послідовно. Спочатку користувач через інтерфейс завантажує модель та тестовий запис. Далі модуль аналізу

передає сигнал до моделі нейронної мережі, отримує ймовірності класів і формує результат. Після цього результат може бути показаний користувачу, збережений до бази даних або використаний для формування статистичного звіту.

Таким чином, організаційна структура програмного забезпечення побудована за модульним принципом. Кожна частина системи виконує окрему функцію: інтерфейс забезпечує взаємодію з користувачем, модель виконує класифікацію, база даних зберігає результати, а модуль звітів формує узагальнену інформацію. Така структура спрощує супровід програми, дозволяє розширювати функціональність і забезпечує логічний поділ системи на окремі компоненти.

3.2 Вибір інструментарію для створення ППЗ

Для розробки прикладного програмного забезпечення системи аналізу кардіосигналів було обрано мову програмування Python[9]. Вибір цієї мови обґрунтований тим, що Python має широкий набір бібліотек для роботи з даними, машинним навчанням, нейронними мережами, побудовою графіків, створенням графічного інтерфейсу та роботою з базами даних. Це дозволяє реалізувати всі основні частини програмної системи в одному середовищі без необхідності використання кількох різних мов програмування.

У розробленому програмному застосунку Python використовується для реалізації графічного інтерфейсу користувача, завантаження підготовлених ЕКГ-записів, роботи з моделлю нейронної мережі, побудови графіка кардіосигналу, збереження результатів у базу даних SQLite та формування звітно-статистичної інформації. У файлі програми використовуються модулі `os`, `csv`, `sqlite3`, `datetime`, `numpy`, `tkinter`, `matplotlib` і `tensorflow`, що відповідає функціональним частинам розробленої системи.

Для реалізації графічного інтерфейсу користувача було обрано бібліотеку Tkinter[10]. Вона входить до стандартного набору Python і не потребує складного додаткового встановлення. Tkinter дозволяє створювати вікна, кнопки, поля введення, списки, таблиці та інші елементи керування. У розробленій системі за

допомогою Tkinter створено головне вікно програми, блок завантаження моделі, блок введення даних пацієнта, блок вибору тестового запису, вікно результату аналізу, вікно бази даних та вікно звітів.

Для покращення вигляду інтерфейсу використовується модуль ttk, який надає сучасніші елементи керування порівняно зі стандартними віджетами Tkinter. У програмі ttk використовується для створення кнопок, рамок, підписів, таблиць Ttreeview, випадаючих списків і смуг прокручування. Це дозволяє зробити інтерфейс більш зрозумілим і зручним для користувача.

Для роботи з числовими масивами було обрано бібліотеку NumPy[11]. Вона використовується для завантаження тестових даних із файлів X_test.npy та y_test.npy, перевірки розмірності масивів, вибору конкретного ЕКГ-запису за індексом і підготовки сигналу до подання на вхід моделі. Оскільки ЕКГ-сигнали у системі подаються у вигляді багатовимірних масивів, NumPy є необхідним інструментом для їх ефективної обробки.

Для реалізації моделі класифікації було використано TensorFlow/Keras. У системі навчена модель зберігається у файлі формату .keras, а завантаження виконується за допомогою методу tf.keras.models.load_model(). TensorFlow/Keras було обрано тому, що цей інструментарій підтримує створення, навчання, збереження та повторне використання моделей глибокого навчання.

У розробленій програмній системі використовується попередньо навчена модель типу CNN-LSTM. Така модель поєднує два підходи до аналізу даних: згорткову нейронну мережу та рекурентну нейронну мережу з LSTM-шарами. Це є доцільним для аналізу ЕКГ-сигналів, оскільки кардіосигнал має як локальні особливості форми, так і часову послідовність.

Згорткова частина моделі, тобто CNN, призначена для автоматичного виділення характерних локальних ознак ЕКГ-сигналу[12]. До таких ознак можна віднести форму окремих ділянок сигналу, зміну амплітуди, повторювані фрагменти та інші особливості, які можуть бути важливими для визначення нормального або патологічного стану. На відміну від класичних методів, де ознаки потрібно задавати вручну, згорткова нейронна мережа може самостійно навчитися виділяти інформативні фрагменти сигналу.

LSTM є різновидом рекурентної нейронної мережі, призначеним для обробки послідовностей даних[13]. Назва LSTM розшифровується як Long Short-Term Memory, тобто «довга короткочасна пам'ять». Особливість LSTM полягає в тому, що вона може зберігати важливу інформацію про попередні елементи послідовності та враховувати її під час аналізу наступних елементів. Для ЕКГ-сигналів це важливо, оскільки значення сигналу в певний момент часу пов'язане з попередніми та наступними ділянками кардіосигналу. У програмній реалізації LSTM може використовуватися як готовий шар Keras для обробки часових послідовностей [14].

Архітектура CNN-LSTM у межах роботи використовується для багатоміткової класифікації ЕКГ-записів. На вхід моделі подається підготовлений ЕКГ-сигнал у вигляді числового масиву, що містить часові відліки та 12 відведень. Спочатку згорткові шари обробляють сигнал і виділяють локальні ознаки. Далі LSTM-шар аналізує послідовність отриманих ознак у часі. Після цього повнозв'язні шари формують вихідні значення для діагностичних класів. Оскільки один ЕКГ-запис може мати кілька патологічних ознак одночасно, на виході моделі використовуються незалежні ймовірності для кожного класу.

У системі розглядаються такі діагностичні класи: NORM — норма, MI — інфаркт міокарда, STTC — зміни сегмента ST/T, CD — порушення провідності, HYP — гіпертрофія. Для кожного з цих класів модель формує числове значення ймовірності. Далі ці значення порівнюються із заданим порогом класифікації. Якщо ймовірність патологічного класу перевищує поріг, система визначає відповідну патологію як виявлену.

Навчання моделі виконувалося окремо на підготовлених даних ЕКГ, сформованих на основі набору PTB-XL. У графічному застосунку навчання моделі не виконується, оскільки програма призначена для практичного використання вже навченої моделі. У програмі модель зберігається у файлі `best_cnn_lstm_ptb-xl.keras` і завантажується через графічний інтерфейс. Такий підхід дозволяє відокремити етап навчання нейронної мережі від етапу

використання програмної системи. Користувач працює тільки з готовою моделлю, завантажує тестовий ЕКГ-запис і отримує результат класифікації.

Файл моделі має формат .keras, що є стандартним форматом збереження моделей Keras. Після завантаження моделі програмна система передає на її вхід вибраний ЕКГ-сигнал, отримує ймовірності класів і формує висновок про стан кардіосигналу. Завдяки цьому модель CNN-LSTM є центральним компонентом інтелектуальної частини програмної системи.

Для побудови графіка кардіосигналу використано бібліотеку Matplotlib[15]. Вона дозволяє будувати графіки сигналів, налаштовувати підписи осей, сітку, назву графіка та оновлювати зображення у вікні програми. У розробленій системі Matplotlib інтегровано в Tkinter за допомогою FigureCanvasTkAgg, що дозволяє відображати графік ЕКГ безпосередньо в інтерфейсі програми. Користувач може вибрати одне з 12 відведень ЕКГ, після чого система будує графік відповідного сигналу.

Для збереження результатів аналізу було обрано SQLite. Робота з базою даних виконується за допомогою стандартного Python-модуля sqlite3. SQLite не потребує встановлення окремого сервера, а база даних зберігається у вигляді одного локального файлу ecg_analysis.db. Це є зручним рішенням для настільного застосунку, оскільки користувач може працювати з програмою локально, без налаштування серверної інфраструктури.

Для формування та експорту звітів використано стандартний модуль csv. Він дозволяє створювати CSV-файли, які можна відкривати у табличних редакторах. У системі експорт звіту включає загальну кількість аналізів, кількість нормальних і патологічних результатів, розподіл патологій за типами та кількість аналізів по пацієнтах.

Для кращого розуміння обраного інструментарію наведено таблицю 3.2.

Таблиця 3.2 – Інструментальні засоби створення програмного забезпечення

Засіб	Призначення в системі	Обґрунтування вибору
Python	Основна мова	Має бібліотеки для ШІ,

	програмування	обробки даних, GUI та БД
Tkinter / ttk	Створення графічного інтерфейсу	Входить до стандартного набору Python, проста у використанні
NumPy	Робота з масивами ЕКГ-сигналів	Ефективна обробка багатовимірних числових даних
TensorFlow/Keras	Завантаження та використання моделі CNN-LSTM	Підтримує моделі глибокого навчання
Matplotlib	Побудова графіків ЕКГ	Дає змогу візуалізувати кардіосигнал у вікні програми
SQLite / sqlite3	Збереження результатів аналізу	Локальна база даних без окремого сервера
CSV / csv	Експорт звітів	Простий формат для відкриття у табличних редакторах
PyInstaller	Створення виконуваного файлу .exe	Дозволяє запускати програму без ручного запуску Python-файлу
Inno Setup	Створення інсталятора	Дає змогу сформувати інсталяційний пакет системи

Під час розробки інтерфейсу було враховано, що користувач повинен мати змогу виконувати всі основні дії без роботи з кодом. Для цього у програмі створено окремі блоки: «Модель», «Дані пацієнта», «Аналіз з тестового набору», «Керування» та область «Візуалізація ЕКГ». Такий поділ інтерфейсу відповідає логіці роботи системи: спочатку користувач завантажує модель, потім вводить дані пацієнта, вибирає тестовий запис, після чого отримує графік і результат аналізу.

