МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Інформаційна система для автоматизації обчислювального процесу хіміко-лабораторних досліджень»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Берест О.Б.**

Студент групи ІН – 71 — **Журавльов Ю.О.**

СУМИ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2021 р.

ЗАВДАННЯ
до випускної роботи

Студента четвертого курсу, групи ІН-71 спеціальності "Комп'ютерні науки " денної форми навчання Журавльова Юрія Олександровича.

Тема: "**Інформаційна система для автоматизації обчислювального процесу хіміко-лабораторних досліджень**"

Затверджена наказом по СумДУ

№_____ от _____2021 р.

Зміст пояснювальної записки: 1) аналітичний огляд існуючих рішень; 2) постановка завдання й формування завдань розробки; 3) створення сценарію роботи інформаційної системи та проектування; 5) розробка програмного забезпечення інформаційної системи; 6) тестування роботи додатку.

Дата видачі завдання "_____"_____2021 р.

Керівник випускної роботи _____ Берест О.Б.

Завдання приняв до виконання _____ Журавльов Ю.О.

РЕФЕРАТ

Записка: 90 стор., 52 рис., 1 додаток, 19 джерел.

Об'єкт дослідження — інформаційна система для автоматизації обчислювального процесу хіміко-лабораторних досліджень.

Мета роботи — розробка додатку інформаційної системи для операційної системи Android для проведення аналізу вмісту сірководню в природному газі.

Результати — Розроблено інформаційну систему у вигляді додатку для операційної системи Android та віддаленої бази даних. Розроблена інформаційна система реалізована у формі програмного забезпечення, створеного за допомогою середовища розробки Android Studio на мові Java та реляційної бази даних на основі діалекту MySQL.

ІНФОРМАЦІЙНА СИСТЕМА, АВТОМАТИЗАЦІЯ ХІМІКО-ЛАБОРАТОРНИХ ДОСЛІДЖЕНЬ, АНАЛІЗ ВМІСТУ СІРКОВОДНЮ В ПИРОДНОМУ ГАЗІ, АПРОКСИМАЦІЯ МЕТОДОМ НАЙМЕНШИХ КВАДРАТІВ, ANDROID ДОДАТОК.

ЗМІСТ

ВСТУП

В сучасній лабораторній практиці підвищуються вимоги до системи якості вимірювань. Так у вимірювальних лабораторіях, які пройшли оцінку відповідності системи керування вимірюваннями згідно вимог ДСТУ ISO 10012:2005 «Система керування вимірюванням. Вимоги до процесів вимірювання та вимірювального обладнання», є вимоги до керування інформаційними ресурсами (методиками та програмним забезпеченням) і протоколами, в тому числі результатів вимірювання.

Окрім цього, в лабораторії необхідно постійно проводити дії по забезпеченню і контролю якості: контролювати операції та результати аналізування, щоб визначити, чи достатньо достовірні результати для того, щоб їх можна було видавати замовнику. Для контролю різних параметрів мінливості процесу можна застосовувати різні способи контролю якості.

При вимірюванні сірководню в природному газі фотоколориметричним методом згідно ГОСТ 22387.2-97, НДТОВ 04-014:2019 необхідно щоразу при аналізі перевіряти стабільність градуювальної характеристики та повторним аналізуванням перевірити збіжність. Так як для даного аналізу в лабораторії не розроблено програмне забезпечення і процес вимірювання та контролювання проводиться в ручну, потребується розробити додаток з наступними функціями:

1. Введення даних для побудови градуювальної характеристики.
2. Побудова графіку градуювальної характеристики.
3. Перевірка стабільності градуювальної характеристики.
4. Обчислення вмісту сірководню в досліджуваній пробі.
5. Перевірка збіжності аналізу.
6. Збереження минулих градуювальних характеристик для аналізування валідації методу в конкретній лабораторії.
7. Збереження результатів вимірювання.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Аналіз аналогів.

В зв’язку з необхідністю створення додатку був проведений аналіз аналогів.

Найбільш поширеними є :

- Report 10.1
- Ask-2
- ОАС-ГТП

Report 10.1

Розробник ТОВ НТЦ «Атом».

Програмне забезпечення Report виконує розрахунки показників природного газу відповідно до ДСТУ ISO 6974 «Визначення складу із заданою невизначеністю методом хроматографії» за методом А, а також ДСТУ ISO 6976 «Природний газ, Обчислення теплоти згорання, густини, відносної густини і числа Воббе на основі компонентного складу» та РМУ 026-2009 «Природний газ. Визначення складу методом газової хроматографії з оцінкою невизначеності результату вимірювання», МВУ 06-023:2011 «Методика виконання вимірювань компонентного складу із застосуванням хроматографів “Кристал” та обчислення густини, теплоти згорання і числа Воббе», МВУ 06-063:2011 «Методика виконання вимірювань компонентного складу із застосуванням хроматографів HP/AC 6890 та обчислення густини, теплоти згорання і числа Воббе».

Переваги:

- Придатний до застосування згідно ДСТУ ISO/IEC 9126-1:2013; ДСТУ 2851-94; ДСТУ 2853-94; ДСТУ ISO 6974; ДСТУ ISO 6976; ДСТУ 7363:2013; РМУ 026-2009; МВУ 06-023:2011; МВУ 06-063:2011.
- Високий рівень захисту ПЗ

Недоліки:

- Не застосовуються для розрахунків при визначенні сірководню.

Програмне забезпечення опитування автоматичних обчислювачів, коректорів та потокових хроматографів Ask-2.

Розробник Інститут транспорту газу.

Програмний комплекс (ПК) Ask-2 призначений для:

- Зчитування газовимірювальних даних (ГВД) та повідомлень з автоматичних обчислювачів витрати газу та коректорів (АО);
- Зчитування даних про фізико-хімічні показники (ФХП) з потокових хроматографів;
- Імпорту даних, щодо складу газу – ФХП, з протоколу вимірювань лабораторних хроматографів;
- Збереження зчитаних (імпортованих) даних до бази даних (також у Hostlib-форматі для АО);
- Віддаленого введення в АО параметрів газу, атмосферного тиску, контрактної години, а також коригування часу;
- Перегляду та аналізу отриманих даних;
- Формування, перегляду і друку (або збереження у файлі) добових і місячних звітів по АО установленої форми;
- Формування, перегляду і друку (або збереження у файлі) паспортів газу за добу та за місяць установленої форми.

Переваги:

- Збирає дані з віддалених потокових хроматографів та передає на робоче місце.
- Зберігає дані до бази даних.
- Формує звіт.

Недоліки:

- Не перевіряє дані на коректність.
- Не застосовується для аналізу сірководню в природному газі.

Обліково-аналітична система газотранспортного підприємства (ОАС-ГТП).

Розробник Інститут транспорту газу.

Призначений для обліку фізико-хімічних параметрів газу та формування паспорту газу.

Переваги:

- Обширна база даних всіх занесених аналізів.
- Формує паспорт фізико-хімічних показників природного газу.
- Формує звіт про недотримання фізико-хімічних показників.
- Перевіряє дані на коректність.

Недоліки:

- Не виконує обчислень (вимагає обраховані дані).
- Не застосовується для аналізу сірководню в природному газі.

Кожен з аналогів має ряд переваг та недоліків. Основним недоліком кожного з них є те, що жоден не застосовується для аналізу сірководню в природному газі.

1.2 Аналіз технологій.

Для створення додатку розглядались наступні технології:

- Java
- C/C++
- C#
- Kotlin

Java

Java на сьогодні є найпоширенішою мовою програмування для Android додатків в світі. На неї посилається більшість документації Google, а також створено найбільше навчальних посібників та веб-ресурсів серед інших мов програмування для даної платформи.

Перевагою використання Java є також IDE Android Studio, яку Google визнало як офіційну IDE для розробки Android додатків [16]. Серед переваг Android Studio можна виділити візуальний UI-редактор та автозаповнення коду, що значно полегшує розробку.

При розробці Java додатків використовують не лише Java-класи. Крім них створюються файли маніфесту за допомогою XML. Також XML використовується для верстки UI. Крім того використовуються інструменти для автоматичного збирання проекту Gradle, Maven або Ant.

Інформацію про Android розробку на Java взято з праці [4]

C/C++

C/C++ - більш низькорівневі мови ніж Java, що дозволяє створювати нативні додатки, що є значною перевагою для створення ігор або ресурсоємних програм. Android Studio підтримує розробку на C/C++ через Android NDK. Це значить що додаток буде запускатись напряму через девайс, а не через JVM, що дасть більше контролю над елементами Android системи.

Недоліком є складність налаштування, тому C/C++ рекомендується для написання модулів де необхідно провести складні операції, наприклад, рендерінг та обробку графіки.

Крім того Microsoft пропонує надбудову для Visual Studio для розробки мобільних додатків, де основною перевагою визначає кроссплатформеність розробки.

Інформацію про Android розробку на C/C++ взято з праці [7].

C#

Для розробки Android додатків на C# використовується фреймворк Xamarin.

Xamarin має ряд переваг:

- Дозволяє створювати кросплатформенні додатки.
- Інтерфейс описується як в коді так і за допомогою XML.
- Має зручні IDE для розробки Visual Studio та Xamarin Studio.

Недоліком Xamarin є великій розмір готових додатків.

Інформацію про Xamatin та Android розробку на C# взято з праці [6].

Kotlin

Kotlin – відносно нова мова програмування. В 2019 році Google визнала Kotlin пріоритетною мовою розробки для Android додатків.

Kotlin, як і Java, вже інтегровано в Android Studio.

Мова основана на Java та має переваги перед нею. Наприклад:

- Автоматичне виявлення типів даних.
- Підтримка функціональної парадигми.
- Null-безпека.

До того ж мова сумісна з Java v1.6 та компілюэться в байткод JVM.

В результаті аналізу мовою розробки було обрано Java оскільки мова має всі потрібні інструменти для створення додатку, зручне IDE для розробки та велику кількість навчальних посібників.

Також для реалізації додатку потрібно створити базу даних.

В залежності від потреб та об’єму даних базу можна зберігати локально або на сервері.

Локальна база даних призначена для збереження не великої кількості даних, крім того доступ до даних можливий лише з конкретної копії додатку на носії. Перевагою такої бази є швидкість виконання запитів та відсутність потреби в підключенні до мережі. В випадку використання такого типу збереження даних більшість посібників рекомендують використовувати базу даних SQLite.

Інформацію про використання MySQL наведено в праці [12].

Серверна база даних навпаки призначена для зберігання більшого обсягу даних та мають можливість збереження даних до сервера та отримання даних з нього копіями додатку на різних пристроях, що мають підключення до мережі. Швидкість виконання запитів до такої бази менша ніж до локальної. Для серверної бази даних можна використовувати MySQL, Oracle, PostgreSQL та інші.

Інформацію про використання MySQL наведено в праці [11].

Загальна інформація про використання баз даних наведено в працях [19, 10, 15].

Для реалізації додатку використаємо MySQL, оскільки потрібно забезпечити безпечний доступ до градуювальних характеристик приладів, а також зберігати дані градуювання та результати аналізу зроблені на різних приладах на сервер, що пізніше може бути використано для формування звіту. До того ж є безліч безкоштовних хостингів баз даних на базі діалекту MySQL, що знадобиться при розробці та тестуванні інформаційної системи.

2 ПРОЕКТУВАННЯ РОБОТИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Сценарій роботи інформаційної системи.

Для роботи додатку було розроблено наступний сценарій:

Користувач авторизується в системі (вводить назву приладу та пароль).

Після авторизації на екран виводиться

1. Дата останнього градуювання.
2. Градуювальна таблиця (концентрація градуювального розчину, оптична густина).
3. Графік апроксимованої прямої на основі даних таблиці.

Користувач може розпочати аналіз, провести процедуру повторного градуювання або вийти з системи.

Процедура повторного градуювання.

1. Вводимо об'єм титрованого розчину тіосульфату натрію, витрачений на контрольне титрування розчину йоду без додавання розчину сульфіду натрію, см3 - V;
2. Вводимо об'єм титрованого розчину тіосульфату натрію, витрачений на титрування розчину йоду з додаванням розчину сульфіду натрію, см3 – вводим тричі, знаходим середнє - V1.
3. Розраховуємо концентрацію розчину сульфіду натрію X в перерахунку на сірководень (мкг/см³): $X = \frac{(V-V_1)\cdot 17}{20}$ (1 см³ робочого розчину Na_2S концентрації C(1/2 Na_2S)=0,001 моль/см³ відповідає X сірководню).
4. Заповнюємо градуювальну таблицю (V_i робочого розчину см³, три значення оптичної густини D які відповідають об’єму), для кожного запису робим розрахунок концентрації x=X*V_i, середнє значення оптичної густини D. Для таблиці потрібно 8 записів.
5. Заносимо отримані дані та дату градуювання до бази даних.

6. Повертаємося на головну сторінку.

Аналіз.

1.Проводимо перевірку стабільності градуювальної характеристики (створюємо градуювальну таблицю з 3-ма записами). На її основі будуємо апроксимовану пряму та знаходимо кут між градуювальною прямою та контрольною. Якщо кут більше 15° пропонуємо зробити нове градуювання або контрольну перевірку по нових розчинах. Якщо кут менше 15° продовжуємо розрахунок.

2.Вводимо температуру газу в лічильнику, t °C, атмосферний барометричний тиск, P мм рт ст.

3.Розраховуємо коефіцієнт приведення об'єму газу до стандартних умов $k = \frac{293 \cdot P}{(273+t) \cdot 760}$

4.Вводимо об'єм газу, пропущеного через поглинальний розчин, дм³ V.

5.Вводимо два значення оптичної густини (перша та друга склянка), отриманої в результаті досліду. Якщо значення оптичної густини в другій склянці становить до 5% від першої склянки для розрахунку концентрації сірководню беруть оптичну густину лише першої склянки. Від 5 до 20% – для розрахунку використовують сумарну оптичну густину першої та другої склянок. При перевищенні 20% випробування слід повторити з меншим об’ємом проби газу.

6.За значенням оптичної густини обрахованої в пункті 5 знаходимо масу сірководню у випробовуваному розчині за апроксимованою прямою градуювальної характеристики x мкг.

7.Далі за формулою $X = \frac{x}{V \cdot k \cdot 1000}$ г/м^3 знаходимо значення концентрації.

8.Повторюємо аналіз починаючи з пункту 4 для паралельного аналізу.

9.Розраховуємо різницю $|X_1 - X_2|$ та перевіряємо збіжність (чи це значення менше 2E-4).

10. Якщо збіжності не досягнуто пропонуємо повторно провести процедуру аналізу.
11. Результат аналізу є середнє значення $X = \frac{X_1 + X_2}{2}$ округлене до Е-4.
12. Результат разом з датою проведення аналізу заноситься до бази даних.

2.2 Дизайн ІС на основі UML-діаграм.

