МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«___» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

Інтелектуальна система раннього виявлення ризику діабету

Спеціальність 121 Інженерія програмного забезпечення
Освітня програма «Інженерія програмного забезпечення»

Здобувачка _____ **Жанна РУДЕНКО**

«__» _____ 20__ р.

Керівник роботи
канд. техн. наук,
доцент _____ **Євген ДАВИДЕНКО**

«__» _____ 20__ р.

Миколаїв – 2025

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Другий (магістерський)
Освітній ступінь	Магістр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген Давиденко
«___» _____ 2025 р.

ЗАВДАННЯ
на кваліфікаційну магістерську роботу здобувача
Руденко Жанна

1. Тема кваліфікаційної роботи «Інтелектуальна система раннього виявлення ризику діабету» затверджена наказом ректора ЧНУ ім. Петра Могили № 182 від «02» липня 2025 р.
2. Строк представлення кваліфікаційної роботи «22» грудня 2025 р.
3. Очікуваний результат роботи та початкові дані якщо такі потрібні.

Очікуваним результатом інтелектуальна система раннього виявлення ризику діабету.

4. Перелік питань, що підлягають розробці:

– огляд і аналіз предметної області раннього виявлення цукрового діабету;

– аналіз та вибір алгоритмів машинного навчання та інтелектуального аналізу даних, які можуть бути застосовані для прогнозування ризику діабету;

– побудова концептуальної моделі інтелектуальної системи оцінювання ризику;

– розробка та впровадження системи з модулем прогнозування;

– тестування системи та оцінка точності її роботи.

5. Перелік графічних матеріалів:

Презентація

6. Консультанти:

Консультант	**Кафедра (організація)**	**Частина роботи**

Дата видачі завдання «____» _____ 20___ р.

Календарний план виконання кваліфікаційної роботи

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: **Інтелектуальна система раннього виявлення ризику діабету**

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КМР	01.09.2025	08.09.2025	Виконано
2.	Огляд літератури за темою роботи	09.09.2025	16.09.2025	Виконано
3.	Складання календарного плану КМР	17.09.2025	24.09.2025	Виконано
4.	Аналіз предметної області	25.09.2025	06.10.2025	Виконано
5.	Розробка проєктних рішень	27.10.2025	13.10.2025	Виконано
6.	Моделювання та конструювання ПЗ	14.10.2025	21.10.2025	Виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	22.10.2025	01.11.2025	Виконано
8.	Відгук керівника КМР	02.11.2025	03.11.2025	Виконано
9.	Оформлення КМР та презентації	04.11.2025	21.11.2025	Виконано
10.	Попередній захист	24.11.2025	24.11.2025	Виконано
11.	Рецензування	01.12.2025	08.12.2025	Виконано
12.	Завершення оформлення КМР та презентації	09.12.2025	19.12.2025	Виконано
13.	Захист кваліфікаційної роботи	22.12.2025	23.12.2025	Виконано

Здобувачка _____ **Жанна РУДЕНКО**

«__» _____ 20__ р.

Керівник роботи

канд. техн. наук,

доцент _____ **Євген ДАВИДЕНКО**

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної магістерської роботи

«Інтелектуальна система раннього виявлення ризику діабету»

Здобувачка 608д гр.: Руденко Жанна

Керівник: канд. техн. наук, доцент Давиденко Євген

Кваліфікаційна магістерська робота присвячена розробці інтелектуальної системи для прогнозування ризику розвитку цукрового діабету II типу з використанням методів машинного навчання. Система забезпечує введення медичних показників користувача, обчислення ймовірності розвитку захворювання за допомогою моделі TensorFlow.js, візуалізацію результатів, ведення історії та моніторинг рівня глюкози.

Об'єктом дослідження є процес раннього виявлення ризику розвитку цукрового діабету.

Предметом дослідження є методи прогнозування ризику розвитку цукрового діабету.

Метою роботи є рання діагностика ризику розвитку цукрового діабету за рахунок розробки інтелектуальної системи з використанням методів машинного навчання.

Для досягнення визначеної мети необхідно вирішити такі завдання:

— провести аналіз предметної галузі та сучасних методів ранньої діагностики цукрового діабету;

— дослідити алгоритми машинного навчання та інтелектуального аналізу даних, що можуть бути застосовані для прогнозування ризику діабету;

— розробити концептуальну модель інтелектуальної системи раннього виявлення ризику діабету;

— реалізувати програмне забезпечення на Angular із використанням TensorFlow для обробки вхідних даних та прогнозування ризику;

— провести тестування та оцінку точності роботи системи.

Структура кваліфікаційної магістерської роботи включає вступ, чотири розділи, висновки, список джерел та додатки. У вступі обґрунтовано актуальність теми, визначено мету, завдання, об'єкт і предмет дослідження.

У першому розділі виконано огляд сучасних досліджень і рішень у сфері прогнозування діабету та розглянуто застосування штучного інтелекту в медицині.

У другому розділі побудовано логічну, структурну та функціональну моделі системи, розроблено діаграми використання, діяльності та потоків даних.

У третьому розділі описано архітектуру системи, стек технологій, реалізацію основних компонентів системи та інтеграцію моделі TensorFlow.js. У четвертому розділі наведено процес тестування, результати апробації, аналіз якості та надійності розробленого програмного забезпечення.

У висновках наведено узагальнення отриманих результатів, сформульовано висновки щодо ефективності системи та перспектив подальшого розвитку.

Кваліфікаційна магістерська робота викладена на 105 сторінках, містить 4 розділи, 38 ілюстрацій, 5 таблиць, 30 джерел у переліку посилань.

Ключові слова: інтелектуальна система, діабет, TensorFlow.js, Angular, машинне навчання, прогнозування, Firebase, медична інформатика.

ABSTRACT

to the qualifying master's thesis

"Intelligent System for Early Detection of Diabetes Risk"

Student of group 608d: Zhanna Rudenko

Supervisor: Ph.D. in Engineering, Associate Professor Yevhen Davydenko

The qualifying master's thesis is devoted to the development of a system for predicting the risk of type 2 diabetes using machine learning methods. The system provides user input of medical indicators, calculates the probability of disease development using a TensorFlow.js model, visualizes results, maintains prediction history, and monitors glucose levels.

The object of the research is the process of early detection of diabetes risk.

The subject of the study is methods for predicting the risk of developing diabetes.

The aim of the work is the early diagnosis of the risk of developing diabetes through the development of an intelligent system using machine learning methods.

To achieve this goal, the following tasks were defined:

– to analyze the domain area and modern methods of early diabetes diagnosis;

– to study machine learning and data mining algorithms applicable to diabetes risk prediction;

– to develop a conceptual model of an intelligent system for early diabetes risk detection;

– to implement software using Angular and TensorFlow for data processing and risk prediction;

– to perform testing and accuracy evaluation of the system using real or synthetic medical data.

The structure of the master's thesis includes an introduction, four chapters, conclusions, a list of references, and appendices.

The **introduction** substantiates the relevance of the topic and defines the purpose, objectives, object, and subject of the research.

The **first chapter** provides an overview of current research and existing solutions in the field of diabetes prediction and discusses the application of artificial intelligence in medicine.

The **second chapter** presents the logical, structural, and functional models of the system, as well as the development of use case, activity, and data flow diagrams.

The **third chapter** describes the architecture of the system, the technology stack, the implementation of key system components, and the integration of the TensorFlow.js model.

The **fourth chapter** outlines the testing process, the results of validation, and an analysis of the quality and reliability of the developed software.

The **conclusions** summarize the obtained results and evaluate the effectiveness of the developed system, as well as prospects for further development.

The qualifying master's thesis consists of 105 pages, includes 4 chapters, 38 figures, 5 tables, and 30 references.

Keywords: *intelligent system, diabetes, TensorFlow.js, Angular, machine learning, prediction, Firebase, medical informatics.*

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

AI – Artificial Intelligence

AUC – Area Under Curve

BMI – Body Mass Index

DFD – Data Flow Diagram

DI – Dependency Injection

DOI – Digital Object Identifier

GPU – Graphics Processing Unit

HbA1c – Glycated Hemoglobin

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

JSON – JavaScript Object Notation

ML – Machine Learning

NN – Neural Network

PNG – Portable Network Graphics

RxJS – Reactive Extensions for JavaScript

SPA – Single Page Application

TF.js – TensorFlow.js

UI – User Interface

URL – Uniform Resource Locator

UX – User Experience

WHO – World Health Organization

КМР – кваліфікаційна магістерська робота

ПЗ – програмне забезпечення

ІС – інтелектуальна система

ВСТУП

Сучасний розвиток медицини нерозривно пов’язаний з активним впровадженням інформаційних технологій, які дозволяють підвищувати ефективність діагностики, лікування та профілактики різних захворювань. Однією з найбільш актуальних проблем сучасної охорони здоров’я є цукровий діабет – хронічне ендокринне захворювання, яке характеризується порушенням обміну глюкози в організмі. За даними Всесвітньої організації охорони здоров’я, кількість хворих на діабет невпинно зростає і вже перевищує 500 мільйонів осіб у світі. Прогнози свідчать, що у найближчі десятиліття ця цифра може подвоїтися. Особливу небезпеку становить той факт, що значна частина випадків захворювання діагностується на пізніх стадіях, коли вже виникають ускладнення.

У зв’язку з цим особливого значення набувають системи раннього виявлення ризику розвитку діабету, які дозволяють своєчасно визначати групи підвищеного ризику та застосовувати профілактичні заходи. Традиційні методи скринінгу часто вимагають значних матеріальних та часових витрат, тоді як інтелектуальні інформаційні системи на основі методів штучного інтелекту і машинного навчання забезпечують автоматизацію процесів аналізу медичних даних, підвищують точність прогнозів та дозволяють мінімізувати людський фактор.

Науково-практичне значення теми полягає в тому, що розробка інтелектуальної системи раннього виявлення ризику діабету сприятиме підвищенню якості медичних послуг, зниженню навантаження на лікарів та медичний персонал, а також дозволить зменшити витрати на лікування за рахунок своєчасної профілактики. Такі системи можуть застосовуватися як у медичних закладах, так і в рамках індивідуального моніторингу стану здоров’я пацієнтів.

Метою роботи є рання діагностика ризику розвитку цукрового діабету за рахунок розробки інтелектуальної системи з використанням методів машинного навчання.

Для досягнення визначеної мети необхідно вирішити такі завдання:

— провести аналіз предметної галузі та сучасних методів ранньої діагностики цукрового діабету;

— дослідити алгоритми машинного навчання та інтелектуального аналізу даних, що можуть бути застосовані для прогнозування ризику діабету;

— розробити концептуальну модель інтелектуальної системи раннього виявлення ризику діабету;

— реалізувати програмне забезпечення на Angular із використанням TensorFlow для обробки вхідних даних та прогнозування ризику;

— провести тестування та оцінку точності роботи системи.

Об’єктом дослідження є процес раннього виявлення ризику розвитку цукрового діабету.

Предметом дослідження є методи прогнозування ризику розвитку цукрового діабету.

Необхідність розробки зумовлена тим, що наявні рішення або дорогі й недоступні, або не забезпечують достатньої точності прогнозування. Хоча провідні університети та компанії активно створюють моделі для медичної діагностики, більшість із них потребують адаптації під локальні умови, зокрема для використання в Україні.

Сфера застосування результатів включає державні та приватні медичні заклади, лабораторії, центри сімейної медицини, а також персональні застосунки для пацієнтів, які прагнуть здійснювати моніторинг власного стану здоров’я.

Таким чином, обрана тема є актуальною, має як наукове, так і практичне значення, а результати дослідження можуть стати підґрунтям для подальшого розвитку систем підтримки прийняття рішень у медицині та підвищення ефективності профілактики цукрового діабету.

Апробація результатів КМР відбулась під час XXVII Всеукраїнської науково-практичної конференції «Могилянські читання – 2025», Миколаїв, 10-14 листопада, 2025 р. (Додаток А).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд предметної області

Діабет є однією з найбільш поширених і серйозних хронічних хвороб сучасності, що становить значну загрозу для здоров’я населення та розвитку систем охорони здоров’я. За даними Всесвітньої організації охорони здоров’я [1], кількість людей, які страждають на цю хворобу, постійно зростає, і вже сьогодні діабет визнаний глобальною епідемією. Особливо тривожним є той факт, що діабет часто виявляється на пізніх стадіях, коли профілактичні заходи стають менш ефективними, а лікування потребує значних ресурсів.

Важливим джерелом інформації є дані National Center for Health Statistics (NCHS), які відображають зміни у віково-скоригованій поширеності діабету серед дорослого населення США за період із 1999–2000 рр. до серпня 2021–2023 рр. Згідно з результатами дослідження:

— поширеність усього діабету зросла з 9,7 % у 1999–2000 рр. до 14,3 % у 2021–2023 рр.;

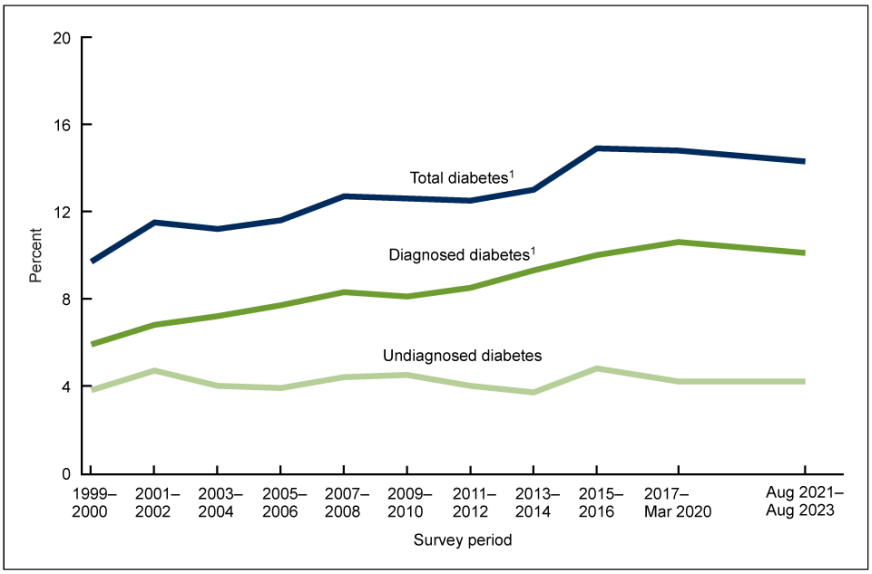
— поширеність діагностованого діабету збільшилася з 5,9 % до 10,1 % за той самий період;

— рівень недіагностованого діабету істотно не змінився, залишаючись на рівні близько 4 %.

Особливу увагу привертає той факт, що в останні роки (між 2017–березень 2020 рр. і серпнем 2021–2023 рр.) суттєвих змін у загальній поширеності не зафіксовано: показники для всіх категорій залишилися приблизно стабільними [2].

На рисунку 1.1 зображено тенденції поширеностданіі загального, діагностованого та недіагностованого діабету серед дорослих віком від 20 років і старше у США [3]. Графік наочно демонструє зростання частки загального та діагностованого діабету протягом останніх двох десятиліть при відносно стабільному рівні недіагностованих випадків.

Figure 5. Trends in age-adjusted prevalence of total, diagnosed, and undiagnosed diabetes in adults age 20 and older: United States, 1999–2000 through August 2021–August 2023



[1]Significant increasing linear trend ($p < 0.05$).
NOTES: Fasting glucose values were adjusted using forward regression equations provided by the National Center for Health Statistics. Estimates for diagnosed diabetes are based on responses to the survey question, "Other than during pregnancy, have you ever been told by a doctor or health professional that you have diabetes or sugar diabetes?" Estimates for undiagnosed diabetes are based on an 8- to 24-hour fasting plasma glucose greater than or equal to 126 mg/dL or hemoglobin A1c greater than or equal to 6.5% in a participant who reported never receiving a diabetes diagnosis from a healthcare provider. Estimates are age adjusted by the direct method to the U.S. Census 2000 population using the age groups 20–39, 40–59, and 60 and older.
SOURCE: National Center for Health Statistics, National Health and Nutrition Examination Surveys, 1999–2000 through August 2021–August 2023.

Рисунок 1.1 – Тенденції віково-скоригованої поширеності діабету серед дорослого населення США у 1999–2000 – 2021–2023 рр.

Об’єктом дослідження у даній кваліфікаційній роботі є процеси прогнозування ймовірності розвитку діабету в людини на основі медичних та соціально-демографічних даних. Предметом дослідження виступають інформаційні технології та програмні засоби, що дозволяють автоматизувати цей процес із застосуванням сучасних методів штучного інтелекту.

Важливим завданням предметної області є створення інструментів для своєчасного виявлення груп ризику, що дає змогу знизити навантаження на медичні заклади та зменшити кількість випадків ускладнень. Прогнозування на основі цифрових технологій дозволяє не лише проводити масовий скринінг, але й індивідуалізувати профілактику та лікування.

По-перше, необхідно враховувати структурні особливості даних, що використовуються у прогнозуванні діабету. Серед ключових факторів можна виділити вік пацієнта, індекс маси тіла, рівень фізичної активності, наявність спадкової схильності, показники артеріального тиску та рівня глюкози в крові [4]. Комбінація цих параметрів дозволяє побудувати прогностичну модель, здатну з високою точністю визначати ймовірність розвитку захворювання.

По-друге, функціональні особливості предметної області визначаються специфікою використання програмного забезпечення. Система має бути орієнтована на різні категорії користувачів:

— лікарів, які отримують аналітичні дані для прийняття клінічних рішень;

— пацієнтів, яким система надає зручний інтерфейс для введення своїх параметрів та отримання прогнозу;

— дослідників, що можуть застосовувати систему для аналізу ефективності моделей прогнозування.

Третім аспектом є сучасний стан розвитку інформаційних технологій у цій галузі. Сьогодні існує широкий спектр рішень для прогнозування захворювань на основі штучного інтелекту. Зокрема, використовуються методи машинного навчання, нейронні мережі та системи на базі TensorFlow, які дозволяють швидко обробляти великі масиви даних і підвищувати точність прогнозів. Серед світових тенденцій виділяється перехід до вебзастосунків, які не потребують складного встановлення та можуть працювати безпосередньо в браузері. Це робить системи доступними широкому колу користувачів, включаючи тих, хто не має спеціальних технічних знань.

Окрім того, важливо враховувати особливості української медичної сфери. В умовах обмеженого фінансування та нестачі кадрів, автоматизовані системи прогнозування можуть стати важливим інструментом для підтримки лікарів первинної ланки. Це відповідає сучасним стратегіям цифрової трансформації медицини, які активно реалізуються в Україні.

Таким чином, предметна область, пов’язана з прогнозуванням діабету, характеризується високим рівнем актуальності, наявністю чітко визначених структурних та функціональних особливостей, а також динамічним розвитком інформаційних технологій, які забезпечують ефективність програмних рішень. Розробка вебзастосунку на основі Angular і TensorFlow дозволить поєднати зручність користування, високу точність прогнозів та доступність для широкої

аудиторії, що робить даний напрям дослідження перспективним як у науковому, так і в практичному вимірах.

1.2 Огляд аналогів

У світі активно розробляються інтелектуальні системи, які дозволяють здійснювати прогнозування ризику діабету з використанням сучасних методів штучного інтелекту та машинного навчання. У цьому підрозділі розглянемо кілька актуальних аналогів подібних систем, їхні архітектури, функції та особливості [5].

DRPM: Advanced Predictive Model for Early Diabetes Detection

Модель DRPM (рис. 1.2) була створена дослідниками Китайського університету традиційної медицини для вирішення завдання раннього виявлення діабету 2 типу. Система використовує дані клінічних обстежень і методи глибокого навчання. Її головна особливість полягає у поєднанні високої точності прогнозування з мінімально необхідним набором вхідних показників, що робить її практичною для застосування в масових скринінгових програмах [6].

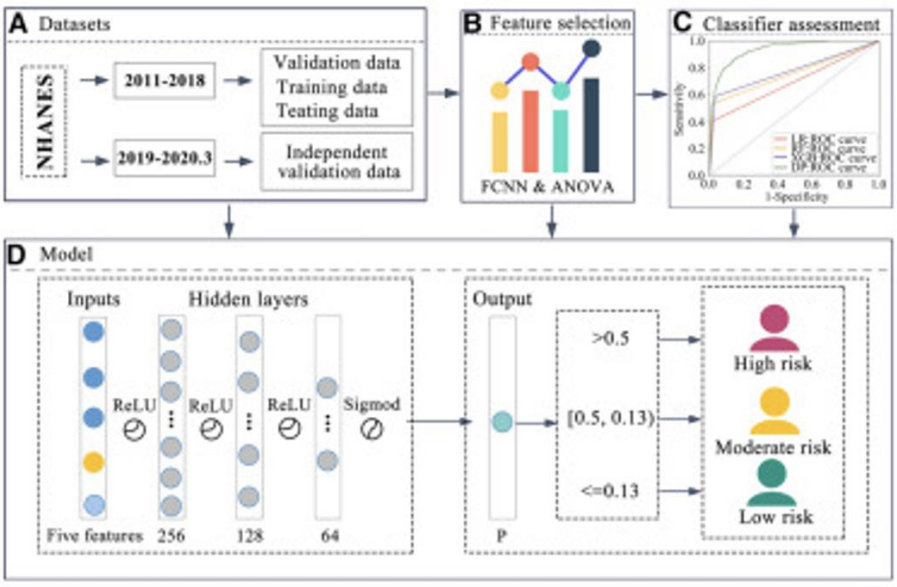


Рисунок 1.2 – Графічне представлення моделі DRPM

Таблиця 1.1 – Опис DRPM

Назва	**DRPM**
Архітектура	Глибоке навчання, алгоритми вибору ознак, аналіз даних фізичних оглядів
Виробник	Група дослідників під керівництвом Nie, Xiaoming Song, Wei Chen
Мова реалізації	Python (TensorFlow/PyTorch)
Функції	1) прогнозування ризику розвитку діабету II типу; 2) стратифікація пацієнтів за ризиком; 3) визначення найбільш важливих ознак (FBG, вік, WtHR, MSP, BMI).
Переваги	1) висока точність (AUC ≈ 0.94–0.96); 2) простота практичного застосування завдяки невеликій кількості параметрів; 3) можливість масового скринінгу.
Недоліки	1) залежність від конкретних даних; 2) обмежена інтерпретованість; 3) потреба перевірки на різних популяціях.
Посилання	DRPM

HealthEdge: Smart Healthcare Framework for Type 2 Diabetes Prediction

HealthEdge (рис. 1.3) – це інноваційна система, орієнтована на використання Інтернету речей, хмарних технологій та обчислень на периферії (Edge). Вона дозволяє збирати показники здоров’я у реальному часі, обробляти їх на локальних пристроях і передавати у хмару для поглибленого аналізу. Такий підхід робить систему придатною для персоналізованого моніторингу

пацієнтів, забезпечуючи швидкий зворотний зв'язок і потенційно високу зручність у використанні [7].

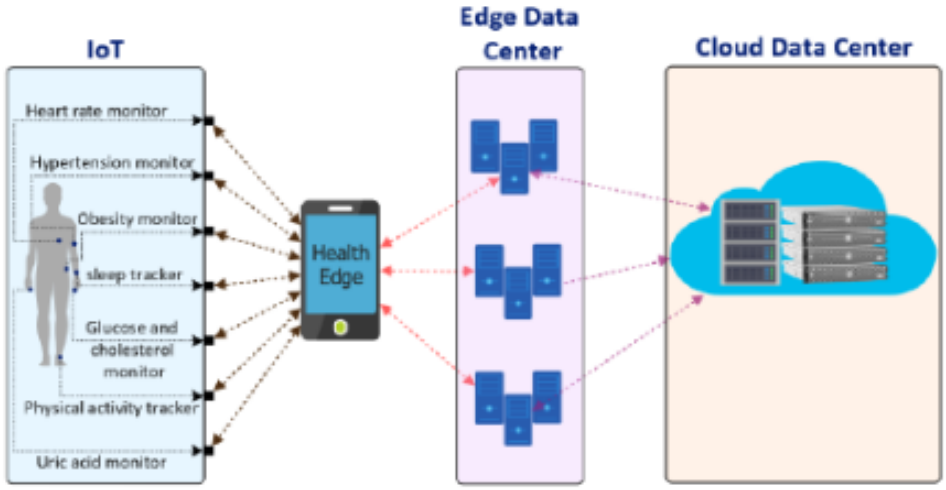


Рисунок 1.3 – Графічне представлення системи HealthEdge

Таблиця 1.2 – Опис HealthEdge

Назва	**HealthEdge**
Архітектура	IoT–Edge–Cloud інтегрована система, застосування Logistic Regression та Random Forest
Виробник	Alain Hennebelle, Huned Materwala, Leila Ismail
Мова реалізації	Python у поєднанні з технологіями для IoT та хмарних сервісів
Функції	1) збір медичних показників за допомогою сенсорів; 2) попередня обробка даних на Edge рівні; 3) прогнозування ризику діабету; 4) захист даних та можливість блокчейн-аутентифікації.