Окремою перевагою обраного інструментарію є можливість створення інсталяційного пакету. Для цього використовується PyInstaller, який дозволяє зібрати Python-застосунок у виконуваний файл, та Inno Setup, за допомогою якого формується інсталятор[16]. Завдяки цьому користувач може запускати систему як звичайний встановлений настільний застосунок Windows без необхідності відкривати програмний код або працювати з командним рядком [16].

Таким чином, вибір інструментарію для створення прикладного програмного забезпечення є обгрунтованим. Python забезпечує єдине середовище для реалізації всіх частин системи, TensorFlow/Keras дозволяє працювати з нейронною мережею, NumPy забезпечує обробку ЕКГ-даних, Matplotlib відповідає за візуалізацію сигналу, Tkinter використовується для створення інтерфейсу, SQLite забезпечує локальне збереження результатів, а CSV дає змогу експортувати звітно-статистичну інформацію.

3.3. Алгоритмізація та програмування програмних модулів

Алгоритмізація програмних модулів є важливим етапом розробки програмної системи, оскільки дозволяє описати послідовність дій, які виконує програма під час обробки даних, класифікації кардіосигналу, збереження результатів у базу даних та формування звітно-статистичної інформації. У розробленому програмному забезпеченні основні алгоритми реалізовані в класі ECGAnalyzerApp.

У програмі можна виділити такі основні алгоритмічні частини:

- завантаження моделі нейронної мережі;
- завантаження ЕКГ-запису з тестового набору;
- побудова графіка кардіосигналу;
- класифікація сигналу за допомогою моделі CNN-LSTM;
- формування результату аналізу;
- збереження результату в базу даних SQLite;

- перегляд збережених результатів;
- формування звітно-статистичної інформації;
- експорт звіту у CSV-файл.

Основний процес роботи системи починається із завантаження моделі. Для цього в програмі використовується метод `load_model()`. Він перевіряє наявність бібліотеки TensorFlow, перевіряє існування файлу моделі та завантажує модель за допомогою `tf.keras.models.load_model()`. Після успішного завантаження система повідомляє користувача, що модель готова до використання.

Після цього користувач завантажує ЕКГ-запис із тестового набору. У програмі це реалізовано методом `load_sample_from_test()`. Метод перевіряє наявність файлу `X_test.npy`, коректність введеного індексу запису, розмірність масиву та відповідність форми сигналу очікуваному вигляду (Т, 12). Після успішного завантаження сигнал зберігається у змінній `current_signal`, після чого автоматично будується графік і запускається класифікація. У коді також передбачено перевірки помилок для некоректного індексу, неправильної форми масиву та відсутнього файлу тестових даних.

Побудова графіка кардіосигналу реалізована методом `plot_current_signal()`. Якщо сигнал не завантажено, програма виводить порожній графік із повідомленням про відсутність сигналу. Якщо сигнал наявний, система визначає вибране користувачем відведення ЕКГ, отримує відповідний канал сигналу та будує графік у вікні програми. Для цього використовується бібліотека Matplotlib, а сам графік вбудовується в інтерфейс Tkinter.

Класифікація сигналу виконується методом `predict_current_signal()`. Перед виконанням прогнозування програма перевіряє, чи завантажена модель і чи завантажений ЕКГ-сигнал. Далі значення порогу бінаризації зчитується з інтерфейсу користувача, а сигнал перетворюється у формат батчу за допомогою `np.expand_dims()`. Після цього модель виконує прогнозування і повертає ймовірності для класів NORM, MI, STTC, CD, HYP. Отримані ймовірності порівнюються з порогом, після чого формується масив передбачених міток.

Формування результату аналізу виконується методом `_build_result_data()`. У цьому методі визначається ймовірність класу NORM, окремо обробляються

ймовірності патологічних класів MI, STTC, CD, HYP, а також обчислюється інтегральна ймовірність патології. Якщо один або кілька патологічних класів перевищують заданий поріг, система формує висновок «Ймовірно наявна патологія». Якщо клас NORM перевищує поріг, а патологічні класи не досягають його, формується висновок «Ймовірно здорова людина (норма)». Якщо патологічні класи не перетнули поріг, але їх сумарна ймовірність вища за ймовірність норми, система формує висновок про підозру на патологію.

Для наочного представлення алгоритму обробки даних у системі використовується блок-схема. Вона відображає послідовність дій від завантаження ЕКГ-запису до отримання результату класифікації та його збереження.

Алгоритм збереження результату аналізу в БД

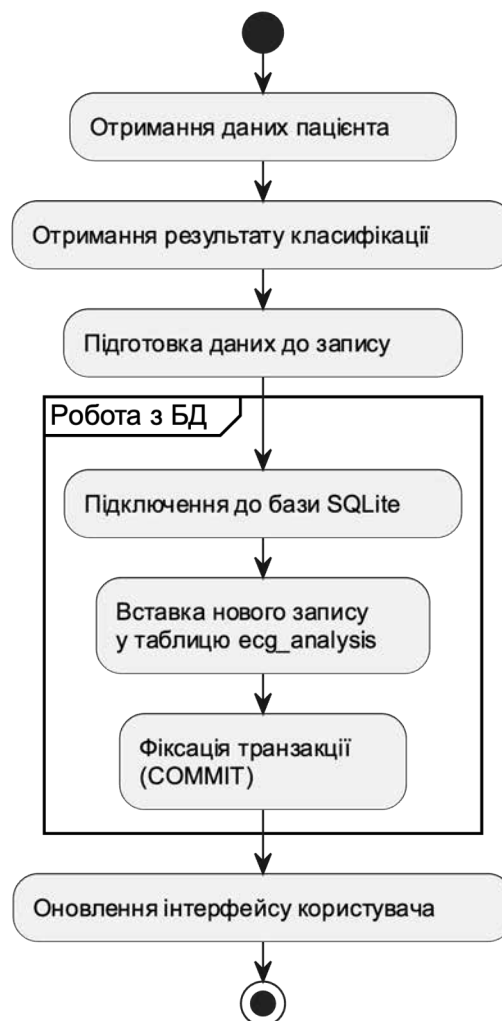


Рисунок 3.2 – Блок-схема алгоритму обробки даних у програмній системі

На рис. 3.2 показано алгоритм обробки даних у системі. Спочатку користувач завантажує модель і вибирає тестовий ЕКГ-запис. Далі система перевіряє коректність вхідних даних, завантажує сигнал, будує графік вибраного відведення та передає сигнал на класифікацію до моделі CNN-LSTM. Після отримання ймовірностей класів формується висновок про стан пацієнта. У разі виявлення патології система визначає її тип, а якщо патологія не виявлена — формує висновок про норму.

Результат класифікації відображається в окремому вікні. У цьому вікні показуються істинні мітки, передбачені мітки, загальний висновок, ймовірність норми, ймовірність патології, виявлені патології, найімовірніша патологія та пояснення рішення. Також у цьому вікні розміщено кнопку «Занести в базу даних», яка запускає алгоритм збереження результату.

Збереження результату в базу даних реалізовано методом `save_current_result_to_db()`. Спочатку метод перевіряє, чи існує сформований результат аналізу. Потім викликається метод `_get_patient_data()`, який перевіряє введені користувачем ПІБ і вік пацієнта. Якщо дані введені коректно, програма відкриває з'єднання з базою SQLite та виконує SQL-запит `INSERT INTO analysis_results`. До бази заносяться дата і час аналізу, ПІБ, вік, результат, патологія, висновок, істинні й передбачені мітки, ймовірність норми, ймовірність патології, поріг, інформація про сигнал і шлях до моделі. Після цього виконується `commit()`, а користувачу виводиться повідомлення про успішне збереження.

Фрагмент коду збереження результату аналізу наведено нижче.

```
def save_current_result_to_db(self):
    if not self.current_result:
        messagebox.showerror("Помилка", "Немає результату для збереження")
        return
```

```

try:

    full_name, age = self._get_patient_data()

except ValueError as exc:

    messagebox.showerror("Помилка", str(exc))

    return

data = self.current_result

with sqlite3.connect(self.db_path) as conn:

    conn.execute(

        """

        INSERT INTO analysis_results (

            created_at, full_name, age, result_type, pathology_name, general_status,

            true_labels, predicted_labels, norm_probability, pathology_probability,

            threshold, source_info, model_path

        ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)

        """,

        (

            datetime.now().strftime("%Y-%m-%d %H:%M:%S"),

            full_name,

            age,

            data["result_type"],

            data["pathology_name"],

            data["conclusion"],

```

```

data["true_labels"],
data["predicted_labels"],
data["norm_probability_raw"],
data["pathology_probability_raw"],
float(data["threshold"]),
data["signal_info"],
self.model_path_var.get().strip(),
),
)
conn.commit()

```

Рисунок 3.3 – Фрагмент коду збереження результату аналізу в базу даних

Для перегляду результатів у програмі реалізовано окреме вікно «База даних». Після відкриття цього вікна метод `refresh_database_view()` виконує SQL-запит `SELECT`, отримує записи з таблиці `analysis_results` і відображає їх у таблиці `Treeview`. Користувач бачить дату аналізу, ПІБ, вік, результат, патологію, висновок, імовірність норми та імовірність патології. При подвійному натисканні на запис відкривається вікно з детальною інформацією.