Для візуалізації сценарію було розроблено декілька діаграм нотації UML: функціональну діаграму (Рисунок 2.1, Рисунок 2.2), діаграму варіантів використання (Рисунок 2.3) та діаграму діяльності (Рисунок 2.4). Інформацію про використання нотації наведено в працях [13, 14]

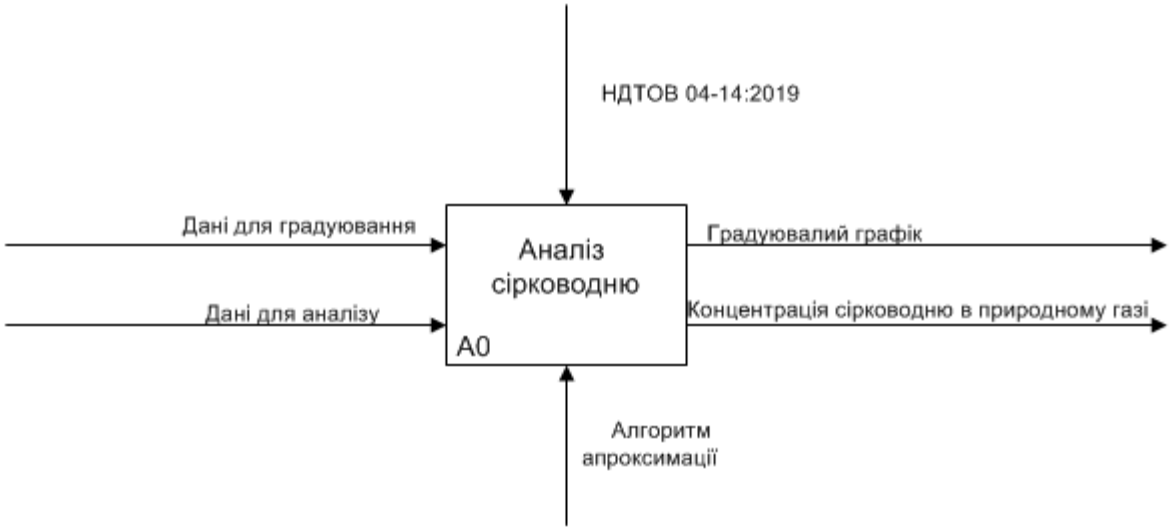

Рисунок 2.1 – Функціональна діаграма, 1 рівень

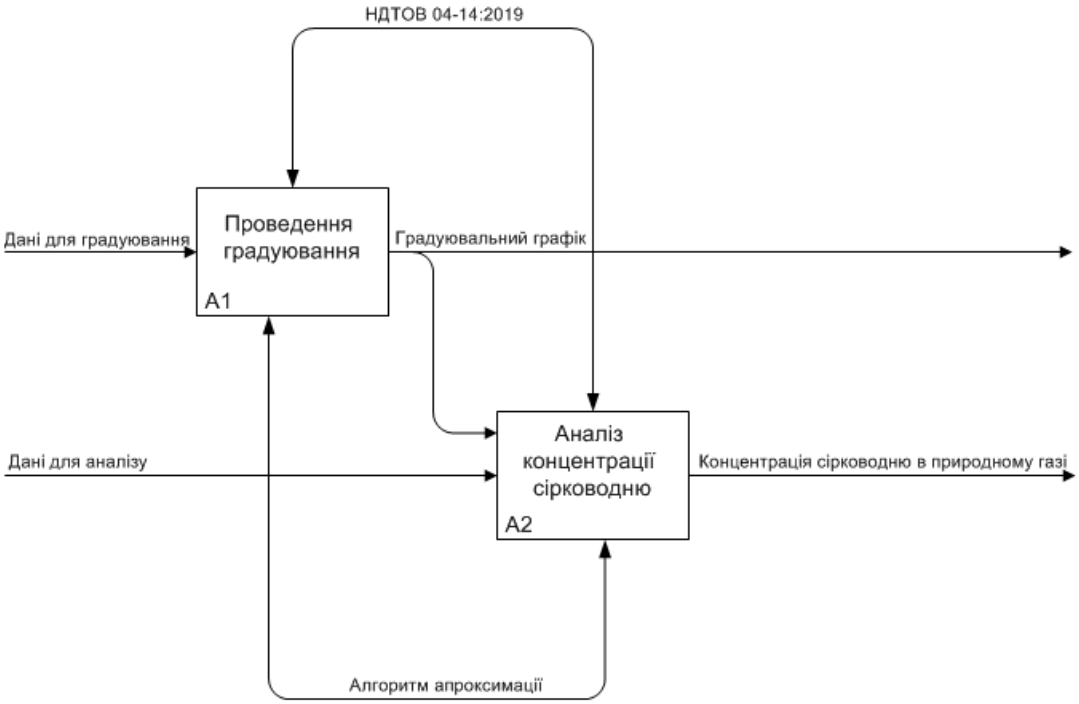

Рисунок 2.2 – Функціональна діаграма, 2 рівень

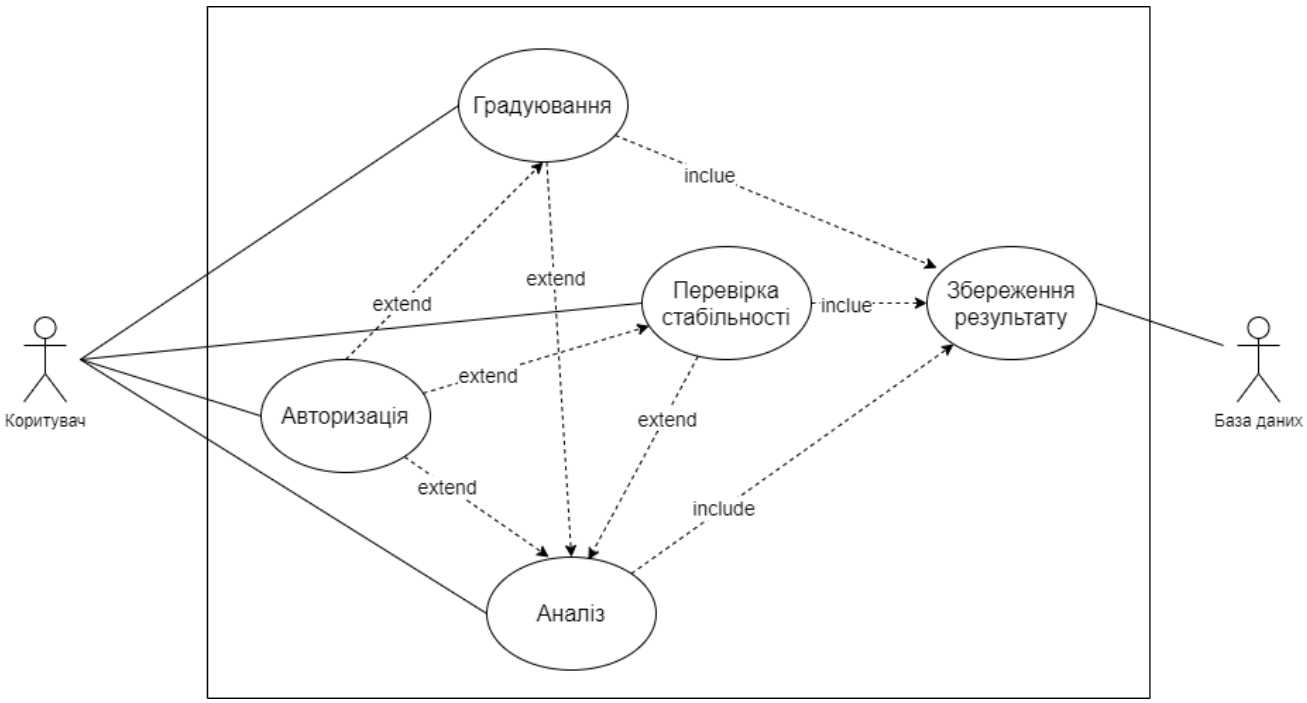

Рисунок 2.3 – Діаграма варіантів використання

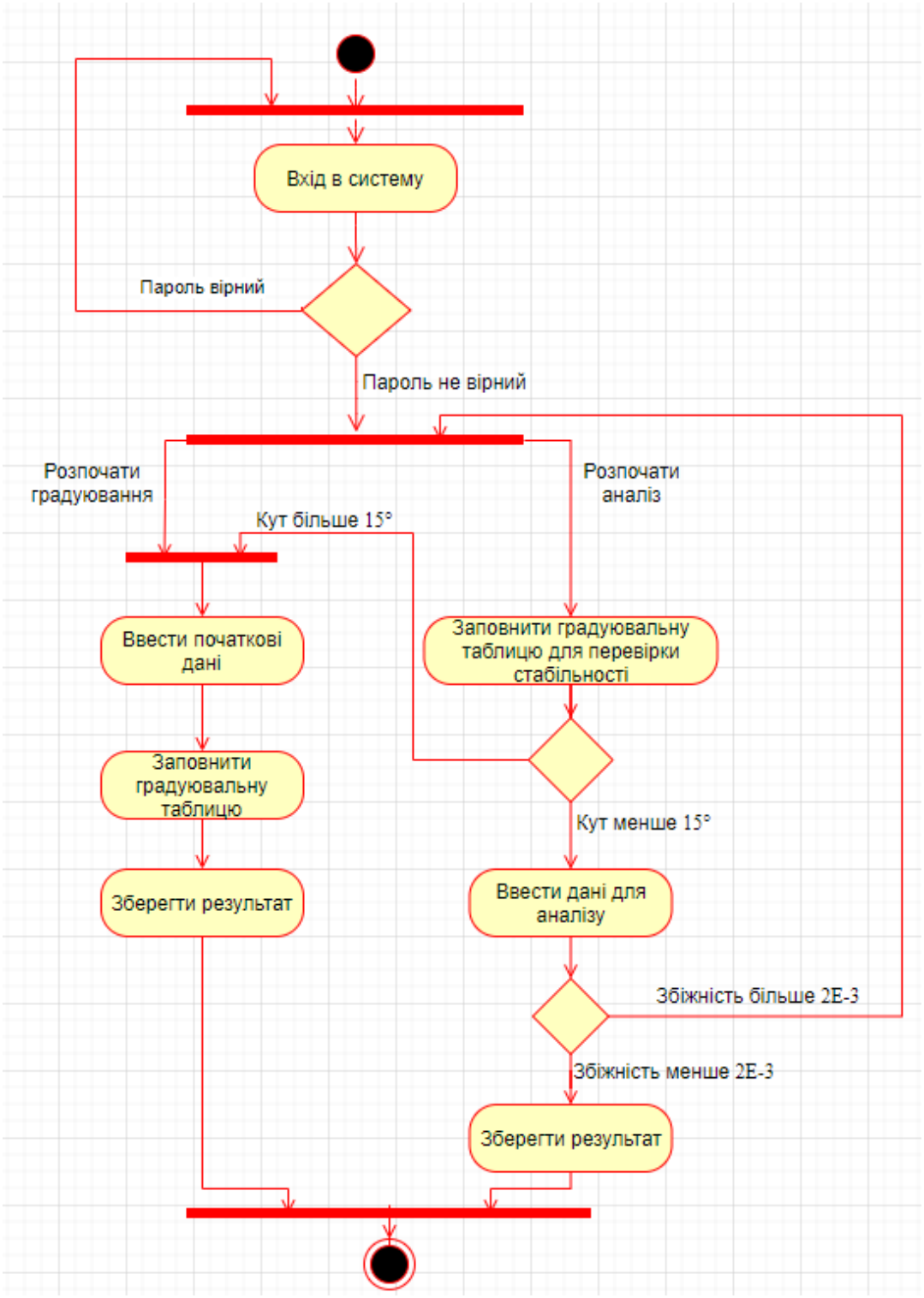

Рисунок 2.4 – Діаграма діяльності

Для візуалізації структури додатку наведемо діаграму класів (Рисунок 2.5). Класи MainActivity, MainPage, GradTable, GradTableFill, StartGrad, ResGrad, AnalysisStart, AnalysisNext, AnalysisResult – класи активності призначені для відображення інтрефейсу, реакції на дії користувача та переходу

до інших активностей. Клас Aprox відповідає за апроксимацію точок отриманих в результаті заповнення градуювальної таблиці, в результаті обробки точок об’єкт класу зберігає в собі коефіцієнти функції апроксимованої прямої. Клас ModelPract призначений для роботи з базою даних. Клас PractUtils містить статичні методи, що використовуються в інших класах.

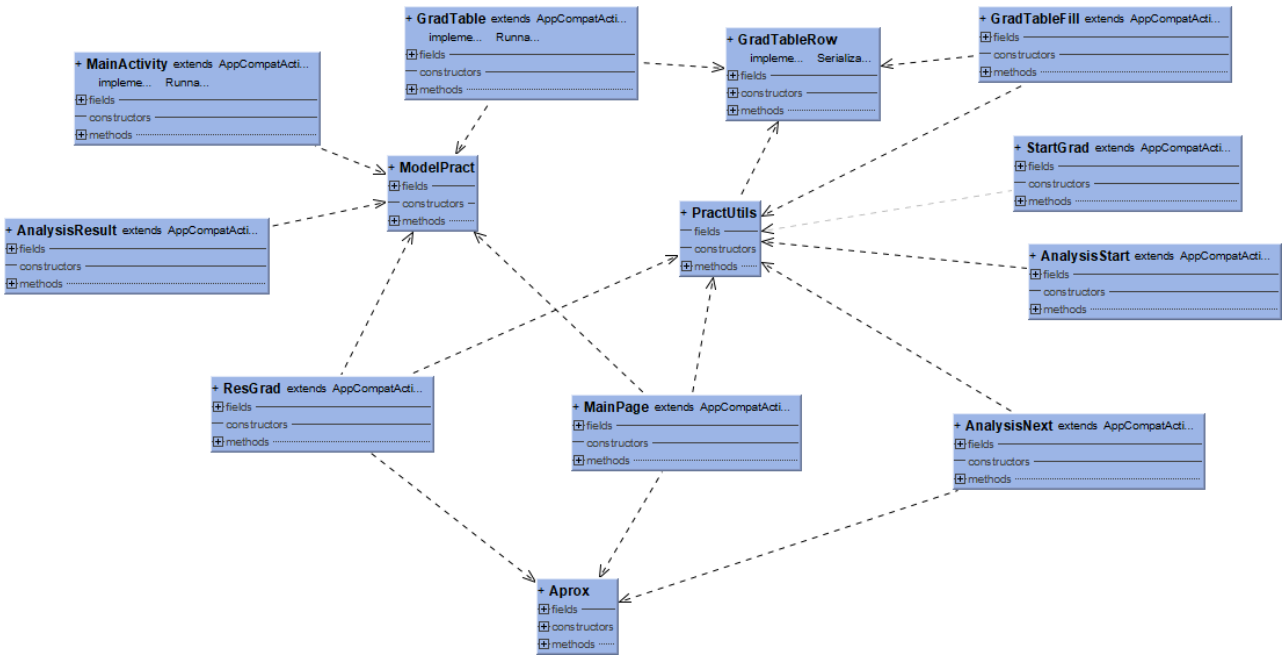

Рисунок 2.5 – Діаграма класів

Для візуалізації структури бази даних наведемо ER діаграму (Рисунок 2.6). Інформація необхідна для проектування бази даних взята з праці [10]. Таблиця device призначена для зберігання назв приладів та паролів до них, що буде використано для авторизації а додатку. Таблиця grad призначена для зберігання дати та інших основних характеристик кожного градуювання. Таблиця grad_table призначена для збереження градуювальних таблиць. Таблиця results призначена для збереження дати та результату аналізу. Таблиця stabilization призначена для збереження інформації що до перевірок стабільності градуювальної характеристики.

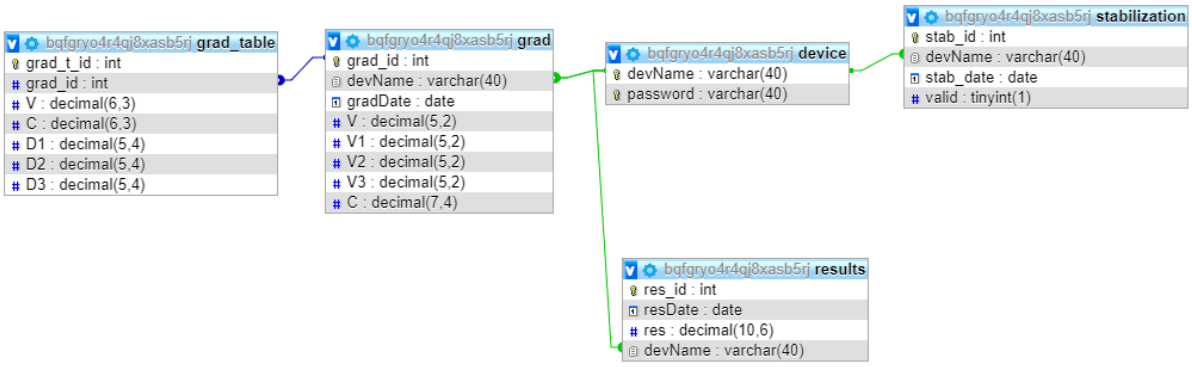

Рисунок 2.6 – Діаграма сутність-зв’язок

2.3 Дизайн користувацького інтерфейсу

Для додатку було розроблено дизайн на основі інформації отриманої з літературних джерел [17, 18]

При запуску додатку користувача зустрічає інтерфейс авторизації (Рисунок 2.7), що відповідає активності MainActivity, в якому користувач обирає потрібний йому прилад та вводить до нього пароль після чого потрапляє до головної сторінки (Рисунок 2.8).