Кінець таблиці 1.2

Переваги	1) швидка обробка даних у реальному часі; 2) можливість застосування з носимими пристроями; 3) відносна простота моделей для інтерпретації результатів.
Недоліки	1) висока вартість інфраструктури (IoT + Edge + Cloud); 2) обмежені ресурси на периферійних пристроях; 3) нижча точність у порівнянні з сучасними нейромережевими підходами.
Посилання	HealthEdge

AIRE-DM: AI-ECG Risk Estimation for Diabetes Mellitus

AIRE-DM – система, розроблена в Imperial College London та Imperial College Healthcare NHS Trust, яка використовує аналіз електрокардіограм (ECG) з метою передбачення ризику розвитку цукрового діабету 2 типу за багато років до появи клінічних симптомів. Система орієнтована на використання вже наявних рутинних ECG-знімків, що робить її потенційно дешевою та неінвазійною – дуже корисною для масового скринінгу чи інтеграції в первинну медичну допомогу [8].

Таблиця 1.3 – Опис AIRE-DM

Назва	**AIRE-DM**
Архітектура	Моделі машинного та глибинного навчання для аналізу ECG

Кінець таблиці 1.3

Виробник	Imperial College London та Imperial College Healthcare NHS Trust
Мова реалізації	Python
Функції	1) аналіз рутинних ECG-записів для оцінки майбутнього ризику діабету; 2) прогнозування розвитку хвороби задовго до появи стандартних клінічних ознак; 3) комбінування результатів ECG із клінічними та генетичними даними для підвищення точності.
Переваги	1) неінвазивність та доступність (ECG робиться практично в кожній клініці); 2) раннє виявлення, що дає можливість профілактики; 3) велика база даних для навчання моделі, наявність зовнішньої валідації.
Недоліки	1) точність близько 70 %, що нижче порівняно з деякими іншими моделями; 2) перебуває у стадії досліджень, ще не впроваджена у медичну практику; 3) залежність від якості ECG-сигналу та умов збору даних.
Посилання	AIRE-DM

Таким чином, можна зробити висновок, що універсального рішення наразі не існує: кожна з розглянутих систем має власні сильні та слабкі сторони. Для популяційних обстежень доцільно застосовувати моделі типу DRPM, для персоналізованого моніторингу – рішення на зразок HealthEdge, а для раннього скринінгу з мінімальним втручанням перспективним є підхід AIRE-DM. У майбутньому найбільш ефективними можуть стати гібридні системи, що поєднуватимуть сильні сторони різних підходів.

Переваги системи на базі Angular та TensorFlow у порівнянні з аналогами:

1) зручність використання – на відміну від DRPM та AIRE-DM, система має готовий вебінтерфейс (Angular), доступний лікарям і пацієнтам без додаткового ПЗ;

2) гнучкість – TensorFlow дозволяє легко оновлювати й донавчати модель;

3) низькі вимоги до обладнання – працює на звичайних ПК чи у хмарі, без IoT-інфраструктури (як HealthEdge) чи спеціальних пристроїв (як AIRE-DM);

4) масштабованість – легке розгортання через веб робить систему придатною для широкого використання.

1.3 Аналіз системи, що розробляється

Розробка призначена для створення інтелектуальної системи, яка дозволяє своєчасно оцінювати ризик розвитку ЦД2 на основі введених користувачем даних. Основною особливістю системи є застосування методів машинного навчання: усі обчислення виконуються у браузері за допомогою TensorFlow.js, а інтерфейс користувача реалізовано на Angular.

Аналіз системи включає в себе огляд існуючих проблем у сфері ранньої діагностики діабету та визначення ключових вимог, які мають бути враховані при розробці. Основними проблемами є відсутність простих у користуванні онлайн-інструментів, складність доступу до професійних діагностичних систем для широкого кола користувачів.

Таблиця 1.4 – Аналіз системи

Пункт аналізу	**Опис**
Функції	1) введення даних – користувач вводить основні медичні показники (вік, вага, зріст, ІМТ, глюкоза тощо); 2) прогноз ризику – модель TensorFlow.js аналізує дані та формує результат; 3) візуалізація – відображення прогнозу у вигляді графіків та індикаторів; 4) історія результатів – збереження попередніх прогнозів у локальному сховищі; 5) трекер глюкози – фіксація змін рівня глюкози, побудова графіків і надання порад.
Користувачі	1) пацієнти, які бажають самостійно перевірити свій ризик захворювання; 2) лікарі, що використовують систему як допоміжний інструмент у профілактичних оглядах; 3) студенти та дослідники для навчальних і наукових цілей.

Кінець таблиці 1.4

Сценарії роботи	1) користувач відкриває вебзастосунок у браузері; 2) вводить особисті дані: вік, вага, зріст, рівень глюкози тощо; 3) система аналізує дані за допомогою моделі TensorFlow.js; 4) користувач отримує результат у вигляді прогнозу ризику та графічної візуалізації; 5) при бажанні користувач зберігає результати в історію. 6) користувач використовує трекер глюкози – вводить поточні вимірювання, переглядає динаміку рівня цукру на графіку та отримує автоматичні поради на основі змін показників.

Першочерговим завданням є визначення потреб користувачів: пацієнтів, які очікують простий та доступний інструмент для самоперевірки; лікарів, яким важлива швидка допоміжна оцінка стану пацієнта; та дослідників, що потребують інструмента для навчальних чи наукових цілей.

В результаті аналізу визначено основні цілі та завдання системи: забезпечення швидкого, зручного та безпечного інструменту для оцінки ризику діабету, що сприятиме підвищенню рівня профілактики та своєчасному виявленню захворювання.

1.4 Методи машинного навчання для прогнозування діабету

Використання машинного навчання у сфері медицини, зокрема для прогнозування цукрового діабету, набуває дедалі більшої актуальності. Це обумовлено потребою у ранньому виявленні ризиків, що дозволяє запобігти розвитку ускладнень і знизити навантаження на систему охорони здоров'я. [9]

Таблиця 1.5 – Основні підходи до застосування алгоритмів машинного навчання для прогнозування ризику діабету

Підхід	**Опис**
Класифікація	Класифікаційні алгоритми, такі як дерева рішень, наївний баєсівський класифікатор і метод опорних векторів (SVM), активно використовуються для визначення, чи належить пацієнт до групи ризику. Вони дозволяють класифікувати користувачів за категоріями (низький, середній або високий ризик), що дає можливість лікарям і пацієнтам вчасно вжити заходів профілактики.
Регресія	Регресійні моделі, зокрема лінійна та логістична регресія, застосовуються для прогнозування ймовірності розвитку діабету на основі кількісних показників (наприклад, рівня глюкози чи індексу маси тіла). Вони також дають змогу дослідити вплив окремих факторів ризику (вік, спадковість, спосіб життя) на розвиток хвороби.

Кінець таблиці 1.5

Аналіз часових рядів	У випадках, коли дані пацієнта збираються протягом певного періоду (наприклад, щоденні вимірювання рівня глюкози), застосовуються моделі аналізу часових рядів, такі як ARIMA або LSTM-мережі. Це дозволяє прогнозувати зміни стану пацієнта у майбутньому та виявляти небезпечні тренди.

Огляд найбільш ефективних моделей:

— **логістична регресія:** один із найпоширеніших методів у медичних дослідженнях, простий для інтерпретації, придатний для базових моделей ризику;

— **дерева рішень:** дає змогу будувати зрозумілі правила класифікації пацієнтів. Використовується для пояснюваних моделей;

— **нейронні мережі: з**абезпечують високу точність за умови наявності великих масивів даних, використовуються для виявлення складних взаємозв’язків між численними показниками;

— **метод випадкового лісу:** є надійним ансамблевим методом, що зменшує ризик переобучення і добре підходить для аналізу медичних даних з різних джерел;

— **глибоке навчання (Deep Learning):** застосовується для роботи з медичними зображеннями (наприклад, ретинальні знімки при діабетичній ретинопатії) та багатовимірними даними; особливо ефективними є рекурентні нейронні мережі (RNN) і згорткові нейронні мережі (CNN).

Переваги та виклики застосування

Переваги:

— автоматизація процесу аналізу даних та зменшення навантаження на лікарів;

— персоналізовані прогнози для конкретних пацієнтів;

— можливість виявлення прихованих закономірностей, які складно знайти традиційними методами;

— підвищення точності ранньої діагностики та своєчасне втручання.

Виклики:

— потреба у великих та якісних наборах медичних даних;

— складність інтерпретації результатів складних моделей (наприклад, нейронних мереж);

— етичні та правові питання, пов’язані із захистом персональних медичних даних;

— висока вартість впровадження та потреба у спеціалізованому програмному забезпеченні.

Застосування методів машинного навчання у прогнозуванні діабету відкриває нові можливості для медицини, зокрема для своєчасної діагностики та профілактики. Вибір конкретної моделі залежить від доступних даних, вимог до точності прогнозу та можливості інтерпретації результатів. Подальший розвиток цієї сфери сприятиме створенню ще більш ефективних інтелектуальних систем для підтримки лікарів і пацієнтів. [10]

Висновки до розділу 1

У підсумку можна зазначити, що проблема цукрового діабету залишається однією з найактуальніших у сфері охорони здоров’я, і ключовим завданням сучасної медицини є своєчасне виявлення груп ризику для запобігання розвитку ускладнень. Інтелектуальні системи, засновані на методах штучного інтелекту та машинного навчання, дозволяють автоматизувати процес аналізу медичних даних, підвищують точність прогнозів та мінімізують вплив людського фактору.

Розробка такої системи має значну наукову й практичну цінність, адже сприятиме підвищенню якості медичних послуг, зниженню навантаження на лікарів і витрат на лікування за рахунок своєчасної профілактики. Вона також має потенціал до масштабування й подальшої інтеграції з медичними інформаційними системами. Необхідність створення подібного рішення зумовлена тим, що більшість існуючих програмних продуктів є комерційними, обмеженими у доступності або не забезпечують достатньої точності прогнозування, тоді як адаптована до українських умов система дозволить зробити діагностику більш доступною та ефективною.

Крім того, впровадження подібних технологій сприятиме формуванню більш персоналізованої медицини, де рішення ухвалюються не лише на основі загальних статистичних даних, а й з урахуванням індивідуальних особливостей пацієнта. Це створює передумови для точнішого контролю стану здоров’я, раннього виявлення негативних тенденцій і своєчасного коригування лікування. Таким чином, розвиток систем прогнозування на основі штучного інтелекту є важливим кроком до модернізації медичної галузі та підвищення ефективності профілактики хронічних захворювань.

2 ПРОГНОЗУВАННЯ РИЗИКУ ДІАБЕТУ

2.1 Специфікація вимог

1. Призначення та межі проєкту

1.1. Призначення системи

Розроблюване програмне забезпечення призначене для збору, аналізу та прогнозування медичних показників користувачів із метою оцінки ризику розвитку цукрового діабету.

Система надає користувачу можливість вводити особисті параметри (вік, вага, зріст, ІМТ, рівень глюкози тощо), отримувати прогноз ризику за допомогою моделі TensorFlow.js, візуалізувати результати, вести історію прогнозів, а також відстежувати зміни рівня глюкози за допомогою трекера глюкози з автоматичними порадами.

1.2. Погодження

Проєкт розробляється відповідно до технічного завдання, погодженого керівником дипломного проєкту та відповідає вимогам стандартів оформлення програмної документації.

1.3. Межі проєкту

Система функціонує як вебзастосунок, який працює у браузері користувача. Дані зберігаються у хмарній базі даних Firebase Firestore. Серверна частина реалізується засобами Firebase. Застосунок не виконує медичної діагностики, а лише надає аналітичні прогнози та рекомендації інформаційного характеру.

2. Загальний опис

2.1. Сфера застосування

Програмне забезпечення використовується у сфері цифрової медицини та персональної аналітики здоров’я. Метою є допомога користувачу у моніторину показників, оцінці тенденцій зміни глюкози та попередженні можливих ризиків розвитку діабету.

2.2. Характеристики користувачів

Користувачі системи – це люди, які:

1) мають потребу контролювати власні медичні показники;
2) не мають спеціальних технічних або медичних знань;
3) володіють базовими навичками користування вебзастосунками.

2.3. Загальна структура і склад системи

Система складається з таких основних модулів:

1) модуль авторизації та реєстрації користувача;
2) модуль введення медичних показників;
3) модуль прогнозування ризику (TensorFlow.js);
4) модуль візуалізації результатів (Chart.js);
5) модуль історії прогнозів;
6) модуль трекера глюкози з порадами;
7) модуль експорту даних (CSV, JSON, PNG);
8) модуль збереження інформації у Firebase.

2.4. Загальні обмеження

1) застосунок працює у браузерах, що підтримують ECMAScript 2020 і WebGL;

2) потрібне активне підключення до Інтернету для синхронізації з Firebase;

3) система не призначена для медичного діагностування, а лише для аналітики.

3. Функції системи

3.1. Авторизація та реєстрація

3.1.1. Опис функції:

Забезпечення входу користувача через електронну пошту.

3.1.2. Вхідна/вихідна інформація:

Вхідна: email, пароль.

Вихідна: токен користувача, доступ до персональних даних у Firebase.

3.1.3. Функціональні вимоги:

1) забезпечити безпечне зберігання даних через Firebase Authentication;
2) підтримувати сесії користувача до виходу з системи.

3.2. Введення медичних показників

3.2.1. Опис функції:

Користувач заповнює форму з особистими параметрами (вік, вага, зріст, ІМТ, глюкоза, артеріальний тиск тощо).

3.2.2. Вхідна/вихідна інформація:

Вхідна: значення полів форми.

Вихідна: об’єкт даних для моделі прогнозування.

3.2.3. Функціональні вимоги:

1) поля повинні мати валідацію (мінімальні/максимальні значення);
2) після відправки дані передаються у TensorFlow.js для аналізу.

3.3. Прогнозування ризику

3.3.1. Опис функції:

Модуль використовує навчальну модель TensorFlow.js для прогнозування ризику діабету.

3.3.2. Вхідна/вихідна інформація:

Вхідна: дані користувача (медичні показники).

Вихідна: числове значення ймовірності ризику та категорія («низький», «середній», «високий»).

3.3.3. Функціональні вимоги:

1) модель має бути виконана локально у браузері;
2) виводити результат одразу після обчислення.

3.4. Візуалізація результатів

3.4.1. Опис функції:

Відображення результатів прогнозу у вигляді діаграм, індикаторів та текстового пояснення.

3.4.2. Вхідна/вихідна інформація:

Вхідна: прогнозні дані з моделі.

Вихідна: графічне представлення результатів.

3.4.3. Функціональні вимоги:

1) використати бібліотеку Chart.js для побудови діаграм;
2) адаптивний інтерфейс для різних пристроїв.

3.5. Історія прогнозів

3.5.1. Опис функції:

Збереження результатів прогнозів користувача у Firebase та можливість експорту у JSON або CSV.

3.5.2. Вхідна/вихідна інформація:

Вхідна: дані прогнозу.

Вихідна: список історії, файли експорту.

3.5.3. Функціональні вимоги:

1) зберігати дані для кожного користувача окремо;
2) забезпечити пошук і фільтрацію історії.

3.6. Трекер глюкози

3.6.1. Опис функції:

Користувач може вносити поточні значення глюкози, переглядати графік змін і отримувати поради.

3.6.2. Вхідна/вихідна інформація:

Вхідна: показники глюкози, час вимірювання.

Вихідна: графік зміни рівня глюкози, текстова порада.

3.6.3. Функціональні вимоги:

1) дані зберігаються у Firebase;
2) поради формуються на основі аналізу змін (тенденцій);
3) реалізувати експорт у JSON та CSV.

4. Вимоги до інформаційного забезпечення

4.1. Джерела і зміст вхідної інформації

Основні дані вводяться користувачем вручну. Додатково використовуються проміжні обчислення моделі TensorFlow.js.

4.2. Нормативно-довідкова інформація

Використовуються медичні референсні діапазони для оцінки норм та відхилень глюкози, ІМТ тощо.

4.3. Вимоги до способів збереження

Інформація зберігається у Firebase Firestore, з розділенням за користувачами (users/{uid}/predictions, users/{uid}/glucose).

5. Вимоги до технічного забезпечення

1) будь-який сучасний комп'ютер або смартфон із підтримкою браузера (Chrome, Edge, Firefox, Safari).

2) підключення до Інтернету.

6. Вимоги до програмного забезпечення

6.1. Архітектура системи

Клієнт-серверна SPA-архітектура з односторінковим інтерфейсом (Angular) і хмарною базою (Firebase).

6.2. Системне ПЗ

ОС користувача: Windows, macOS, Android або iOS.

6.3. Мережне ПЗ

HTTP(S)-з'єднання, API Firebase, REST-запити.

6.4. ПЗ ведення інформаційної бази

Firebase Firestore, Firebase Authentication.

6.5. Мова і технологія

TypeScript, Angular 17, HTML5, SCSS.

TensorFlow.js, Chart.js, Firebase SDK.

7. Вимоги до зовнішніх інтерфейсів

7.1. Інтерфейс користувача

Інтуїтивно зрозумілий, адаптивний дизайн із використанням Angular Material.

7.2. Апаратний інтерфейс

Не потребує додаткового обладнання.

7.3. Програмний інтерфейс

Використовує API Firebase для CRUD-операцій та TensorFlow.js API для інференсу моделі.

7.4. Комунікаційний протокол

HTTPS, WebSocket (при необхідності).

8. Властивості програмного забезпечення

Таблиця 2.1 – Властивості програмного забезпечення

Властивість	**Характеристика**
Доступність	Цілодобово, у браузері на будь-якому пристрої.
Супроводжуваність	Код розбитий на модулі Angular, легко оновлюється.
Переносимість	Працює у більшості сучасних браузерів.
Продуктивність	Швидкий аналіз даних у браузері (TensorFlow.js).
Надійність	Автоматичне збереження даних у Firebase.
Безпека	Авторизація через Firebase, доступ до даних лише власнику.

9. Інші вимоги

1) система має бути протестована на кросбраузерну сумісність.
2) інтерфейс – українською мовою.
3) код повинен відповідати стилю Angular і вимогам ESLint.

2.2 Опис набору даних та ознак

Набір даних для прогнозування діабету – це колекція медичних і демографічних даних пацієнтів, а також інформації про їхній діабетичний статус (позитивний або негативний). Дані включають такі характеристики, як вік, стать, індекс маси тіла (ІМТ), наявність гіпертонії та серцево-судинних захворювань, історія куріння, рівень глікованого гемоглобіну (HbA1c) та рівень глюкози в крові. Набір даних для прогнозування діабету було отримано з платформи **Kaggle** — відкритого онлайн-ресурсу, що надає доступ до великої кількості датасетів для аналізу даних і задач машинного навчання [11].

Цей набір даних може бути використаний для побудови моделей машинного навчання, які прогнозують ймовірність розвитку діабету у пацієнтів на основі їхніх медичних показників і демографічної інформації. Такі моделі можуть бути корисними для медичних працівників у виявленні пацієнтів із підвищеним ризиком, що сприятиме ранній діагностиці, індивідуальному підходу до лікування та профілактичним заходам. Крім того, набір даних може бути використаний дослідниками для вивчення зв'язків між різними факторами та ризиком виникнення діабету.

Датасет містить 1000 записів та 9 ознак.

Короткий опис ознак:

1) **Gender** – біологічна стать пацієнта (Male/Female), яка може впливати на ймовірність розвитку діабету;

2) **Age** – вік пацієнта (від 0 до 80 років); з віком ризик діабету зростає;

3) **Hypertension** – гіпертонія (0 – відсутня, 1 – наявна); високий тиск є фактором ризику;

4) **Heart_disease** – серцево-судинні захворювання (0 – немає, 1 – є); часто супроводжують або передують діабету;

5) **Smoking_history** – інформація про куріння (наприклад, *never*, *former*, *current*, *No Info*); куріння підвищує ризик ускладнень при діабеті;

6) **BMI** (*Body Mass Index*) – індекс маси тіла, що визначає наявність зайвої ваги або ожиріння:

- < 18.5 – недостатня вага;
- 18.5–24.9 – норма;
- 25–29.9 – надмірна вага;
- ≥30 – ожиріння.

7) **HbA1c_level** – рівень глікованого гемоглобіну (%), що показує середній рівень цукру в крові за останні 2–3 місяці. Значення >6.5% зазвичай свідчить про наявність діабету;

8) **Blood_glucose_level** – миттєвий рівень глюкози в крові; ключовий біомаркер для діагностики діабету;

9) **Diabetes** – цільова змінна: 0 – діабет відсутній, 1 – діабет наявний.

Типи ознак:

1) категоріальні: gender, smoking_history;
2) числові: age, bmi, HbA1c_level, blood_glucose_level;
3) бінарні: hypertension, heart_disease;
4) ціль: diabetes (0 – немає, 1 – є).

2.3 Первинний аналіз даних

На цьому етапі виконується первинне знайомство з датасетом для прогнозування діабету. Ми завантажимо дані, переглянемо основні характеристики та оцінимо структуру ознак. Для цього скористаємося базовими функціями бібліотек **pandas**, **numpy**, **seaborn** та **matplotlib** (рис. 2.1).

```
[1]: import pandas as pd
     import numpy as np
     import seaborn as sns
     import matplotlib.pyplot as plt

     %matplotlib inline
```

Рисунок 2.1 – Імпорт основних бібліотек

Спочатку переглянемо, як виглядають дані (рис. 2.2):

```
data = pd.read_csv("diabetes_prediction_dataset.csv")
data.head()
```

[4]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	80.0	0	1	never	25.19	6.6	140	0
1	Female	54.0	0	0	No Info	27.32	6.6	80	0
2	Male	28.0	0	0	never	27.32	5.7	158	0
3	Female	36.0	0	0	current	23.45	5.0	155	0
4	Male	76.0	1	1	current	20.14	4.8	155	0

Рисунок 2.2 – Первинні данні

Оцінимо розмірність датасету (рис. 2.3):

```
data = pd.read_csv("diabetes_prediction_dataset.csv")
data.shape
```

```
: (100000, 9)
```

Рисунок 2.3 – Розмірність датасету

Таким чином, датафрейм містить 10000 записів та 9 ознак. Поглянемо детальну інформацію про ознаки (рис. 2.4).

```
data = pd.read_csv("diabetes_prediction_dataset.csv")
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 9 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   gender               100000 non-null  object 
 1   age                  100000 non-null  float64
 2   hypertension         100000 non-null  int64  
 3   heart_disease        100000 non-null  int64  
 4   smoking_history      100000 non-null  object 
 5   bmi                  100000 non-null  float64
 6   HbA1c_level          100000 non-null  float64
 7   blood_glucose_level  100000 non-null  int64  
 8   diabetes             100000 non-null  int64  
dtypes: float64(3), int64(4), object(2)
memory usage: 6.9+ MB
```

Рисунок 2.4 – Детальна інформація про ознаки

Отже, після перегляду детальної інформації про ознаки можна визначити основні пункти:

1) усі стовпці заповнені, відсутні пропущені значення;

2) деякі ознаки мають категоріальний тип (gender, smoking_history) – їх потрібно буде кодувати на етапі передобробки;

3) цільова змінна – diabetes – є бінарною (0 або 1).

Подивимося на статистику даних (рис. 2.5).

```
data = pd.read_csv("diabetes_prediction_dataset.csv")
data.describe()
```

[7]:

	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	diabetes
count	100000.000000	100000.00000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000
mean	41.885856	0.07485	0.039420	27.320767	5.527507	138.058060	0.085000
std	22.516840	0.26315	0.194593	6.636783	1.070672	40.708136	0.278883
min	0.080000	0.00000	0.000000	10.010000	3.500000	80.000000	0.000000
25%	24.000000	0.00000	0.000000	23.630000	4.800000	100.000000	0.000000
50%	43.000000	0.00000	0.000000	27.320000	5.800000	140.000000	0.000000
75%	60.000000	0.00000	0.000000	29.580000	6.200000	159.000000	0.000000
max	80.000000	1.00000	1.000000	95.690000	9.000000	300.000000	1.000000

Рисунок 1.5 – Статистична інформація за первинними даними

На цьому етапі можна звернути увагу на наступне:

1) мінімальні та максимальні значення ознак, наприклад: age: від 0 до 80 років, bmi: від 10.16 до 71.55, HbA1c_level: варіанти > 6.5% вказують на діабет;

2) наявність потенційно **аномальних значень** (наприклад, вік = 0 або ІМТ > 60).

Проведемо аналіз демографічної ознаки «age» (рис. 2.6).

```
data = pd.read_csv("diabetes_prediction_dataset.csv")
data[['age']].describe()
```

[8]:

	age
count	100000.000000
mean	41.885856
std	22.516840
min	0.080000
25%	24.000000
50%	43.000000
75%	60.000000
max	80.000000

Рисунок 2.6 – Статистична інформація за віком

Вік варіюється в межах від **0 до 80 років**, середнє значення – близько **43 років**, що відповідає типовому віку для прояву діабету 2 типу. Виявлення пацієнтів з дуже низьким віком (0–5 років) може свідчити про наявність шумів у даних.

Розгянемо аналіз медичних кількісних ознак (рис. 2.7).

```
data = pd.read_csv("diabetes_prediction_dataset.csv")
data[['bmi', 'HbA1c_level', 'blood_glucose_level']].describe()
```

[9]:

	bmi	HbA1c_level	blood_glucose_level
count	100000.000000	100000.000000	100000.000000
mean	27.320767	5.527507	138.058060
std	6.636783	1.070672	40.708136
min	10.010000	3.500000	80.000000
25%	23.630000	4.800000	100.000000
50%	27.320000	5.800000	140.000000
75%	29.580000	6.200000	159.000000
max	95.690000	9.000000	300.000000

Рисунок 2.7 – Статистика за медичними кількісними ознаками

Після перегляду статистики за медичними кількісними ознаками можна звернути увагу на наступне:

– **індекс маси тіла (BMI)**: середнє значення перевищує 30, що вказує на тенденцію до ожиріння серед пацієнтів;

– **HbA1c_level**: значення вище 6.5% часто асоціюються з діабетом. Частина записів перевищує цей поріг;

– **blood_glucose_level**: деякі пацієнти мають значення понад 180 мг/дл – ознака неконтрольованого діабету.