Окремим важливим модулем є формування звітно-статистичної інформації. У системі цей алгоритм реалізовано методом `_get_report_stats()`. Він виконує SQL-запити до таблиці `analysis_results` і підраховує загальну кількість аналізів, кількість результатів «Норма», кількість результатів «Патологія», розподіл патологій за типами та кількість аналізів по кожному пацієнту.

Для наочного представлення алгоритму формування звітно-статистичної інформації використовується окрема блок-схема.

Алгоритм збереження результату аналізу в БД

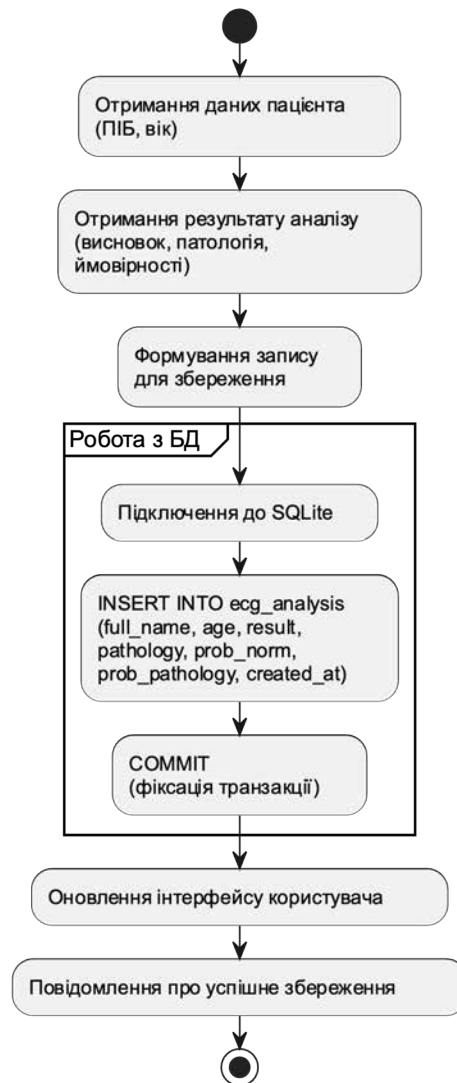


Рисунок 3.4 – Блок-схема алгоритму формування звітно-статистичної інформації

На рис. 3.4 показано алгоритм формування звітно-статистичної інформації. Спочатку користувач відкриває вікно «Звіти». Після цього програма підключається до бази даних SQLite і виконує вибірку записів із таблиці `analysis_results`. Якщо записи знайдено, система підраховує основні статистичні показники: загальну кількість аналізів, кількість нормальних результатів, кількість патологічних результатів, розподіл патологій за типами та кількість аналізів по пацієнтах. Після завершення обробки статистика відображається у вікні програми.

Фрагмент коду формування статистики наведено нижче.

```
def _get_report_stats(self):

    with sqlite3.connect(self.db_path) as conn:

        total = conn.execute("SELECT COUNT(*) FROM
analysis_results").fetchone()[0]

        normal_count = conn.execute(

            "SELECT COUNT(*) FROM analysis_results WHERE result_type =
'Норма'"

        ).fetchone()[0]

        pathology_count = conn.execute(

            "SELECT COUNT(*) FROM analysis_results WHERE result_type =
'Патологія'"

        ).fetchone()[0]

        pathology_rows = []

        for code in PATHOLOGY_CLASSES:

            count = conn.execute(

                "SELECT COUNT(*) FROM analysis_results WHERE pathology_name
LIKE ?",

                (f"%{code}%",),

            ).fetchone()[0]

            pathology_rows.append((code, CLASS_LABELS[code], count))

        patient_rows = conn.execute(
```

```

"""

SELECT full_name, COUNT(*) AS cnt

FROM analysis_results

GROUP BY full_name

ORDER BY cnt DESC, full_name ASC

"""

).fetchall()

```

Рисунок 3.5 – Фрагмент коду формування звітної-статистичної інформації

У програмі також передбачено експорт сформованого звіту у формат CSV. Для цього використовується метод `export_reports_csv()`. Він отримує статистичні дані, відкриває діалог збереження файлу, після чого записує у CSV-файл загальну статистику, розподіл патологій і кількість аналізів по пацієнтах. У файлі використовується кодування `utf-8-sig`, що забезпечує коректне відкриття українського тексту в табличних редакторах.

Фрагмент коду експорту звіту наведено нижче.

```

with open(save_path, "w", encoding="utf-8-sig", newline="") as f:

    writer = csv.writer(f, delimiter=';')

    writer.writerow(["Показник", "Значення"])

    writer.writerow(["Усього аналізів", stats["total"]])

    writer.writerow(["Норма", stats["normal"]])

    writer.writerow(["Патологія", stats["pathology"]])

    writer.writerow([])

    writer.writerow(["Код патології", "Назва", "Кількість"])

    for row in stats["pathology_rows"]:

```

```
writer.writerow(row)

writer.writerow([])

writer.writerow(["ПІБ", "Кількість аналізів"])

for row in stats["patient_rows"]:

    writer.writerow(row)
```

Рисунок 3.6 – Фрагмент коду експорту звітно-статистичної інформації у CSV-файл

Таким чином, у програмній системі реалізовано повний алгоритмічний цикл роботи з кардіосигналом: завантаження даних, побудова графіка, класифікація, формування висновку, збереження результату в базу даних, перегляд записів і формування статистичного звіту. Побудовані блок-схеми відображають основні алгоритми роботи системи, а наведені фрагменти коду підтверджують їхню практичну реалізацію в програмному файлі `ecg_gui_app.py`.

4 ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ СИСТЕМИ

4.1 Тестування системи

Тестування програмної системи є одним із завершальних етапів розробки, який дозволяє перевірити правильність роботи основних функцій, виявити можливі помилки та переконатися, що програмне забезпечення відповідає поставленим вимогам[17]. Для розробленої системи аналізу кардіосигналів тестування проводилось з метою перевірки завантаження моделі, завантаження ЕКГ-запису, побудови графіка, виконання класифікації, збереження результату в базу даних, перегляду збережених записів, формування звітів та експорту звітно-статистичної інформації у CSV-файл.

У процесі тестування використовувався підхід функціонального тестування. Основна увага приділялася перевірці того, чи правильно система реагує на дії користувача та чи коректно виконує основні операції. Оскільки користувач взаємодіє із системою через графічний інтерфейс, тестування проводилось шляхом послідовного виконання типових сценаріїв роботи з програмою.

Першим етапом тестування була перевірка запуску головного вікна програми. Після запуску файлу `ecg_gui_app.py` повинно відкриватися головне вікно застосунку з блоками «Модель», «Дані пацієнта», «Аналіз з тестового набору», «Керування» та областю «Візуалізація ЕКГ». Також було перевірено, що ліва частина інтерфейсу має вертикальну прокрутку, завдяки чому всі елементи керування залишаються доступними навіть при меншій висоті вікна.