Рисунок 2.7 – Активність MainActivity

Головна сторінка відповідає активності MainPage (Рисунок 2.8). На цій сторінці користувач може переглянути поточний градуювальний графік приладу та дату останнього градуювання. З цієї активності користувач починає процедуру градуювання та аналізу, а також може переглянути поточну градуювальну таблицю, що відповідає активності GradTable (Рисунок 2.9).

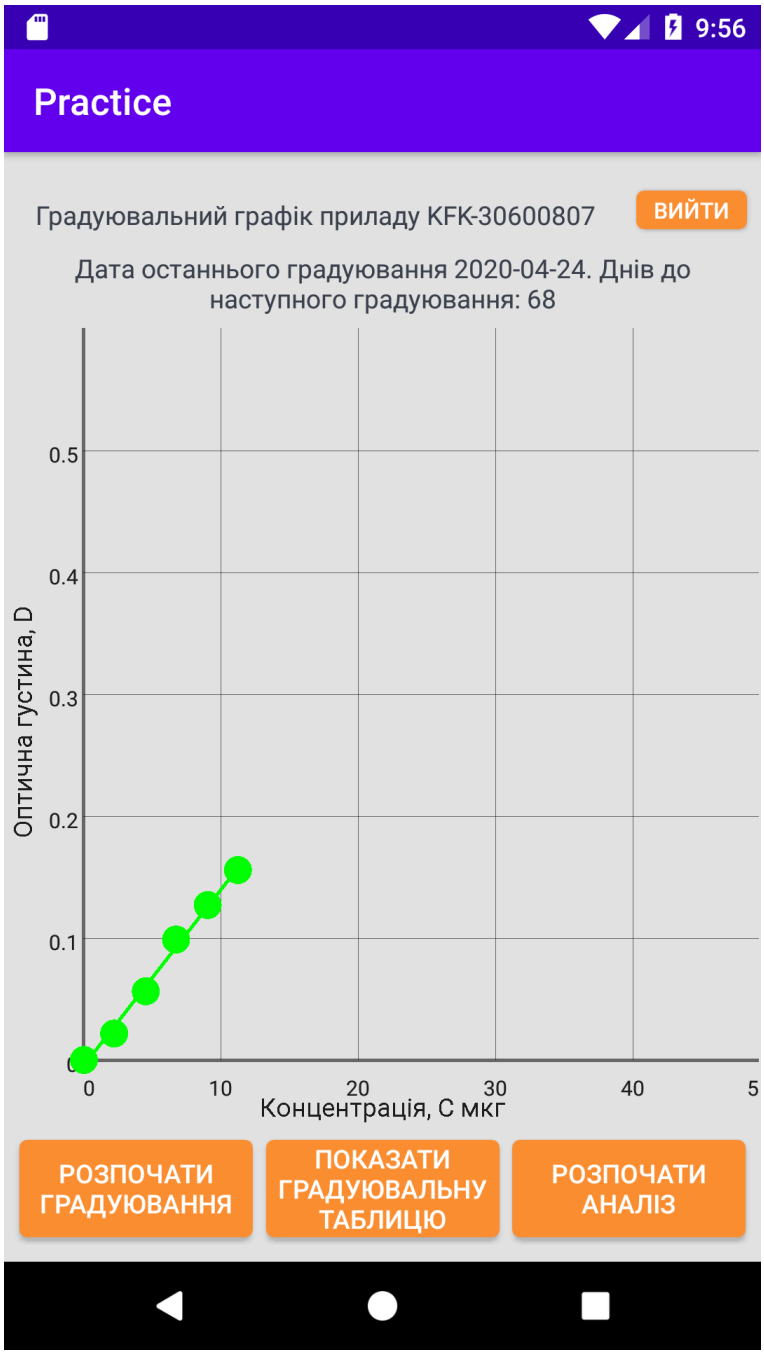

Рисунок 2.8 – Активність MainPage

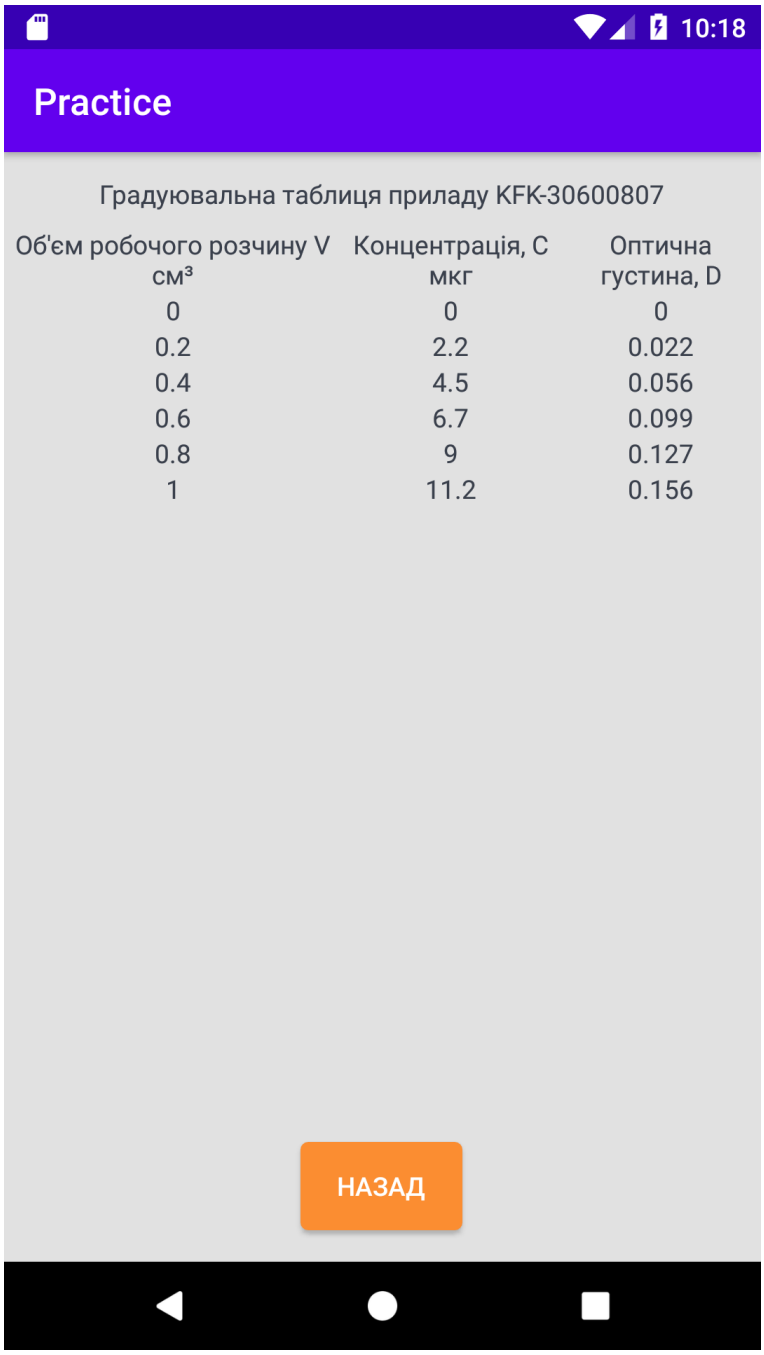

Рисунок 2.9 – Активність GradTable

Для проведення процедури градуювання застосовуються активності StartGrad (Рисунок 2.10), GradTableFill (Рисунок 2.11), ResGrad (Рисунок 2.12). Ці активності також застосовуються для перевірки стабільності градуювальної характеристики при аналізі. При перевірці стабільності в активностях StartGrad та GradTableFill слово «Градуювання» замінюється словом «Аналіз», а ResGrad має інший вигляд (Рисунок 2.13).