Переглянемо частотний аналіз бінарних та категоріальних ознак (рис. 2.8).

```
data = pd.read_csv("diabetes_prediction_dataset.csv")
data['gender'].value_counts(normalize=True)
data['smoking_history'].value_counts(normalize=True)
data['hypertension'].value_counts(normalize=True)
data['heart_disease'].value_counts(normalize=True)
```

.0]:

```
heart_disease
0    0.96058
1    0.03942
Name: proportion, dtype: float64
```

Рисунок 2.8 – Частоти категоріальних ознак

Після перегляду частоти категоріальних ознак можна звернути увагу на наступне:

– стать майже збалансована, хоча може мати невелике зміщення;

– куріння представлено кількома категоріями: *never*, *current*, *former*, *No Info*;

– гіпертонія і захворювання серця зустрічаються рідше, але мають значний вплив на цільову змінну.

Висновки до статистичного аналізу:

1) демографічні та медичні ознаки мають реалістичні значення, проте потребують детальнішої візуалізації для виявлення потенційних аномалій;

2) ознаки bmi, HbA1c_level та blood_glucose_level демонструють значну варіативність і є перспективними для побудови моделей;

3) категоріальні ознаки не мають пропусків, але потребують кодування;

4) дані готові до етапу візуального аналізу та попередньої обробки.

2.4 Первинний візуальний аналіз даних

Для глибшого розуміння структури медичних даних, виявлення можливих залежностей між ознаками, а також підготовки до побудови моделей машинного навчання, було проведено первинний візуальний аналіз числових змінних. Це дозволяє виявити наявність кореляцій, аномалій, нелінійних зв’язків та потенціал до класифікації у зниженому просторі.

Для початку дослідимо лінійні залежності між числовими ознаками за допомогою коефіцієнта кореляції Пірсона (рис. 2.9) [12]. До аналізу включено такі змінні: age, hypertension, heart_disease, bmi, HbA1c_level, blood_glucose_level, diabetes.

```
[2]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline

data = pd.read_csv("diabetes_prediction_dataset.csv")
numeric_features = data[['age', 'hypertension', 'heart_disease', 'bmi', 'HbA1c_level', 'blood_glucose_level', 'diabetes']]
corr_matrix = numeric_features.corr()

plt.figure(figsize=(10, 7))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Кореляційна матриця числових ознак")
plt.show()
```

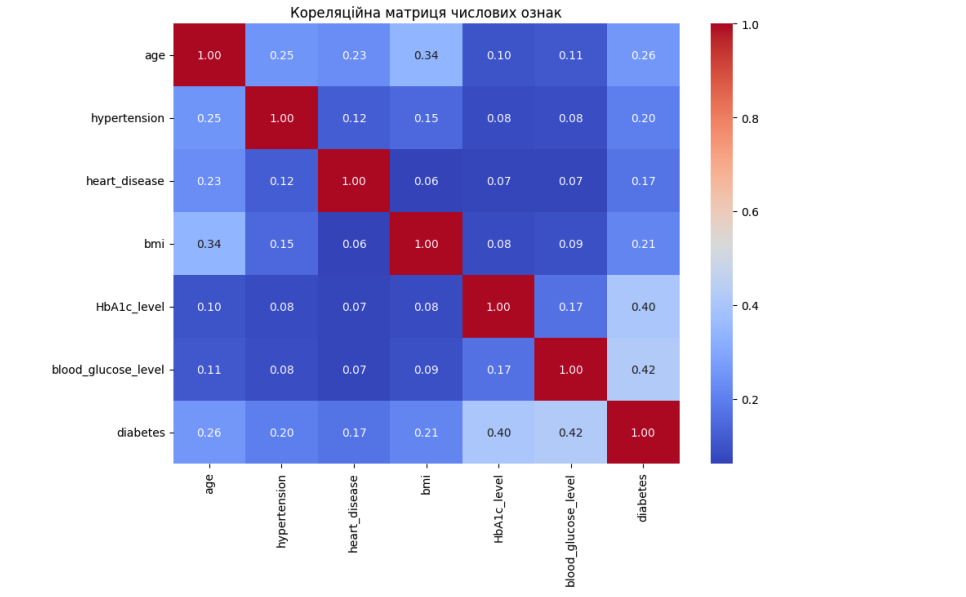


Рисунок 2.9 – Кореляційна матриця Пірсона для числових ознак

За результатами аналізу виявлено, що найбільш значущі зв’язки з цільовою змінною diabetes мають ознаки:

1) **HbA1c_level** – сильна позитивна кореляція;
2) **blood_glucose_level** – висока кореляція з діабетом;
3) **bmi та age** демонструють помірні позитивні зв’язки.

Це свідчить про потенціал використання цих ознак для побудови якісної моделі класифікації.

Для візуального аналізу розподілу ключових числових ознак побудовано гістограми (рис. 2.10–2.12).

```
data = pd.read_csv("diabetes_prediction_dataset.csv")
sns.histplot(data=data, x='bmi', bins=30, kde=True)
plt.title("Розподіл індексу маси тіла (BMI)")
plt.xlabel("BMI")
plt.ylabel("Кількість пацієнтів")
plt.show()
```

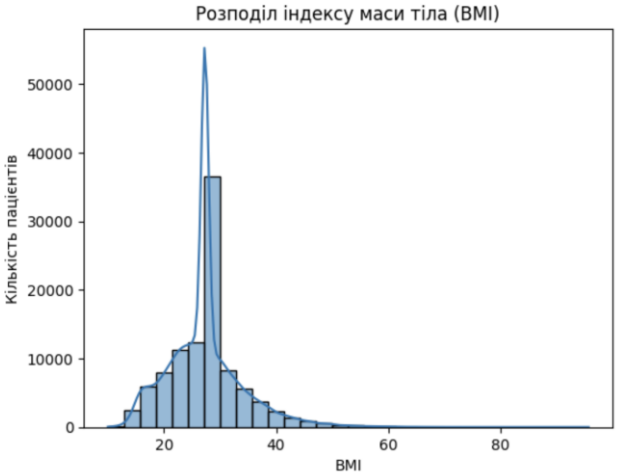


Рисунок 2.10 – Розподіл BMI

Розподіл індексу маси тіла (BMI):

1) **форма**: сильно скошена вправо (right-skewed);

2) **пік**: найбільше пацієнтів мають BMI приблизно між 25 і 30, що відповідає **надмірній масі тіла**;

3) **аномалії**: присутні поодинокі випадки з дуже високим BMI (>50), що можуть бути артефактами або рідкісними випадками.

Висновок: більшість пацієнтів мають надлишкову вагу або ожиріння, що потенційно корелює з ризиком діабету.

```
data = pd.read_csv("diabetes_prediction_dataset.csv")
sns.histplot(data=data, x='HbA1c_level', bins=30, kde=True)
plt.title("Розподіл рівня HbA1c")
plt.xlabel("HbA1c (%)")
plt.ylabel("Кількість пацієнтів")
plt.show()
```

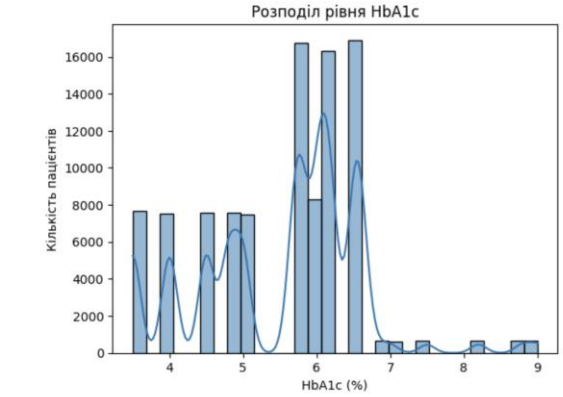


Рисунок 2.11 – Розподіл HbA1c

Розподіл рівня HbA1c:

1) **форма**: мультимодальний розподіл (кілька піків), найбільший пік біля 5.5–6.5%.

2) **пояснення**:

– значення <5.7% – **нормальні**;

– 5.7–6.4% – **прediабет**;

– ≥6.5% – **діабет**.

3) **спостереження**: дуже багато пацієнтів мають HbA1c у прикордонному або діабетичному діапазоні.

Висновок: значна частка вибірки вже має або ризикує мати діабет, судячи з рівня глікозильованого гемоглобіну.

```
data = pd.read_csv("diabetes_prediction_dataset.csv")
sns.histplot(data=data, x='blood_glucose_level', bins=30, kde=True)
plt.title("Розподіл рівня глюкози в крові")
plt.xlabel("Глюкоза (мг/дл)")
plt.ylabel("Кількість пацієнтів")
plt.show()
```

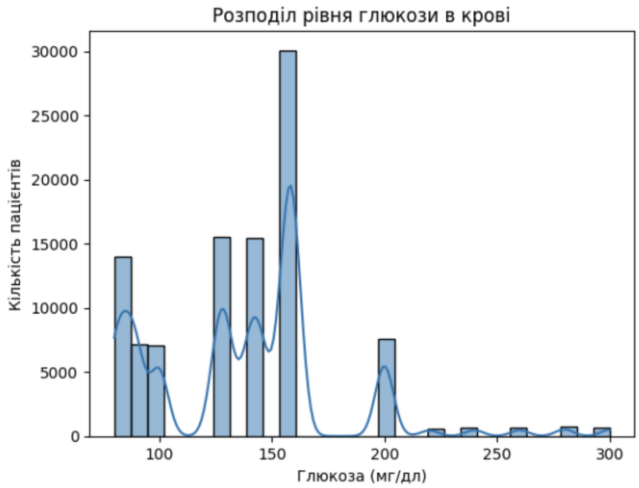


Рисунок 2.12 – Розподіл глюкози в крові

Розподіл рівня глюкози в крові:

1) **форма**: сильно скошена вправо, з піком біля 150 мг/дл.

2) **інтерпретація**: норма: до ~140 мг/дл (залежить від контексту – натще чи після їжі), значення 150–200+ мг/дл – вказують на **гіперглікемію**, **аномалії**: є значення до 300 мг/дл, що може бути критичним.

Висновок: у великої кількості пацієнтів спостерігається підвищений рівень глюкози, що також узгоджується з діабетичним ризиком.

Для візуалізації простору ознак і оцінки лінійної роздільності між класами застосовано метод головних компонент (PCA) (рис. 2.13).

```
data = pd.read_csv("diabetes_prediction_dataset.csv")
# Масштабування ознак
features = data[['age', 'hypertension', 'heart_disease', 'bmi', 'HbA1c_level', 'blood_glucose_level']]
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Застосування PCA
pca = PCA(n_components=2)
pca_components = pca.fit_transform(features_scaled)

# Візуалізація з розфарбуванням за класом
labels = data['diabetes']
plt.figure(figsize=(10, 8))
sns.scatterplot(x=pca_components[:, 0], y=pca_components[:, 1], hue=labels, palette='Set1', alpha=0.7)
plt.title("PCA-візуалізація: 2D простір ознак з поділом за класами діабету")
plt.xlabel("PCA 1")
plt.ylabel("PCA 2")
plt.legend(title="Diabetes")
plt.grid(True)
plt.show()
```

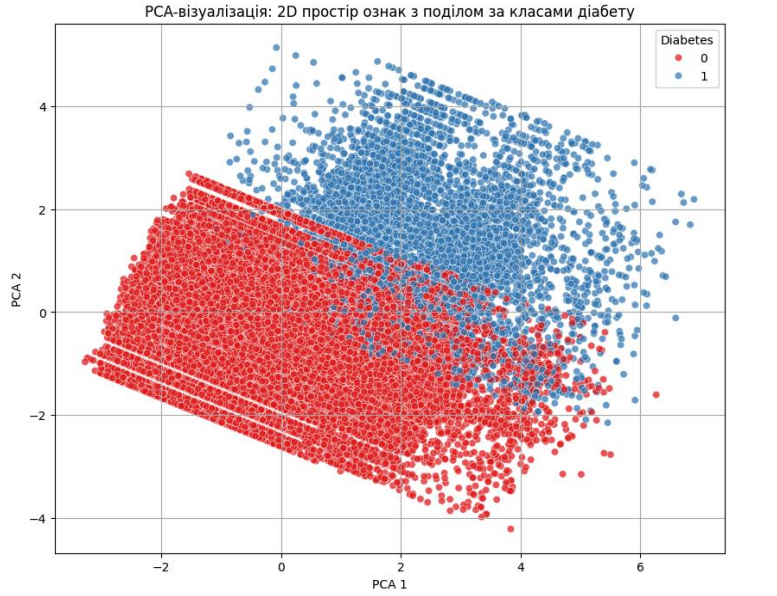


Рисунок 2.13 – Візуалізація PCA-простору

Висновки за результатами візуального аналізу:

– дані не містять пропусків, розподілені досить рівномірно, але мають ознаки **асиметрії** (особливо HbA1c_level та blood_glucose_level);

– виявлено **високі кореляції між ключовими медичними показниками** та наявністю діабету;

– візуалізація PCA засвідчила, що класи частково розділяються, що дозволяє застосовувати як **лінійні**, так і **нелінійні моделі** класифікації;

– надалі рекомендовано перевірити ефективність моделей на основі всього простору ознак, а також із використанням зниження розмірності та відбору найінформативніших ознак.

2.5 Побудова моделі

Метою даного етапу є побудова нейронної мережі, здатної прогнозувати ризик розвитку цукрового діабету на основі сукупності медичних показників користувача. Модель реалізована за допомогою фреймворку Keras (з бібліотеки TensorFlow) із вбудованими засобами препроцесингу в графі обчислень, що дозволяє мінімізувати ризики розсинхронізації даних між етапами підготовки та інференсу (передбачення).

Архітектура передбачає:

1) роботу з різнотипними ознаками (числовими, бінарними, категоріальними, рядковими);

2) автоматичне нормування числових даних і обробку відсутніх значень;

3) вбудоване кодування категоріальних ознак через StringLookup;

4) побудову багатошарового перцептрона (MLP) з функцією активації ReLU та сигмоїдним виходом для класифікації;

5) інтеграцію всієї логіки в єдиний граф TensorFlow, що забезпечує сумісність із TensorFlow.js.

Підготовка даних та препроцесинг

Дані користувачів складаються з кількох груп зображених в таблиці 2.2.

Таблиця 2.2 – Опис груп даних

Тип ознаки	**Приклад**	**Обробка**
Числові (num_features)	вік, індекс маси тіла (IMT), рівень глюкози, HbA1c	нормалізація, заповнення пропусків середнім
Бінарні (bin_features)	наявність гіпертонії, серцевих захворювань	заміна пропусків на 0
Категоріальні (string)	стать, історія куріння	перетворення у one-hot через StringLookup

Для числових змінних виконано обчислення середніх значень по тренувальній вибірці та адаптацію шару Normalization:

```
num_norm = layers.Normalization(axis=-1)
num_norm.adapt(np.nan_to_num(Xn_tr, nan=np.nanmean(Xn_tr, axis=0)))
```

Під час виконання моделі всі пропущені значення автоматично замінюються середніми:

```
x_num_filled = layers.Lambda(
    lambda t: K.where(K.isnan(t), K.cast(num_norm.mean, t.dtype), t),
    name="impute_num_with_mean"
)(inp_num)
```

Для категоріальних змінних використано шари StringLookup з режимом output_mode="one_hot", що забезпечує автоматичне перетворення текстових значень у вектори:

```
gender_lookup = layers.StringLookup(output_mode="one_hot")
smoke_lookup = layers.StringLookup(output_mode="one_hot")
gender_lookup.adapt(Xg_tr.astype(str))
smoke_lookup.adapt(Xs_tr.astype(str))
```

Бінарні змінні очищуються від пропусків (NaN → 0) засобами keras.ops:

```
x_bin = layers.Lambda(
    lambda t: K.where(K.isnan(t), K.zeros_like(t), t),
    name="impute_bin_with_zero"
)(inp_bin)
```

Архітектура нейронної мережі

Після злиття ознак використовується багатошаровий перцептрон із двома прихованими шарами:

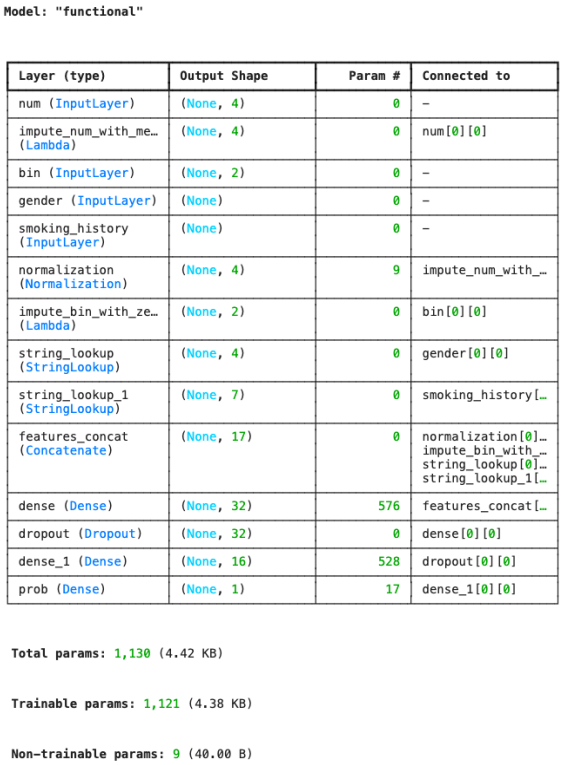
```
x = layers.Dense(32, activation="relu")(x)
x = layers.Dropout(0.2)(x)
x = layers.Dense(16, activation="relu")(x)
out = layers.Dense(1, activation="sigmoid", name="prob")(x)
```

Вихід моделі – ймовірність ризику діабету, нормована у діапазоні [0,1].

Модель компілюється з такими параметрами:

```
model.compile(
    optimizer=keras.optimizers.Adam(1e-3),
    loss="binary_crossentropy",
    metrics=["accuracy", keras.metrics.AUC(name="auc")]
)
```

Отже модель має **функціональну архітектуру** (Functional API) [13], що дозволяє обробляти кілька входів одночасно – числові, бінарні та категоріальні дані (рис. 2.14).



Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
num (InputLayer)	(None, 4)	0	–
impute_num_with_me… (Lambda)	(None, 4)	0	num[0][0]
bin (InputLayer)	(None, 2)	0	–
gender (InputLayer)	(None)	0	–
smoking_history (InputLayer)	(None)	0	–
normalization (Normalization)	(None, 4)	9	impute_num_with_…
impute_bin_with_ze… (Lambda)	(None, 2)	0	bin[0][0]
string_lookup (StringLookup)	(None, 4)	0	gender[0][0]
string_lookup_1 (StringLookup)	(None, 7)	0	smoking_history[…
features_concat (Concatenate)	(None, 17)	0	normalization[0]… impute_bin_with_… string_lookup[0]… string_lookup_1[…
dense (Dense)	(None, 32)	576	features_concat[…
dropout (Dropout)	(None, 32)	0	dense[0][0]
dense_1 (Dense)	(None, 16)	528	dropout[0][0]
prob (Dense)	(None, 1)	17	dense_1[0][0]

Total params: 1,130 (4.42 KB)

Trainable params: 1,121 (4.38 KB)

Non-trainable params: 9 (40.00 B)

Рисунок 2.14 – Функціональна модель

Архітектура моделі поєднує класичний підхід багатошарового перцептрона (MLP) із сучасним підходом до препроцесингу безпосередньо в графі TensorFlow. Така структура:

1) дозволяє приймати одночасно різні типи даних;
2) гарантує узгодженість під час інференсу;
3) є оптимальною для завдань **прогнозування ризику діабету** на основі анкетних і клінічних параметрів.

2.6 Навчання моделі

Навчання виконується на тренувальній вибірці з валідацією:

```
history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=50,
    batch_size=32
)
```

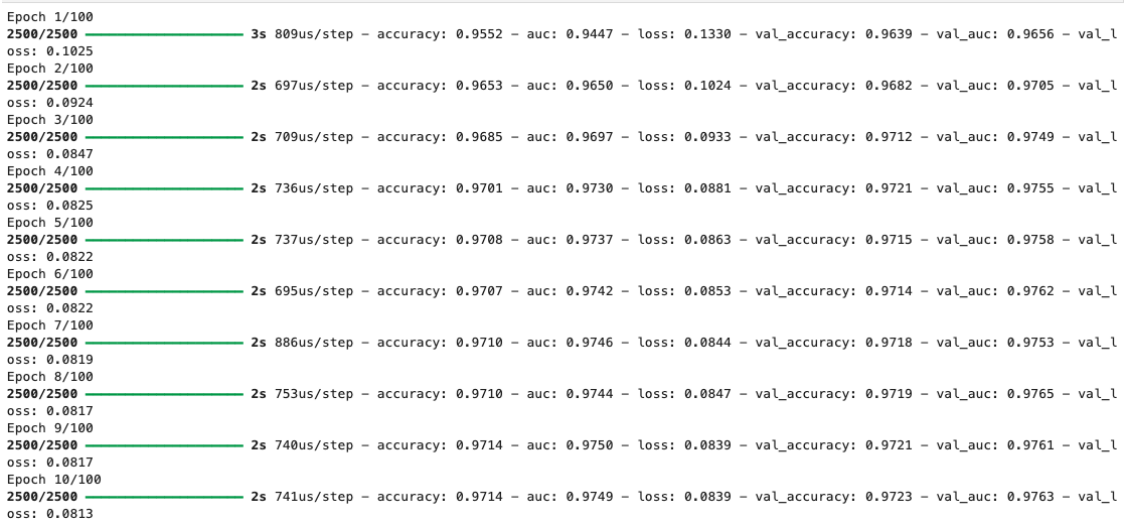


Рисунок 2.15 – Візуалізація навчання вибірки

Розглянемо динаміку навчання нейронної мережі для прогнозування ризику діабету за трьома основними метриками – **Loss (функція втрат)**, **Accuracy (точність класифікації)** та **AUC (Area Under Curve)**.

Loss (функція втрат)

На графіку (рис. 2.16) спостерігається стрімке зниження функції втрат на перших епохах як для тренувальної, так і для валідаційної вибірки. Уже після 5–7 епох значення стабілізується на рівні близько 0.08, що свідчить про ефективне навчання моделі та відсутність ознак перенавчання. Криві train_loss і val_loss розташовані близько одна до одної, отже модель добре узагальнює дані.

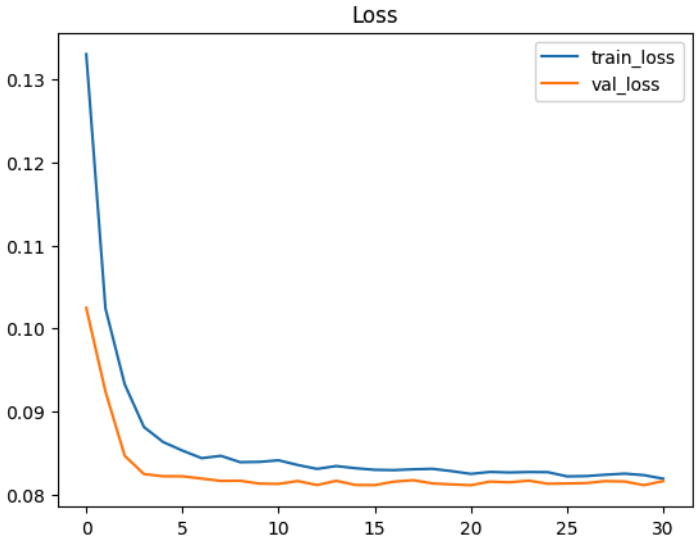


Рисунок 2.16 – Динаміка навчання нейронної мережі для прогнозування ризику діабету за функцією втрат

Accuracy (точність класифікації)

На графіку (рис. 2.17) видно, що точність моделі швидко зростає протягом перших епох і досягає понад **97%**. Криві навчання (train_acc) та валідації (val_acc) практично збігаються, що свідчить про стабільну поведінку мережі. Модель не демонструє різкого розходження метрик між тренувальними та тестовими даними, тому узагальнення працює коректно.