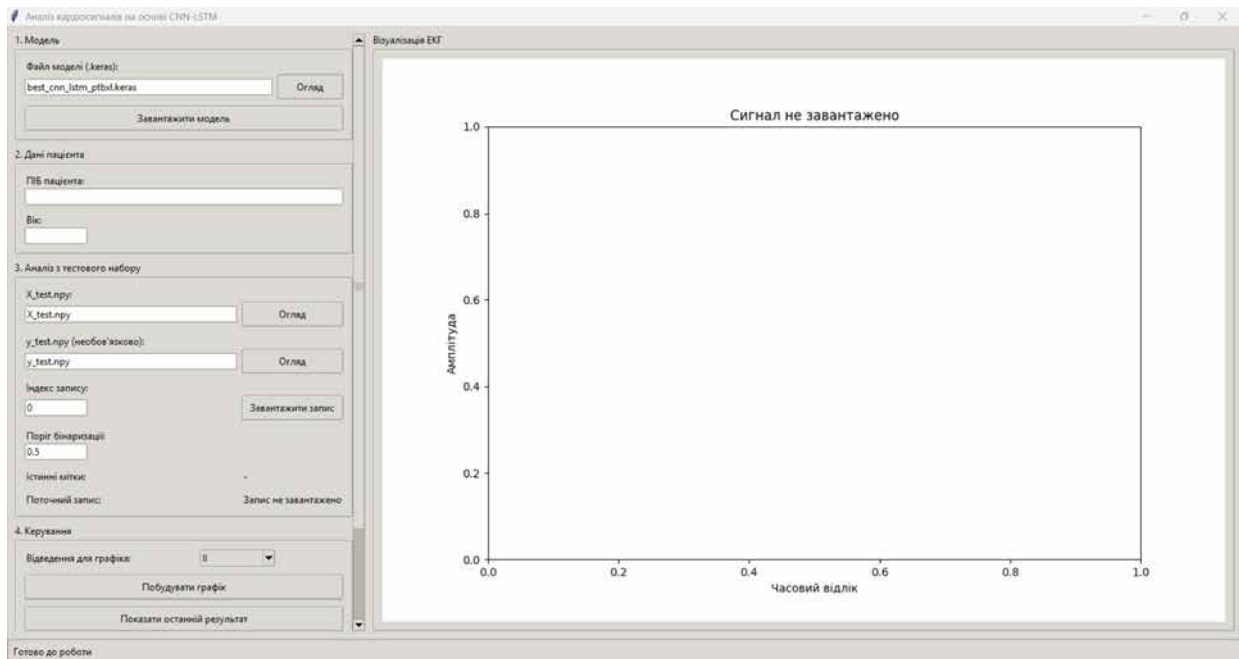


Рисунок 4.1 – Головне вікно програмної системи аналізу кардіосигналів

Наступним етапом було перевірено завантаження моделі нейронної мережі. Користувач вибирає файл моделі у форматі .keras або залишає стандартний шлях до файлу `best_cnn_lstm_ptb1.keras`, після чого натискає кнопку «Завантажити модель». Якщо файл існує та TensorFlow встановлено коректно, система завантажує модель і виводить повідомлення про успішне завантаження. Якщо файл моделі не знайдено, програма повинна вивести повідомлення про помилку.

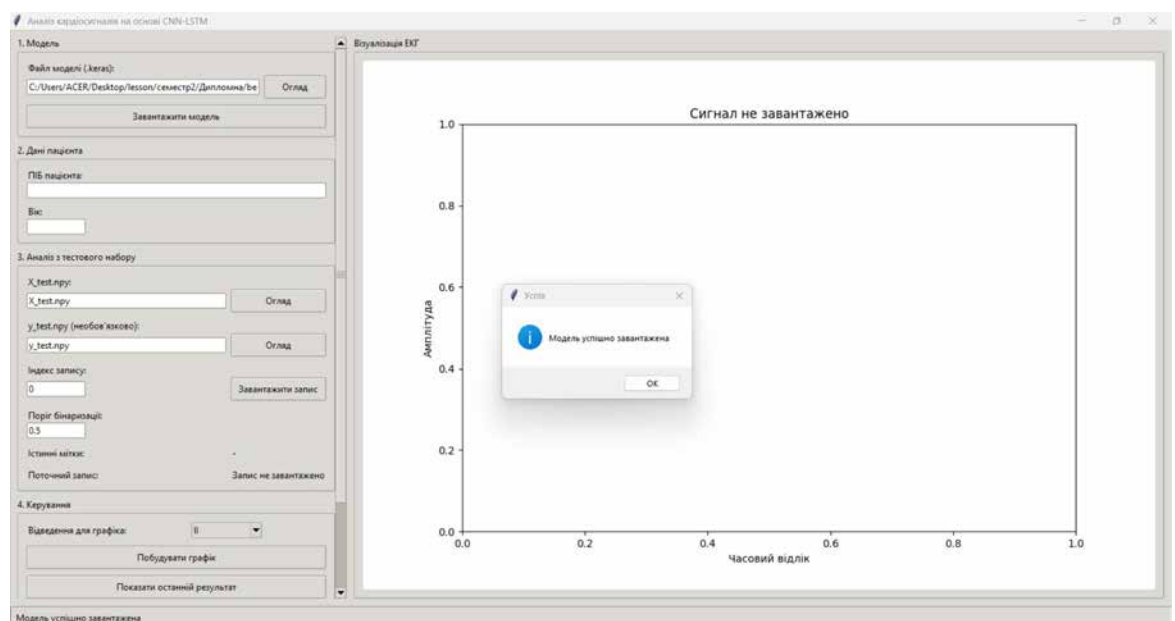


Рисунок 4.2 – Повідомлення про успішне завантаження моделі

Окремо було протестовано завантаження тестового ЕКГ-запису. Для цього у програмі вибирається файл `X_test.npy`, вказується індекс запису та натискається кнопка «Завантажити запис». Після цього програма перевіряє наявність файлу, правильність введеного індексу та форму масиву. Якщо дані коректні, система завантажує сигнал, відображає інформацію про поточний запис і будує графік вибраного відведення ЕКГ.

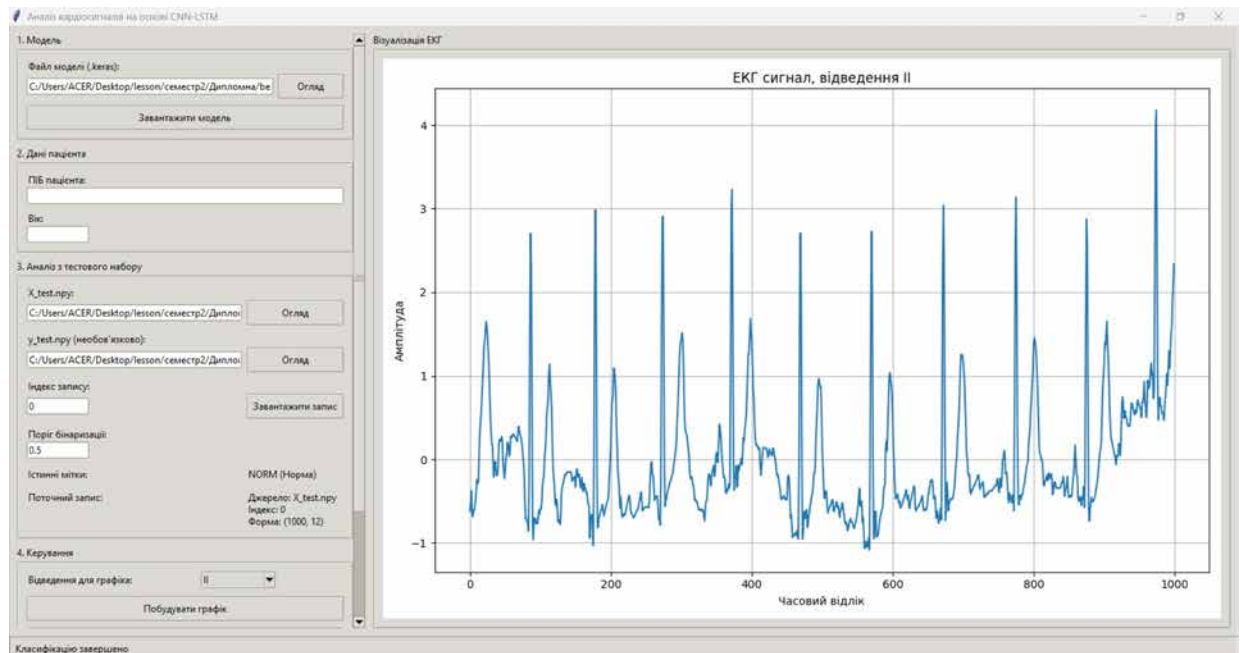


Рисунок 4.3 – Побудова графіка ЕКГ-сигналу

Після завантаження запису система автоматично виконує класифікацію сигналу за допомогою навченої моделі CNN-LSTM. У результаті формується окреме вікно, де відображається медичний висновок, істинні мітки, передбачені мітки, ймовірність норми, ймовірність патології, виявлені патології та найімовірніший патологічний клас. Було перевірено, що результат відображається у зрозумілому вигляді, а значення ймовірностей подаються як у числовому форматі, так і у відсотках.