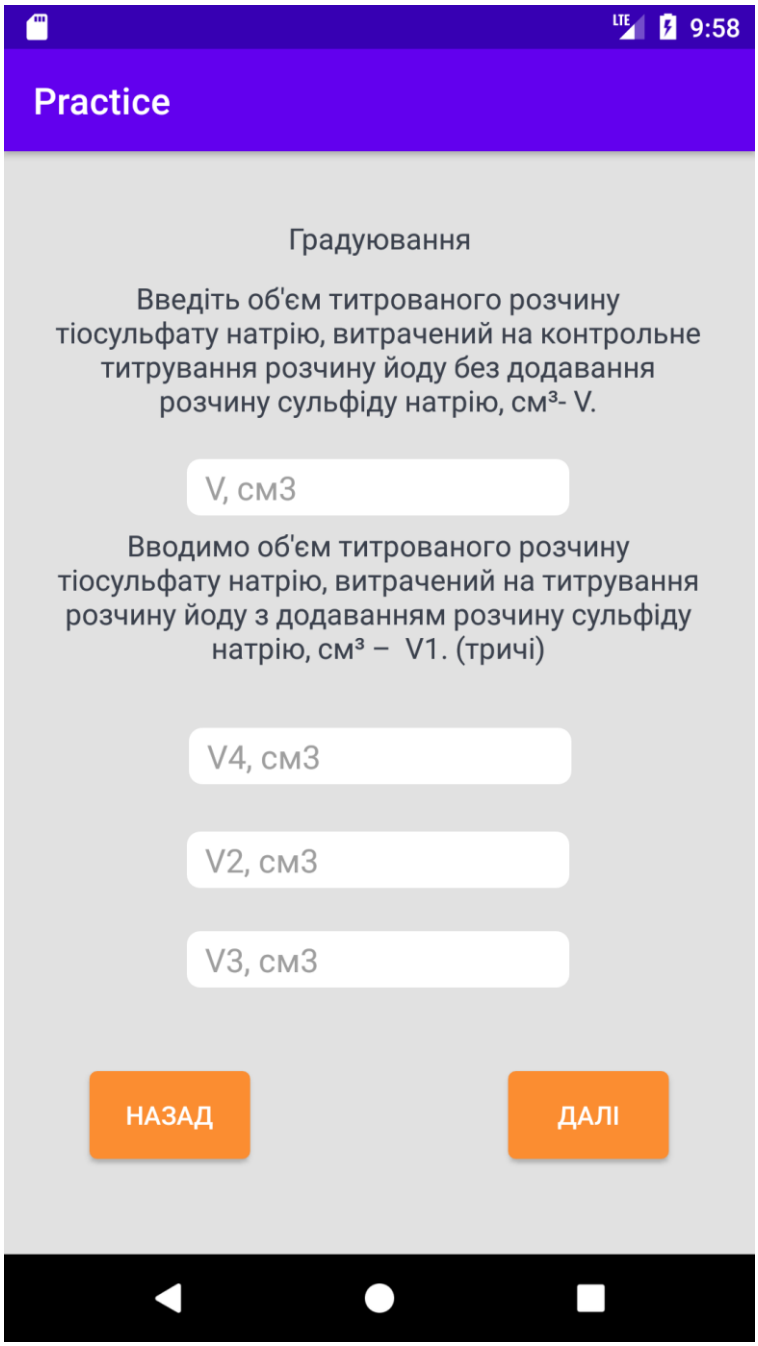

Рисунок 2.10 – Активність StartGrad

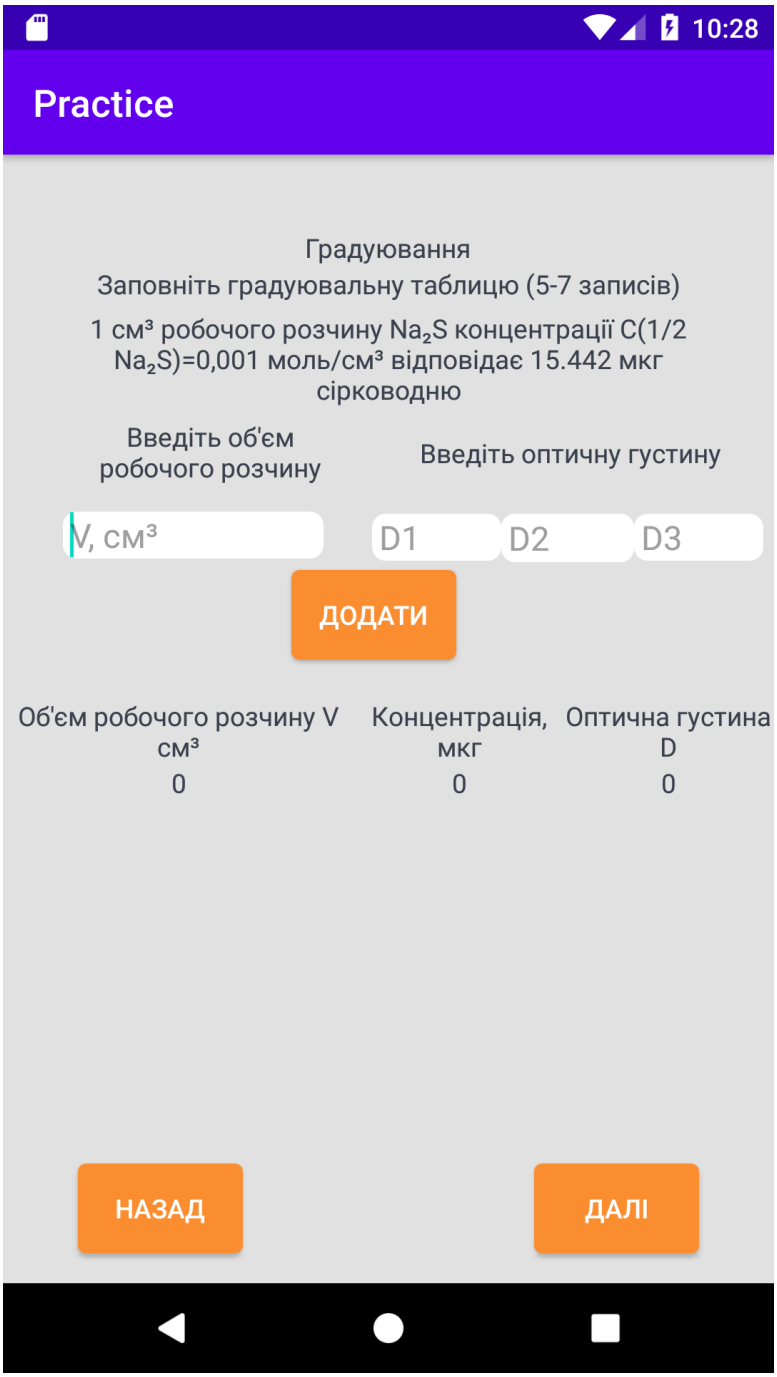

Рисунок 2.11 – Активність GradTableFill

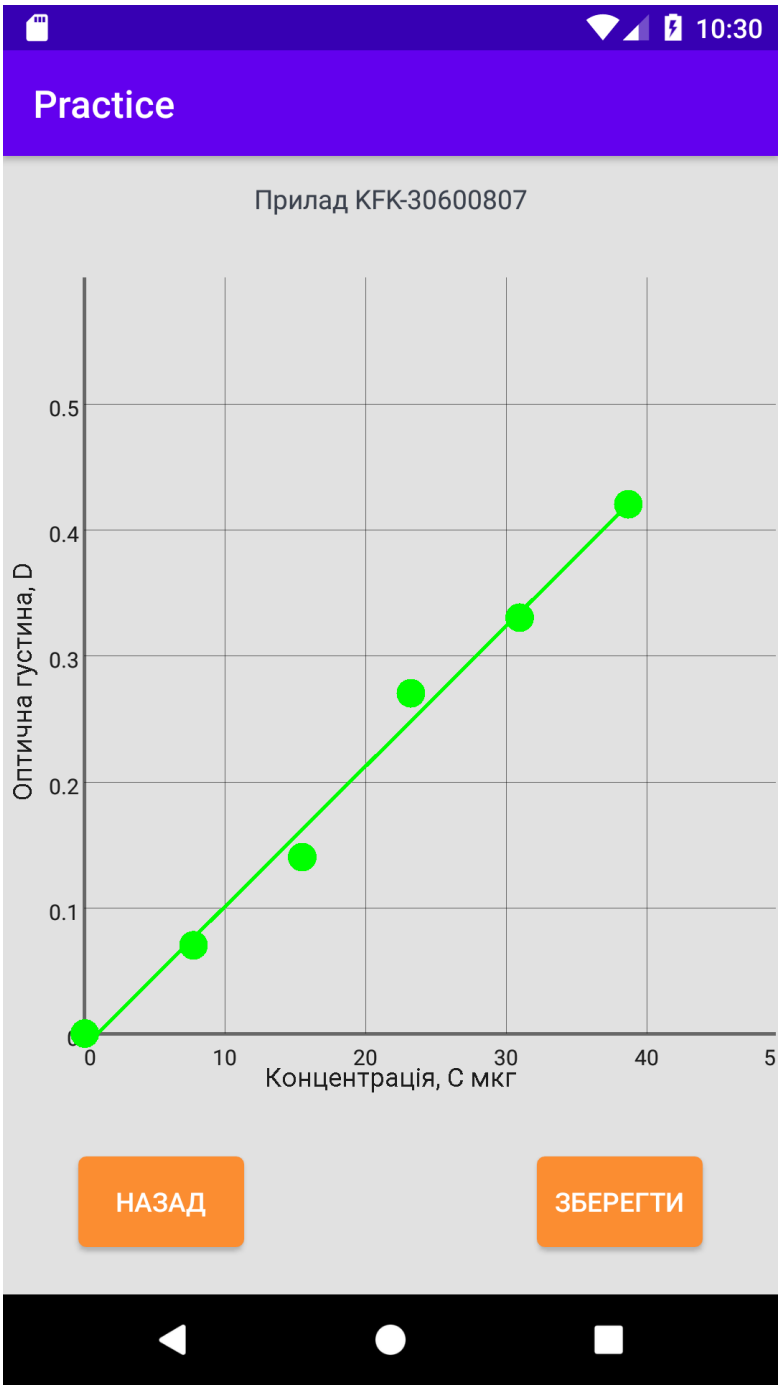

Рисунок 2.12 – Активність ResGrad при градуюванні

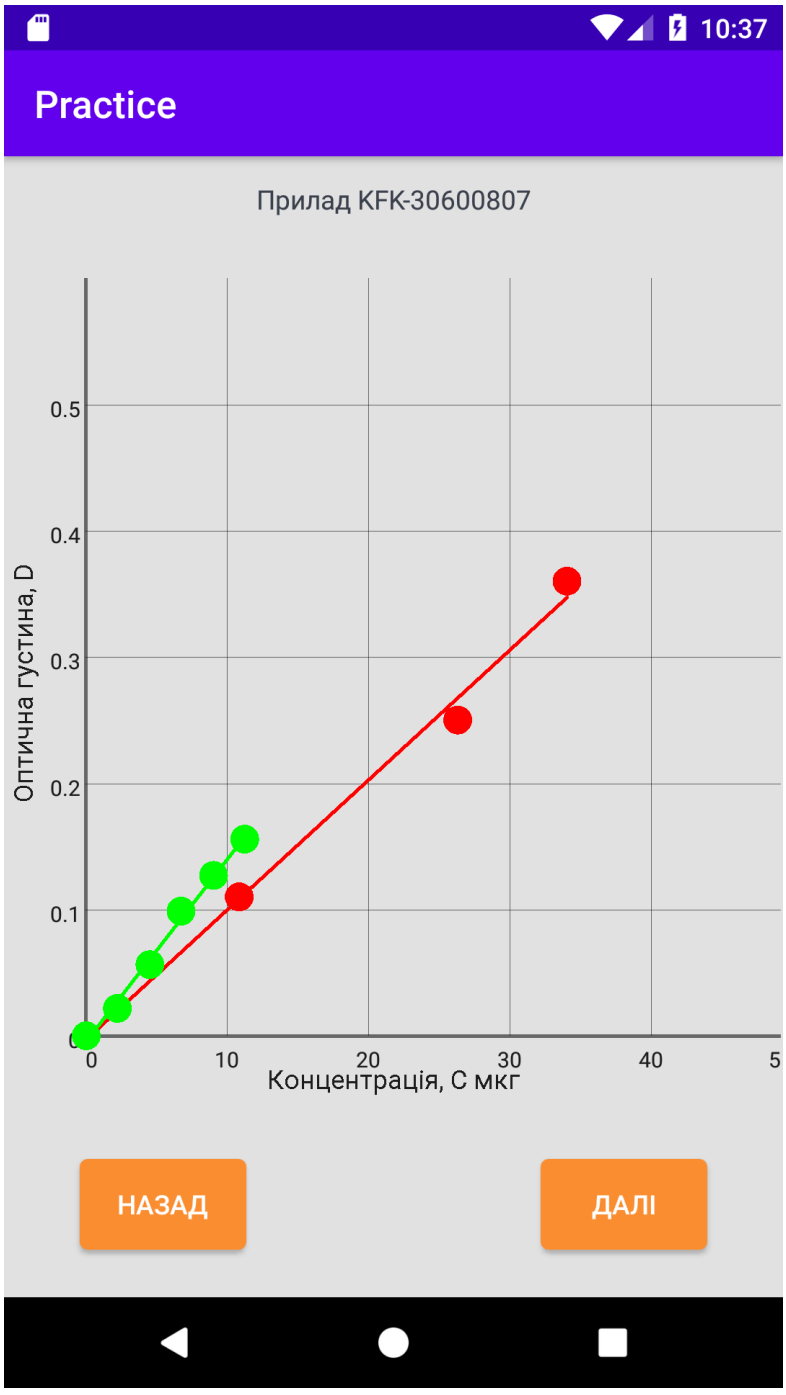

Рисунок 2.13 – Активність ResGrad при аналізі

Після перевірки стабільності проводиться сам аналіз. Під час аналізу застосовуються активності AnalysisStart (Рисунок 2.14), AnalysisNext (Рисунок 2.15) та AnalysisResult (Рисунок 2.16).

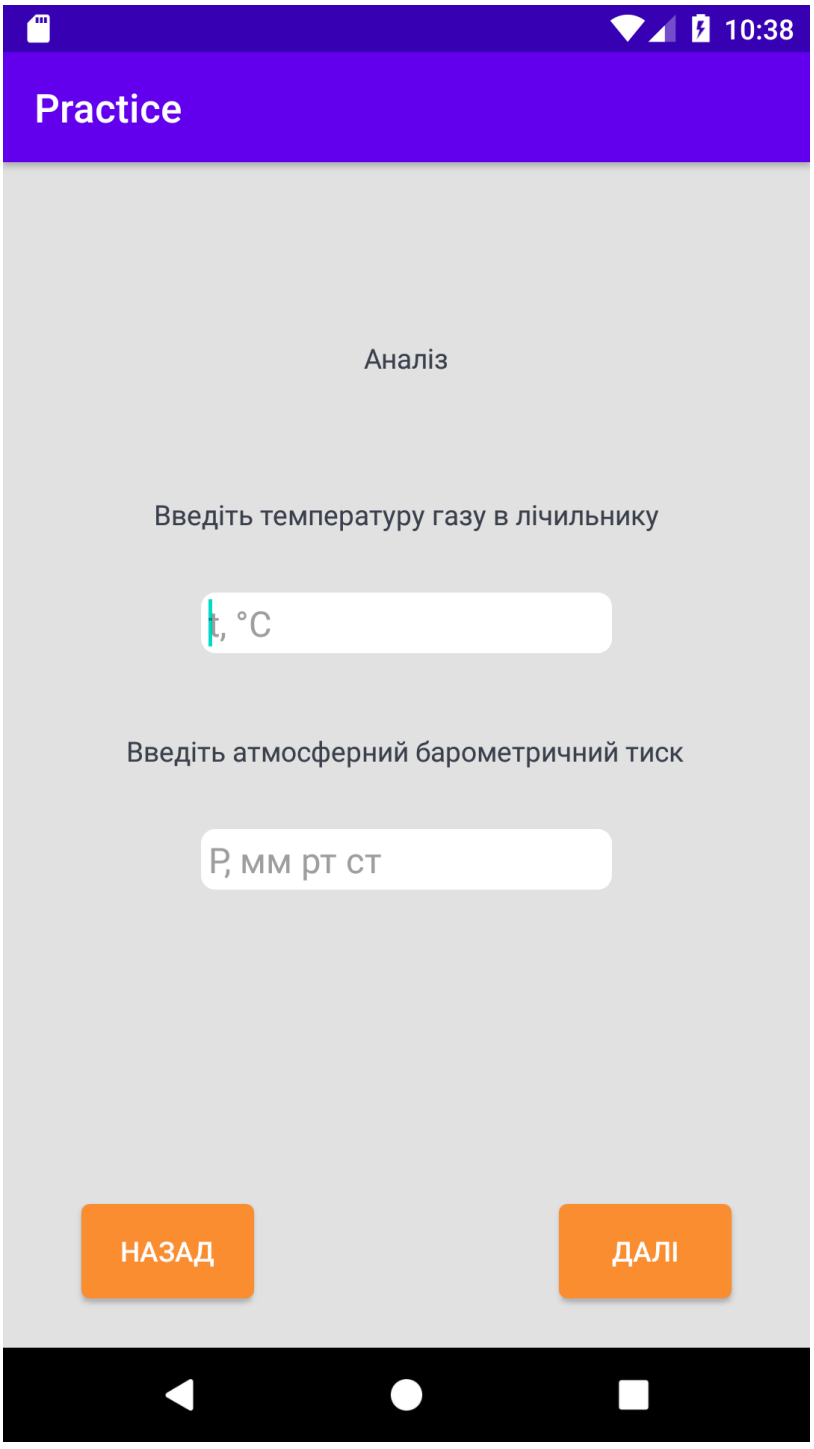

Рисунок 2.14 – Активність AnalysisStart

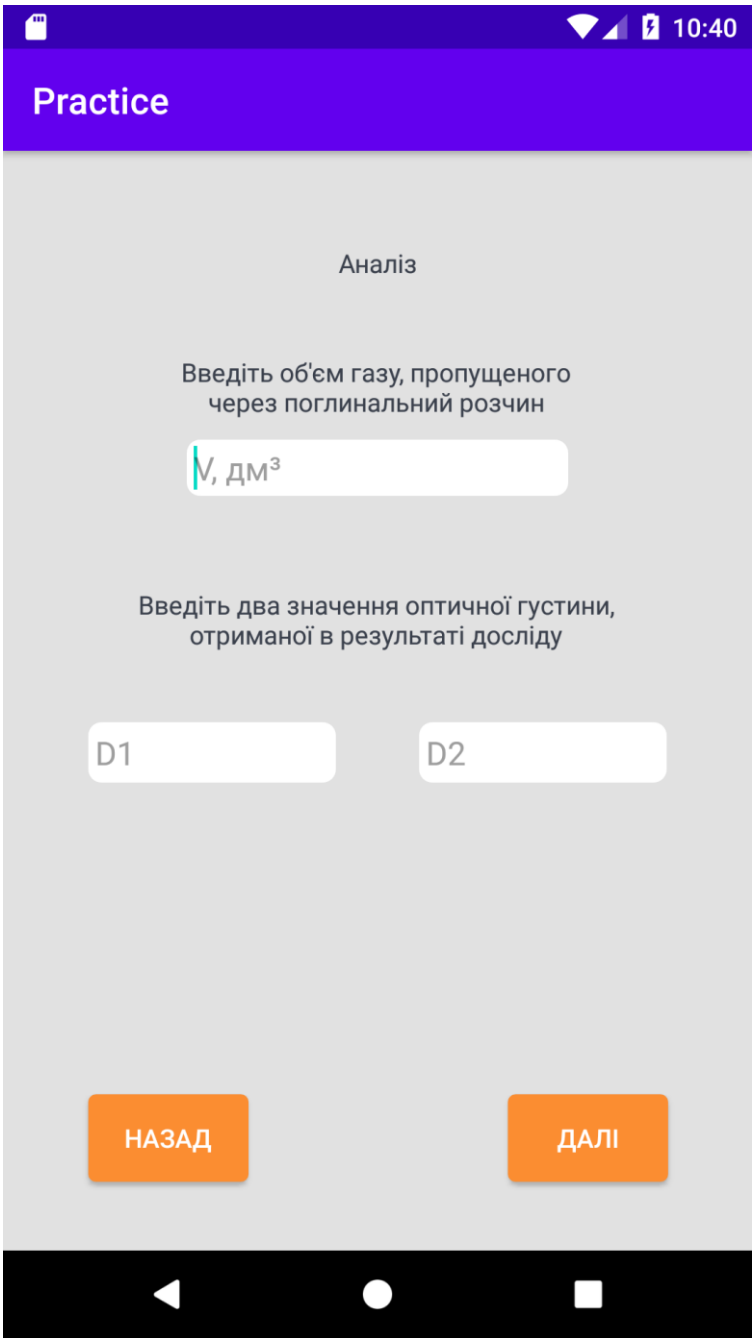

Рисунок 2.15 – Активність AnalysisNext

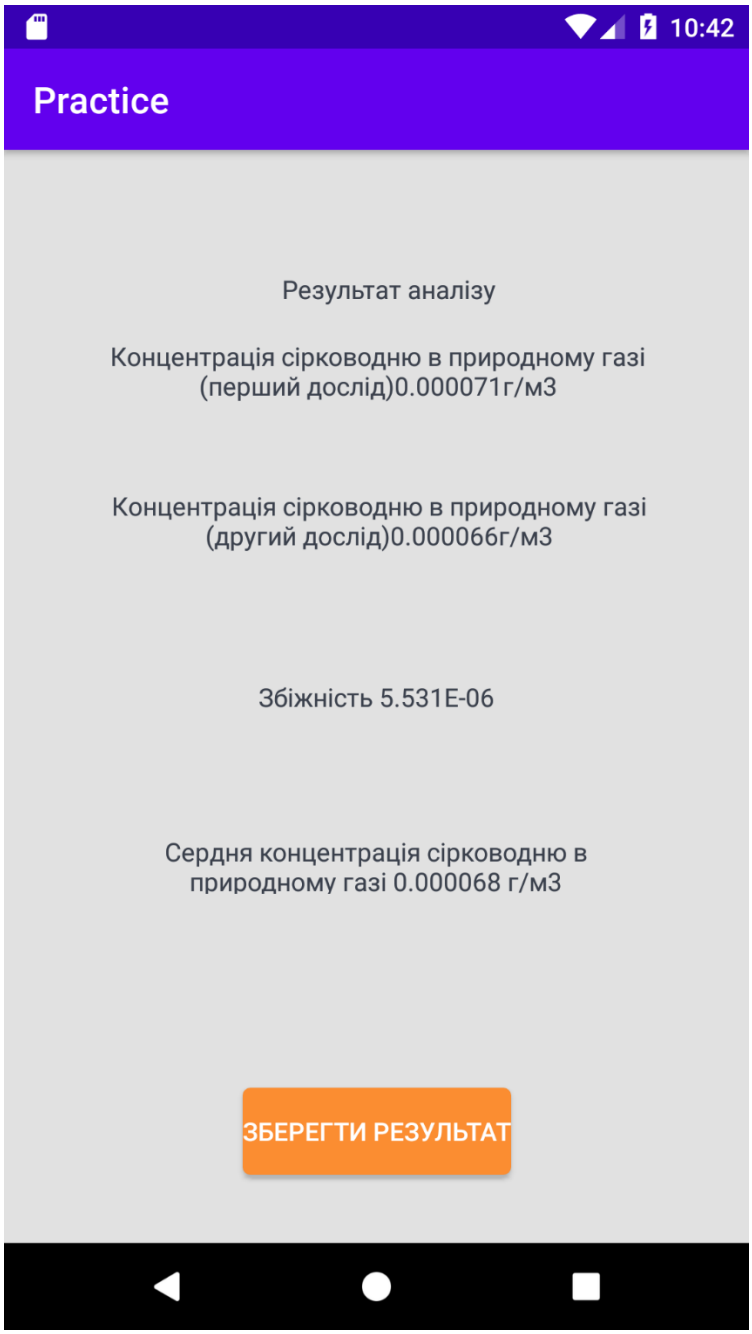

Рисунок 2.16 – Активність AnalysisResult

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Опис алгоритму апроксимації

Будь яка лінійна функція може бути представлена у вигляді

$$y = kx + b$$

Задача апроксимації полягає в знаходженні коефіцієнтів рівняння прямої k та b таких, що всі точки лежали якомога ближче до прямої.