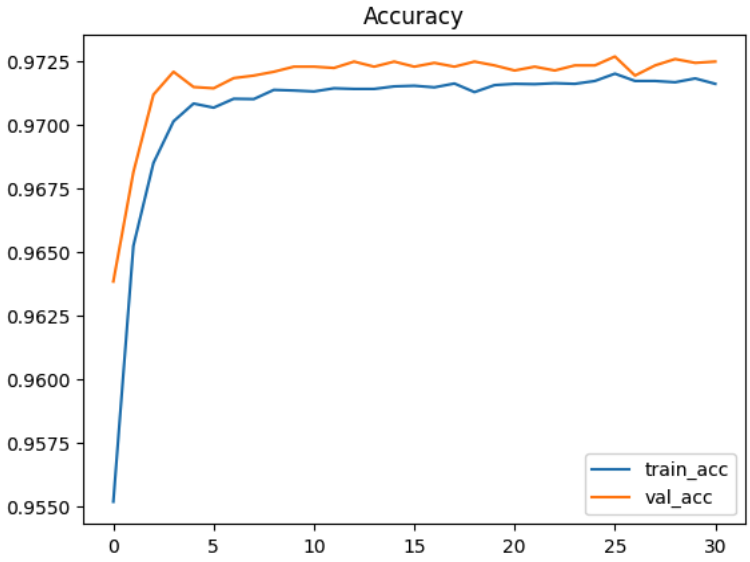


Рисунок 2.17– Динаміка навчання нейронної мережі для прогнозування ризику діабету за точністю класифікації

AUC (Area Under Curve)

Аналізуючи графік (рис. 2.18) можна зазначити, що значення AUC перевищує **0.97** вже після кількох епох, що говорить про високу здатність моделі відокремлювати позитивні та негативні класи. Криві train_auc і val_auc майже паралельні, що свідчить про високу стабільність класифікації незалежно від набору даних. Такий рівень AUC є показником відмінної якості моделі для медичних задач прогнозування.

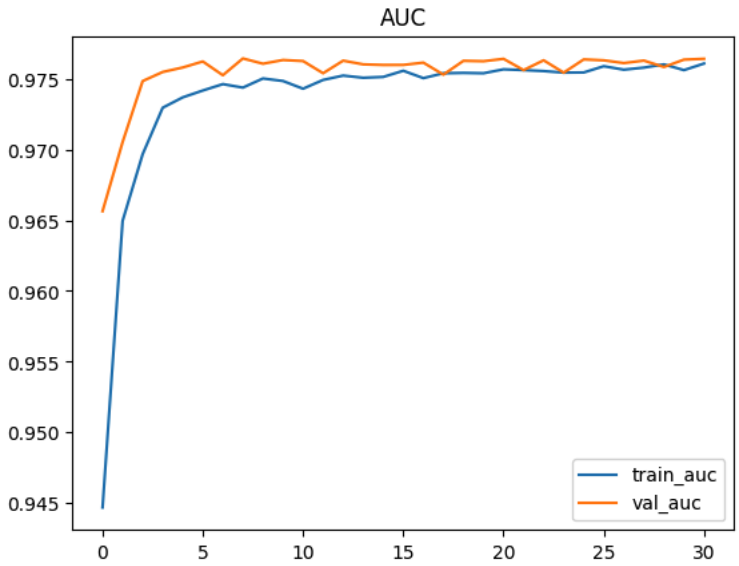


Рисунок 2.17– Динаміка навчання нейронної мережі для прогнозування ризику діабету за Area Under Curve

Збереження та експорт моделі

Після навчання модель зберігається у форматі Keras:

model.save("model.h5")

Для інтеграції у вебзастосунок виконується конвертація у формат TensorFlow.js:

!tensorflowjs_converter --input_format=keras --weight_shard_size_bytes=300000000 model.h5 ./web_model_new

У результаті створюється набір файлів (model.json і ваги .bin) (рис. 2.18), які можуть бути безпосередньо використані у браузері через API tf.loadLayersModel().

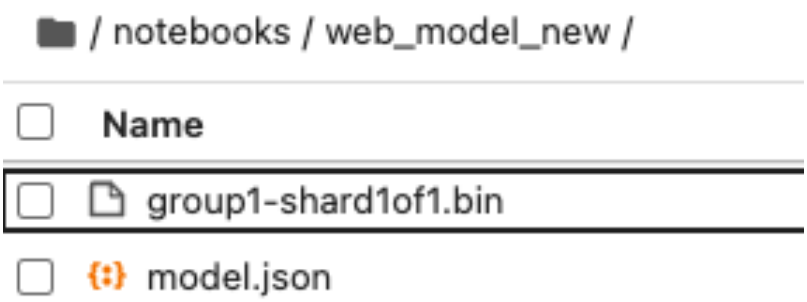


Рисунок 2.18 – Згенеровані файли

Згенерована модель (рис. 2.19) – нейромережа, яка визначає ймовірність належності до одного з двох класів. Вона отримує чотири типи даних: числа, бінарні значення (0 або 1), стать і історію куріння. У числових даних пропуски замінюються середнім, у бінарних – нулями, а текстові значення перетворюються у набір чисел (one-hot). Потім усі дані об’єднуються в один вектор і проходять через два шари нейронів із ReLU та шар Dropout, який запобігає перенавчанню. На виході – один нейрон із функцією sigmoid, який видає ймовірність результату. Модель навчається з оптимізатором Adam і метриками точності (accuracy) та AUC.

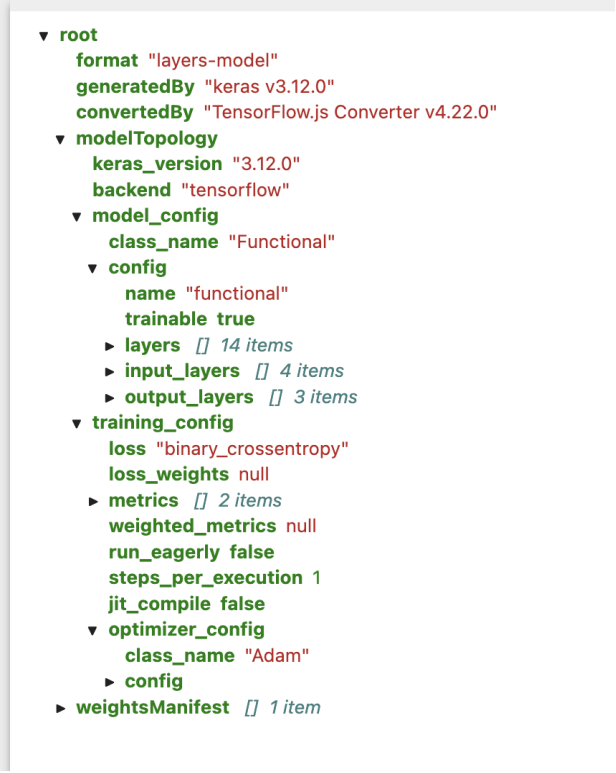


Рисунок 2.19 – Структура файлу model.json

Переваги реалізованого підходу:

1) повний препроцесинг усередині графа моделі, що гарантує відсутність розбіжностей між етапами навчання і прогнозування;

2) автоматичне оброблення пропусків та нормалізація без необхідності зовнішніх бібліотек;

3) гнучка інтеграція у вебзастосунок, оскільки модель експортується у формат TensorFlow.js;

4) висока швидкодія завдяки виконанню обчислень безпосередньо у браузері;

5) простота супроводу – уся логіка обробки даних і передбачення реалізована в єдиній моделі.

Висновки до розділу 2

У другому розділі розроблено систему прогнозування ризику цукрового діабету II типу з використанням Angular і TensorFlow.js. Проведено аналіз набору з 1000 записів із числовими, категоріальними та бінарними ознаками; основними факторами ризику визначено рівень глікованого гемоглобіну, глюкозу та індекс маси тіла.

Побудовано й навчено модель із вбудованим препроцесингом (StringLookup, Normalization), що спростило підготовку даних і забезпечило узгодженість обробки. Модель продемонструвала добру точність і AUC та була експортована у формат TensorFlow.js для використання у вебзастосунку як інструмент раннього виявлення ризику діабету.

3 МОДЕЛЮВАННЯ ВЕБЗАСТОСУНКУ

3.1 Створення сценаріїв

Сценарії використання є важливими, оскільки вони дозволяють описати взаємодію користувача із системою зрозумілою мовою, формалізувати вимоги, зменшити ризик неоднозначного трактування функціональності та забезпечити основу для подальшого тестування. Вони допомагають замовникам, аналітикам і розробникам мати єдине бачення того, як система повинна поводитися у різних ситуаціях, а також перевіряти відповідність програмного забезпечення потребам користувачів. [14]

Коротка форма:

Usecase: Очистка історії прогнозів

Актор: Пацієнт/Лікар.

Приклад:

Користувач відкриває вебзастосунок і переходить у розділ «Історія». Він бачить список попередніх результатів прогнозів. Користувач натискає кнопку «Очистити історію», після чого система запитує підтвердження дії. Погодившись, користувач підтверджує очищення, і система видаляє всі записи з LocalStorage. На екрані з’являється повідомлення «Історію очищено», а список результатів стає порожнім.

Поверхнева форма:

Usecase: Отримати прогноз ризику цукрового діабету II типу

Суть: Користувач вводить медичні показники у вебзастосунку та отримує категорію ризику (низький/середній/високий) з візуалізацією.

Користувачі: Пацієнт/Лікар/Гість.

Передумови: Сучасний браузер з підтримкою WebGL; модель TensorFlow.js завантажена; користувач підтвердив дисклеймер про орієнтовний характер оцінки.

Основний успішний сценарій:

1) користувач відкриває вебзастосунок;
2) переходить на сторінку «Оцінка ризику»;

3) вводить показники (вік, зріст і вагу або BMI, рівень глюкози натще, за наявності – артеріальний тиск тощо);

4) натискає кнопку «Розрахувати»;

5) система виконує локальний inference моделі TensorFlow.js у браузері;

6) система відображає категорію ризику з індикатором/діаграмою та короткими підказками інтерпретації;

7) користувач може зберегти результат до історії;

8) за потреби користувач експортує підсумок у PDF/PNG або повертається до головної сторінки.

Альтернативні сценарії

Сценарій 1: Користувач не прийняв дисклеймер. У цьому випадку система блокує розрахунок і пропонує переглянути умови ще раз.

Сценарій 2: Користувач не ввів жодних показників. У цьому випадку система підсвічує обов’язкові поля й показує підказки щодо формату/одиниць.

Сценарій 3: Користувач ввів некоректні дані (негативні значення, нереалістичні межі, помилкові одиниці). У цьому випадку система відображає повідомлення про помилку та приклади валідних значень.

Сценарій 4: Обчислення неможливе через брак ресурсів/відсутність WebGL. У цьому випадку система повідомляє про обмеження, пропонує спростити набір ознак або спробувати інший браузер/пристрій.

Сценарій 5: Користувач не зацікавлений у результаті. У цьому випадку він може закрити вебзастосунок або повернутися на головну без збереження запису.

Повна форма

Таблиця 2.1 – Повна форма для сценарію «Перегляд історії прогнозів»

Usecase section	Опис
Use Case Name	Перегляд історії прогнозів
Scope	Система
Level	User-goal
Primary Actor	Пацієнт
Stakeholders and interests	Пацієнт/Лікар (користувач): бажає бачити динаміку стану здоров'я, аналізувати зміни у прогнозах, швидко знаходити потрібні записи. Розробники: забезпечити простий доступ з використанням Firebase
Preconditions	У Firebase Firestore вже існують записи з прогнозами.
Success guarantee	Система відображає список прогнозів із такими даними: 1) дата створення прогнозу; 2) категорія ризику (низький/середній/високий); 3) ключові параметри (наприклад, показники з вимірювань).
Main Success Scenario	1) користувач відкриває розділ «Історія»; 2) система отримує список прогнозів**;** 3) система сортує записи за датою (останній зверху); 4) система відображає список із короткою інформацією (дата, категорія ризику, короткі параметри, міні-діаграма);
Extensions	Історія порожня: система показує повідомлення *«Записів немає»*. Запис пошкоджений/неповний: система відображає його зі спеціальним маркером *«Неповні дані»*. Фільтрація: користувач може застосувати пошук за датою, категорією або ключовим параметром.

Кінець таблиці 2.1

Special Requirements	1) автоматичне сортування за датою (новіші – зверху); 2) пошук та фільтрація без перезавантаження сторінки; 3) міні-діаграми (наприклад, тренди ризиків) у списку для наочності.
Technology and Data Variations List	1) зберігання: Firebase Firestore. 2) фільтрація: за діапазоном дат, за категоріями ризику; 3) візуалізація: міні-діаграми, що відображають зміни ключових параметрів.
Frequency of Occurrence	Кілька разів на тиждень
Miscellaneous	Історія прогнозів зберігається у Firebase; доступна лише авторизованим користувачам. Дані відображаються у вигляді списку з кольоровими індикаторами ризику та підтримкою фільтрації й сортування.

3.2 Створення діаграми використання

Діаграма використання – поведінкова діаграма UML, що фіксує функціональні вимоги системи з точки зору зовнішніх акторів і показує межу системи як прямокутник, варіанти використання як еліпси всередині та зв’язки акторів із цими варіантами, а також залежності include і extend. Її мета – узгодити що робить система і з ким взаємодіє на високому рівні, слугувати основою для текстових специфікацій use case і подальшого тестування. Таким чином, діаграма використання дозволяє на високому рівні зрозуміти, які функції має виконувати система і як з нею взаємодіють користувачі [15].

Діаграма використання (рис. 3.1) відображає взаємодію користувачів із системою, зокрема пацієнтів, лікарів і гостей. Основна мета системи – забезпечити зручне введення медичних даних, автоматичне прогнозування ризику захворювання, перегляд і аналіз результатів, а також відстеження змін

рівня глюкози. Кожен випадок використання описує окрему функціональну можливість системи – від реєстрації користувача до експорту результатів і перегляду історії прогнозів. Такий підхід дає змогу чітко визначити ролі користувачів, їхні дії та очікувані результати роботи системи.

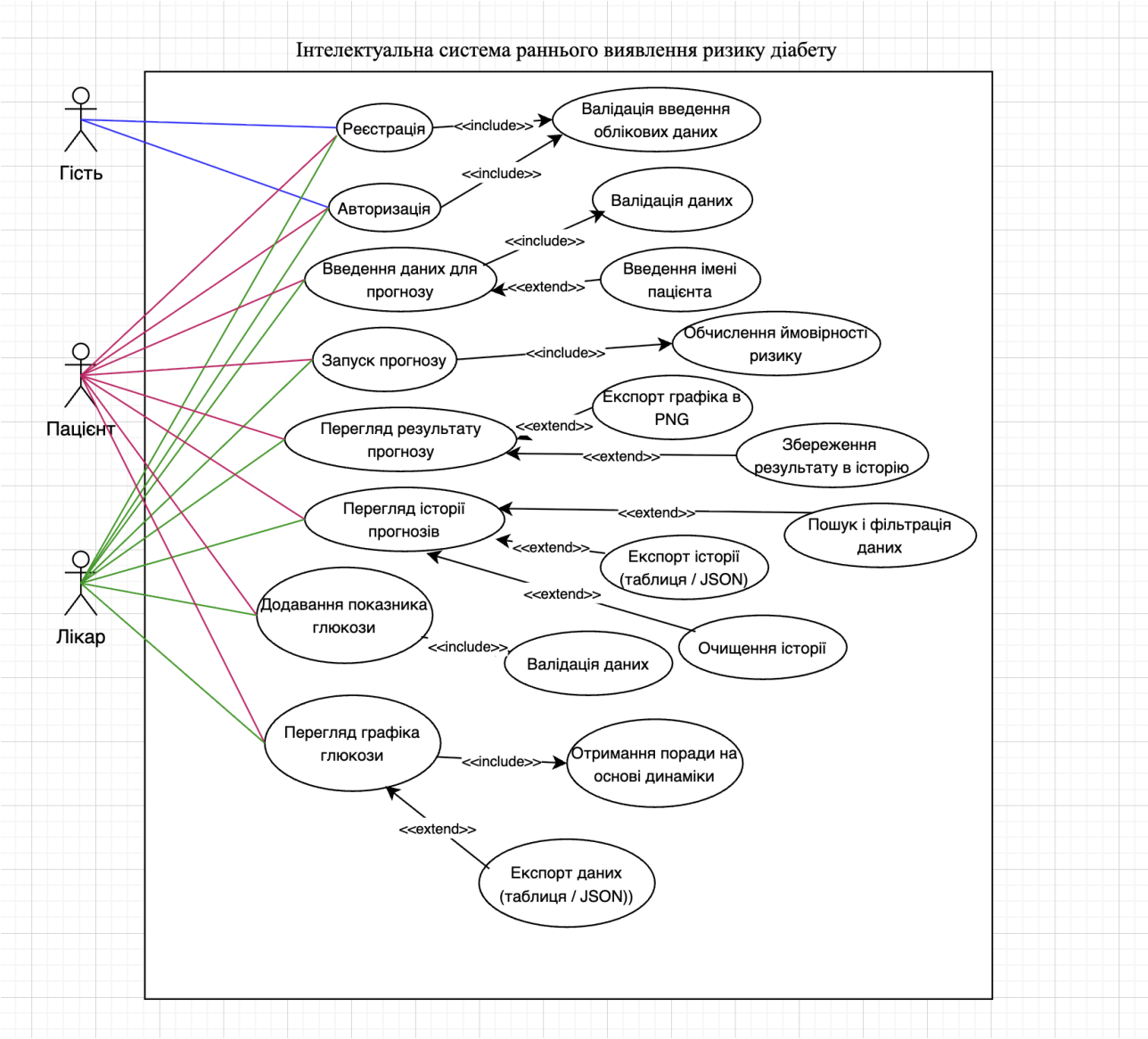


Рисунок 3.1 – Діаграма використання

Актори включають:

1) гість;
2) пацієнт;
3) лікар.

Випадки використання системи прогнозування ризику діабету включають:

1) реєстрація та авторизація користувача (Register / Login) – створення нового облікового запису або вхід до системи для отримання доступу до основних функцій;

2) введення даних (Enter Data) – введення необхідних показників, таких як вік, IMT, рівень глюкози, глікований гемоглобін, стать та історія куріння;

3) валідація даних (Validate Data) – перевірка правильності форматів і діапазонів значень перед обчисленням прогнозу;

4) обчислення ризику діабету (Compute Diabetes Risk) – виконання локального прогнозування за допомогою моделі TensorFlow.js;

5) перегляд результату (View Result) – відображення текстової оцінки ризику (низький, середній, високий) та відповідної графічної візуалізації;

6) збереження результату в історію (Save to History) – додавання прогнозу до локальної історії користувача;

7) перегляд історії прогнозів (View History) – перегляд збережених результатів із можливістю пошуку та фільтрації;

8) експорт результатів (Export Results) – збереження результатів або історії у форматах PNG, таблиці чи JSON для подальшого використання;

9) очищення історії (Clear History) – видалення всіх записів прогнозів користувача з локальної пам’яті;

10) введення імені пацієнта (Enter Patient Name) – опціональна дія для лікаря при створенні прогнозу для конкретного пацієнта;

11) трекинг глюкози (Glucose Tracking) – введення рівня глюкози та дати вимірювання для відстеження динаміки;

12) перегляд графіка глюкози (View Glucose Chart) – візуалізація змін рівня глюкози з часом;

13) отримання поради (Get Recommendation) – автоматичне формування короткої рекомендації на основі тенденції рівня глюкози.

3.3 Створення діаграм діяльності

Діаграма діяльності – це тип поведенської UML-діаграми, який слугує для моделювання порядку виконання кроків або операцій в системі чи процесі. Вона показує, які дії виконуються, у якій послідовності, а також як переходи між ними реалізовані через вибір (умови), розгалуження чи об’єднання потоків.

Такі діаграми активно застосовують у розробці програмного [16] забезпечення, бізнес-аналізі, моделюванні робочих процесів та інших областях, де потрібно візуалізувати сценарії виконання завдань, зрозуміти послідовність кроків, виявити потенційні проблемні місця або оптимізувати процес [18].

Діаграма діяльності: Обчислення ризику діабету (рис. 3.2)

Мета: показати основний потік – від введення даних до отримання результату.

Кроки діяльності:

1) користувач (гість, пацієнт або лікар) вводить дані: вік, вагу, зріст, BMI, рівень глюкози тощо;

2) система виконує валідацію: перевірка коректності форматів і діапазонів;

3) якщо помилка → повідомлення про некоректні дані, повернення до введення;

4) якщо дані валідні → TensorFlow.js модель виконує обчислення;

5) отримується результат (низький/середній/високий ризик);

6) система відображає результат на екрані з візуалізацією.

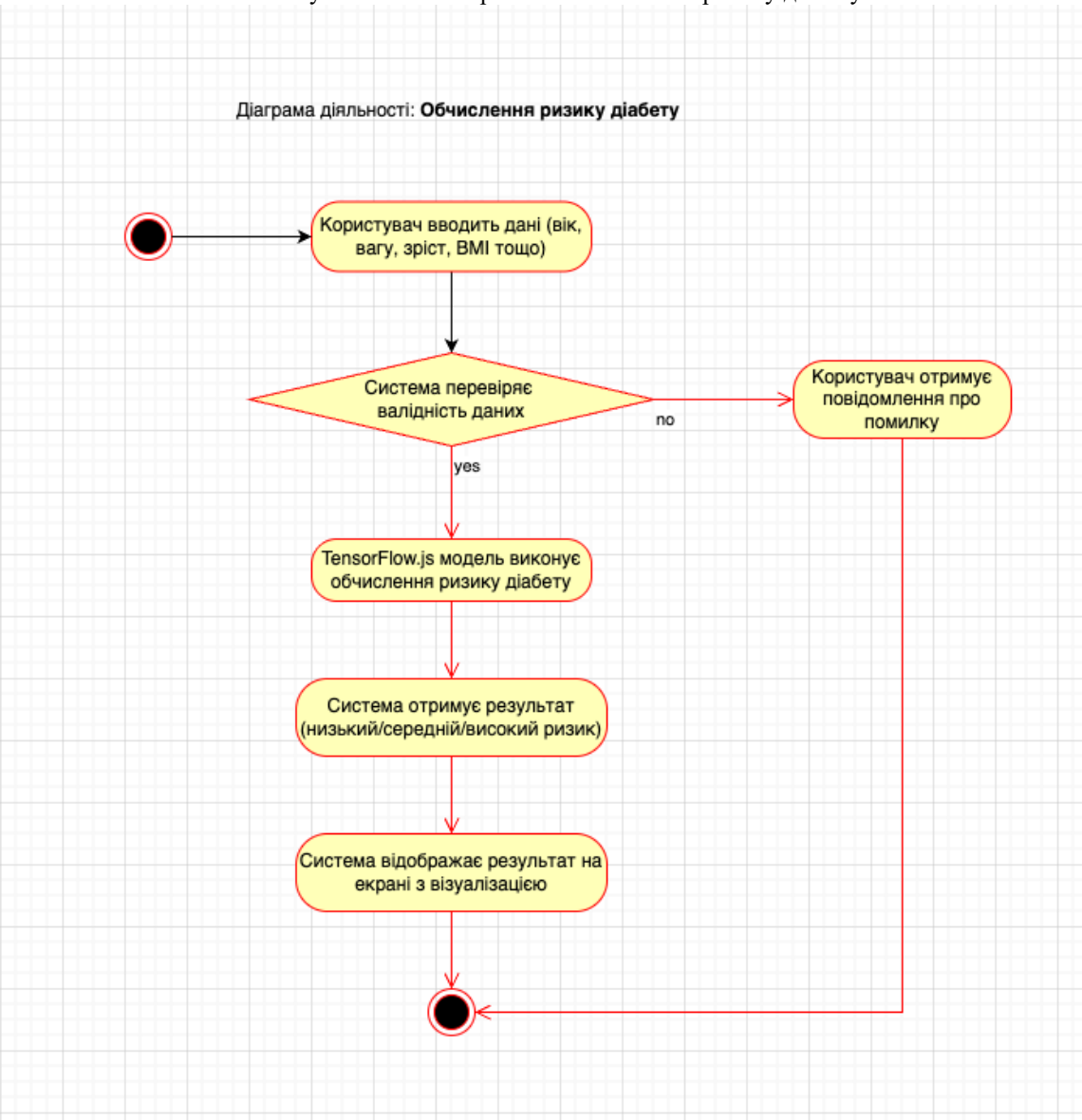


Рисунок 3.2 – Діаграма діяльності для usecase «**Обчислення ризику діабету**»

Діаграма діяльності: Робота з історією результатів (рис. 3.3)

Мета: відобразити, як користувач може працювати зі збереженими прогнозами.

Кроки діяльності:

1) користувач переходить на вкладку «Історія»

2) користувач обирає одну з опцій:

– перегляд історії → система показує список попередніх записів (з датами та показниками);

– очищення історії → система повністю видаляє всі записи;

– експорт результатів → формування звітів у форматі таблиці або JSON.

3) якщо користувач не має історії, то він бачить повідомлення про відсутність записів.