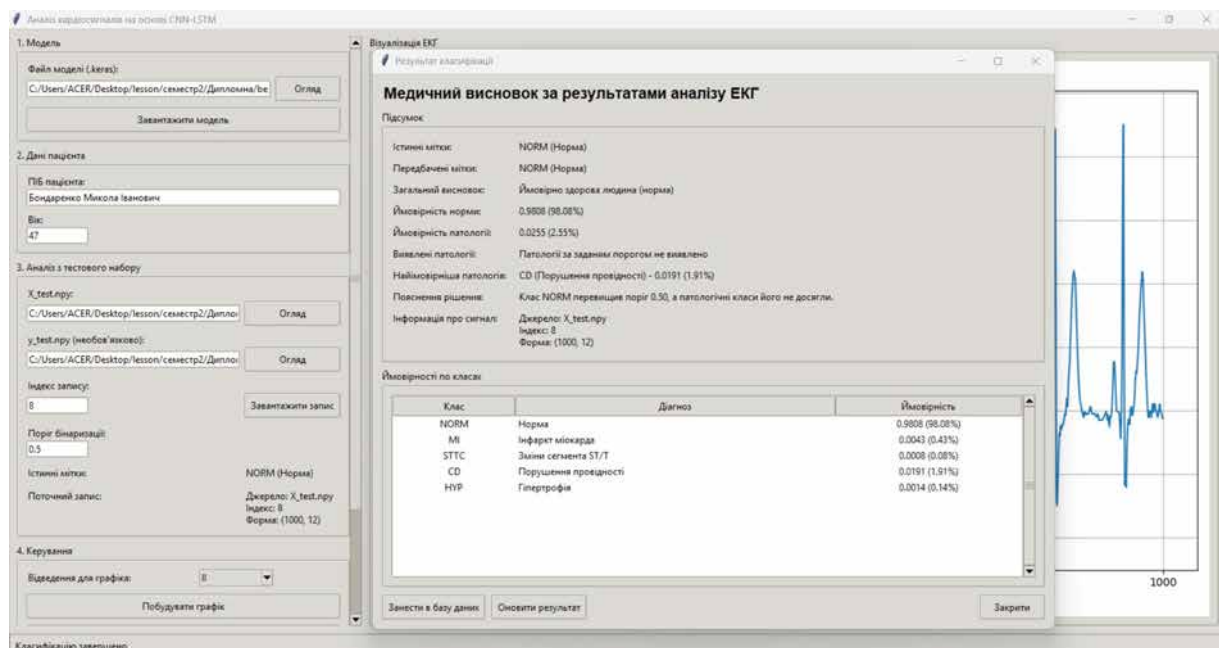


Рисунок 4.4 – Вікно результату аналізу кардіосигналу

Також було перевірено роботу механізму збереження результату в базу даних. Для цього після отримання результату аналізу користувач вводить ПІБ і вік пацієнта, після чого натискає кнопку «Занести в базу даних». Якщо всі необхідні дані введено правильно, програма додає новий запис до таблиці analysis_results у базі даних SQLite та виводить повідомлення про успішне збереження. Якщо ПІБ або вік не введено, система показує повідомлення про помилку та не виконує запис у базу даних.

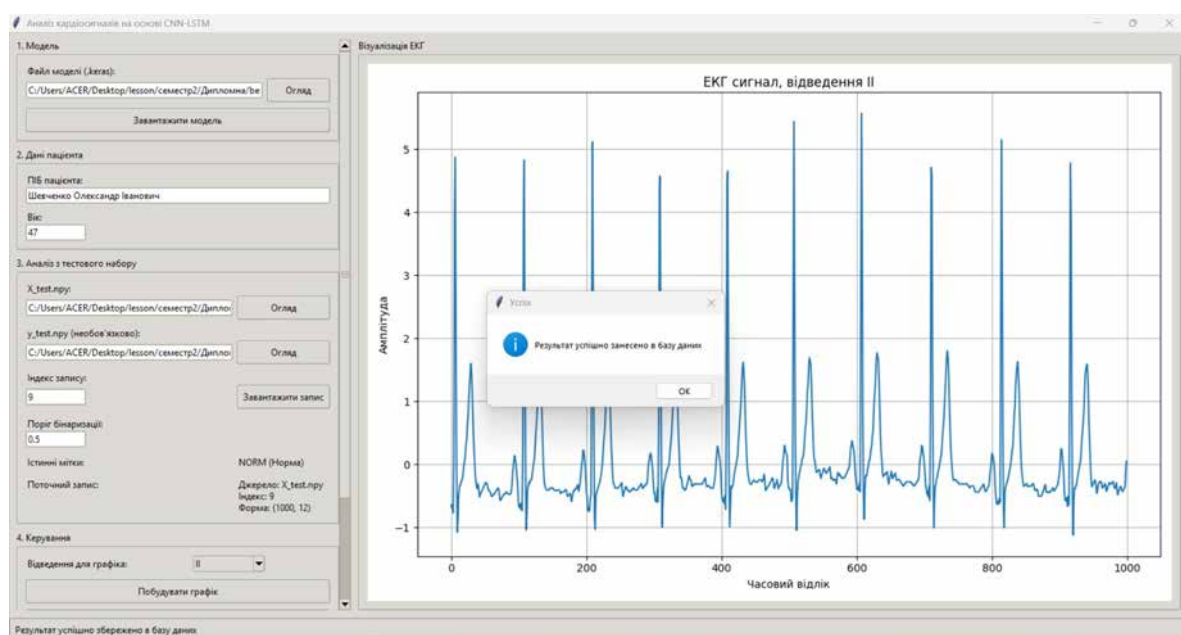


Рисунок 4.5 – Збереження результату аналізу в базу даних

Після цього було протестовано перегляд збережених результатів. При натисканні кнопки «База даних» відкривається окреме вікно, у якому відображаються всі записи, збережені в таблиці `analysis_results`. У таблиці виводяться ID запису, дата, ПІБ пацієнта, вік, результат, патологія, висновок, ймовірність норми та ймовірність патології. Також перевірено можливість перегляду детальної інформації про запис подвійним натисканням на відповідний рядок.

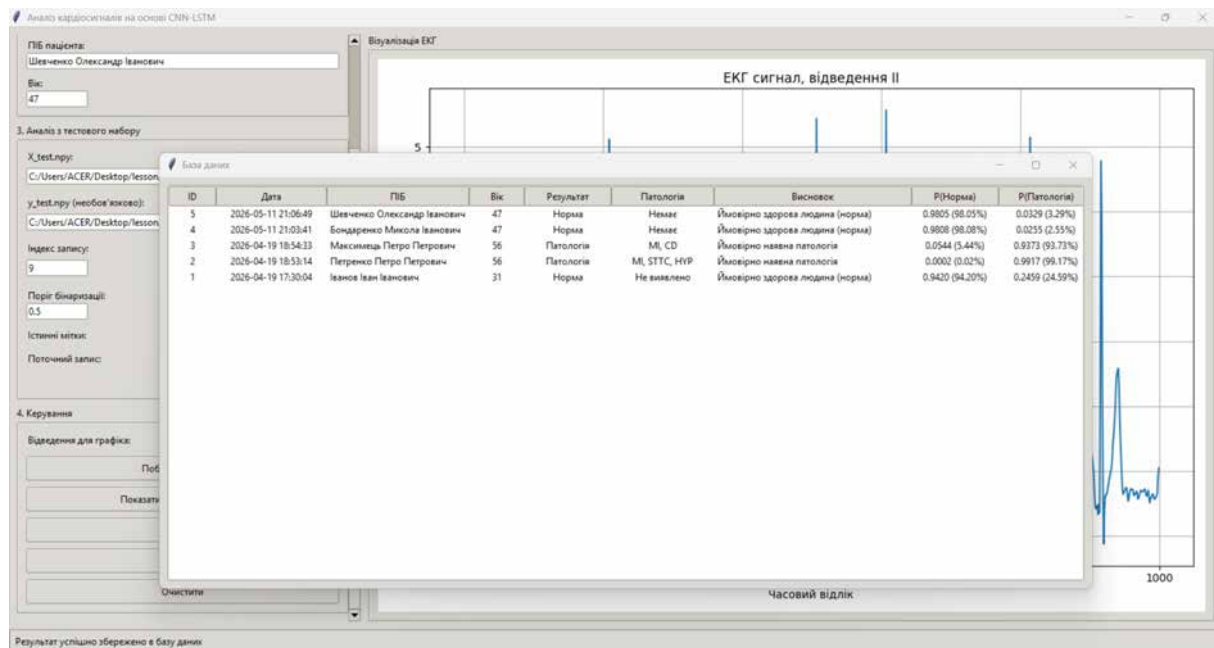


Рисунок 4.6 – Вікно перегляду збережених результатів аналізу

Окремо було протестовано модуль формування звітно-статистичної інформації. Після натискання кнопки «Звіти» відкривається вікно, у якому система відображає загальну кількість аналізів, кількість результатів типу «Норма», кількість результатів типу «Патологія», розподіл патологій за типами та кількість аналізів по пацієнтах. Дані для звіту формуються на основі записів, що зберігаються в базі SQLite.