Для цього використовується метод найменших квадратів (МНК), що полягає в тому, щоб сума квадратів відхилень від апроксимуючої прямої була мінімальна. Метод наведено в працях [1, 2].

$$F(a,b) = \sum_{i=1}^{n} (y_i - (ax_i + b))^2$$

Рішення даної задачі зводиться до пошуку екстремуму функції двох змінних.

$$\begin{cases} \dfrac{\partial F(a,b)}{\partial a} = -2\sum_{i=1}^{n} \big(y_i - (ax_i + b)\big)x_i = 0 \\ \dfrac{\partial F(a,b)}{\partial b} = -2\sum_{i=1}^{n} \big(y_i - (ax_i + b)\big) = 0 \end{cases}$$

Обчислюємо значення коефіцієнтів

$$\begin{cases} a = \dfrac{n \cdot \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \cdot \sum_{i=1}^{n} y_i}{n \cdot \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2} \\ b = \dfrac{\sum_{i=1}^{n} y_i - a \cdot \sum_{i=1}^{n} x_i}{n} \end{cases}$$

Для обчислення коефіцієнтів використаємо спрощення

$$sumxy = \sum_{i=1}^{n} x_i y_i \quad sumx = \sum_{i=1}^{n} x_i$$

$$sumy = \sum_{i=1}^{n} y_i \quad sumx2 = \sum_{i=1}^{n} x_i^2$$

Значення коефіцієнтів можуть бути знайдені як

$$\begin{cases} a = \dfrac{n \cdot sumxy - sumx \cdot sumy}{n \cdot sumx2 - sumx^2} \\ b = \dfrac{sumy - a \cdot sumx}{n} \end{cases}$$

Програмна реалізація алгоритму пердставлена класом Aprox.

```
public class Aprox {
    private double k;
    private double b;
    public DataPoint[] getAprox (DataPoint[] dataPoints){
        int length = dataPoints.length;
        DataPoint[] result = new DataPoint[length];
        double sumx = 0;
        double sumy = 0;
        double sumx2 = 0;
        double sumxy = 0;
        for (DataPoint dataPoint: dataPoints) {
            sumx += dataPoint.getX();
            sumy += dataPoint.getY();
            sumxy += dataPoint.getY() * dataPoint.getX();
            sumx2 += Math.pow(dataPoint.getX(),2);
        }
        k = (length * sumxy - (sumx * sumy))/(length * sumx2 -
Math.pow(sumx,2));
        b = (sumy - k * sumx) / (length * 1.0);
        for (int i = 0; i < length; i++){
            result[i] = new DataPoint(dataPoints[i].getX(),k *
dataPoints[i].getX() + b);
        }
        return result;
    }
    public Aprox() {
    }
    public Aprox(DataPoint[] dataPoints){
        getAprox(dataPoints);
```

```
    }
    public double getX(double y){
        return (y - b)/k;
    }
}
```

3.2 Створення таблиць бази даних додатку

В результаті пошуку та аналізу безкоштовних рішень для хостингу MySQL бази даних було знайдено ресурс Clever Cloud. На ресурсі було створено таблиці відповідно до діаграми сутність-зв’язок, описаної в звіті виробничої практики, та заповнено початковими значеннями необхідними для використання бази даних в додатку. Опис роботи з діалектом MySQL наведено в праці [3].

Скрипт для створення таблиць наведено в Лисікова К.О. Апроксимація функцій методом найменших квадратів / С.О. Цибульник, К.О. Лисікова // Вісник інженерної академії України. – Київ, 2017. – № 1. – С. 106-110. (фахове видання)

1.Брановицька С.В. «Обчислювальна математика та програмування» / С.В. Брановицька, Р.Б. Медведєв, Ю.А. Фіалков – К.: Політехніка, 2004. — 248 с.

2.Дюбуа, Поль MySQL; М.: Вильямс; Издание 2-е - Москва, 2010. - 185 с.

3.Griffiths, Dawn & Griffiths, David (2017) Head First Android Development: A Brain-Friendly Guide. Sebastopol, California: O'Reilly Media, Inc.

4.Шилдт, Герберт Java. Полное руководство, 10-е изд. — СПб.: ООО “Альфакнига”; 2018. - 1488 с.: ил. — Парал. тит. англ. ISBN 978-5-6040043-6-4

5. Beerbohm, Max (2020) Xamarin: Xamarin for beginners, Building Your First Mobile App with C# .NET and Xamarin - 3nd Edition. Independently published [in English].

6. Cinar, Onur (2013) Pro Android C++ with the NDK. New York City, NY: aPress [in English].

7. Griffiths, Dawn & Griffiths, David (2017) Head First Android Development: A Brain-Friendly Guide. Sebastopol, CA: O'Reilly Media, Inc [in English].

8. Leiva, Antonio (2016) Kotlin for Android Developers: Learn Kotlin the easy way while developing an Android App. Scotts Valley, CA: CreateSpace Independent Publishing Platform [in English].

9. Coronel, Carlos & Morris, Steven (2018) Database Systems Design, Implementation, & Management, Loose-Leaf Version. Boston, MA: CENGAGE Learning [in English].

10. Bell, Charles (2018) Introducing InnoDB Cluster: Learning the MySQL High Availability Stack. New York City, NY: aPress [in English].

11. Kreibich, Jay A. (2010) Using SQLite. Sebastopol, CA: O'Reilly Media, Inc [in English].

12. Табунщик Г. В. Проектування та моделювання програмного забезпечення сучасних інформаційних систем / Г. В. Табунщик, Т. І. Каплієнко, О.А. Петрова − Запоріжжя : Дике Поле, 2016. – 250 с.

13. Ларман, Крэг Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итеративную разработку / Крэг Ларман. - Москва: Гостехиздат, 2017. - 736 с.

14. Анісімов А.В. Інформаційні системи та бази даних: Навчальний посібник для студентів факультету комп’ютерних наук та кібернетики. / А.В. Анісімов, П.П. Кулябко − Київ. – 2017. – 110 с.

15. Android Studio Development [Електронний ресурс] // Android Platform & Tools Survey. – 2017. – Режим доступу до ресурсу: https://developer.android.com/.

16. Варфел Т. Прототипирование. Практическое руководство. - М.: Ман, Иванов и Фербер, 2013. - 240 с.

17.Гринберг С., Карпендейл Ш., Маркардт Н., Бакстон Б. UX-дизайн. Идея – эскиз – воплощение. – СПб.: Издательство "Питер", 2014. - 272 с.

18. Мюллер, Р.Дж. Базы данных и UML. Проектирование / Р.Дж. Мюллер. - М.: ЛОРИ, 2017. - 420 с.

ДОДАТОК.

Для роботи з базою даних в додатку створено клас DAOProj.

```
public class DAOProj {
    private static void connect(){
        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            con = DriverManager.getConnection(url, user, password);
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        } catch (InstantiationException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void disconnect(){
        try {
            con.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static ArrayList<String> getDevices(){
        ArrayList<String> result = new ArrayList<String>();
        try {
            connect();
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT devname FROM device");
            while (rs.next()) {
```

```
                result.add(rs.getString(1));
            }
        } catch (SQLException sqlEx) {
            sqlEx.printStackTrace();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        finally {
            disconnect();
        }
        return result;
    }

    public static boolean checkPass(String login, String pass){
        boolean result = false;
        try {
            connect();
            PreparedStatement preparedStatement = con.prepareStatement("SELECT
password FROM device where devName = ?");
            preparedStatement.setString(1, login);
            ResultSet rs = preparedStatement.executeQuery();
            rs.next();
            if (rs.getString(1).equals(pass)){
                result = true;
            }
        } catch (SQLException sqlEx) {
            sqlEx.printStackTrace();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        finally {
            disconnect();
        }
        return result;
    }

    public static int getDaysLeft(String device){
        int result = 0;
        try {
            connect();
```

```
            PreparedStatement preparedStatement = con.prepareStatement("SELECT
IF (DATEDIFF(DATE_ADD(MAX(gradDate),Interval 6 MONTH),SYSDATE())<0,
0,DATEDIFF(DATE_ADD(MAX(gradDate),Interval 6 MONTH),SYSDATE())) days_left " +
                    "FROM grad WHERE devName= ? GROUP BY devName");
            preparedStatement.setString(1, device);
            ResultSet rs = preparedStatement.executeQuery();
            rs.next();
            result = rs.getInt(1);
        } catch (SQLException sqlEx) {
            sqlEx.printStackTrace();
        }
        catch (Exception e) {
            e.printStackTrace();
        } finally {
            disconnect();
        }
        return result;
    }

    public static int getDaysLeftStabilization(String device){
        int result = 0;
        try {
            connect();
            PreparedStatement preparedStatement = con.prepareStatement("SELECT
IF (DATEDIFF(DATE_ADD(stab_date,Interval 3 MONTH),SYSDATE())<0 OR valid = false,
0,DATEDIFF(DATE_ADD(stab_date,Interval 3 MONTH),SYSDATE())) days_left " +
                    "FROM stabilization WHERE devName= ?");
            preparedStatement.setString(1, device);
            ResultSet rs = preparedStatement.executeQuery();
            rs.next();
            result = rs.getInt(1);
        } catch (SQLException sqlEx) {
            sqlEx.printStackTrace();
        }
        catch (Exception e) {
            e.printStackTrace();
        } finally {
            disconnect();
        }
        return result;
    }
```

```
    public static ArrayList<GradTableRow> getGradRows(String devName){
        ArrayList<GradTableRow> result = new ArrayList<>();
        try {
            connect();
            PreparedStatement preparedStatement = con.prepareStatement("SELECT
gt.V, gt.C, gt.D1, gt.D2, gt.D3 " +
                    "FROM grad_table gt JOIN grad g USING (grad_id) " +
                    "where (g.gradDate = (SELECT MAX(gradDate) FROM grad where
devName = ? GROUP BY devName)) and (g.devName = ?)");
            preparedStatement.setString(1,devName);
            preparedStatement.setString(2,devName);
            ResultSet rs = preparedStatement.executeQuery();
            while (rs.next()) {
                result.add(new
GradTableRow(rs.getDouble(1),rs.getDouble(2),rs.getDouble(3),rs.getDouble(4),rs.
getDouble(5)));
            }
        } catch (SQLException sqlEx) {
            sqlEx.printStackTrace();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        finally {
            disconnect();
        }
        return result;
    }

    public static void insertGrad(double[] v, double x, ArrayList<GradTableRow>
gradTableRows, String devName){
        try {
            connect();
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT MAX(grad_id) grad_id FROM
grad");
            rs.next();
            int gradId = rs.getInt(1)+1;
            try {
                Thread.sleep(1000);
            } catch (InterruptedException ex) {
                ex.printStackTrace();
```

```
            }
            PreparedStatement gradStatement = con.prepareStatement("Insert into
grad(grad_id, devName, gradDate, v, v1, v2, v3, c) " +
                    "values (?, ?, SYSDATE(), ?, ?, ?, ?, ?)");
            gradStatement.setInt(1,gradId);
            gradStatement.setString(2,devName);
            gradStatement.setDouble(3,v[0]);
            gradStatement.setDouble(4,v[1]);
            gradStatement.setDouble(5,v[2]);
            gradStatement.setDouble(6,v[3]);
            gradStatement.setDouble(7,x);
            gradStatement.executeUpdate();
            try {
                Thread.sleep(1000);
            } catch (InterruptedException ex) {
                ex.printStackTrace();
            }
            for (GradTableRow row: gradTableRows) {
                PreparedStatement statement = con.prepareStatement("Insert into
grad_table(grad_id, v, c, d1, d2, d3) values (?,?,?,?,?,?)");
                statement.setInt(1,gradId);
                statement.setDouble(2,row.getV());
                statement.setDouble(3,row.getC());
                statement.setDouble(4,row.getD1());
                statement.setDouble(5,row.getD2());
                statement.setDouble(6,row.getD3());
                statement.executeUpdate();
            }
        } catch (SQLException sqlEx) {
            sqlEx.printStackTrace();
        }
        catch (Exception e){
            System.out.println(e);
        }
        finally {
            disconnect();
        }
    }

    public static void insertResult(double result, String devName){
        try {
            connect();
```

```
            PreparedStatement preparedStatement = con.prepareStatement("Insert
into results(devName, resDate, res) values (?, SYSDATE(), ?)");
            preparedStatement.setString(1,devName);
            preparedStatement.setDouble(2,result);
            preparedStatement.executeUpdate();
            try {
                Thread.sleep(1000);
            } catch (InterruptedException ex) {
                ex.printStackTrace();
            }
        } catch (SQLException sqlEx) {
            sqlEx.printStackTrace();
        }
        catch (Exception e){
            e.printStackTrace();
        }
        finally {
            disconnect();
        }
    }

    public static void setStabilizationInvalid(String devName){
        try {
            connect();
            PreparedStatement preparedStatement = con.prepareStatement("UPDATE
stabilization SET valid = false WHERE devName = ?");
            preparedStatement.setString(1,devName);
            preparedStatement.executeUpdate();
            try {
                Thread.sleep(1000);
            } catch (InterruptedException ex) {
                ex.printStackTrace();
            }
        } catch (SQLException sqlEx) {
            sqlEx.printStackTrace();
        }
        catch (Exception e){
            e.printStackTrace();
        }
        finally {
            disconnect();
        }
```

```
    }

    public static void updateStabilization(String devName){
        try {
            connect();
            PreparedStatement preparedStatement = con.prepareStatement("UPDATE
stabilization SET valid = true, stab_date = SYSDATE() WHERE devName = ?");
            preparedStatement.setString(1,devName);
            preparedStatement.executeUpdate();
            try {
                Thread.sleep(1000);
            } catch (InterruptedException ex) {
                ex.printStackTrace();
            }
        } catch (SQLException sqlEx) {
            sqlEx.printStackTrace();
        }
        catch (Exception e){
            e.printStackTrace();
        }
        finally {
            disconnect();
        }
    }
}
```

Методи connect та disconnect призначені для підключення та розриву з'єднання з базою даних. Інші методи будуть описані при описі роботи контролерів.

3.3 Налаштування механізму авторизації

Розробка додатку відбувалась у середовищі Android Studio з використанням мови програмування Java. В якості посібників використовувались джерела [**Ошибка! Источник ссылки не найден.**, 22]

Після запуску додатку необхідно пройти процедуру авторизації. Для авторизації використовуються записи таблиці device. Алгоритм виконання авторизації представлений на блок-схемі (Рисунок 3.1).