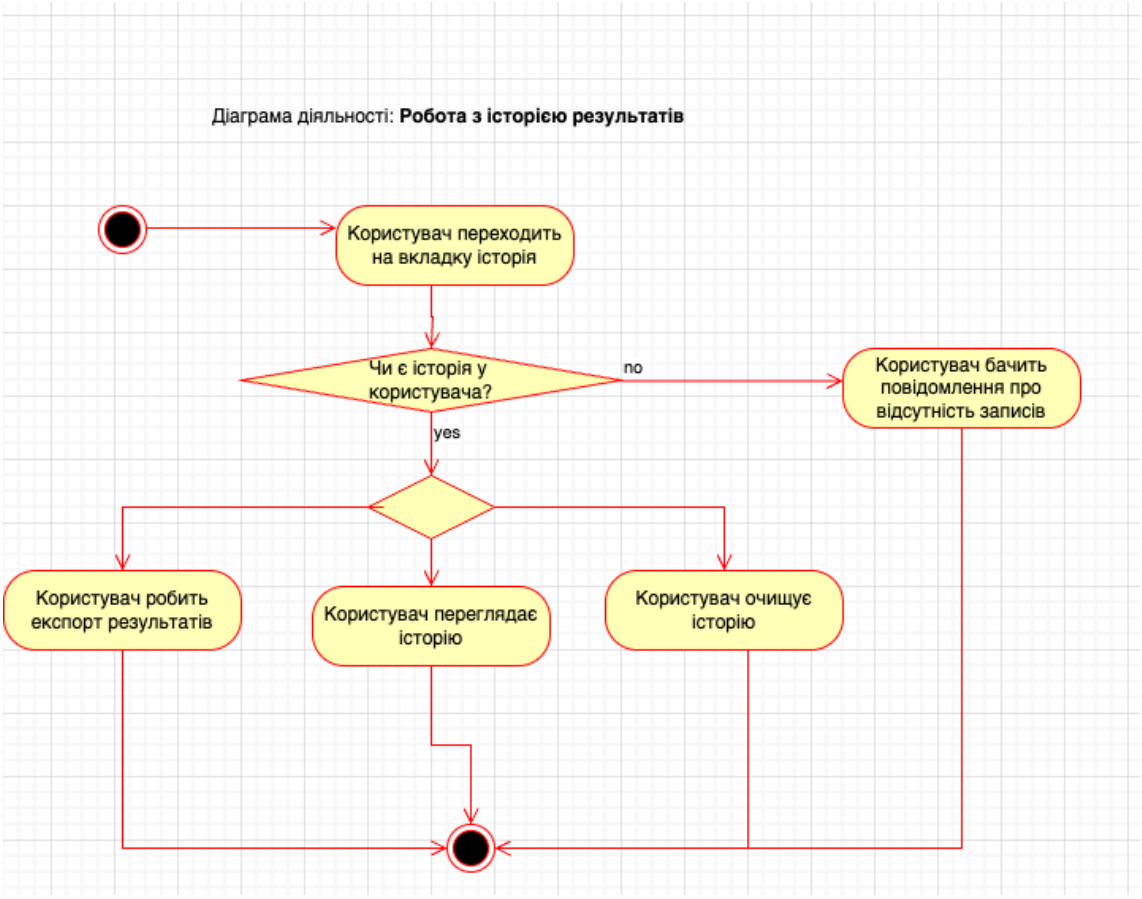


Рисунок 3.3 – Діаграма діяльності для usecase «Робота з історією результатів»

Діаграма діяльності: Ведення трекера глюкози (рис. 3.4)

Мета: зафіксувати вимір глюкози та показати користувачу оновлену візуалізацію і пораду.

Основний потік:

1) користувач переходить на сторінку трекера глюкози;

2) користувач вводить рівень глюкози та дату вимірювання;

3) система перевіряє коректність введених даних (формат, діапазони);

4) якщо дані валідні → користувач натискає кнопку збереження рівня глюкози;

5) система зберігає запис і відображає: діаграму змін рівня глюкози, збережений результат, коротку пораду щодо покращення самопочуття.

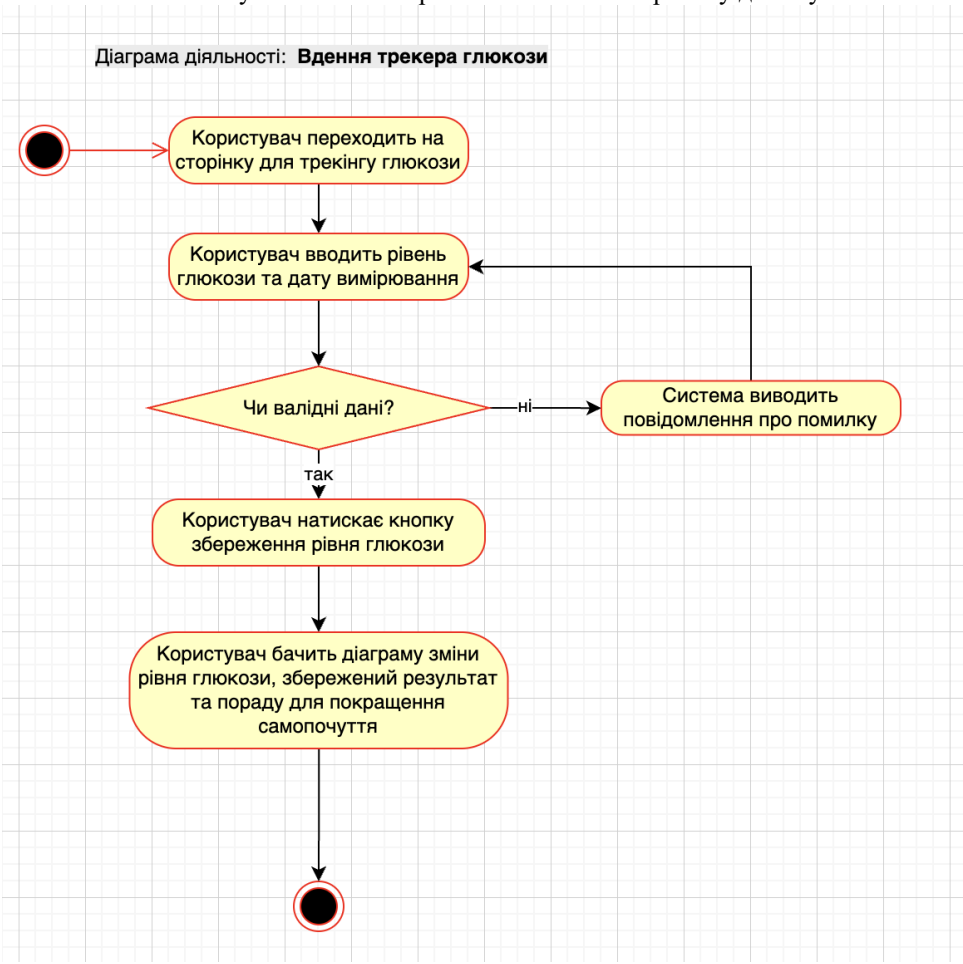


Рисунок 3.4 – Діаграма діяльності для usecase «Ведення трекера глюкози»

Альтернативний потік (помилка валідації): якщо дані невалідні, система показує повідомлення про помилку, після чого користувач може виправити значення і повторити кроки 2–4.

3.4 Створення DFD

DFD (Data Flow Diagram) – це графічна модель, яка показує, як дані рухаються в інформаційній системі чи бізнес-процесі. Вона відображає зовнішні сутності, що взаємодіють із системою, процеси, які перетворюють дані, сховища, де інформація зберігається, та потоки, якими дані передаються. Існують різні рівні деталізації: від контекстної діаграми, що показує систему загалом, до розгорнутих діаграм, де кожен процес описується все детальніше [17].

DFD допомагає зрозуміти логіку роботи системи, виявити зайві або відсутні потоки та зробити модель доступною для технічних і бізнес-

користувачів. Водночас вона не відображає часової послідовності подій і може ставати складною при надмірній деталізації.

DFD-0 системи прогнозування діабету (рис. 3.5) показує, як відбувається обмін даними між основними процесами:

1) керування введенням даних – приймає медичні показники користувача;

2) керування обчисленням ризику – виконує валідацію та розрахунок ризику за моделлю TensorFlow.js;

3) керування візуалізацією результатів – формує текстові та графічні відображення прогнозу;

4) керування історією – відповідає за збереження, перегляд і очищення попередніх прогнозів;

5) керування трекером глюкози – забезпечує введення вимірів глюкози, побудову графіка змін та генерацію порад;

6) керування експортом результатів – дає змогу користувачу формувати звіти у форматах PNG, таблиці чи JSON.

Рисунок 3.5 – DFD нульового рівня

DFD першого рівня (рис. 3.6) відображає основні процеси обробки даних у системі прогнозування ризику діабету. Центральним елементом є процес «Система прогнозування діабету», який координує роботу всіх підсистем та забезпечує повний цикл – від введення даних до формування результатів і звітів.

До складу системи входять такі основні підпроцеси:

1) керування введенням медичних показників – приймає від користувача дані (вік, вага, зріст, IMT, рівень глюкози та інші параметри) і передає їх для подальшої обробки;

2) керування обчисленням ризику – виконує перевірку введених даних і проводить прогнозування за допомогою моделі TensorFlow.js, визначаючи категорію ризику розвитку діабету;

3) керування візуалізацією результатів – відповідає за відображення прогнозу у текстовій і графічній формах, забезпечуючи зрозуміле представлення інформації для користувача;

4) керування історією – дозволяє зберігати, переглядати, фільтрувати або очищати результати попередніх прогнозів;

5) керування експортом результатів – забезпечує створення звітів у форматах PNG, таблиці або JSON для подальшого використання;

6) керування трекером глюкози – дає змогу користувачу вводити рівні глюкози, переглядати графік змін та отримувати рекомендації на основі динаміки показників.

На виході система формує основні результати: звіт про введені дані, прогноз ризику, візуалізацію результатів, звіт про історію обчислень, дані трекера глюкози та експортовані файли.

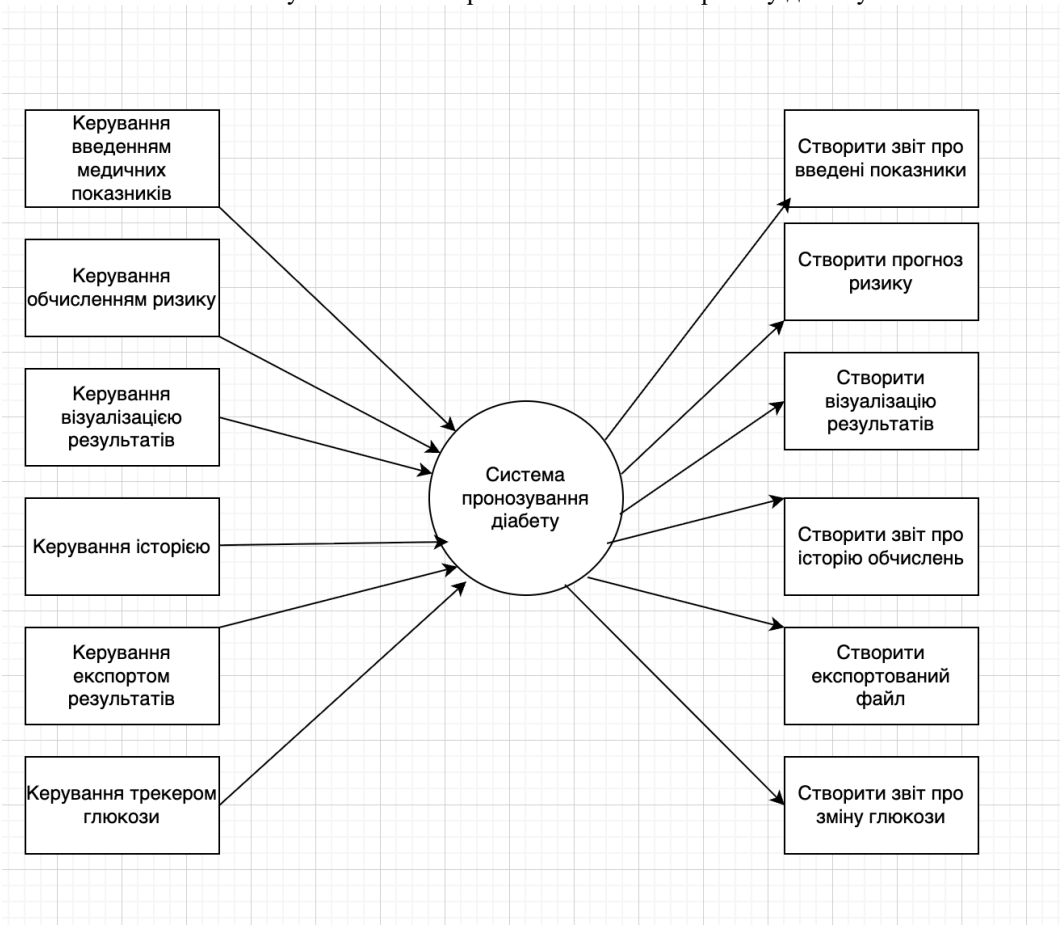


Рисунок 3.6 – DFD першого рівня

DFD другого рівня (рис. 3.7) деталізує роботу підпроцесу «Керування введенням медичних показників» у системі прогнозування діабету. Основна увага приділена послідовності обробки даних, які користувач вводить для подальшого розрахунку ризику захворювання.

Процес починається з введення базових даних, де користувач надає основну інформацію – вік, зріст, вагу та інші початкові параметри. Далі ці значення надходять до підпроцесу розрахунку похідних показників, у якому система автоматично визначає узагальнені параметри, наприклад, індекс маси тіла. Потім дані передаються до введення клінічних показників, де додаються медичні значення, такі як рівень глюкози, артеріальний тиск тощо.

Після збору усіх параметрів виконується валідація значень – система перевіряє правильність форматів і діапазонів, а також виявляє можливі помилки у введених даних. Далі вся перевірена інформація надходить до формування узагальненого набору даних, де показники структуруються для подальшої обробки.

Отриманий набір передається до центрального процесу керування модулями, який координує взаємодію з іншими частинами системи – керуванням обчисленням ризику, керуванням візуалізацією результатів, керуванням історією, керуванням експортом результатів і керуванням трекером глюкози.

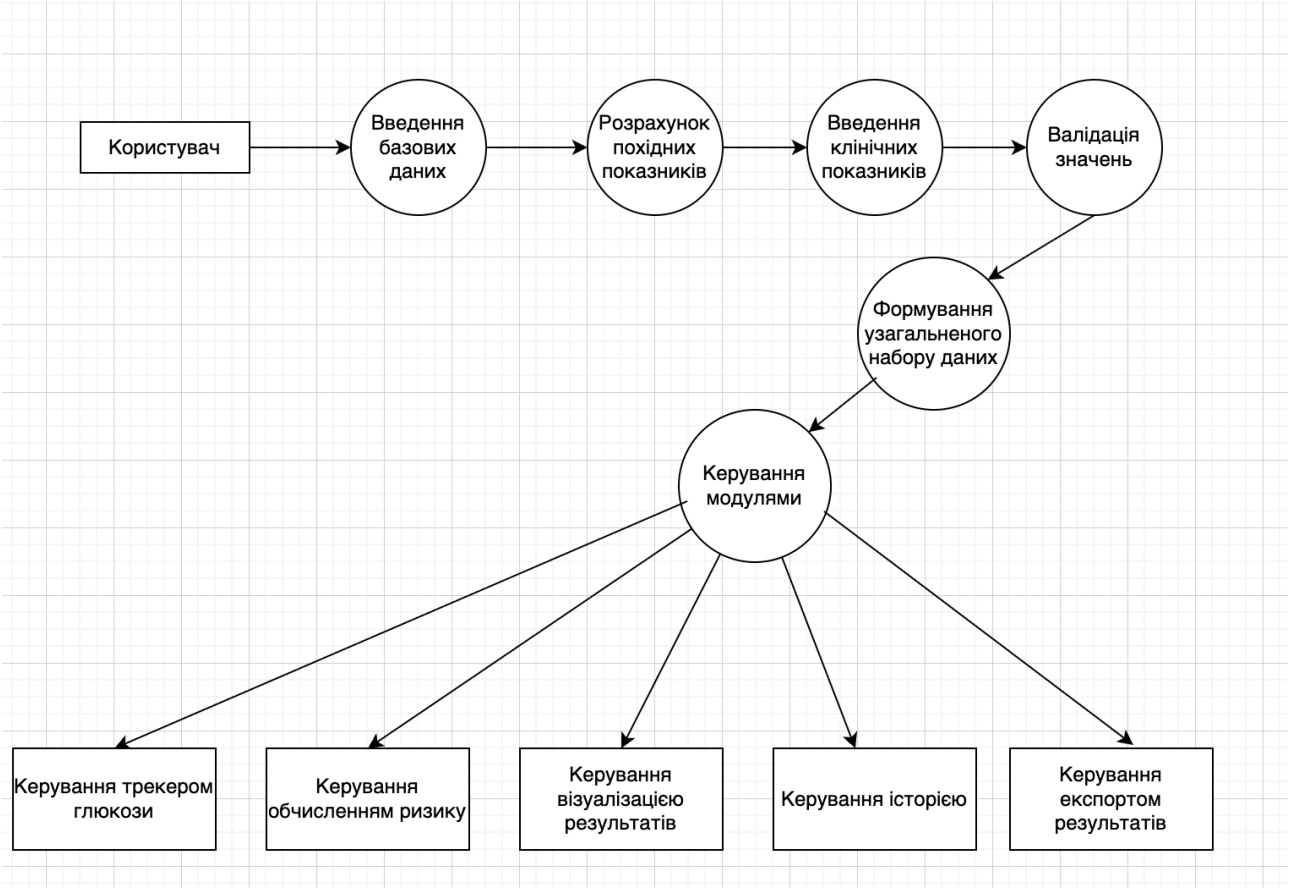


Рисунок 3.7– DFD другого рівня

Таким чином, DFD 2-го рівня описує повний цикл обробки введених користувачем даних: від введення початкових і клінічних показників, через їх розрахунок і перевірку, до підготовки структурованого набору, який надалі використовується всіма основними модулями системи.

3.5 Створення діаграми розгортання

Діаграма розгортання (Deployment diagram) – це тип діаграми UML, яка використовується для моделювання фізичної архітектури системи. Вона показує, як програмні артефакти (програми, компоненти, бази даних, бібліотеки) розміщуються на апаратних вузлах (пристроях, серверах,

клієнтських машинах, мобільних пристроях тощо) і як ці вузли з'єднані між собою каналами зв'язку [18].

Була створена діаграма розгортання для інтелектуальної системи раннього виявлення ризику діабету (рис. 3.8). На діаграмі зображено архітектуру розгортання вебсистеми прогнозування ризику діабету. Центральним елементом є користувач (User), який взаємодіє із застосунком через веббраузер на своєму клієнтському пристрої (Client Device).

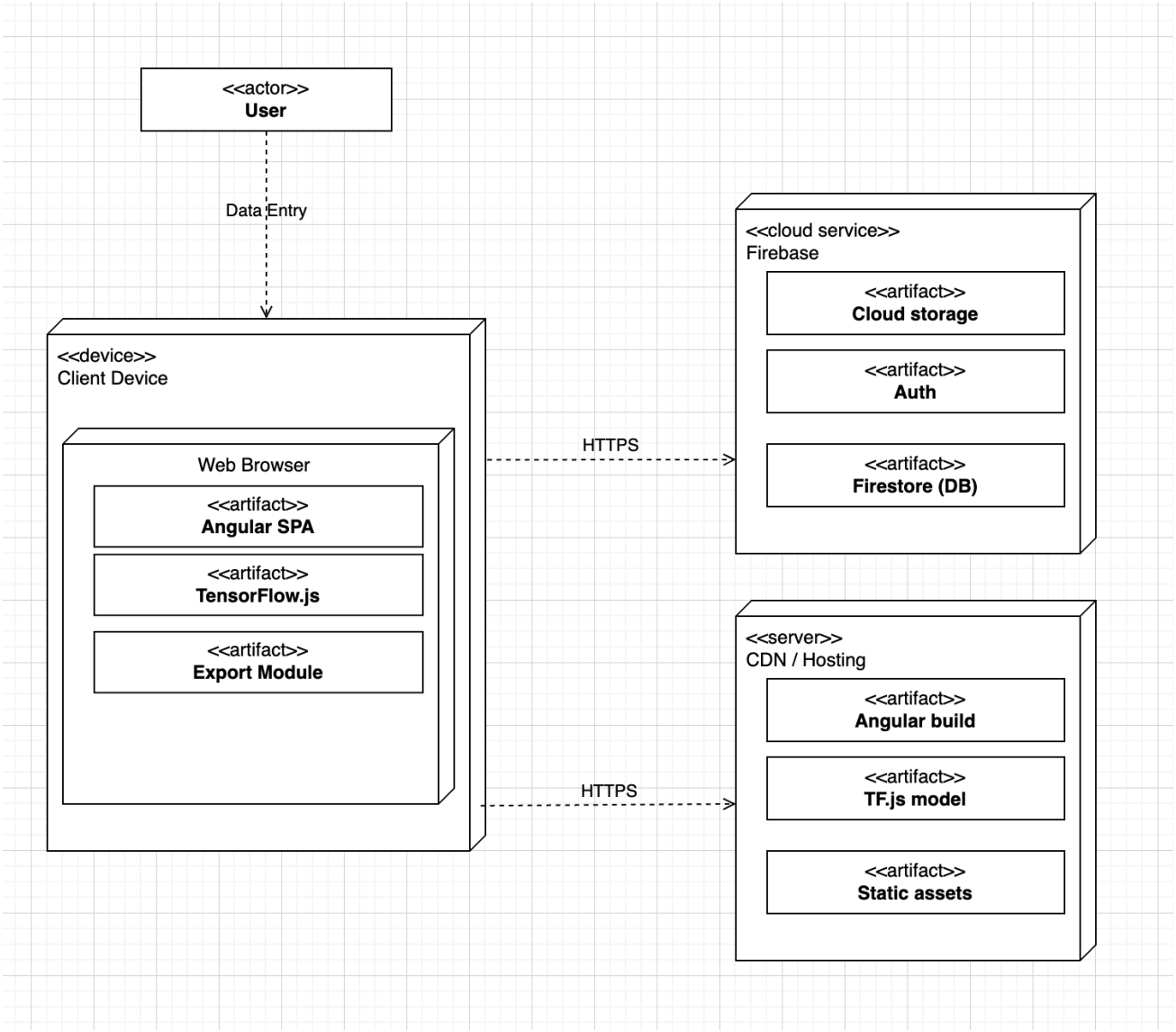


Рисунок 3.8 – Діаграма розгортання системи

У браузері виконуються основні компоненти системи:

1) Angular SPA – односторінковий вебзастосунок, що забезпечує інтерфейс користувача та логіку взаємодії;

2) TensorFlow.js – бібліотека для виконання локальних обчислень моделі прогнозування ризику діабету безпосередньо на клієнті;

3) Export Module – модуль, який відповідає за формування звітів і графічних результатів у форматах PNG або таблиці.

Комунікація із зовнішніми сервісами здійснюється через HTTPS-з’єднання.

Додаток отримує ресурси з CDN / Hosting, де розміщено:

1) Angular build – зібрана версія вебзастосунку;

2) TF.js model – навчена модель нейромережі у форматі TensorFlow.js;

3) Static assets – допоміжні статичні ресурси (зображення, стилі, скрипти).

Другий серверний вузол – Firebase (cloud service) – забезпечує хмарні функції системи:

1) Firestore (DB) – база даних для збереження історії прогнозів, показників глюкози та профілів користувачів;

2) Auth – сервіс автентифікації для реєстрації та входу користувачів;

3) Cloud Storage – сховище для файлів, наприклад, збережених експортів або зображень результатів.

Таким чином, діаграма демонструє взаємодію користувача, клієнтського застосунку та хмарних сервісів. Обчислення ризику діабету виконуються безпосередньо в браузері, а Firebase відповідає за автентифікацію, збереження даних та синхронізацію результатів, забезпечуючи повністю веборієнтовану архітектуру без потреби у власному серверному бекенді.

Висновки до розділу 3

У третьому розділі було виконано моделювання системи для прогнозування ризику цукрового діабету. Були розроблені сценарії використання в різних формах, що дозволило описати взаємодію користувачів із системою, визначити функціональні вимоги. Далі побудовано діаграму використання з акторами (гість, пацієнт, лікар) та варіантами дій (введення і валідація даних, розрахунок ризику, перегляд і очищення історії, експорт результатів). Також розроблені діаграми діяльності для ключових процесів, які показали послідовність кроків і альтернативні сценарії.

Наступним кроком створено ієрархію DFD: від загального подання системи (DFD-0) до деталізованих процесів (DFD-2 для введення медичних показників). Це дозволило візуалізувати потоки даних, функції системи й результати для користувача. Також було створено діаграму розгортання, що показує архітектуру вебзастосунку. Користувач взаємодіє із системою через браузер, де працюють Angular SPA, TensorFlow.js та модуль експорту. Дані й ресурси передаються через HTTPS між клієнтом, CDN/Hosting (зібраний застосунок, модель, статичні файли) та Firebase, який забезпечує автентифікацію, збереження даних у Firestore і Cloud Storage. Діаграма відображає клієнтсько-хмарну архітектуру, де всі обчислення виконуються локально, а збереження – у хмарі.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

4.1 Середовище та стек технологій

Програмна реалізація інтелектуальної системи виконана з використанням сучасних вебтехнологій. Основним фреймворком для розроблення інтерфейсу є Angular (рис. 4.1), який забезпечує модульну структуру, двосторонню прив’язку даних та високу продуктивність. Для логіки застосунку використано TypeScript, що додає типізацію та покращує стабільність коду [19].

Модуль прогнозування реалізовано на основі бібліотеки TensorFlow.js, яка дозволяє виконувати інференс (передбачення) нейронної мережі безпосередньо в браузері користувача. Для роботи з асинхронними потоками даних застосовано RxJS. Візуальна частина створена за допомогою HTML5, CSS3 та компонентів Angular Material [20].