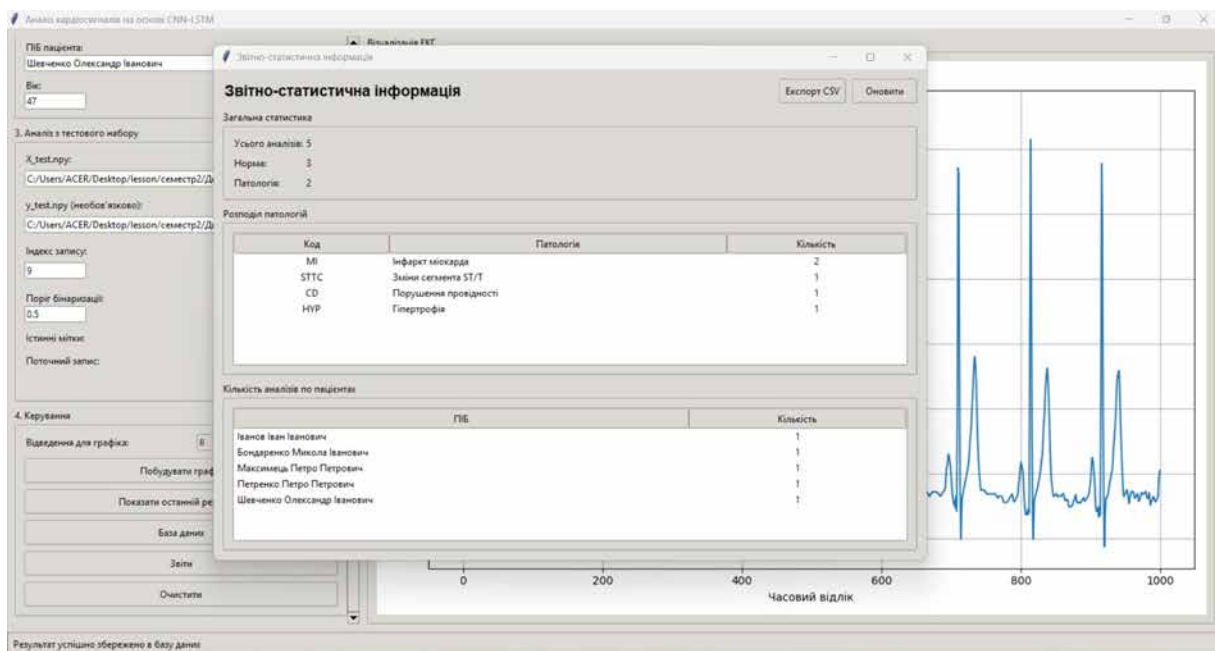


Рисунок 4.7 – Вікно звітно-статистичної інформації

Також було перевірено експорт звіту у формат CSV. Після натискання кнопки «Експорт CSV» користувач вибирає місце збереження файлу, після чого програма формує CSV-файл із загальною статистикою, розподілом патологій та кількістю аналізів по пацієнтах. Було перевірено, що файл створюється коректно та відкривається в табличному редакторі. Для коректного відображення українського тексту використовується кодування utf-8-sig.

111.csv - Excel

Файл | Основні | Вставлення | Макет сторінки | Формули | Дані | Рецензування | Подання | Довідка | Схожість, що потрібно зробити | Службовий доступ

Вставити | Буфер обміну | Шрифт | Веріфікація | Число | Умовне форматування | Стилі | Вставити | Виділити | Формат | Автоsuma | Заповнити | Очистити | Сортувати й фільтрувати | Знайти й замінити

Скопіювати текст | Об'єднати та розташувати в центрі | Залишений | Умовне форматування - таблиця - комірки - Стилі | Вставити | Виділити | Формат | Автоsuma | Заповнити | Очистити | Сортувати й фільтрувати | Знайти й замінити

1	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
2		Усього ан	5																				
3		Норма	3																				
4		Патологія	2																				
5																							
6		Код патол	Назва	Кількість																			
7		MI	Інфаркт м	2																			
8		STTC	Зміни сег	1																			
9		CD	Порушенн	1																			
10		HVP	Гіпертроф	1																			
11																							
12		ПІБ	Кількість аналізів																				
13		Іванов Ів	1																				
14		Бондарен	1																				
15		Максимен	1																				
16		Петренко	1																				
17		Шевченк	1																				
18																							
19																							
20																							
21																							
22																							
23																							
24																							
25																							
26																							
27																							
28																							
29																							

Рисунок 4.8 – Результат експорту звітно-статистичної інформації у CSV-файл

У процесі тестування було перевірено основні сценарії роботи програмної системи. Зокрема, було протестовано запуск програми, завантаження моделі, завантаження тестового ЕКГ-запису, побудову графіка сигналу, виконання класифікації, збереження результату в базу даних, перегляд збережених записів, формування звітів та експорт звітно-статистичної інформації у CSV-файл.

Окремо перевірялась реакція системи на помилкові дії користувача. Якщо користувач не вибрав файл моделі, вказав неправильний індекс запису, не ввів ПІБ або вік пацієнта, програма не завершує роботу аварійно, а виводить відповідне повідомлення про помилку. Це підвищує надійність системи та робить її використання більш зрозумілим для користувача.

Для перевірки коректності введення даних пацієнта використовується метод `_get_patient_data()`. Він перевіряє, чи введено ПІБ, чи введено вік, чи є вік цілим числом і чи більше він за нуль. Така перевірка дозволяє уникнути збереження некоректних записів у базу даних.

Фрагмент перевірки даних пацієнта наведено нижче.

```
def _get_patient_data(self):  
  
    full_name = self.patient_name_var.get().strip()  
  
    age_text = self.patient_age_var.get().strip()  
  
    if not full_name:  
  
        raise ValueError("Введи ПІБ пацієнта")  
  
    if not age_text:  
  
        raise ValueError("Введи вік пацієнта")  
  
    try:  
  
        age = int(age_text)  
  
    except ValueError as exc:  
  
        raise ValueError("Вік має бути цілим числом") from exc
```

```
if age <= 0:  
  
    raise ValueError("Вік має бути більше нуля")  
  
return full_name, age
```

Рисунок 4.9 – Перевірка коректності введення даних пацієнта

Результати тестування показали, що програмне забезпечення коректно виконує основні функції, передбачені постановкою завдання. Усі ключові сценарії роботи були виконані успішно, а виявлені помилкові ситуації обробляються засобами графічного інтерфейсу. Це підтверджує працездатність розробленої системи та її готовність до практичного використання в межах поставлених задач.

4.2 Вимоги до апаратного та програмного забезпечення

Для впровадження та експлуатації програмної системи аналізу кардіосигналів необхідно визначити апаратні та програмні вимоги. Оскільки розроблена система є настільним застосунком, вона призначена для роботи на персональному комп'ютері користувача без необхідності підключення до віддаленого сервера бази даних. Усі основні компоненти системи розміщуються локально: графічний інтерфейс, модель нейронної мережі, тестові дані, база даних SQLite та модуль формування звітів.

У програмній системі використовується локальна архітектура. Користувач працює з графічним інтерфейсом, який забезпечує завантаження моделі, вибір тестового ЕКГ-запису, введення даних пацієнта, побудову графіка сигналу, перегляд результату класифікації, збереження результатів у базу даних і формування звітно-статистичної інформації.

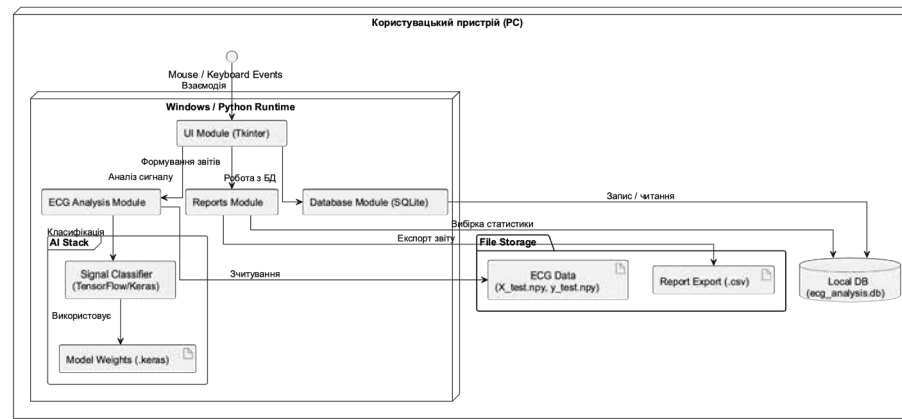


Рисунок 4.10 – Діаграма розміщення програмної системи аналізу кардіосигналів

На рис. 4.10 показано розміщення основних компонентів програмної системи. На персональному комп'ютері користувача розташовано операційну систему, середовище виконання Python, графічний інтерфейс програми, модуль аналізу кардіосигналів, модуль роботи з базою даних, модуль звітів, файл навченої моделі, тестові дані та локальну базу SQLite. Така структура не потребує окремого сервера, оскільки всі операції виконуються локально.

Якщо в роботі використовується окрема діаграма розгортання, її можна вставити після діаграми розміщення. Вона деталізує, які програмні компоненти входять до складу системи та як вони взаємодіють між собою.