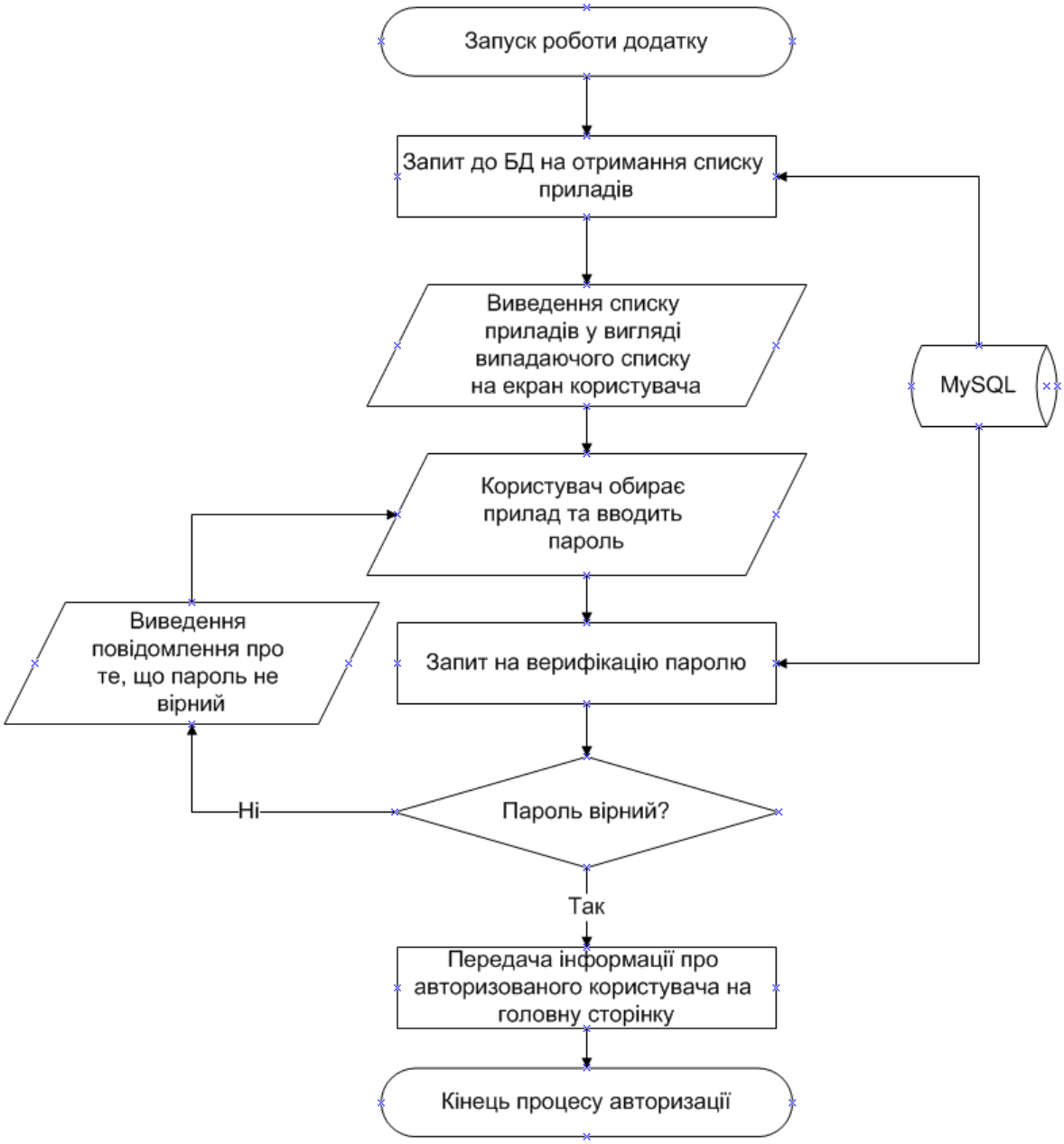

Рисунок 3.1 – Блок-схема механізму авторизації

Інтерфейс користувача виконаний в візуальному редакторі Android Studio (Рисунок 3.2).

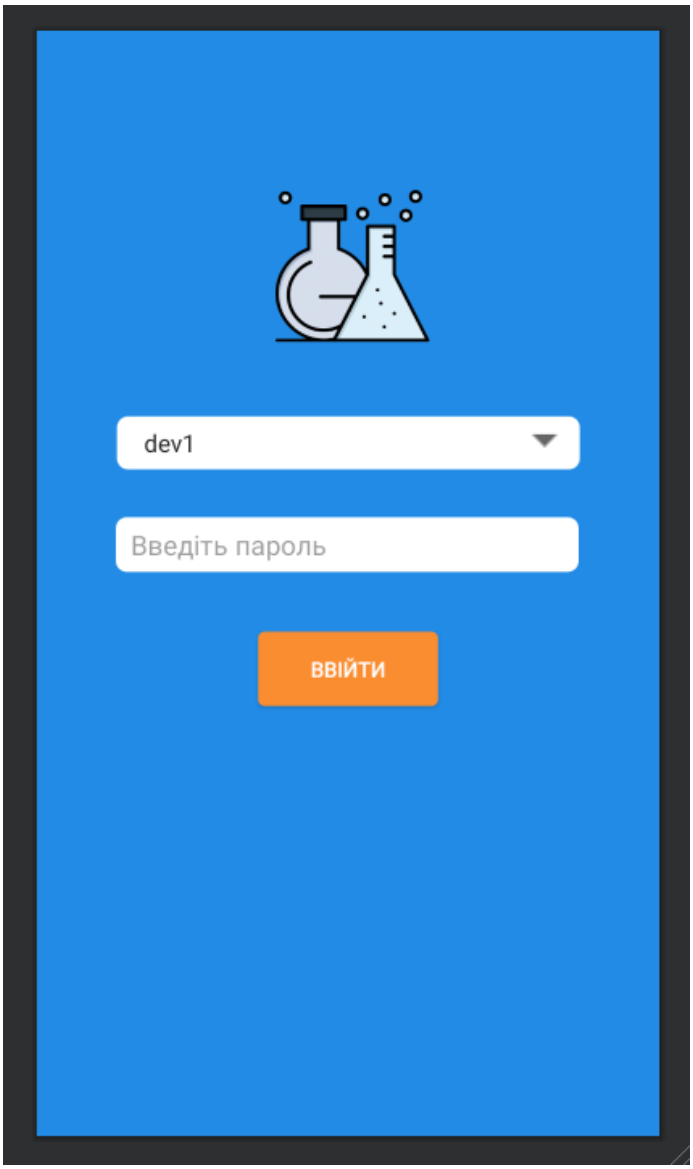

Рисунок 3.2 – Інтерфейс авторизації

Для програмної реалізації використовуються класи Authorization в якості контролера та DAOProj в якості класу підключення до бази даних.

При переході на сторінку виконується метод onCreate класу Authorization.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    ImageView iv = findViewById(R.id.colb);
    iv.setImageResource(R.drawable.colb);
    Thread t = new Thread(() -> {
        arrayList = DAOProj.getDevices();
        handlerOnCreate.sendEmptyMessage(0);
```

```
    });
    t.start();
    try {
        t.join();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}


@SuppressLint("HandlerLeak")
private Handler handlerOnCreate = new Handler() {
    public void handleMessage(Message msg) {
            Spinner spinner = findViewById(R.id.device);
            ArrayAdapter<String> arrayAdapter = new
ArrayAdapter<String>(Authorization.this, android.R.layout.simple_spinner_item,
arrayList);
arrayAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_it
em);
            spinner.setAdapter(arrayAdapter);
    }
};
```

В даному методі використовується метод getDevices класу DAOProj, який дозволяє отримати список приладів, після чого контролер встановлює отриманий список як контент випадаючого списку та логотип додатку.

Для авторизації користувач вибирає потрібний йому прилад та вводить пароль, після чого натискає кнопку ввійти, яка запускає метод контролера onClickLogIn.

```
public void onClickLogIn(View view){
    Spinner device = findViewById(R.id.device);
    deviceName = String.valueOf(device.getSelectedItem());
    EditText pass = findViewById(R.id.pass);
    String passVal = pass.getText().toString();
    Thread t = new Thread(() -> {
        result = DAOProj.checkPass(deviceName,passVal);
        handlerOnLogIn.sendEmptyMessage(0);
    });
    t.start();
```

```
    try {
        t.join();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

@SuppressLint("HandlerLeak")
private Handler handlerOnLogIn = new Handler() {
    public void handleMessage(Message msg) {
        if (!result){
            AlertDialog.Builder builder = new
AlertDialog.Builder(Authorization.this);
            builder.setMessage("Введіть вірний пароль")
                    .setTitle("Не вірний пароль");
            AlertDialog dialog = builder.create();
            dialog.show();
        }
        else{
            Intent intent = new Intent(Authorization.this,MainPage.class);
            intent.putExtra(MainPage.DEV_NAME, deviceName);
            startActivity(intent);
        }

    }
};
```

В даному методі використовується метод checkPass класу DAOProj, який перевіряє чи відповідає введений користувачем пароль паролю до обраного приладу, після чого, якщо результат вірний, відбувається перехід на головну сторінку додатку, якщо ні – виводиться діалогове вікно, в якому повідомляється що пароль не вірний, після чого користувач може спробувати авторизуватись знову.

3.4 Дизайн головної сторінки додатку

Після авторизації користувач потрапляє на головну сторінку. Головна сторінка містить градуювальний графік та дані про дату останнього градуювання та кількість днів до наступного обов’язкового градуювання.

Також з головної сторінки користувач може подивитись дані про останнє градуювання в тестовому форматі, розпочати процедуру градуювання та процедуру аналізу.

Інтерфейс користувача виконаний в візуальному редакторі Android Studio (Рисунок 3.3).

Рисунок 3.3 – Інтерфейс головної сторінки

Для програмної реалізації використовуються класи MainPage в якості контролера, DAOProj в якості класу підключення до бази даних, PractUtils для перетворення даних отриманих з бази в точки для графіку, Aprox для ароксимації занесених в базу даних точок.

При переході на сторінку виконується метод onCreate класу MainPage.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main_page);
    TextView device = findViewById(R.id.devName);
    Intent intent = getIntent();
    this.devName = intent.getStringExtra(DEV_NAME);
    device.setText("Градуювальний графік приладу " + this.devName);

    Thread t = new Thread(() -> {
        dateGrad = DAOProj.getDaysLeft(devName);
        gradTableRows = DAOProj.getGradRows(devName);
        String[] subStr = dateGrad.split(" ");
        daysLeft = Integer.parseInt(subStr[subStr.length-1]);
        onCreateHandler.sendEmptyMessage(0);
    });
    t.start();
    try {
        t.join();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

@SuppressLint("HandlerLeak")
private Handler onCreateHandler = new Handler() {
    @SuppressLint("SetTextI18n")
    public void handleMessage(Message msg) {
        TextView dateGradView = findViewById(R.id.gradDate);
        dateGradView.setText("До наступного градуювання " + daysLeft + " днів.\n
До наступної перевірки стабільності " + stabDaysLeft + " днів.");
        GraphView graph = findViewById(R.id.graph);
        graph.getViewport().setYAxisBoundsManual(true);
        graph.getGridLabelRenderer().setVerticalAxisTitle("Оптична густина, D");
        graph.getGridLabelRenderer().setHorizontalAxisTitle("Концентрація, С
мкг");
        graph.getViewport().setMinY(0);
        graph.getViewport().setMaxY(0.6);
        graph.getViewport().setXAxisBoundsManual(true);
        graph.getViewport().setMinX(0);
        graph.getViewport().setMaxX(50);
```

```
        StaticLabelsFormatter staticLabelsFormatter = new
StaticLabelsFormatter(graph);
        staticLabelsFormatter.setVerticalLabels(new String[] {"0", "0.1",
"0.2","0.3","0.4","0.5","0.6"});
        graph.getGridLabelRenderer().setLabelFormatter(staticLabelsFormatter);
        staticLabelsFormatter.setHorizontalLabels(new String[]
{"0","10","20","30","40","50"});
        graph.getGridLabelRenderer().setLabelFormatter(staticLabelsFormatter);
        graph.getGridLabelRenderer().setTextSize(30);
        graph.getGridLabelRenderer().reloadStyles();
        GradTableRows.getInstance().setGradTableRows(gradTableRows);
        DataPoint[] dots = PractUtils.getDataPoints(gradTableRows);
        PointsGraphSeries<DataPoint> series = new PointsGraphSeries<>(dots);
        Aprox aprox = new Aprox();
        LineGraphSeries<DataPoint> series2 = new
LineGraphSeries<>(aprox.getAprox(dots));
        series.setColor(Color.GREEN);
        series2.setColor(Color.GREEN);
        graph.addSeries(series);
        graph.addSeries(series2);
    }
};
```

В даному методі використовуються методи getDaysLeft, getDaysLeftStabilization та getGradRows класу DAOProj. getDaysLeft дозволяє отримати строку, що містить в собі повідомлення про дату останнього градуювання та кількість днів до наступного обов’язкового градуювання, getDaysLeftStabilization – до наступноі перевірки градуювальної характеристики. getGradRows дозволяє отримати дані градуювальної таблиці у вигляді списку об’єктів класу GradTableRow.

```
public class GradTableRow implements Serializable {
    private double v;
    private double c;
    private double d1;
    private double d2;
    private double d3;
    public GradTableRow(double v, double c, double d1, double d2, double d3) {
        this.v = v;
        this.c = c;
```

```
        this.d1 = d1;
        this.d2 = d2;
        this.d3 = d3;
    }
    public double getV() {
        return v;
    }
    public double getC() {
        return c;
    }
    public double getD1() {
        return d1;
    }
    public double getD2() {
        return d2;
    }
    public double getD3() {
        return d3;
    }
    public double getAvD(){
        return (d1 + d2 + d3)/3;
    }
}
```

Після отримання градуювальної таблиці вона записується до класу GradTableRows, який являє собою сінглетон клас для збереження записів градуювальної таблиці.

```
public class GradTableRows {
    private static GradTableRows instance = new GradTableRows();
    private ArrayList<GradTableRow> gradTableRows;

    private GradTableRows() {}

    public static GradTableRows getInstance() { return instance; }

    public ArrayList<GradTableRow> getGradTableRows() { return gradTableRows; }

    public void setGradTableRows(ArrayList<GradTableRow> gradTableRows) {
        this.gradTableRows = gradTableRows;
    }
}
```

Після виконання даних методів викликається метод getDataPoints класу PractUtils, що перетворює список об’єктів класу GradTableRow в масив DataPoints, що можуть бути використані для заповнення графіку. Після цього створюється об’єкт класу Aprox та через метод getAprox повертаються апроксимовані точки.

```
public class PractUtils {
    public static boolean isEmpty(EditText etText) {
        return etText.getText().toString().trim().length() == 0;
    }
    public static DataPoint[] getDataPoints(ArrayList<GradTableRow>
gradTableRows){
        int length = gradTableRows.size();
        DataPoint[] dataPoints = new DataPoint[length];
        for (int i = 0; i < length; i++){
            dataPoints[i] = new
DataPoint(gradTableRows.get(i).getC(),gradTableRows.get(i).getAvD());
        }
        return dataPoints;
    }
}
```

Після загрузки сторінки користувач може вийти з системи натиснувши кнопку Вийти. Після цього запуститься подія onLogOutClick після чого користувач переходить на сторінку авторизації.

```
public void OnLogOutTClick (View view){
    Intent intent = new Intent(this, Authorization.class);
    startActivity(intent);
}
```

Користувач може натиснути кнопку Градуювання, після чого запуститься подія onStartGradClick та розпочнеться процедура градуювання.

```
public void onStartGradClick (View view){
    Intent intent = new Intent(this, StartGrad.class);
    intent.putExtra(StartGrad.DEV_NAME, devName);
    intent.putExtra(StartGrad.STATE,1);
    startActivity(intent);
}
```

Користувач може натиснути кнопку Стабілізація, після чого запуститься подія onStartStabilization та розпочнеться процедура перевірки стабільності градуювальної характеристики.

```
public void onStartStabilization (View view){
    Intent intent = new Intent(this, StartGrad.class);
    intent.putExtra(StartGrad.DEV_NAME, devName);
    intent.putExtra(StartGrad.STATE, 2);
    startActivity(intent);
}
```

Користувач може натиснути кнопку Аналіз, після чого запуститься подія onStartAnalysisClick.

```
public void onStartAnalysisClick (View view){
    if (daysLeft <= 0){
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage("Для початку аналізу проведіть процедуру
градуювання")
                .setTitle("")
                .setPositiveButton("Перейти до початку градуювання", (dialog,
id) -> {
                    Intent intent = new Intent(MainPage.this,StartGrad.class);
                    intent.putExtra(StartGrad.DEV_NAME, devName);
                    intent.putExtra(StartGrad.STATE,1);
                    startActivity(intent);
                });
        AlertDialog dialog = builder.create();
        dialog.show();
    } else {
        if (stabDaysLeft <= 0){
            AlertDialog.Builder builder = new AlertDialog.Builder(this);
            builder.setMessage("Для початку аналізу проведіть процедуру
стабілізації")
                    .setTitle("")
                    .setPositiveButton("Перейти до початку стабілізації",
(dialog, id) -> {
                        Intent intent = new
Intent(MainPage.this,StartGrad.class);
                        intent.putExtra(StartGrad.DEV_NAME, devName);
                        intent.putExtra(StartGrad.STATE,2);
```

```
                        startActivity(intent);
                    });
            AlertDialog dialog = builder.create();
            dialog.show();
        } else {
            Intent intent = new Intent(this, AnalysisStart.class);
            intent.putExtra(AnalysisStart.DEV_NAME, devName);
            startActivity(intent);
        }
    }
}
```

Метод перевірить що кількість днів що залишились до наступного градуювання та наступної перевірки стабільності більше 0. Якщо будь яке з цих значень дорівнює 0 виводить відповідне повідомлення про це. Якщо обидва значення більше 0 розпочинає процедуру аналізу.