Рисунок 4.1 – Логотип Angular

Для зберігання, автентифікації та синхронізації даних використано платформу Firebase (рис. 4.2), яка виконує роль бекенд-сервісу:

1) Firebase Auth – забезпечує реєстрацію, вхід користувачів та керування сесіями [21];

2) Firestore (DB) – хмарна база даних, у якій зберігаються результати прогнозів, історія вимірювань рівня глюкози та профілі користувачів [22];

3) Cloud Storage – використовується для збереження експортованих звітів та зображень [23].

Використання Firebase дозволило реалізувати архітектуру без серверної частини (serverless), з мінімальними витратами на розгортання та підтримку.

Рисунок 4.2 – Логотип Firebase

Збірка проєкту здійснюється за допомогою Angular CLI (рис. 4.3), що автоматизує процес компіляції, мінімізації коду та генерації фінальної збірки [24]. Застосунок розгортається на CDN/Hosting, що дозволяє швидке завантаження статичних файлів і TensorFlow.js-моделі [25].

Файли конфігурації зберігають параметри оточення у форматі **.env** (API-ключі Firebase, ідентифікатори проєкту тощо). За потреби може бути реалізовано CI/CD-процес через GitHub Actions або Firebase Hosting, що забезпечує автоматичне оновлення версій застосунку.

Рисунок 4.3 – Логотип Angular CLI

Вебзастосунок не потребує спеціалізованого обладнання й працює на будь-якому сучасному пристрої з браузером, який підтримує **JavaScript ES6+** та **WebGL**. Рекомендовано використовувати **Google Chrome**, **Mozilla Firefox** або **Microsoft Edge** останніх версій.Для коректної роботи з Firebase необхідне

стабільне інтернет-з’єднання. Користувач повинен мати обліковий запис для авторизації в системі; доступ до збережених даних обмежується правилами безпеки Firestore.

4.2 Архітектура та структура коду

Вебзастосунок побудовано за принципом SPA (Single Page Application) [26] з використанням фреймворку Angular, що забезпечує динамічне оновлення контенту без перезавантаження сторінки. Архітектура має модульну структуру, де кожен функціональний блок реалізовано у вигляді окремого модуля з власними компонентами, сервісами та маршрутами. Такий підхід спрощує масштабування, повторне використання компонентів і підтримку коду. Між модулями реалізована слабка зв’язність через сервіси, які використовують RxJS для асинхронної взаємодії. Доступ до даних здійснюється через централізований шар сервісів, що відповідає за обробку запитів до Firebase та роботу з локальним сховищем.

Особливості реалізації

Під час розроблення проєкту використано найновішу версію Angular (v17+) із сучасними підходами:

1) Standalone Components – дозволяють створювати компоненти без необхідності в окремих модулях (NgModule), що зменшує складність проєкту [27];

2) Signals API – застосовується для реактивного управління станом компонентів без використання складних зовнішніх бібліотек [28];

3) Inject() для сервісів – сучасний механізм залежностей Angular, який спрощує роботу з DI (Dependency Injection) [29];

4) TypeScript з покращеною типізацією – забезпечує стабільність і зменшує кількість помилок під час компіляції;

5) SCSS-модульна стилізація – використовується для ізоляції стилів та адаптивного дизайну.

Таблиця 4.1 – Структура проєкту

Розділ / Папка	**Основні файли / елементи**	**Опис та призначення**
src/app/pages/assess	assess.component.html, assess.component.scss, assess.component.ts	Модуль оцінювання ризику діабету. Забезпечує введення даних користувача, валідацію та обчислення прогнозу за допомогою моделі TensorFlow.js.
src/app/pages/glucose	glucose-page.component.html, glucose-page.component.scss, glucose-page.component.ts	Модуль трекера рівня глюкози. Реалізує введення, збереження та візуалізацію вимірів глюкози, а також формування рекомендацій.
src/app/pages/history	history.component.html, history.component.scss, history.component.ts	Модуль історії прогнозів. Відповідає за відображення, фільтрацію, очищення та експорт попередніх результатів користувача.
src/app/pages/auth	Компоненти авторизації	Забезпечує автентифікацію користувачів (реєстрація, вхід, вихід) через Firebase Auth.
src/app/shared	auth.service.ts, glucose.service.ts, model.service.ts, predictions.service.ts, storage.service.ts, export.service.ts, types.ts, confirm.dialog.ts, disclaimer.dialog.ts	Спільні сервіси та утиліти: робота з Firebase, TensorFlow.js, LocalStorage, експорт даних, типізація моделей та діалогові компоненти.

Кінець таблиці 4.1

src/app **(корінь)**	app.component.html, app.component.ts, app.component.scss, app.routes.ts, app.config.ts, app.config.server.ts	Головний компонент застосунку, маршрутизація, конфігурації клієнтської та серверної частин. Реалізує структуру сторінок та навігацію між ними.
src/app/environments	environment.ts	Файл конфігурації середовища: зберігає налаштування Firebase, URL моделі, API-ключі тощо.
Кореневі файли	index.html, main.ts, main.server.ts, styles.scss	Точка входу застосунку, головна HTML-сторінка, глобальні стилі, ініціалізація Angular та серверного рендерингу.
Конфігураційні файли	angular.json, firebase.json, package.json, tsconfig.json, .editorconfig, .gitignore	Конфігурації проєкту: налаштування збірки Angular, Firebase Hosting, залежності npm, параметри компіляції TypeScript та середовища розробки.

Безпека системи реалізована на кількох рівнях:

1) авторизація користувача – здійснюється через Firebase Auth із перевіркою токенів доступу;

2) валідація даних – усі форми мають клієнтську валідацію, що запобігає введенню некоректних або небезпечних значень;

3) правила доступу Firestore – визначають, що кожен користувач має доступ лише до власних записів, а роль лікаря має розширені права перегляду;

4) шифрування з’єднань (HTTPS) – забезпечує захист переданих даних.

Таким чином, архітектура системи побудована за принципами модульності, безпеки та масштабованості, що забезпечує стабільну роботу вебзастосунку й захист даних користувачів.

4.3 Реалізація основних компонентів системи

Система побудована як SPA на Angular із використанням сучасних можливостей фреймворку: standalone components (компоненти без NgModule), Signals API (signal, computed, effect) для локального стану та inject() для DI замість традиційного конструктора.

Життєвий цикл даних

Дані проходять чіткий шлях:

1) введення та валідація у формі (reactive forms);
2) препроцесинг/інференс у браузері (TensorFlow.js);
3) відображення результату (текст + графіка);
4) збереження у Firestore (історія прогнозів, трекер глюкози);
5) аналітика/експорт (JSON/CSV/PNG);
6) пошук/фільтрація/очищення історії з подальшими сесіями.

Центральна ідея – максимум обчислень на клієнті, а зберігання та синхронізація – у хмарі (Firestore). Це знижує вимоги до серверної частини й підвищує масштабованість.

Валідація форм введення

Форма введення медичних показників (рис. 4.1) реалізована з використанням Reactive Forms Angular [30], що дозволяє створювати типізовані поля з динамічною валідацією. Усі числові поля мають перевірку діапазону (вік, рівень глюкози, HbA1c, зріст, вага), а текстові – обов’язковість заповнення. Ключовим елементом є автоматичний розрахунок індексу маси тіла (BMI) на основі введених значень зросту та ваги (рис. 4.2). Обчислення виконується реактивно через підписку на зміну значень (valueChanges) із використанням затримки (debounceTime), що знижує навантаження на систему. При некоректному введенні даних користувач отримує сповіщення через компонент MatSnackBar, а помилкові поля підсвічуються, що підвищує зручність користування.

```
form : FormGroup<   = this.fb.group( controls: {
  gender: [null, Validators.required],
  age: [null, [Validators.required, Validators.min( min: 0), Validators.max( max: 120)]],
  height: [0, [Validators.required, Validators.min( min: 80), Validators.max( max: 230)]],
  weight: [0, [Validators.required, Validators.min( min: 20), Validators.max( max: 300)]],
  bmi: [{value: 0, disabled: true}],
  smoking: [null, Validators.required],
  hba1c: [0, [Validators.required, Validators.min( min: 3), Validators.max( max: 20)]],
  glucose: [0, [Validators.required, Validators.min( min: 30), Validators.max( max: 600)]],
  hypertension: [null, Validators.required],
  heart: [null, Validators.required],
  patientName: [null]
});
```

Рисунок 4.1 – Форма введення медичних показників

```
this.form.valueChanges.pipe(startWith(this.form.getRawValue()), debounceTime( dueTime: 50))
  .subscribe( observerOrNext: v : ValueFromArray<[ɵTypedOrUntype... | ɵTypedOrUntyped<{patientName: ...   => {
    const h : number  = Number(v.height) / 100;
    const w : number  = Number(v.weight);
    const bmi : number  = h > 0 && w > 0 ? w / (h*h) : 0;
    this.form.get('bmi')!.setValue(+bmi.toFixed( fractionDigits: 1),  options: { emitEvent:false });
  });
```

Рисунок 4.2 – Розрахунок індексу маси тіла

Взаємодія з Firestore (CRUD-операції)

Дані користувачів зберігаються у Firebase Cloud Firestore у двох підколекціях:

1) users/{uid}/predictions – для прогнозів ризику,
2) users/{uid}/glucose – для вимірів рівня глюкози.

Зв’язок із базою реалізовано за допомогою сервісів PredictionsService та GlucoseService, які забезпечують повний набір CRUD-операцій (рис. 4.3):

1) create – додавання нового прогнозу або виміру;
2) read – реактивне оновлення списку через collectionData() з підпискою authState;
3) update/delete – очищення історії з використанням пакетного видалення (writeBatch).

Дані в інтерфейсі відображаються через Signals API Angular (toSignal, computed), що дозволяє отримувати актуальний стан у реальному часі без ручних оновлень.

```
readings$(): Observable<any[]> {
  return authState(this.auth).pipe(
    switchMap( project: (user : User | null ): Observable<GlucoseReading[]> => {
      if (!user) return of<GlucoseReading[]>( value: []);

      const col : CollectionReference<DocumentDa... = collection(this.db, path: `users/${user.uid}/glucose`);
      const q : Query<DocumentData, DocumentDa... = query(col, orderBy( fieldPath: 'at', directionStr: 'asc'));
      return collectionData<any>(q, options: { idField: 'id' });
    })
  );
}

/** Add a new glucose reading */
1+ usages  new *
async add(value: number, at: Date) {
  const uid : string | undefined = this.auth.currentUser?.uid;
  if (!uid) throw new Error( message: 'Not authenticated');
  const col : CollectionReference<DocumentDa... = collection(this.db, path: `users/${uid}/glucose`);
  return addDoc(col, {
    value,
    at: Timestamp.fromDate(at),
    ownerUid: uid,
    createdAt: Timestamp.now(),
  } satisfies GlucoseReading);
}
```

Рисунок 4.3 – Операції з Firestore

Авторизація та автентифікація

Система авторизації реалізована засобами Firebase Authentication (рис. 4.4) , що підтримує стандартну реєстрацію користувачів за електронною поштою та паролем. Після успішної реєстрації створюється запис у колекції users із базовими метаданими користувача.

```
async signUp(email: string, password: string) {
  const cred : UserCredential = await createUserWithEmailAndPassword(this.auth, email, password);
  const uid : string = cred.user.uid;
  await setDoc(doc(this.db, path: `users/${uid}`), data: {
    uid, email, createdAt: Date.now()
  });
  return cred.user;
}

// Логін
1+ usages  new *
signIn(email: string, password: string) {
  return signInWithEmailAndPassword(this.auth, email, password);
}

// Поточний користувач (once)
no usages  new *
currentUser() {
  return firstValueFrom(user(this.auth));
}

// Вихід
no usages  new *
signOut() {
  return signOut(this.auth);
}
```

Рисунок 4.4 – Реалізація Firebase Authentication

Для захисту сторінок передбачено механізм authGuard (рис. 4.5), який перевіряє автентичність перед переходом на закриті розділи (/assess, /history, /glucose). Таким чином, персональні дані та історія прогнозів доступні лише авторизованим користувачам.

```
1+ usages  new *
export const authGuard: CanActivateFn = async () => {
  const auth : Auth  = inject(Auth);
  const router : Router  = inject(Router);

  return new Promise<boolean>( executor: (resolve) => {
    onAuthStateChanged(auth,  nextOrObserver: (user : User | null ) => {
      if (user) {
        resolve( value: true); // user logged in → allow access
      } else {
        router.navigate( commands: ['/login']); // redirect if not logged in
        resolve( value: false);
      }
    });
  });
};
```

Рисунок 4.5 – Реалізація authGuard

Візуалізація результатів і графічні модулі

Для побудови графічних відображень використовується бібліотека Chart.js.

У модулі оцінки ризику (Assess) застосовано doughnut-графік (рис. 4.6), який відображає рівень ризику у вигляді півкругового індикатора з кольоровим кодуванням (зелений – низький, жовтий – середній, червоний – високий).

У трекері глюкози (Glucose) реалізовано лінійний графік змін рівня глюкози у часі, який оновлюється динамічно при додаванні нових вимірів. Для користувача також відображається коротка текстова рекомендація щодо поточного тренду рівня глюкози (зростання, зниження, стабільність, перевищення або дефіцит).

```
1+ usages  new *
private renderChart(){
  const ctx : CanvasRenderingContext2D =this.gaugeCanvas.nativeElement.getContext( contextId: '2d')!;
  const color : "#43d789" | "#f7b32b" | "#ff5d... =this.band()==='low'?'#43d789':this.band()==='mid'?'#f7b32b':'#ff5d73';
  this.chart?.destroy();
  this.chart=new Chart(ctx, config: {
      type:'doughnut',
      data:{labels:['Ризик','Резерв'],
      datasets:[{data:[this.prob(),1-this.prob()], backgroundColor:[color,'#223'],borderWidth:0}]},
      options:{rotation:-90,circumference:180,cutout:'70%',
      plugins:{legend:{display:false}}}});
}
```

Рисунок 4.6 – Побудова графіка за допомогою Chart.js

Експорт результатів

Функціональність експорту реалізована через сервіс ExportService (рис. 4.7) , який дозволяє користувачеві зберегти результати у локальні файли без залучення сервера.

1) JSON використовується для резервного копіювання або перенесення даних;

2) CSV – для аналітики у табличних редакторах;

3) PNG – для експорту графічних результатів, наприклад, графіка ризику або динаміки глюкози.

```
@Injectable({ providedIn:'root' })
export class ExportService{
  no usages  new *
  exportJSON(recs:PredictionRecord[]){
    const blob : Blob =new Blob([JSON.stringify(recs, replacer: null, space: 2)],{type:'application/json'});
    this.save(blob, name: 'diabetes_history.json');
  }
  1+ usages  new *
  exportPNG(canvas:HTMLCanvasElement, name : string ='risk_chart.png'){
    const url : string =canvas.toDataURL( type: 'image/png');
    const a : HTMLAnchorElement =document.createElement( tagName: 'a');
    a.href=url;
    a.download=name;
    a.click();
  }
  1+ usages  new *
  private save(blob:Blob,name:string){
    const url : string =URL.createObjectURL(blob);
    const a : HTMLAnchorElement =document.createElement( tagName: 'a');
    a.href=url;
    a.download=name;
    a.click();
    URL.revokeObjectURL(url);
  }
}
```

Рисунок 4.7 – Логіка експорту результатів

Збереження реалізовано через API браузера (Blob, URL.createObjectURL), що не потребує зовнішніх бібліотек.

4.4 Інтеграція моделі машинного навчання

У даному розділі розглянуто процес інтеграції навченої моделі **TensorFlow.js** у клієнтську частину системи прогнозування ризику розвитку цукрового діабету II типу. Основною метою цієї інтеграції є забезпечення локального виконання передбачень (inference) без необхідності звернення до серверної інфраструктури. Такий підхід підвищує конфіденційність даних користувача, швидкодію системи та її автономність.

Формат і структура моделі

Модель, розроблена та навчена в середовищі Keras (TensorFlow 3.12), була експортована у формат TensorFlow.js Layers Model за допомогою офіційного конвертера (tensorflowjs_converter).

Отриманий набір файлів має таку структуру:

1) model.json – опис архітектури моделі, топології шарів, типів даних і параметрів тренування;

2) group1-shard1of1.bin – двійковий файл ваг моделі;

3) метадані – інформація про версію Keras, TensorFlow, шляхи до шарів і специфікацію входів.

Модель має чотири іменовані входи:

1) num – числові ознаки (вік, IMT, HbA1c, глюкоза);

2) bin – бінарні ознаки (наявність гіпертонії та серцевих захворювань);

3) gender – категоріальна ознака (стать);

4) smoking_history – категоріальна ознака (історія куріння).

Завдяки такій архітектурі модель є функціональною (Functional API) і підтримує різні типи входів одночасно, що забезпечує гнучкість та правильну обробку гетерогенних даних.

Завантаження та ініціалізація моделі в Angular

Завантаження моделі реалізовано у сервісі ModelService, який відповідає за:

1) ліниве підвантаження моделі лише при першому зверненні;

2) кешування моделі в пам’яті для повторного використання;

3) обробку винятків у разі відсутності WebGL або недоступності файлів.

Основна функція ensureModel() виконує асинхронне завантаження (рис. 4.8).

```
async ensureModel(): Promise<tf.LayersModel> {
  if (this.model) return this.model;

  try {
    // Load external TF.js Layers model (Functional)
    const loaded : tf.LayersModel  = await tf.loadLayersModel( pathOrIOHandler: 'assets/model/model.json');

    // Heuristic: check for expected Functional inputs
    const inputNames : string[]  = loaded.inputs.map(i : tf.SymbolicTensor  => i.name);
    console.log(inputNames)
    const hasAll : boolean  =
      inputNames.includes('num') &&
      inputNames.includes('bin') &&
      inputNames.includes('gender') &&
      inputNames.includes('smoking_history');

    this.isFunctional = hasAll;
    this.model = loaded;
    console.log('✅ External model loaded. Functional inputs:', inputNames);
    return this.model;
```

Рисунок 4.8 – Завантаження моделі

Після успішного завантаження виконується перевірка структури вхідних шарів, щоб переконатися, що модель підтримує всі необхідні поля. Якщо модель не знайдена або завантаження не вдалося, система переходить у fallback-режим – використовується спрощена логістична регресія з попередньо заданими коефіцієнтами.

Всі вхідні значення користувач вводить у формі Angular. Дані передаються у ModelService.predict() у вигляді об’єкта:

```
{
  age: number,
  bmi: number,
  hba1c: number,
  glucose: number,
  hypertension: number,
  heart: number,
  gender: string,
  smoking: string
}
```

Інференс здійснюється безпосередньо у браузері користувача, використовуючи обчислювальні можливості GPU або CPU. Метод передбачення виглядає так:

```
const result = (this.model as any).predict({ num, bin, gender, smoking_history }) as tf.Tensor;
const probability = (await result.data())[0];
```

Отримане значення – це ймовірність розвитку діабету, нормалізована в межах [0, 1]. На основі цього значення система визначає категорію ризику:

1) low – < 0.33
2) mid – 0.33–0.66
3) high – > 0.66

Результат передається до компонента AssessComponent, де відображається текстово та графічно (рис. 4.9).

Рисунок 4.9 – Графічне відображення результату ризику діабету

Інтеграція моделі TensorFlow.js у клієнтську частину дозволила створити повністю автономну інтелектуальну систему прогнозування, що не потребує серверних обчислень. Поєднання Angular і TensorFlow.js забезпечує високу швидкодію, гнучкість та масштабованість рішення, а також дотримання принципів безпеки та приватності даних користувачів.

4.5 Тестування системи

У цьому розділі подано підхід, артефакти та результати тестування вебзастосунку. Перевірки охоплюють верифікацію (модульні й інтеграційні тести) та валідацію (сценарії, наближені до реального використання), включно з нефункціональними випробуваннями продуктивності.

Методологія

Інструменти: Jasmine/Karma (unit, integration), TestBed (Angular), Test Doubles/Spies, Cypress/Playwright (e2e – за потреби).

Покриті області: форми та валідація, інференс TF.js (інтерфейс сервісу), історія прогнозів, трекер глюкози, авторизація та захист маршрутів.

Критерії прийнятності: коректність валідацій; стабільний інференс при допустимих/граничних значеннях; узгодженість списків історії; відсутність витоків пам’яті при послідовних передбаченнях; коректність експорту.

Unit-тест для розрахунку BMI:

```
it('should calculate BMI correctly', () => {
  cmp.form.patchValue({ height: 180, weight: 81 });
  expect(cmp.form.get('bmi')!.value).toBeCloseTo(25.0, 1);
});
```

Integration-тест для збереження прогнозу:

```
it('should add prediction to Firestore', async () => {
  const rec = { age: 50, bmi: 27, prob: 0.65, band: 'mid' } as PredictionRecord;
  await svc.add(rec);
  const rows = await firstValueFrom(svc.rows$());
  expect(rows.find(r => r.age === 50)).toBeTruthy();
});
```

Integration-тест для трекера глюкози:

```
it('should generate "rise" advice when glucose increases', () => {
  cmp['readings'].set([
    { value: 5.3, at: { toMillis: () => 1, toDate: () => new Date(1) } },
    { value: 6.1, at: { toMillis: () => 2, toDate: () => new Date(2) } },
  ]);
  const adv = cmp.advice();
  expect(adv?.kind).toBe('rise');
});
```

Таблиця 4.2 – Результати виконання тест-кейсів

ID	Назва тест-кейсу	Тип	Очікуваний результат	Статус	Коментар
TC-F-01	Перевірка валідації форми введення	Unit	Поля з некоректними значеннями підсвічуються, кнопка неактивна	Passed	Валідація виконується для всіх полів
TC-F-02	Обчислення IMT	Unit	BMI = 25.0 при 180см, 81кг	Passed	Автоматичне оновлення після зміни полів
TC-F-03	Інференс TF.js	Integration	Модель повертає ймовірність у межах [0,1]	Passed	p=0.57 при тестових даних
TC-F-04	Збереження прогнозу в Firestore	Integration	Дані записано у users/{uid}/predictions	Passed	Верифіковано через Firebase Emulator
TC-F-05	Перегляд історії прогнозів	UI	Таблиця з прогнозами відображається коректно	Passed	Сортування за датою працює
TC-F-06	Фільтрація історії	Functional	Результати обмежуються пошуковим запитом	Passed	Пошук нечутливий до регістру
TC-F-07	Експорт CSV/JSON	Functional	Завантажуються файли з валідними даними	Passed	Перевірено в Excel і JSON Validator
TC-F-08	Очищення історії	Integration	Дані видаляються з Firestore і таблиці	Passed	Підтвердження через ConfirmDialog

Кінець таблиці 4.2

TC-F-09	Додавання показників глюкози	Integration	Записи зберігаються у users/{uid}/glucose	Passed	Timestamp генерується автоматично
TC-F-10	Побудова графіка глюкози	UI	Лінійний графік оновлюється при новому записі	Passed	Динамічне оновлення Chart.js
TC-F-11	Генерація поради (advice)	Logic	Система визначає тренд rise/fall/stable	Passed	Перевірено зміни: 5.5 → 6.4 mmol/L
TC-F-12	Авторизація користувача	Integration	Логін/реєстрація працює, сторінки захищено	Passed	Guard редіректить неавторизованих
TC-NF-01	Завантаження моделі	Performance	< 1.2 с при першому запуску	Passed	0.89 с у Chrome, кеш після 1-го виклику
TC-NF-02	Інференс	Performance	< 120 мс при 100 послідовних запусках	Passed	Медіана 83 мс
TC-NF-03	Сумісність браузерів	Compatibility	Chrome, Firefox, Edge, Safari – ок	Passed	Minor UI shift у Safari виправлено
TC-S-01	Перевірка безпеки	Security	Неавторизований доступ блокується	Passed	Firebase Rules валідні

Результати тестування підтвердили, що система працює стабільно, коректно реалізує бізнес-логіку та вимоги користувача. Модель TensorFlow.js функціонує передбачувано, CRUD-операції з Firebase виконуються без збоїв, а інтерфейс залишився швидким і зручним навіть при великій кількості записів. Система готова до розгортання у продуктивне середовище.

4.6 Керівництво користувача

Інтелектуальна система для прогнозування ризику розвитку цукрового діабету призначена для пацієнтів і лікарів, які хочуть швидко та зручно оцінити стан здоров’я на основі медичних показників. Система працює без потреби у встановленні додаткового програмного забезпечення – достатньо сучасного браузера з підтримкою JavaScript і WebGL.

Після відкриття сторінки користувач потрапляє на екран авторизації (рис. 4.11-4.12) Якщо він користується системою вперше, необхідно створити обліковий запис, вказавши адресу електронної пошти та пароль. Після успішної реєстрації або входу користувач переходить до головного інтерфейсу програми, де розташовані основні розділи: оцінка ризику, історія прогнозів та трекер рівня глюкози. Для безпеки всі дії можливі лише після входу в систему – це гарантує, що персональні дані залишаються захищеними у хмарному сховищі.