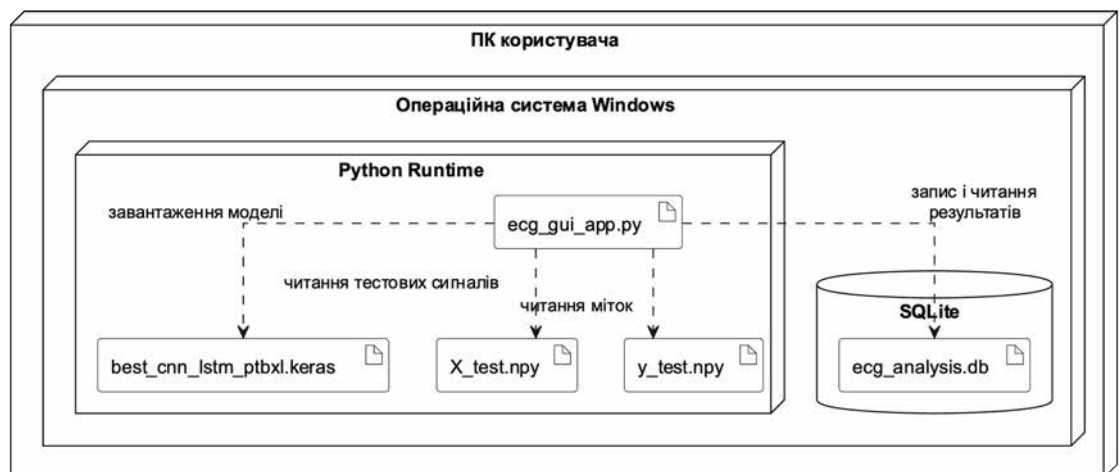


Рисунок 4.11 – Діаграма розгортання програмної системи

Діаграма розгортання показує, що програмна система складається з кількох взаємопов'язаних компонентів: графічного інтерфейсу, модуля

класифікації, модуля візуалізації ЕКГ, модуля бази даних і модуля формування звітів. Графічний інтерфейс є центральною частиною взаємодії користувача із системою. Через нього користувач виконує всі основні дії, а інші компоненти працюють як внутрішні модулі програми.

До основних програмних компонентів системи належать:

- графічний інтерфейс користувача;
- модуль завантаження моделі;
- модуль завантаження тестових ЕКГ-записів;
- модуль побудови графіка кардіосигналу;
- модуль класифікації сигналу;
- модуль формування висновку;
- модуль роботи з базою даних SQLite;
- модуль перегляду збережених записів;
- модуль формування звітно-статистичної інформації;
- модуль експорту звіту у CSV-файл.

Для роботи програмної системи необхідне таке програмне забезпечення:

- операційна система Windows 10 або Windows 11;
- встановлена програмна система аналізу кардіосигналів;
- бібліотека TensorFlow/Keras для завантаження та використання моделі нейронної мережі;
- бібліотека NumPy для роботи з масивами ЕКГ-сигналів;
- бібліотека Matplotlib для побудови графіків кардіосигналів;
- бібліотека Tkinter для реалізації графічного інтерфейсу користувача;
- модуль sqlite3 для роботи з локальною базою даних;
- модуль csv для експорту звітно-статистичної інформації;
- файл навченої моделі `best_cnn_lstm_ptbxl.keras`;
- файл тестових сигналів `X_test.npy`;
- файл істинних міток `y_test.npy`;
- локальна база даних `ecg_analysis.db`.

Експлуатація програмної системи користувачем передбачає запуск встановленого настільного застосунку. Після встановлення через інсталяційний пакет користувач запускає програму за допомогою ярлика або виконуваного файлу застосунку. Такий спосіб є зручним для практичного використання, оскільки не потребує відкриття середовища розробки, налаштування Python або введення команд у командному рядку.

До мінімальних апаратних вимог належать:

- процесор: двоядерний процесор із тактовою частотою від 2 ГГц;
- оперативна пам'ять: не менше 4 ГБ;
- вільне місце на диску: не менше 1 ГБ;
- монітор із роздільною здатністю не нижче 1366×768;
- клавіатура та миша для взаємодії з інтерфейсом.

Рекомендовані апаратні вимоги:

- процесор: чотириядерний процесор Intel Core i5 / AMD Ryzen 5 або кращий;
- оперативна пам'ять: 8 ГБ або більше;
- вільне місце на диску: 2 ГБ або більше;
- монітор із роздільною здатністю Full HD;
- твердотільний накопичувач SSD для швидшого завантаження файлів;
- стабільне середовище Python або встановлена версія програми через інсталятор.

Для роботи системи не є обов'язковою наявність дискретної відеокарти, оскільки аналіз виконується на вже навчений моделі, а не на етапі її навчання. Це означає, що програму можна запускати на звичайному ноутбукі або персональному комп'ютері. Водночас для навчання моделі нейронної мережі бажано використовувати продуктивніше обладнання, оскільки навчання потребує значно більших обчислювальних ресурсів.

Окрему увагу слід приділити розміщенню файлів системи. Для коректної роботи програми в одній директорії мають бути розташовані основний файл застосунку, файл навченої моделі, тестові дані та база даних. Якщо база даних відсутня, програма створює її автоматично під час запуску. Якщо файл моделі

або тестові дані відсутні, користувач може вибрати їх вручну через графічний інтерфейс.

Під час експлуатації системи користувач повинен дотримуватися такого порядку роботи:

- запустити програму;
- завантажити модель нейронної мережі;
- ввести дані пацієнта;
- вибрати тестовий ЕКГ-запис;
- виконати аналіз;
- переглянути результат;
- за потреби зберегти результат у базу даних;
- переглянути базу даних або сформувати звіт.

Таким чином, розроблена програмна система має помірні апаратні та програмні вимоги. Вона може працювати на звичайному персональному комп'ютері або ноутбукі під керуванням Windows. Використання локальної бази SQLite, графічного інтерфейсу Tkinter і готової моделі TensorFlow/Keras дозволяє забезпечити просте встановлення, автономну роботу та зручну експлуатацію програмного забезпечення.

4.3 Склад інсталяційного пакету

Для впровадження розробленої програмної системи було підготовлено інсталяційний пакет, який дозволяє встановити застосунок на персональний комп'ютер користувача. Наявність інсталяційного пакету спрощує запуск програми, оскільки користувачу не потрібно вручну відкривати середовище розробки або налаштовувати структуру проєкту. Після встановлення система може запускатися як звичайний настільний застосунок.

Інсталяційний пакет програмної системи аналізу кардіосигналів містить основні файли, необхідні для роботи програми: виконуваний файл застосунку,

файл навченої моделі, тестові дані, файл бази даних SQLite та допоміжні файли, які забезпечують запуск системи.

 best_cnn_lstm_ptbxxl.keras	01.04.2026 19:07	Файл KERAS	1 165 КБ
 ecg_analysis.db	19.04.2026 18:54	Data Base File	12 КБ
 ECGAnalysisApp.exe	19.04.2026 19:54	Застосунок	470 604 КБ
 unins000.dat	11.05.2026 21:18	Файл DAT	2 КБ
 unins000.exe	11.05.2026 21:17	Застосунок	4 347 КБ
 X_test.npy	01.04.2026 18:57	Файл NPY	101 391 КБ
 y_test.npy	01.04.2026 18:57	Файл NPY	85 КБ

Рисунок 4.12 – Інсталяційний пакет програмної системи

До складу інсталяційного пакету входять такі основні компоненти:

- ECGAnalysisApp.exe — виконуваний файл програмної системи;
- best_cnn_lstm_ptbxxl.keras — файл навченої моделі CNN-LSTM;
- X_test.npy — файл тестових ЕКГ-сигналів;
- y_test.npy — файл істинних міток тестових записів;
- ecg_analysis.db — локальна база даних SQLite;
- Setup.exe — інсталяційний файл для встановлення програми;
- службові файли, необхідні для запуску застосунку.

Основним файлом застосунку є виконуваний файл ECGAnalysisApp.exe. Він створений на основі Python-файлу ecg_gui_app.py, у якому реалізовано графічний інтерфейс, завантаження моделі, аналіз кардіосигналів, роботу з базою даних і формування звітів. Після запуску виконуваного файлу відкривається головне вікно програми, через яке користувач може виконувати всі основні дії.

Файл best_cnn_lstm_ptbxxl.keras містить навчену модель нейронної мережі. Цей файл необхідний для виконання класифікації ЕКГ-сигналів. У програмі користувач може вказати шлях до файлу моделі через інтерфейс або використати стандартне значення, яке встановлене в полі завантаження моделі.

Файли X_test.npy та y_test.npy використовуються для тестування роботи системи. Файл X_test.npy містить ЕКГ-сигнали, які подаються на вхід моделі, а

файл `y_test.npy` містить істинні мітки класів, що дозволяє порівнювати результат прогнозування моделі з очікуваними значеннями.

Файл `ecg_analysis.db` є локальною базою даних SQLite. У ньому зберігаються результати аналізу кардіосигналів, зокрема ПІБ пацієнта, вік, дата аналізу, результат класифікації, визначена патологія, загальний висновок, ймовірність норми та патології. Якщо файл бази даних відсутній, програма створює його автоматично під час запуску.

Для створення виконуваного файлу використовується PyInstaller. Він дозволяє перетворити Python-застосунок на .exe-файл, який можна запускати на комп'ютері користувача без відкриття коду програми. На основі цього виконуваного файлу було сформовано інсталяційний пакет за допомогою Inno Setup[18]. Такий підхід дозволяє встановлювати програму як звичайний настільний застосунок Windows.

Після встановлення програмної системи користувач запускає застосунок через створений ярлик або виконуваний файл. Усі необхідні компоненти розміщуються у каталозі встановлення програми, тому користувачу не потрібно вручну запускати Python-файл або виконувати додаткові команди. Такий підхід спрощує експлуатацію системи та робить її зручною для кінцевого користувача.