3.5 Реалізація алгоритму проведення градуювання

Алгоритм градуювання представлений на блок-схемі (Рисунок 3.4).

Рисунок 3.4 – Блок-схема градуювання

Для градуювання використано декілька сторінок. Ці ж сторінки використано для проведення перевірки стабільності градуювальної характеристики (дивись розділ 3.6). Після початку процедури користувач потрапляє на сторінку, що керується контролером StartGrad. Інтерфейс користувача виконаний в візуальному редакторі Android Studio (Рисунок 3.5).

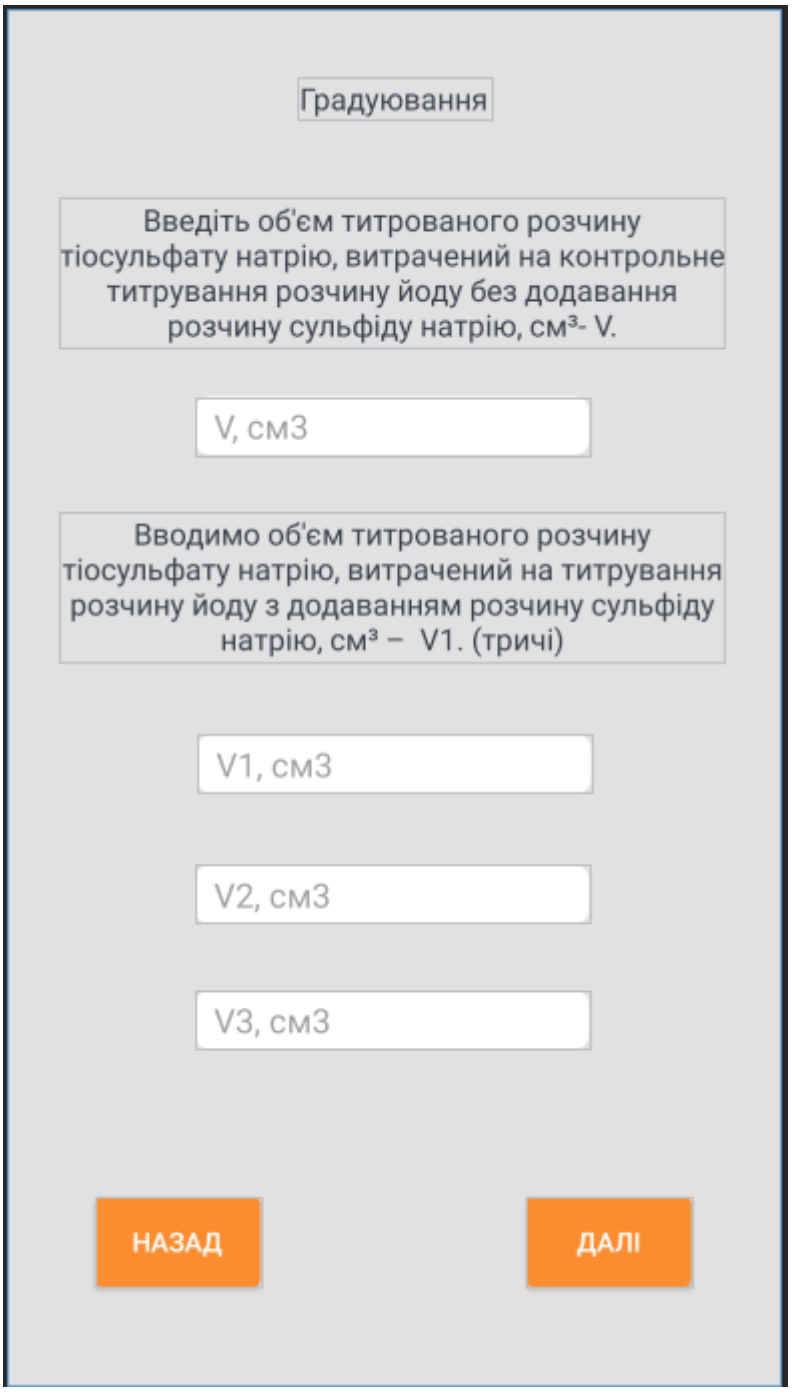

Рисунок 3.5 – Інтерфейс сторінки початку градуювання.

Після того, як користувач заповнить відповідні поля він натискає кнопку Далі, та викликає подію onNextClick.

```
public void onNextClick (View view){
    EditText v = findViewById(R.id.v);
    EditText v1 = findViewById(R.id.v1);
    EditText v2 = findViewById(R.id.v2);
    EditText v3 = findViewById(R.id.v3);
    if (PractUtils.isEmpty(v) || PractUtils.isEmpty(v1) ||
PractUtils.isEmpty(v2) || PractUtils.isEmpty(v3)){
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage("Заповніть всі поля")
                .setTitle("Не заповнені поля");
        AlertDialog dialog = builder.create();
        dialog.show();
    } else {
        double[] valuesV = new double[4];
        valuesV[0] = Double.parseDouble(v.getText().toString());
        valuesV[1] = Double.parseDouble(v1.getText().toString());
        valuesV[2] = Double.parseDouble(v2.getText().toString());
        valuesV[3] = Double.parseDouble(v3.getText().toString());
        double vAv = (valuesV[1] + valuesV[2] + valuesV[3]) / 3;
        double x = (valuesV[0] - vAv) * 17 / 20;
        Intent intent = new Intent(this, GradTableFill.class);
        intent.putExtra(GradTableFill.DEV_NAME, message);
        intent.putExtra(GradTableFill.V_ARR,valuesV);
        intent.putExtra(GradTableFill.CONC,x);
        intent.putExtra(GradTableFill.STATE,state);
        startActivity(intent);
    }
}
```

Метод перевіряє чи заповнені всі поля, та виводить повідомлення, якщо деяке поле не заповнене. Якщо всі поля заповнені проводиться розрахунок V середнього та концентрації, після чого дані передаються на наступну сторінку.

Наступна сторінка керується контролером GradTableFill. Інтерфейс користувача виконаний в візуальному редакторі Android Studio (Рисунок 3.6).

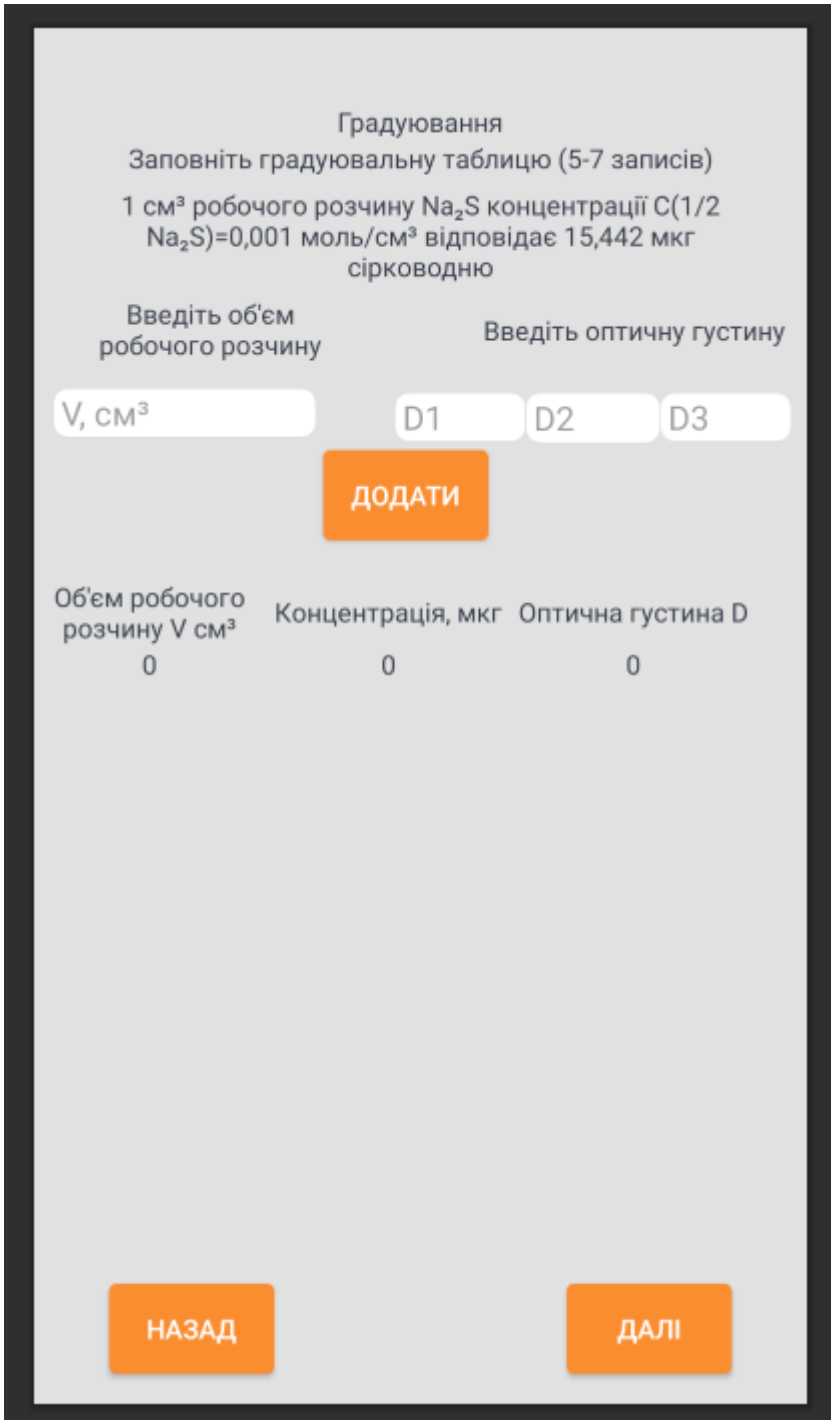

Рисунок 3.6 – Інтерфейс заповнення градуювальної таблиці.

Заповнивши всі поля користувач натискає кнопку Додати, та запускає подію onAddRowClick.

```
public void onAddRowClick(View view){
    EditText vEdit = findViewById(R.id.numberV);
    EditText d1 = findViewById(R.id.numberD1);
    EditText d2 = findViewById(R.id.numberD2);
    EditText d3 = findViewById(R.id.numberD3);
```

```
    if (PractUtils.isEmpty(vEdit) || PractUtils.isEmpty(d1) ||
PractUtils.isEmpty(d2) || PractUtils.isEmpty(d3)){
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage("Заповніть всі поля")
                .setTitle("Не заповнені поля");
        AlertDialog dialog = builder.create();
        dialog.show();
    } else {
        i++;
        double vValue = Double.parseDouble(vEdit.getText().toString());
        double d1Value = Double.parseDouble(d1.getText().toString());
        double d2Value = Double.parseDouble(d2.getText().toString());
        double d3Value = Double.parseDouble(d3.getText().toString());
        gradTableRows.add(new GradTableRow(vValue,vValue * x, d1Value, d2Value,
d3Value));
        TableLayout tableLayout = (TableLayout)
findViewById(R.id.gradTableFill);
        TableRow tableRow = new TableRow(this);
        tableRow.setLayoutParams(new
TableLayout.LayoutParams(TableLayout.LayoutParams.MATCH_PARENT,
                TableLayout.LayoutParams.WRAP_CONTENT));
        TextView v = new TextView(this);
        v.setText(decimalFormat.format(gradTableRows.get(i).getV()));
        v.setGravity(Gravity.CENTER);
        v.setTextColor(ContextCompat.getColor(this, R.color.text));
        TextView c = new TextView(this);
        c.setText(decimalFormat.format(gradTableRows.get(i).getC()));
        c.setGravity(Gravity.CENTER);
        c.setTextColor(ContextCompat.getColor(this, R.color.text));
        TextView d = new TextView(this);
        d.setText(decimalFormat.format(gradTableRows.get(i).getAvD()));
        d.setGravity(Gravity.CENTER);
        d.setTextColor(ContextCompat.getColor(this, R.color.text));
        tableRow.addView(v, 0);
        tableRow.addView(c, 1);
        tableRow.addView(d, 2);
        tableLayout.addView(tableRow, i + 1);
        vEdit.getText().clear();
        d1.getText().clear();
        d2.getText().clear();
        d3.getText().clear();
```

```
    }
}
```

Метод перевіряє чи заповнені всі поля, та виводить повідомлення, якщо деяке поле не заповнене. Якщо всі поля заповнені проводиться розрахунок середнього значення оптичної густини та концентрації, після чого значення виводяться в таблицю.

Після того, як користувач вніс в таблицю щонайменше 5 записів він натискає кнопку Далі та запускає подію onGradFillNextClick.

```
public void onGradFillNextClick (View view){
    if (i<goNextI) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage("Занесіть до таблиці як мінімум "+goNextI+" записів")
                .setTitle("Не достатньо даних");
        AlertDialog dialog = builder.create();
        dialog.show();
    }
    else {
        Intent intent = new Intent(this, ResGrad.class);
        intent.putExtra(ResGrad.DEV_NAME, devName);
        intent.putExtra(ResGrad.V_ARR,vValues);
        intent.putExtra(ResGrad.CONC,x);
        intent.putExtra(ResGrad.STATE,state);
        Bundle args = new Bundle();
        args.putSerializable("ARRAYLIST",gradTableRows);
        intent.putExtra(ResGrad.ROWS,args);
        startActivity(intent);
    }
}
```

Метод перевіряє щоб в таблиці було щонайменше 5 записів. Якщо записів менше виводить повідомлення про це. Якщо кількість записів більше або дорівнює 5 передає інформацію до наступної сторінки.

Наступна сторінка керується контролером ResGrad. Інтерфейс користувача виконаний в візуальному редакторі Android Studio (Рисунок 3.7).

Рисунок 3.7 – Інтерфейс результату градуювання.