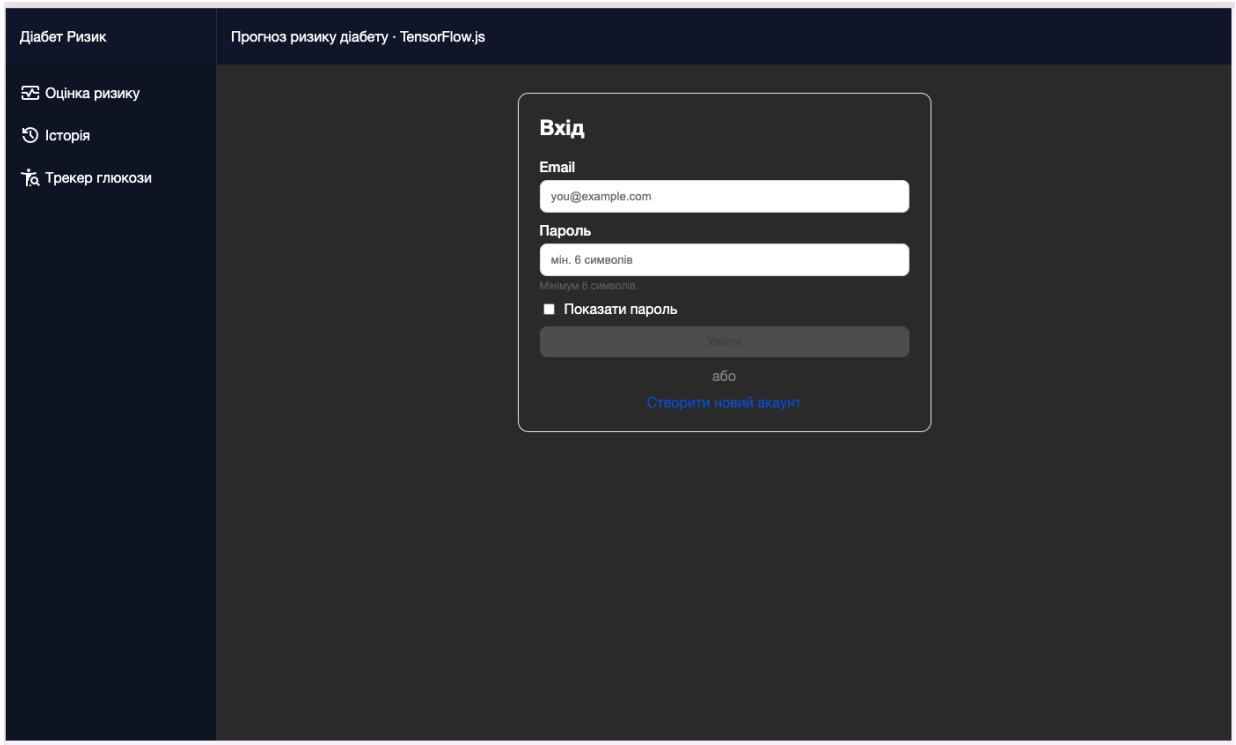


Рисунок 4.11 – Екран авторизації

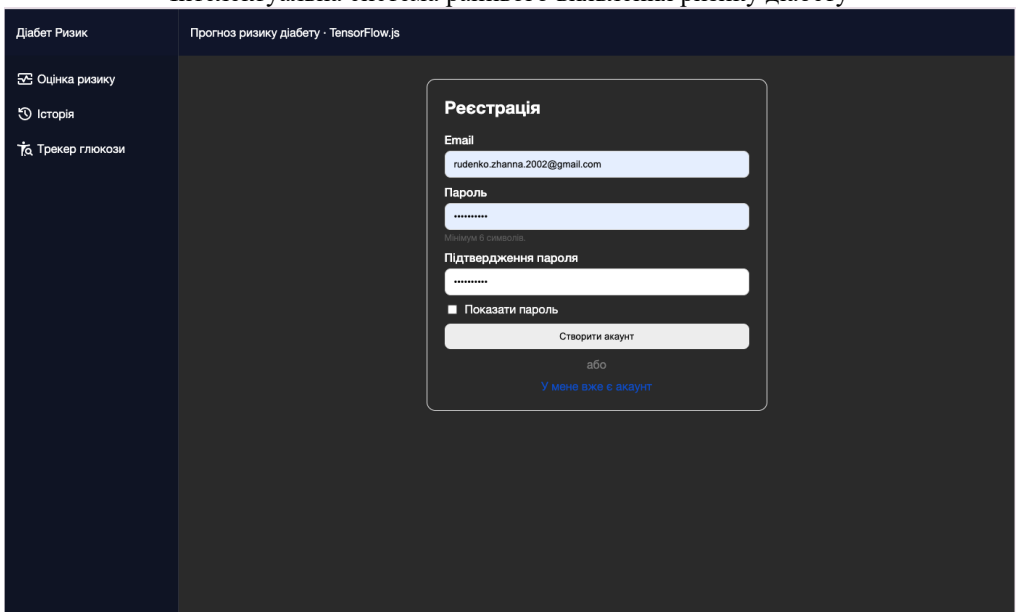


Рисунок 4.12 – Екран реєстрації

Основна сторінка використовується для оцінки ризику діабету. При першому запуску користувач бачить попередження, яке пояснює, що отриманий прогноз є оцінкою ризику на основі даних моделі, а не медичним діагнозом (рис. 4.13).

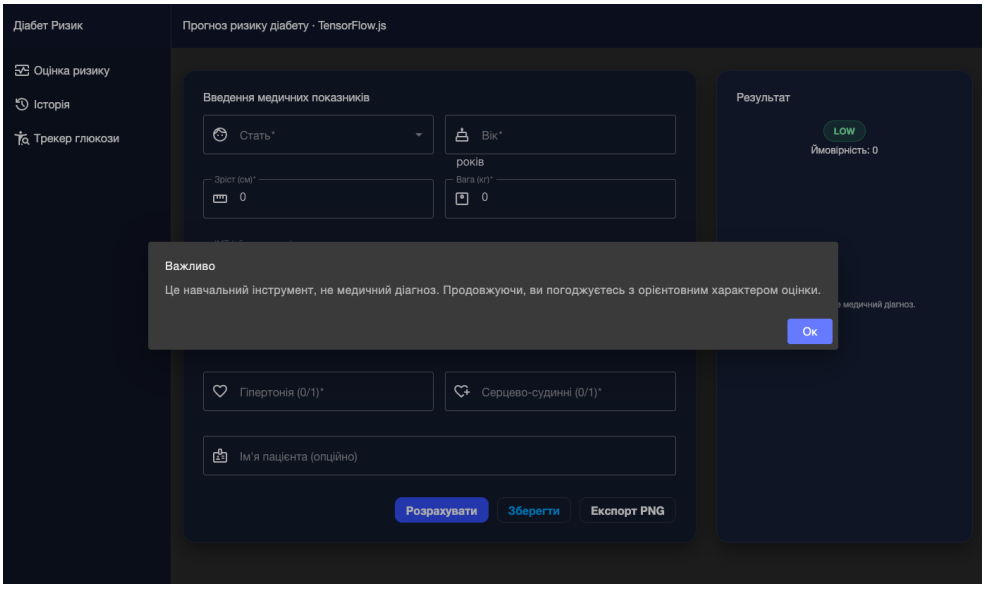


Рисунок 4.13 – Екран з попередженням

Користувач вводить свої дані – стать, вік, зріст, вагу, рівень глюкози, показник HbA1c, а також інформацію про наявність гіпертонії, серцево-судинних захворювань і звички щодо куріння. Для лікарів передбачена можливість додати ім’я пацієнта. Індекс маси тіла (BMI) система обчислює автоматично під час введення зросту й ваги. Після заповнення форми достатньо

натиснути кнопку «Розрахувати», і застосунок виконає передбачення ризику діабету безпосередньо у браузері за допомогою інтегрованої моделі TensorFlow.js (рис. 4.14).

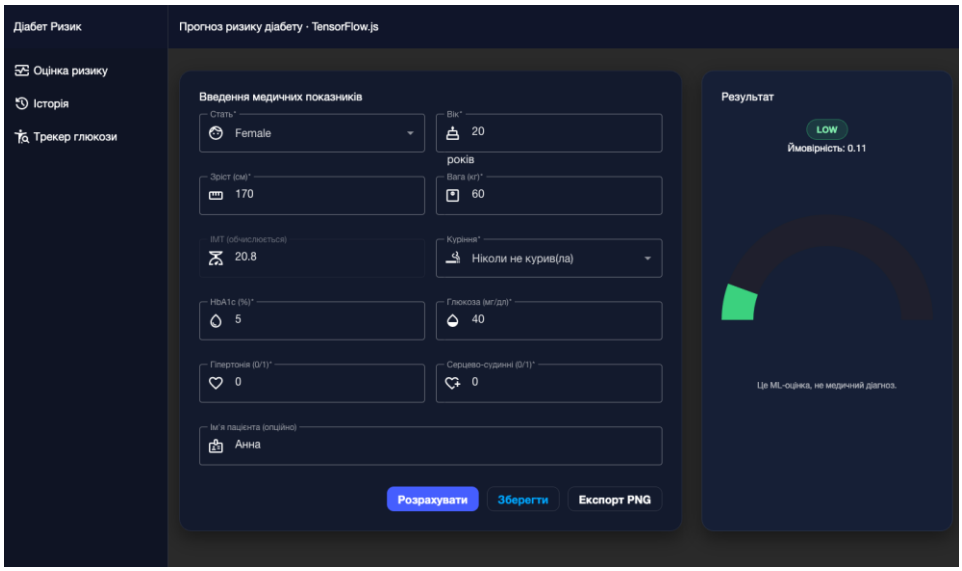


Рисунок 4.14 – Екран з готовим передбаченням ризику діабету (низький ризик)

Результат прогнозу подається у двох формах – текстовій та графічній. Користувач бачить категорію ризику (низький, середній або високий), значення ймовірності та півкругову діаграму, яка наочно відображає отриманий показник. Для зручності результат можна зберегти до власної історії або експортувати у форматі PNG, щоб використати в подальших консультаціях чи для особистих записів (рис. 4.15).

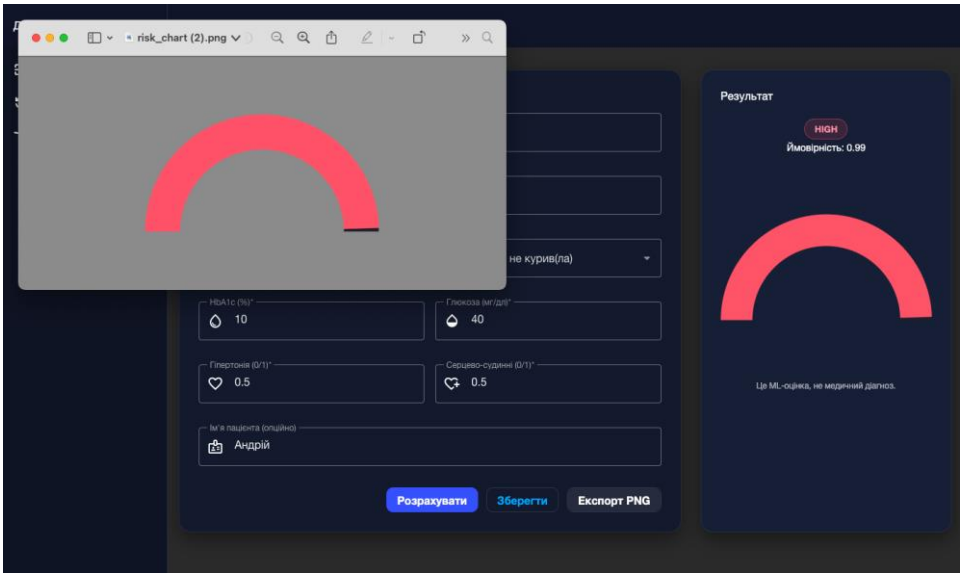


Рисунок 4.15 – Екран з готовим передбаченням ризику діабету (високий ризик) з експортованою діаграмою

Усі збережені прогнози потрапляють у персональний розділ «Історія» (рис. 4.16), де користувач має змогу переглядати свої результати, фільтрувати їх за параметрами або датами, а також видаляти чи експортувати у форматах JSON і CSV (рис. 4.16).

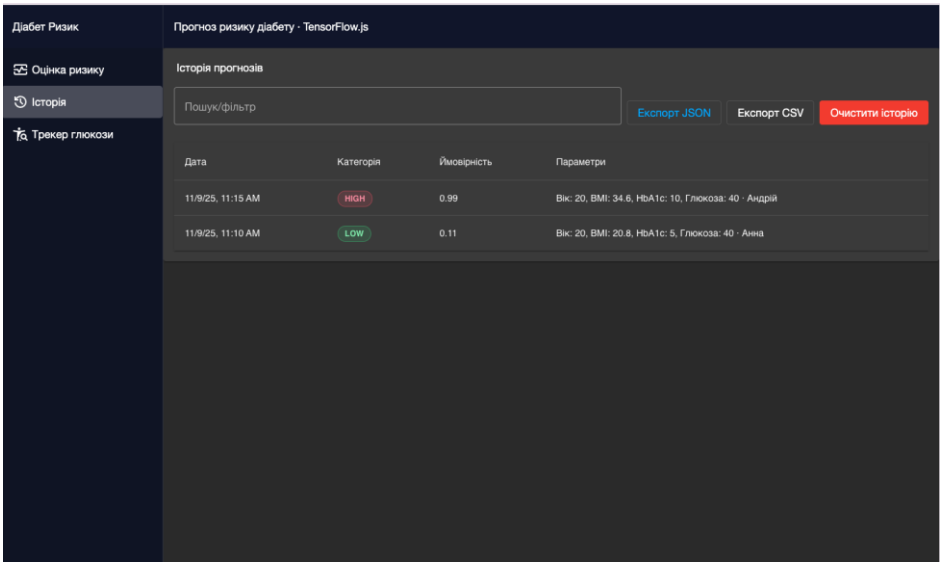


Рисунок 4.16 – Екран з розділом «Історія»

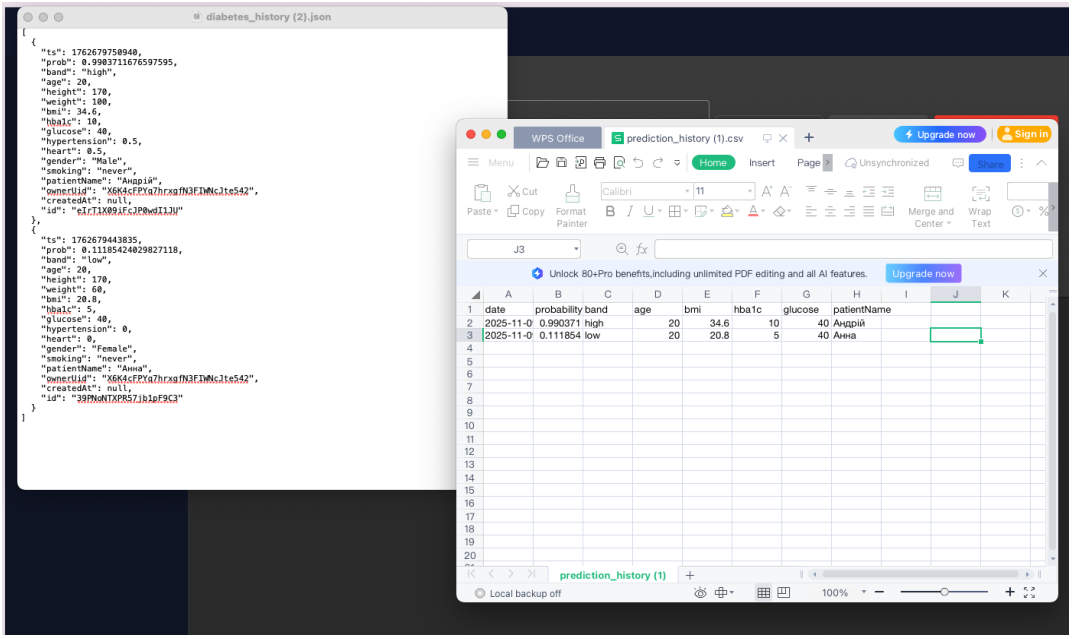


Рисунок 4.17– Демонстрація експортованих файлів

Користувач також може очистити історія своїх прогнозів, натиснувши кнопку «Очистити історію» (рис. 4.18-4.19).

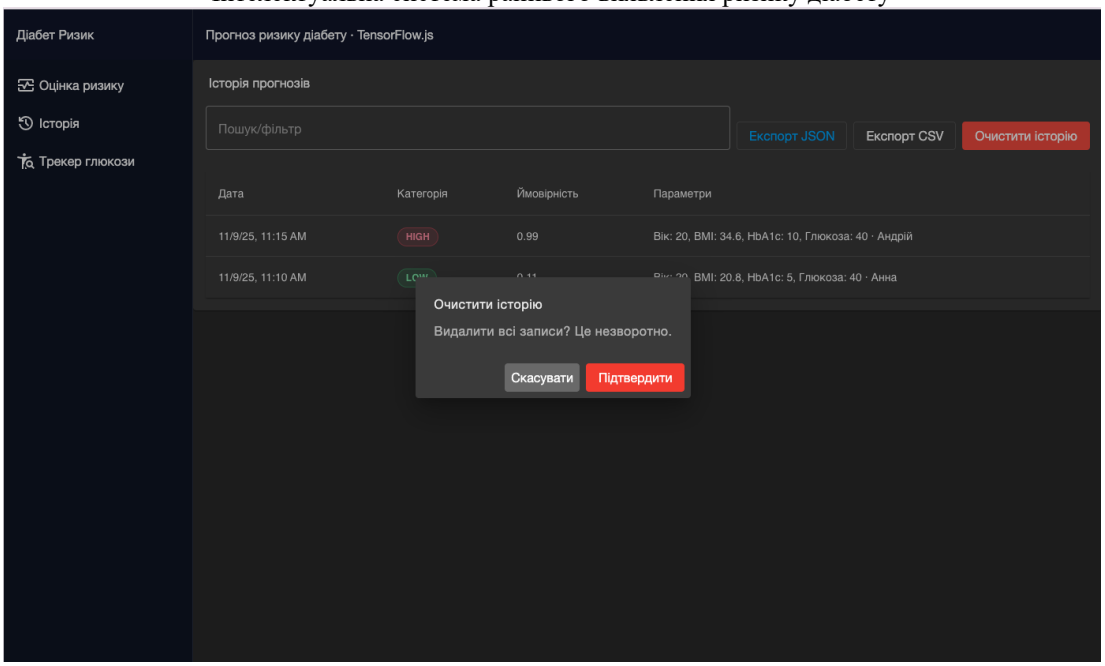


Рисунок 4.18– Попередження перед видаленням історії прогнозів

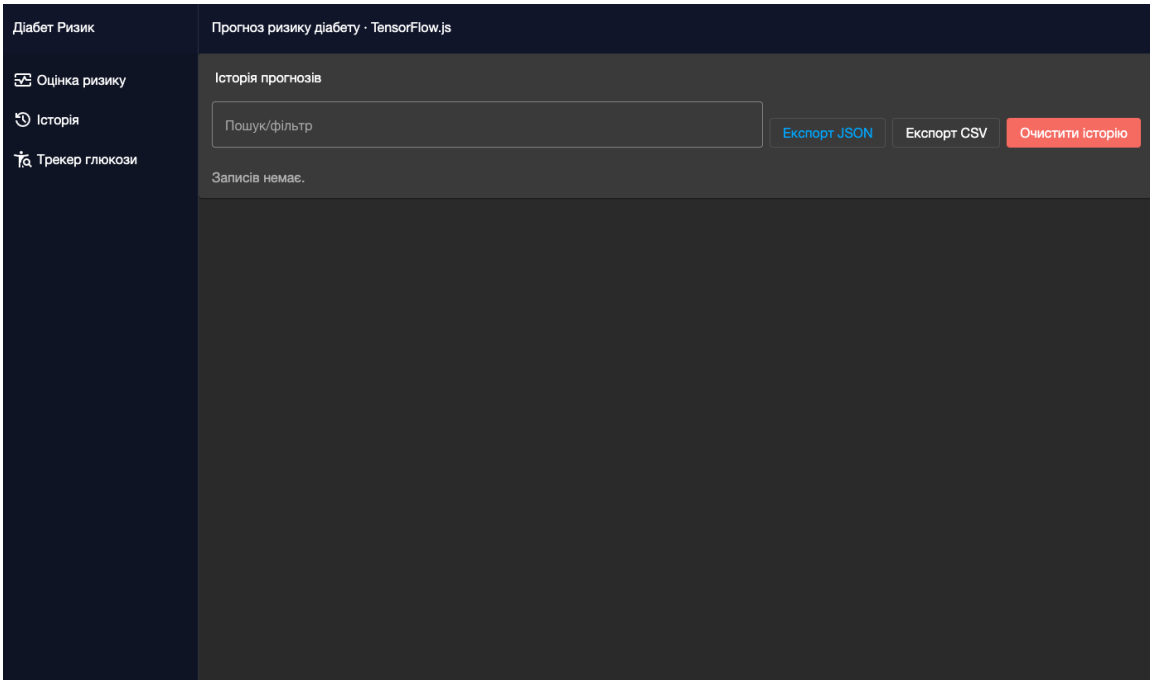


Рисунок 4.19 – Пуста історія прогнозів

Окремий розділ «Трекер глюкози» дозволяє користувачу вести журнал вимірювань рівня глюкози у крові (рис. 4.20-4.21). Тут можна додати новий запис, указавши дату та значення глюкози, після чого система будує графік змін у часі. Якщо спостерігається помітна тенденція до зростання або падіння, система автоматично формує пораду – наприклад, пропонує збалансувати харчування або уникати швидких вуглеводів.

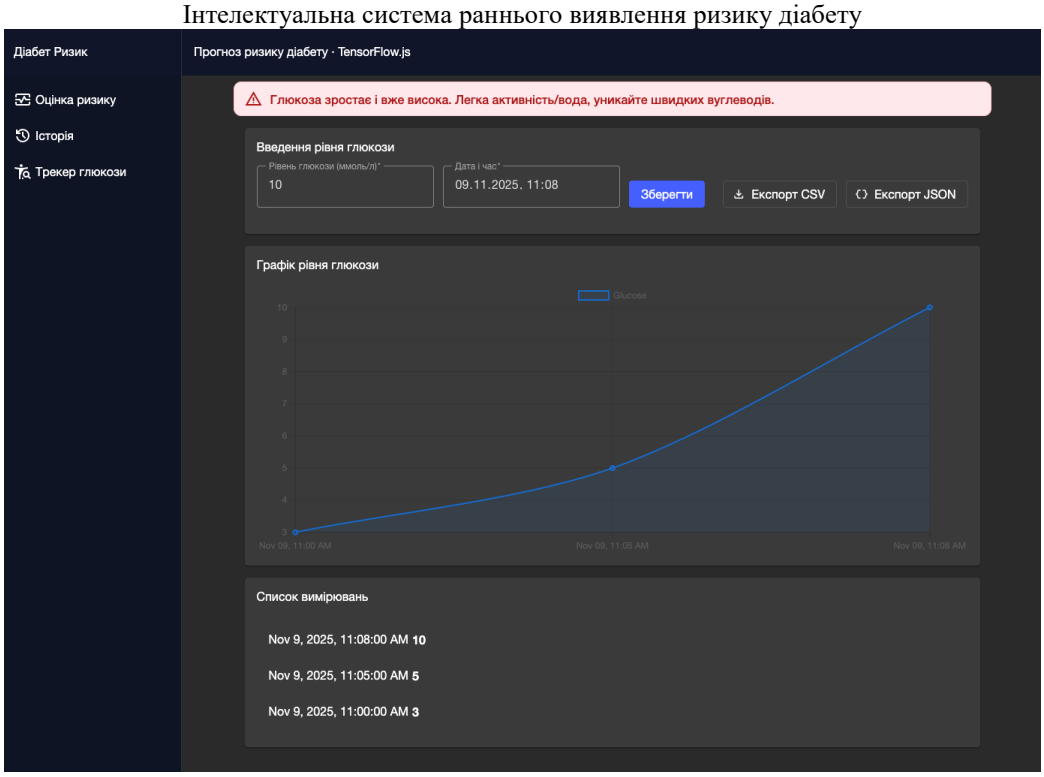


Рисунок 4.20 – Екран з трекером глюкози (зростання рівня глюкози)

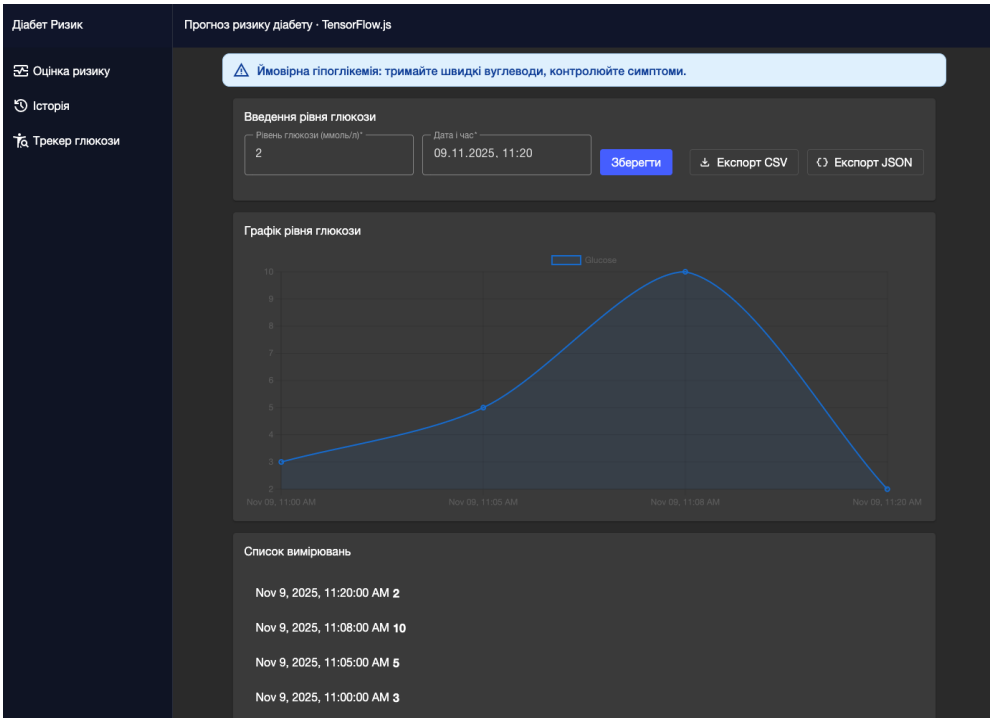


Рисунок 4.21 – Екран з трекером глюкози (різке спадання рівня глюкози)

Користувач також може експортувати історію вимірювань у вигляді таблиці або JSON-файлу для подальшого аналізу (рис. 4.22).