Для коректної роботи встановленої системи бажано, щоб у каталозі програми були доступні файл моделі, тестові дані та база даних. Якщо тестові дані або модель розташовані в іншому місці, користувач може вибрати їх вручну через кнопки «Огляд» у графічному інтерфейсі.

Склад інсталяційного пакету наведено в таблиці 4.1.

Компонент	Призначення
<code>ECGAnalysisApp.exe</code>	Основний виконуваний файл програми
<code>best_cnn_lstm_ptbxl.keras</code>	Навчена модель CNN-LSTM для класифікації ЕКГ
<code>X_test.npy</code>	Тестові ЕКГ-сигнали
<code>y_test.npy</code>	Істинні мітки тестових записів
<code>ecg_analysis.db</code>	Локальна база даних SQLite

Setup.exe	Файл інсталятора програмної системи
Службові файли	Компоненти, необхідні для запуску застосунку

Інсталяційний пакет забезпечує просте встановлення програмної системи на комп'ютер користувача. Після встановлення користувач отримує готовий до роботи застосунок для аналізу кардіосигналів, перегляду графіка ЕКГ, отримання результатів класифікації, збереження їх у базу даних та формування звітно-статистичної інформації.

Таким чином, склад інсталяційного пакету повністю відповідає функціональним потребам розробленої системи. Він містить основний застосунок, модель нейронної мережі, тестові дані та локальну базу даних. Це дозволяє використовувати систему як у режимі демонстрації та тестування, так і у вигляді встановленої настільної програми.

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи було розроблено програмне забезпечення для аналізу кардіосигналів на основі згорткових нейронних мереж. Створена система забезпечує автоматизоване завантаження ЕКГ-записів, побудову графіка кардіосигналу, класифікацію сигналу за допомогою навченої моделі CNN-LSTM, формування висновку про стан пацієнта, збереження результатів у базу даних SQLite та формування звітної-статистичної інформації.

У першому розділі було проведено аналіз предметної області та визначено постановку завдання. Було встановлено, що автоматизація аналізу ЕКГ-сигналів є актуальною задачею, оскільки ручна обробка кардіосигналів потребує часу та спеціальної кваліфікації. У межах роботи визначено основні функціональні можливості системи: завантаження моделі, вибір тестового запису, введення даних пацієнта, класифікація сигналу, відображення результату, збереження даних у базу та формування звітів. Також було побудовано UML-діаграми, які описують взаємодію користувача із системою, послідовність виконання аналізу та загальний алгоритм роботи програмного забезпечення.

У другому розділі було розглянуто інформаційне забезпечення системи. Було побудовано логічну модель даних, яка описує інформаційні об'єкти, необхідні для роботи системи. Для реалізації інформаційної бази обрано SQLite, оскільки ця система управління базами даних не потребує окремого сервера, зберігає дані в одному локальному файлі та добре інтегрується з Python. У фактичній реалізації база даних представлена файлом `ecg_analysis.db` і таблицею `analysis_results`, у якій зберігаються дані пацієнта, результат аналізу, визначена патологія, ймовірності, істинні та передбачені мітки, дата аналізу й інформація про використану модель.

У третьому розділі було описано прикладне програмне забезпечення системи. Розроблена програма реалізована мовою Python із використанням бібліотек Tkinter, TensorFlow/Keras, NumPy, Matplotlib, SQLite3 та CSV. Було обґрунтовано вибір інструментальних засобів, описано організаційну структуру програмного забезпечення та основні модулі системи. До них належать модуль

графічного інтерфейсу, модуль завантаження моделі, модуль роботи з ЕКГ-даними, модуль класифікації, модуль візуалізації, модуль роботи з базою даних і модуль формування звітів. Також було наведено алгоритми обробки даних і формування звітно-статистичної інформації.

У четвертому розділі було розглянуто питання впровадження та експлуатації системи. Проведено тестування основних функцій програмного забезпечення: запуску програми, завантаження моделі, завантаження ЕКГ-запису, побудови графіка, виконання класифікації, збереження результату в базу даних, перегляду записів, формування звітів та експорту статистичної інформації у CSV-файл. Результати тестування показали, що система коректно виконує основні функції та обробляє помилкові дії користувача шляхом виведення відповідних повідомлень. Також було визначено апаратні й програмні вимоги до роботи системи та описано склад інсталяційного пакету.

Практична цінність розробленої системи полягає в тому, що вона поєднує засоби машинного навчання, графічний інтерфейс користувача, локальну базу даних і модуль звітності. Користувач може завантажити ЕКГ-запис, отримати результат класифікації, переглянути ймовірність норми або патології, зберегти результат у базу даних і сформувати статистичний звіт. Це робить програмне забезпечення зручним для демонстрації, навчального використання та подальших досліджень у сфері автоматизованого аналізу кардіосигналів.

Негативним результатом роботи є те, що система орієнтована на аналіз підготовлених тестових даних і не виконує пряме підключення до реального медичного обладнання. Крім того, отримані результати не можуть використовуватися як самостійний медичний діагноз, а мають розглядатися як допоміжна інформація для попереднього аналізу. У подальшому систему можна вдосконалити шляхом додавання підтримки завантаження зовнішніх ЕКГ-файлів, розширення бази патологій, покращення точності моделі, додавання авторизації користувачів та інтеграції з медичними інформаційними системами.

Таким чином, поставлена мета кваліфікаційної роботи була досягнута. Було створено програмне забезпечення для аналізу кардіосигналів на основі згорткових нейронних мереж, яке забезпечує візуалізацію ЕКГ, класифікацію

сигналу, визначення норми або патології, збереження результатів у базу даних SQLite та формування звітно-статистичної інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Wagner P., Strodthoff N., Bousseljot R.-D., Kreiseler D., Lunze F. I., Samek W., Schaeffter T. PTB-XL, a large publicly available electrocardiography dataset. *Scientific Data*. 2020. Vol. 7, Article number: 154. DOI: 10.1038/s41597-020-0495-6. URL: <https://www.nature.com/articles/s41597-020-0495-6>
2. PTB-XL, a large publicly available electrocardiography dataset : PhysioNet. URL: <https://physionet.org/content/ptb-xl/>
3. Moody G. B., Mark R. G. The impact of the MIT-BIH Arrhythmia Database. *IEEE Engineering in Medicine and Biology Magazine*. 2001. Vol. 20, Issue 3. P. 45–50. DOI: 10.1109/51.932724. URL: <https://physionet.org/content/mitdb/>
4. WFDB Python Package : офіційна документація. URL: <https://wfdb.readthedocs.io/>
5. Keras: The high-level API for TensorFlow : офіційна документація TensorFlow. URL: <https://www.tensorflow.org/guide/keras>
6. SQLite Documentation : офіційна документація SQLite. URL: <https://www.sqlite.org/docs.html>
7. sqlite3 — DB-API 2.0 interface for SQLite databases : офіційна документація Python. URL: <https://docs.python.org/3/library/sqlite3.html>
8. csv — CSV File Reading and Writing : офіційна документація Python. URL: <https://docs.python.org/3/library/csv.html>
9. Python 3 Documentation : офіційна документація Python. URL: <https://docs.python.org/3/>
10. tkinter — Python interface to Tcl/Tk : офіційна документація Python. URL: <https://docs.python.org/3/library/tkinter.html>
11. NumPy Documentation : офіційна документація бібліотеки NumPy. URL: <https://numpy.org/doc/>
12. Conv1D layer : офіційна документація Keras. URL: https://keras.io/api/layers/convolution_layers/convolution1d/
13. LSTM layer : офіційна документація Keras. URL: https://keras.io/api/layers/recurrent_layers/lstm/

14. Model saving & serialization : офіційна документація Keras. URL: https://keras.io/guides/serialization_and_saving/
15. Matplotlib Documentation : офіційна документація бібліотеки Matplotlib. URL: <https://matplotlib.org/stable/index.html>
16. PyInstaller Manual : офіційна документація PyInstaller. URL: <https://pyinstaller.org/>
17. Авраменко А. С., Авраменко В. С., Косенюк Г. В. Тестування програмного забезпечення : навчальний посібник. Черкаси : ЧНУ імені Богдана Хмельницького, 2017.
18. Inno Setup Documentation : офіційна документація Inno Setup. URL: <https://jrsoftware.org/ishelp/>

ДОДАТОК А
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Гордашевський Василь Васильович, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проект) на тему «Система аналізу кардіосигналів на основі згорткових нейронних мереж» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
 - ☐ інструменти генеративного ШІ не використовувалися.
 - ☒ інструменти ШІ (вказати назву: ChatGPT, Gemini, Claude, DeepL, _____ інші (вказати назву)) були використані для:
 - ☒ перекладу іншомовних джерел;
 - ☒ стилістичного редагування та перевірки граматики;
 - ☒ допомоги у написанні/відлагодженні програмного коду;
 - ☐ інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

« ____ » _____ 2026 р. _____ **Василь ГОРДАШЕВСЬКИЙ**
(підпис)