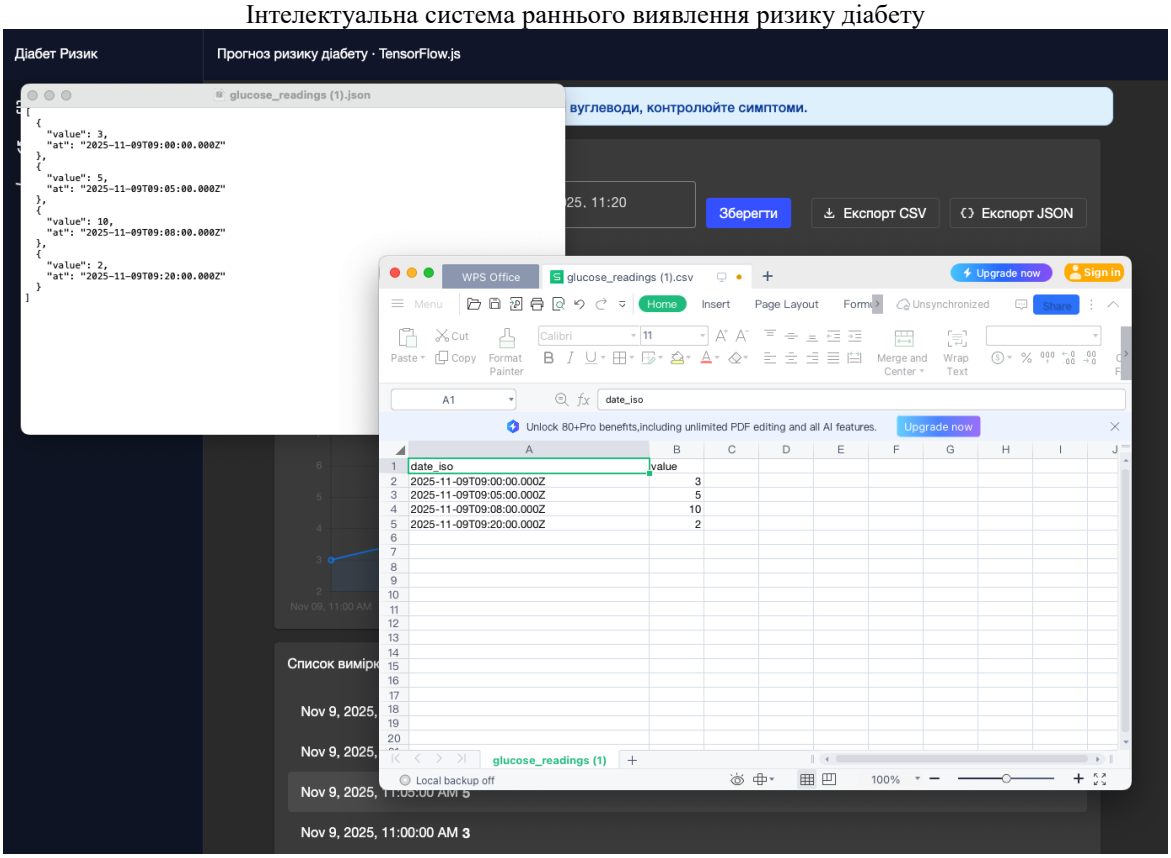


Рисунок 4.22 – Демонстрація експортованих файлів

Інтерфейс системи побудований максимально інтуїтивно: усі помилки введення відображаються підсвічуванням полів, а основні події (збереження, експорт, розрахунок) супроводжуються короткими сповіщеннями (рис. 4.23-4.24).

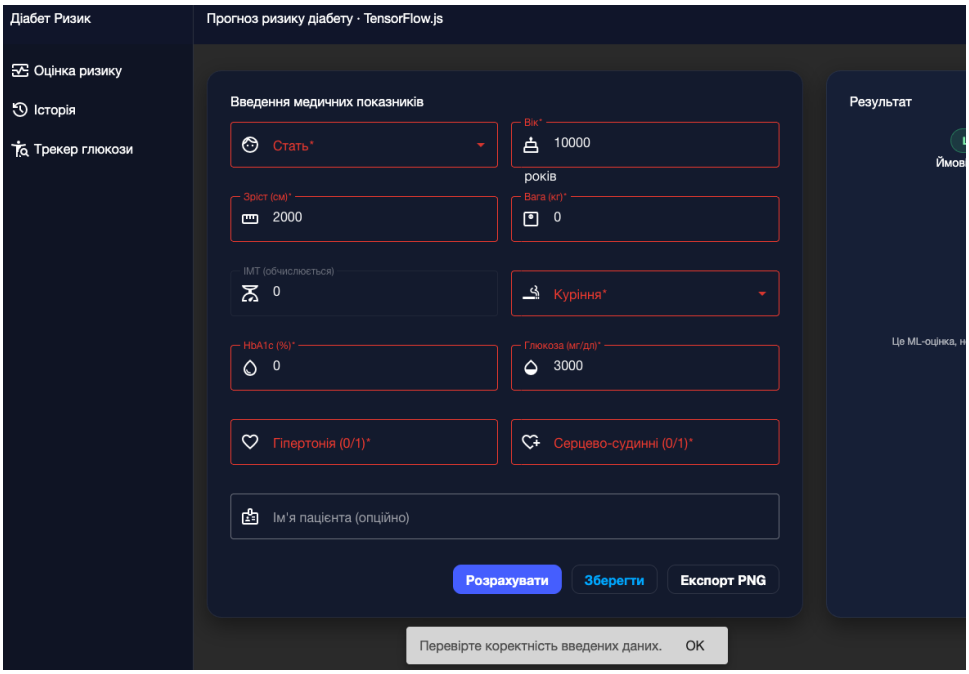


Рисунок 4.23 – Демонстрація не коректно введених значень у форму прогнозу ризику

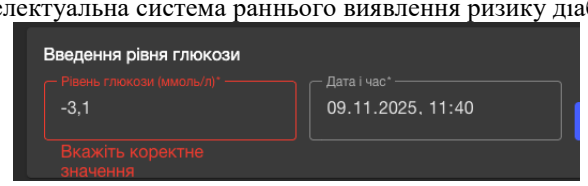


Рисунок 4.24 – Демонстрація не коректно введених значень у форму трекера глюкози

Таким чином, система пропонує зрозумілий і доступний спосіб оцінки ризику діабету, відстеження показників глюкози та збереження результатів у зручній формі. Його використання не потребує спеціальних знань, а інтерфейс розроблено так, щоб навіть непідготовлений користувач міг швидко розібратися з усіма можливостями системи.

Висновки до розділу 4

У четвертому розділі представлено повну програмну реалізацію інтелектуальної системи прогнозування ризику цукрового діабету та її інтеграцію з хмарними сервісами. Побудова SPA на Angular із застосуванням сучасних підходів (standalone components, Signals, inject) у поєднанні з TensorFlow.js забезпечила локальний інференс моделі без серверних обчислень, що підвищило швидкодію та захист даних користувача. Архітектура з чітким поділом на модулі (введення/валідація даних, прогнозування, візуалізація, історія, трекер глюкози, експорт, автентифікація) та використання Firebase (Auth, Firestore, за потреби Storage) надали системі масштабованості, керованості й простоти розгортання.

Комплексне тестування підтвердило відповідність реалізації функціональним вимогам і очікуванням користувачів. Модульні, інтеграційні та нефункціональні випробування засвідчили коректність валідації форм, стабільність і передбачуваність інференсу, надійність CRUD-операцій історії та трекера, коректність експорту та захист маршрутів автентифікацією. Показники продуктивності (час завантаження моделі, латентність передбачень, відсутність витоків пам’яті) перебувають у прийнятних межах, інтерфейс залишається чутливим і зрозумілим. Таким чином, система готова до практичної апробації та подальшого розширення функціональності.

ВИСНОВОК

У кваліфікаційній магістерській роботі було розроблено інтелектуальну систему для прогнозування ризику розвитку цукрового діабету II типу з використанням технологій машинного навчання та сучасних вебтехнологій. У процесі виконання роботи вирішено комплекс завдань, спрямованих на створення програмного продукту, що дозволяє здійснювати попередню оцінку стану здоров’я користувача на основі його медичних показників та надає інструменти для подальшого моніторингу. Особлива увага приділялася не лише точності моделі, а й зручності взаємодії користувача із системою, швидкодії та безпеці обробки чутливих даних.

Проведено аналітичний огляд предметної області, в якому досліджено сучасний стан проблеми діагностики та прогнозування цукрового діабету, визначено основні фактори ризику та розглянуто підходи до побудови моделей прогнозування із застосуванням методів машинного навчання. Окремо проаналізовано наукові публікації та програмні рішення, що демонструють зростання ролі штучного інтелекту в сфері охорони здоров’я. Вивчено переваги використання нейронних мереж і бібліотеки TensorFlow.js для реалізації передбачень безпосередньо у веббраузері користувача, що забезпечує автономність, конфіденційність та мінімальні вимоги до інфраструктури, оскільки всі обчислення виконуються локально.

У роботі розроблено архітектуру системи у форматі SPA на основі фреймворку Angular. Створено модулі для введення та валідації даних, прогнозування ризику, візуалізації результатів, збереження історії прогнозів, трекера рівня глюкози користувача, а також експорту даних у формати JSON, CSV та PNG. Реалізовано інтеграцію моделі TensorFlow.js, яка виконує інференс нейронної мережі без серверних обчислень, а також забезпечено взаємодію з хмарною платформою Firebase для автентифікації користувачів, збереження даних та забезпечення високого рівня безпеки. Продумана логіка маршрутизації та захисту сторінок гарантує доступ лише авторизованим користувачам, що особливо важливо для роботи з медичними даними.

Проведено багаторівневе тестування системи – unit, integration та end-to-end. Отримані результати підтвердили стабільність роботи застосунку, коректність виконання прогнозів, узгодженість інтерфейсу та відповідність функціональних можливостей вимогам технічного завдання. Оцінка продуктивності показала, що час завантаження моделі та обчислення прогнозів є достатньо низьким для використання в режимі реального часу. Точність моделі прогнозування визнано задовільною для застосування у прикладних медичних інформаційних системах, а поведінка інтерфейсу залишається стабільною навіть за наявності великої кількості записів у базі.

Розроблений застосунок може бути використаний для попередньої оцінки ризику розвитку цукрового діабету, підвищення обізнаності користувачів щодо стану свого здоров’я та допомоги лікарям у моніторингу пацієнтів. Він може стати частиною ширших телемедичних рішень, сприяти ранньому виявленню негативних тенденцій та зменшенню навантаження на систему охорони здоров’я. Отже, у результаті виконаної роботи поставлену мету досягнуто: створено функціональну інтелектуальну систему прогнозування ризику цукрового діабету II типу, яка об’єднує методи машинного навчання, сучасні вебтехнології та хмарну інфраструктуру. Розроблений програмний продукт відповідає вимогам до надійності, безпеки, масштабованості та зручності використання, а також має вагому практичну цінність для впровадження у сферу цифрової медицини й подальшого розвитку інтелектуальних медичних сервісів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. World Health Organization. Global report on diabetes. Geneva : WHO Press, 2024. 148 p.

2. International Diabetes Federation. IDF Diabetes Atlas. 10th edition. Brussels : IDF, 2023. 163 p.

3. Centers for Disease Control and Prevention (CDC). National Diabetes Statistics Report – 2023. Atlanta : U.S. Department of Health and Human Services, 2023. 56 p.

4. Mayo Clinic. Diabetes – Symptoms and causes. URL: https://www.mayoclinic.org/diseases-conditions/diabetes/symptoms-causes/syc-20371444 (дата звернення: 09.11.2025).

5. American Diabetes Association. Standards of Medical Care in Diabetes – 2024. Diabetes Care. 2024, Vol. 47 (Suppl. 1), pp. S1–S200. DOI: 10.2337/dc24-SINT.

6. Panwar, H., Reddy, G. T., Lakshmanna, K. et al. Machine learning applications for early diabetes prediction: A review. International Journal of Medical Informatics, 2021, 149, 104414. DOI: 10.1016/j.ijmedinf.2021.104414.

7. Koye, D. N., Shaw, J. E., Reutens, A. T. Diabetes and kidney disease: A review of current and future therapies. Diabetes Research and Clinical Practice, 2018, 143, pp. 85–96.

8. Sinha, R., Warnecke, J. Predicting diabetes using machine learning on health and lifestyle data. BMC Medical Informatics and Decision Making, 2022, 22 (1): 43. DOI: 10.1186/s12911-022-01845-7.

9. Alghamdi, M. Developing an ensemble machine learning model for diabetes prediction. Scientific Reports (Nature Portfolio), 2022, 12 (1): 10036. DOI: 10.1038/s41598-022-14045-2.

10. Dutta, D., Bandyopadhyay, S. Integrating AI in healthcare: Predictive systems for chronic disease management. Health Informatics Journal, 2021, 27 (4), pp. 1–17. DOI: 10.1177/14604582211062365.

11. Kaggle. Diabetes prediction dataset. URL: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database (дата звернення: 09.11.2025).

12. Hennebelle, A., Materwala, H., Ismail, L. HealthEdge: Smart Healthcare Framework for Type 2 Diabetes Prediction. IEEE Access, 2023, 11, pp. 123456–123470. DOI: 10.1109/ACCESS.2023.3256789.

13. Imperial College London. AIRE-DM Project: AI-ECG Risk Estimation for Diabetes Mellitus. URL: https://www.imperial.ac.uk/ai-healthcare/projects/aire-dm (дата звернення: 09.11.2025).

14. Basalii, I. V. Application of machine learning in medical diagnostics. Scientific Bulletin of Kherson State University, Series «Computer Science», 2022, Issue 4(41), pp. 45–52.

15. Reddy, G. T., Reddy, M. P. K., Lakshmanna, K. Machine learning techniques for diabetes prediction: A systematic review. Applied Sciences, 2019, 9 (23): 5124. DOI: 10.3390/app9235124.

16. Islam, M. M., Rafi, S. K. S., Turin, T. C., Talukder, A. H. M. Systematic review of machine learning algorithms for diabetes prediction and risk factor identification. JMIR Medical Informatics, 2020, 8 (7): e16811. DOI: 10.2196/16811.

17. Munshi, A., Sreerama, R., Krishna, P. Deep learning models in diabetes risk prediction. Computers in Biology and Medicine, 2021, 135: 104574. DOI: 10.1016/j.compbiomed.2021.104574.

18. AI for early detection of metabolic disorders. URL: https://pubmed.ncbi.nlm.nih.gov/39969797/ (дата звернення: 09.11.2025).

19. UCI Machine Learning Repository. Diabetes Data Set. – University of California, Irvine, 2023. URL: https://archive.ics.uci.edu/ml/datasets/diabetes (дата звернення: 09.11.2025).

20. Kaggle. Pima Indians Diabetes Database. – Kaggle Datasets, 2023. URL: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database (дата звернення: 09.11.2025).

21. Firebase Authentication. URL: https://firebase.google.com/docs/auth (дата звернення: 13.11.2025).

22. Firebase Firestore – Manage Databases. URL: https://firebase.google.com/docs/firestore/manage-databases (дата звернення: 13.11.2025).

23. Firebase Storage. URL: https://firebase.google.com/docs/storage (дата звернення: 13.11.2025).

24. Angular CLI. URL: https://angular.dev/tools/cli (дата звернення: 13.11.2025).

25. TensorFlow.js. URL: https://www.tensorflow.org/js (дата звернення: 13.11.2025).

26. DOU. Обговорення: https://dou.ua/forums/topic/50894/ (дата звернення: 13.11.2025).

27. Angular Standalone Components (v17). URL: https://v17.angular.io/guide/standalone-components (дата звернення: 13.11.2025).

28. Angular Signals. URL: https://angular.dev/guide/signals (дата звернення: 13.11.2025).

29. Angular Dependency Injection. URL: https://angular.dev/guide/di (дата звернення: 13.11.2025).

30. Angular Reactive Forms. URL: https://angular.dev/guide/forms/reactive-forms (дата звернення: 13.11.2025).

ДОДАТОК А

Апробація кваліфікаційної магістерської роботи

Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили
ДНУ «Інститут модернізації змісту освіти»
Південний науковий центр НАН та МОН
Інститут української археографії та джерелознавства
імені М. С. Грушевського НАН України
Первинна профспілкова організація ЧНУ ім. Петра Могили

ЧНУ

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

XXVIII Всеукраїнська науково-практична конференція

ТЕЗИ ДОПОВІДЕЙ

ТЕХНІЧНІ НАУКИ

Миколаїв, 10–14 листопада 2025 року

Миколаїв – 2025

Рисунок А.1 – Обкладинка збірника тез доповідей конференції

ДОДАТОК Б

Лістинг конфігураційного файлу моделі


```
{
 "format": "layers-model",
 "generatedBy": "keras v3.12.0",
 "convertedBy": "TensorFlow.js Converter v4.22.0",
 "modelTopology": {
  "keras_version": "3.12.0",
  "backend": "tensorflow",
  "model_config": {
   "class_name": "Functional",
   "config": {
    "name": "functional",
    "trainable": true,
    "layers": [
     {
      "class_name": "InputLayer",
      "config": {
       "batch_shape": [null, 4],
       "dtype": "float32",
       "sparse": false,
       "ragged": false,
       "name": "num"
      },
      "name": "num",
      "inbound_nodes": []
```

},
{
"class_name": "Lambda",
"config": {
"name": "impute_num_with_mean",
"trainable": true,
"dtype": {
"module": "keras",
"class_name": "DTypePolicy",
"config": { "name": "float32" },
"registered_name": null
},
"function": {
"class_name": "__lambda__",
"config": {
"code":
"4wEAAAAAAAAAAAAAAAcAAAADAAAA864AAACVAFsAAAAAAAAAAABSA
wAAAAAAAAAAAAAAAAAAAAA...",
"defaults": null,
"closure": null
}
},
"arguments": {}
},
"name": "impute_num_with_mean",
"inbound_nodes": [
{
"args": [

```
        {
          "class_name": "__keras_tensor__",
          "config": {
            "shape": [null, 4],
            "dtype": "float32",
            "keras_history": ["num", 0, 0]
          }
        }
      ],
      "kwargs": { "mask": null }
    }
  ]
},
{
  "class_name": "InputLayer",
  "config": {
    "batch_shape": [null, 2],
    "dtype": "float32",
    "sparse": false,
    "ragged": false,
    "name": "bin"
  },
  "name": "bin",
  "inbound_nodes": []
},
{
  "class_name": "InputLayer",
```

```
  "config": {
    "batch_shape": [null],
    "dtype": "string",
    "sparse": false,
    "ragged": false,
    "name": "gender"
  },
  "name": "gender",
  "inbound_nodes": []
},
{
  "class_name": "InputLayer",
  "config": {
    "batch_shape": [null],
    "dtype": "string",
    "sparse": false,
    "ragged": false,
    "name": "smoking_history"
  },
  "name": "smoking_history",
  "inbound_nodes": []
},
{
  "class_name": "Normalization",
  "config": {
    "name": "normalization_1",
    "trainable": true,
```

```
  "dtype": {
    "module": "keras",
    "class_name": "DTypePolicy",
    "config": { "name": "float32" },
    "registered_name": null
  },
  "axis": [-1],
  "invert": false,
  "mean": null,
  "variance": null
},
"name": "normalization_1",
"inbound_nodes": [
  {
    "args": [
      {
        "class_name": "__keras_tensor__",
        "config": {
          "shape": [null, 4],
          "dtype": "float32",
          "keras_history": ["impute_num_with_mean", 0, 0]
        }
      }
    ],
    "kwargs": {}
  }
]
```

```
},
{
 "class_name": "Lambda",
 "config": {
  "name": "impute_bin_with_zero",
  "trainable": true,
  "dtype": {
   "module": "keras",
   "class_name": "DTypePolicy",
   "config": { "name": "float32" },
   "registered_name": null
  },
  "function": {
   "class_name": "__lambda__",
   "config": {
    "code":
"4wEAAAAAAAAAAAAAAAYAAAADAAAA83wAAACVAFsAAAAAAAAAAABSA
wAAAAAAAAAAAAAAAAAAAAA...",
     "defaults": null,
     "closure": null
   }
  },
  "arguments": {}
 },
 "name": "impute_bin_with_zero",
 "inbound_nodes": [
  {
   "args": [
```

```
        {
          "class_name": "__keras_tensor__",
          "config": {
            "shape": [null, 2],
            "dtype": "float32",
            "keras_history": ["bin", 0, 0]
          }
        }
      ],
      "kwargs": { "mask": null }
    }
  ]
},
{
  "class_name": "StringLookup",
  "config": {
    "name": "string_lookup",
    "trainable": true,
    "dtype": {
      "module": "keras",
      "class_name": "DTypePolicy",
      "config": { "name": "float32" },
      "registered_name": null
    },
    "invert": false,
    "num_oov_indices": 1,
    "oov_token": "[UNK]",
```

```
    "output_mode": "one_hot",
    "vocabulary_size": 4
   },
   "name": "string_lookup",
   "inbound_nodes": [
    {
     "args": [
      {
       "class_name": "__keras_tensor__",
       "config": {
        "shape": [null],
        "dtype": "string",
        "keras_history": ["gender", 0, 0]
       }
      }
     ],
     "kwargs": {}
    }
   ]
  },
  {
   "class_name": "StringLookup",
   "config": {
    "name": "string_lookup_1",
    "trainable": true,
    "dtype": {
     "module": "keras",
```

```
      "class_name": "DTypePolicy",
      "config": { "name": "float32" },
      "registered_name": null
    },
    "invert": false,
    "num_oov_indices": 1,
    "oov_token": "[UNK]",
    "output_mode": "one_hot",
    "vocabulary_size": 7
  },
  "name": "string_lookup_1",
  "inbound_nodes": [
    {
      "args": [
        {
          "class_name": "__keras_tensor__",
          "config": {
            "shape": [null],
            "dtype": "string",
            "keras_history": ["smoking_history", 0, 0]
          }
        }
      ],
      "kwargs": {}
    }
  ]
},
```

{
"class_name": "Concatenate",
"config": {
"name": "features_concat",
"trainable": true,
"axis": -1
},
"name": "features_concat",
"inbound_nodes": []
},
{
"class_name": "Dense",
"config": {
"name": "dense",
"units": 32,
"activation": "relu",
"use_bias": true
},
"name": "dense",
"inbound_nodes": []
},
{
"class_name": "Dropout",
"config": {
"name": "dropout",
"rate": 0.2
},

```
    "name": "dropout",
    "inbound_nodes": []
   },
   {
    "class_name": "Dense",
    "config": {
     "name": "dense_1",
     "units": 16,
     "activation": "relu",
     "use_bias": true
    },
    "name": "dense_1",
    "inbound_nodes": []
   },
   {
    "class_name": "Dense",
    "config": {
     "name": "prob",
     "units": 1,
     "activation": "sigmoid",
     "use_bias": true
    },
    "name": "prob",
    "inbound_nodes": []
   }
  ],
  "input_layers": [
```

```
        ["num", 0, 0],
        ["bin", 0, 0],
        ["gender", 0, 0],
        ["smoking_history", 0, 0]
      ],
      "output_layers": ["prob", 0, 0]
    }
  },
  "training_config": {
    "loss": "binary_crossentropy",
    "metrics": ["accuracy", { "class_name": "AUC", "config": { "curve": "ROC" } }],
    "optimizer_config": {
      "class_name": "Adam",
      "config": { "learning_rate": 0.001, "beta_1": 0.9, "beta_2": 0.999, "epsilon": 1e-07 }
    }
  }
},
"weightsManifest": [
  {
    "paths": ["group1-shard1of1.bin"],
    "weights": [
      { "name": "dense/kernel", "shape": [17, 32], "dtype": "float32" },
      { "name": "dense/bias", "shape": [32], "dtype": "float32" },
      { "name": "dense_1/kernel", "shape": [32, 16], "dtype": "float32" },
      { "name": "dense_1/bias", "shape": [16], "dtype": "float32" },
      { "name": "normalization_1/mean", "shape": [4], "dtype": "float32" },
      { "name": "normalization_1/variance", "shape": [4], "dtype": "float32" },
```

{ "name": "normalization_1/count", "shape": [], "dtype": "int32" },

{ "name": "prob/kernel", "shape": [16, 1], "dtype": "float32" },

{ "name": "prob/bias", "shape": [1], "dtype": "float32" }

]

}

]

}