Після переходу на сторінку користувач бачить градуювальний графік отриманий на основі градуювальної таблиці, яку він заповнив. За виведення графіку відповідає подія onCreate.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_res_grad);
```

```
    Intent intent = getIntent();
    devName = intent.getExtras().getString(DEV_NAME);
    vValues = intent.getExtras().getDoubleArray(V_ARR);
    x = intent.getExtras().getDouble(CONC);
    state = intent.getExtras().getInt(STATE);
    Bundle args = intent.getBundleExtra(ROWS);
    gradTableRows = (ArrayList<GradTableRow>) args.getSerializable("ARRAYLIST");
    TextView heading = findViewById(R.id.header);
    heading.setText("Прилад " + devName);
    GraphView graph = findViewById(R.id.graphGradRes);
    graph.getViewport().setScalable(true);
    graph.getViewport().setYAxisBoundsManual(true);
    graph.getGridLabelRenderer().setVerticalAxisTitle("Оптична густина, D");
    graph.getGridLabelRenderer().setHorizontalAxisTitle("Концентрація, С мкг");
    graph.getViewport().setMinY(0);
    graph.getViewport().setMaxY(0.6);
    graph.getViewport().setXAxisBoundsManual(true);
    graph.getViewport().setMinX(0);
    graph.getViewport().setMaxX(50);
    StaticLabelsFormatter staticLabelsFormatter = new
StaticLabelsFormatter(graph);
    staticLabelsFormatter.setVerticalLabels(new String[] {"0", "0.1",
"0.2","0.3","0.4","0.5","0.6"});
    graph.getGridLabelRenderer().setLabelFormatter(staticLabelsFormatter);
    staticLabelsFormatter.setHorizontalLabels(new String[]
{"0","10","20","30","40","50"});
    graph.getGridLabelRenderer().setLabelFormatter(staticLabelsFormatter);
    graph.getGridLabelRenderer().setTextSize(30);
    graph.getGridLabelRenderer().reloadStyles();
    DataPoint[] points = PractUtils.getDataPoints(gradTableRows);
    PointsGraphSeries<DataPoint> series = new PointsGraphSeries<>(points);
    Aprox aprox = new Aprox();
    dataPoints1 =aprox.getAprox(points);
    LineGraphSeries<DataPoint> series2 = new LineGraphSeries<>(dataPoints1);

    graph.addSeries(series);
    graph.addSeries(series2);
  }
```

Якщо користувач задоволений отриманим результатом він натискає кнопку зберегти та запускає подію onSaveClick.

```
public void onSaveClick (View view){
        Thread t = new Thread(() -> {
            DAOProj.insertGrad(vValues,x,gradTableRows,devName);

            DAOProj.setStabilizationInvalid(devName);
            handlerSaveGradResults.sendEmptyMessage(0);
        });
        t.start();
        try {
            t.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
}
@SuppressLint("HandlerLeak")
private Handler handlerSaveGradResults = new Handler() {
    public void handleMessage(Message msg) {
        Intent intent = new Intent(ResGrad.this, MainPage.class);
        intent.putExtra(MainPage.DEV_NAME, devName);
        startActivity(intent);
    }
};
```

Даний метод викликає методи insertGrad та setStabilizationInvalid класу DAOProj після чого переходить на головну сторінку. Метод insertGrad додає інформацію про проведене градуювання до бази даних. Метод setStabilizationInvalid переводить поточний запис про останню перевірку стабільності в не активний статус, тобто вимагає повторного проведення процедури перевірки стабільності градуювальної характеристики.

3.6 Процедура перевірки стабільності градувальної характеристики

Дана процедура схожа на процедуру проведення градуювання (дивись розділ 3.5). Тому для неї було використано ті самі класи. Алгоритм перевірки стабільності градуювальної характеристики представлений на блок-схемі (Рисунок 3.8).

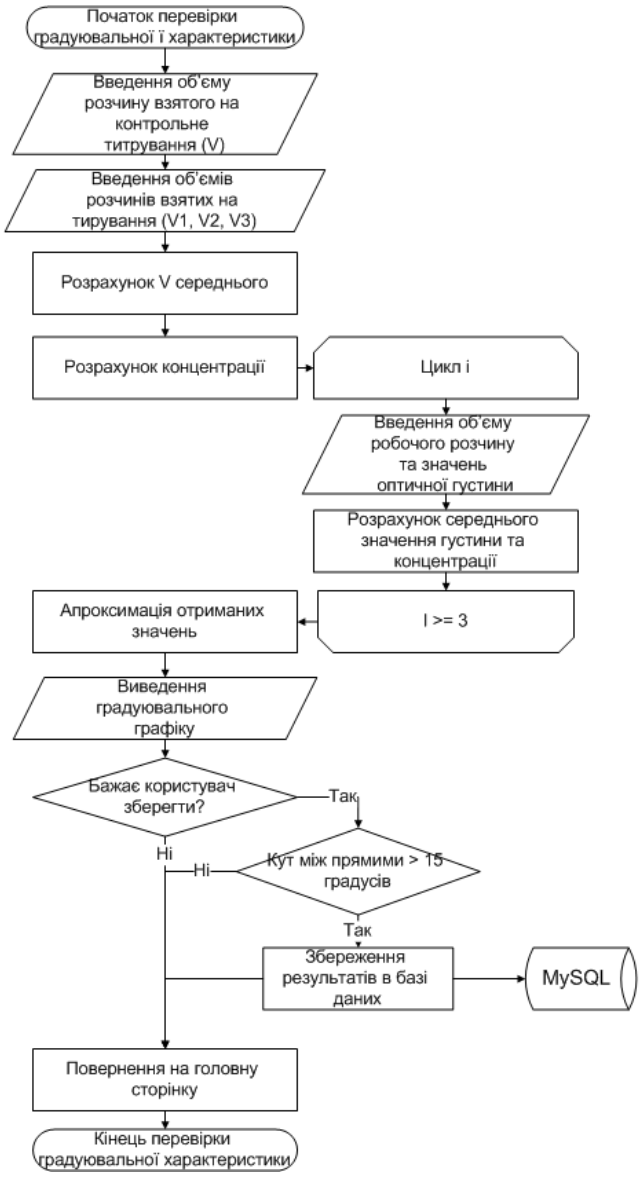

Рисунок 3.8 – Блок-схема перевірки градуювальної характеристики

Класи StartGrad та GradTableFill повністю ідентичні описаним в процедурі градуювання. Змін зазнав лише клас ResGrad.

До методу onCreate класу ResGrad додали наступний код:

```
if (state == 2) {
    series.setColor(Color.RED);
    series2.setColor(Color.RED);
} else{
    series.setColor(Color.GREEN);
    series2.setColor(Color.GREEN);
}
graph.addSeries(series);
```

```
graph.addSeries(series2);
if (state == 2){
    gradTableRowsFromDB = GradTableRows.getInstance().getGradTableRows();
    graph = findViewById(R.id.graphGradRes);
    DataPoint[] dots = PractUtils.getDataPoints(gradTableRowsFromDB);
    Aprox aprox1 = new Aprox();
    dataPoints2 = aprox1.getAprox(dots);
    PointsGraphSeries<DataPoint> series3 = new PointsGraphSeries<>(dots);
    LineGraphSeries<DataPoint> series4 = new LineGraphSeries<>(dataPoints2);
    series3.setColor(Color.GREEN);
    series4.setColor(Color.GREEN);
    graph.addSeries(series3);
    graph.addSeries(series4);
    DataPoint vec1 = new DataPoint(0.1*(dataPoints1[0].getX()-
dataPoints1[1].getX()),10*(dataPoints1[0].getY()-dataPoints1[1].getY()));
    DataPoint vec2 = new DataPoint(0.1*(dataPoints2[0].getX()-
dataPoints2[1].getX()),10*(dataPoints2[0].getY()-dataPoints2[1].getY()));
    double cosAngle = (vec1.getX()*vec2.getX() +
vec1.getY()*vec2.getY())/(Math.sqrt(Math.pow(vec1.getX(),2) +
Math.pow(vec1.getY(),2)) * Math.sqrt(Math.pow(vec2.getX(),2) +
Math.pow(vec2.getY(),2)));
    angle = Math.toDegrees(Math.acos(Math.abs(cosAngle)));
    AlertDialog.Builder builder = new AlertDialog.Builder(ResGrad.this);
    builder.setMessage( decimalFormat.format(angle))
            .setTitle("Кут між прямими");
    AlertDialog dialog = builder.create();
    dialog.show();
}
```

Даний код знаходить апроксимовану пряму на основі введених користувачем даних та шукає кут між нею та апроксимованою прямою побудованою на основі градуювальної характеристики.

При виклику методу onSaveClick для даного алгоритму виконується наступний код:

```
if (angle > 15){
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setMessage("Кут більше 15 градусів")
            .setTitle("Стабільності не досягнено")
            .setPositiveButton("Повернутись до початку перевірки стабільності",
```

```
(dialog, id) -> {
                Intent intent = new Intent(ResGrad.this,StartGrad.class);
                intent.putExtra(StartGrad.DEV_NAME, devName);
                intent.putExtra(StartGrad.STATE,2);
                startActivity(intent);
            })
            .setNegativeButton("Розпочати градування", (dialog, id) -> {
                Intent intent = new Intent(ResGrad.this,StartGrad.class);
                intent.putExtra(StartGrad.DEV_NAME, devName);
                intent.putExtra(StartGrad.STATE,1);
                startActivity(intent);
            });

    AlertDialog dialog = builder.create();
    dialog.show();
} else {
    Thread t = new Thread(() -> {
        DAOProj.updateStabilization(devName);
        handlerSaveGradResults.sendEmptyMessage(0);
    });
    t.start();
    try {
        t.join();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

@SuppressLint("HandlerLeak")
private Handler handlerSaveGradResults = new Handler() {
    public void handleMessage(Message msg) {
        Intent intent = new Intent(ResGrad.this, MainPage.class);
        intent.putExtra(MainPage.DEV_NAME, devName);
        startActivity(intent);
    }
};
```

Якщо кут між прямими більше 15 градусів виводиться повідомлення, яке пропонує користувачу повернутись до початку проведення перевірки стабільності градуювальної характеристики або перейти до самої процедури проведення градування.

Якщо кут менше 15 градусів викликає метод updateStabilization класу DAOProj, що оновлює дані про перевірку стабільності в базі даних, після чого повертає користувача на головну сторінку.

3.7 Реалізація процедури проведення аналізу

Алгоритм аналізу представлений на блок-схемі (Рисунок 3.9).

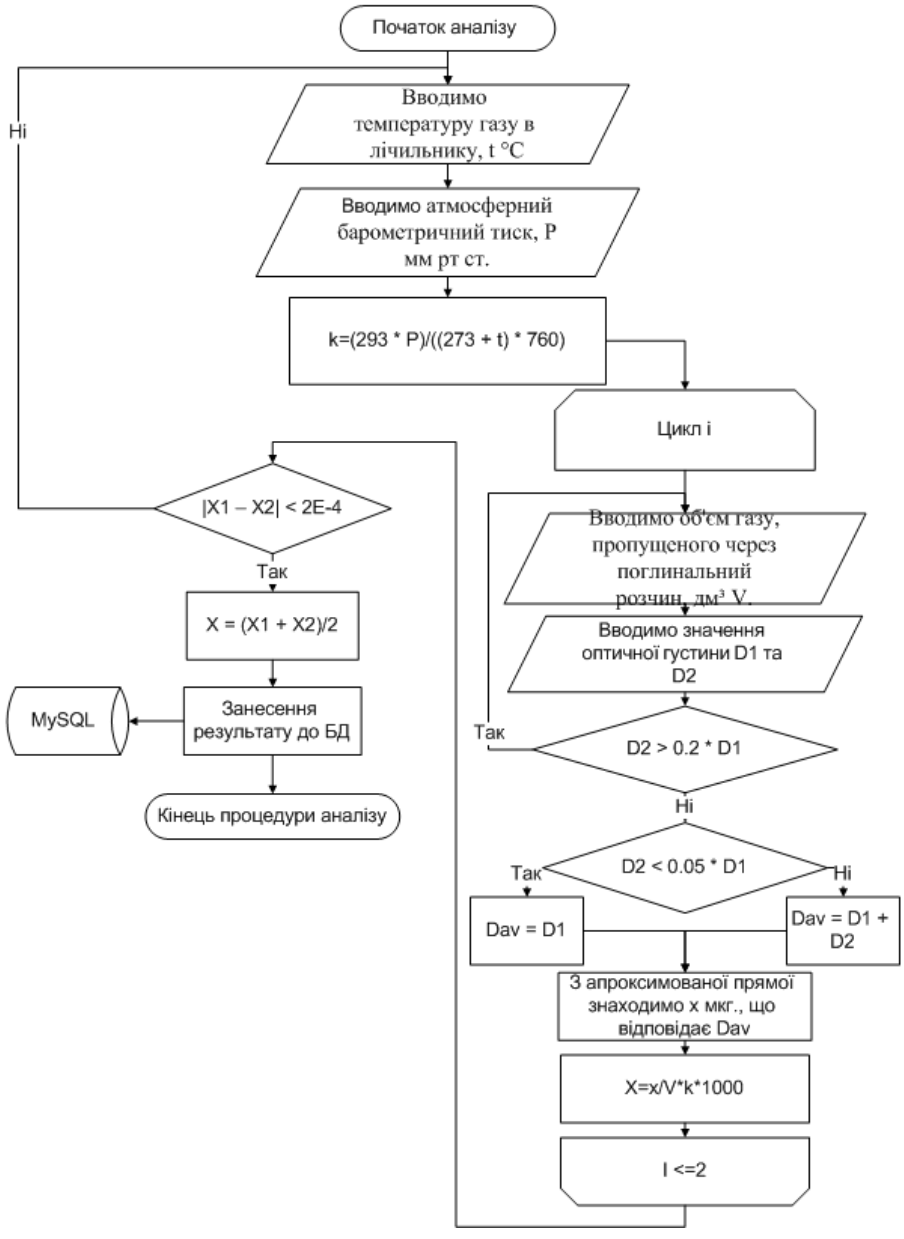

Рисунок 3.9 – Блок-схема процедури аналізу

Алгоритм аналізу використовує декілька сторінок. Після переходу до початку аналізу користувач потрапляє на сторінку, що керується контролером

AnalysisStart. Інтерфейс користувача виконаний в візуальному редакторі Android Studio (Рисунок 3.10).

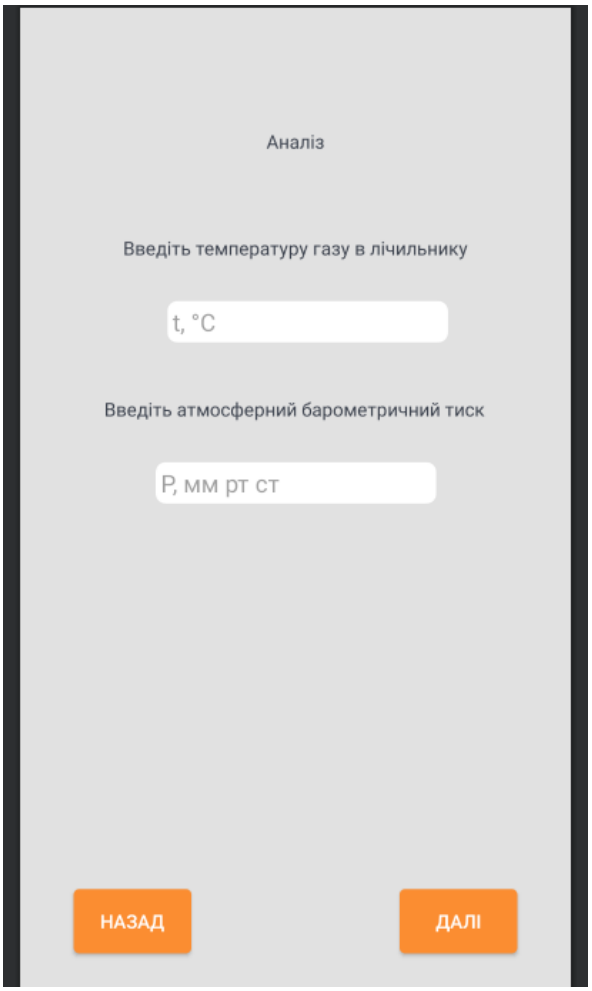

Рисунок 3.10 – Інтерфейс сторінки AnalysisStart

Користувач вводить температуру газу в лічильнику та тиск у відповідні поля, після чого натискає кнопку Далі та запускає подію onAnalysisStartNextClick.

```
public void onAnalysisStartNextClick (View view){
    EditText t = findViewById(R.id.t);
    EditText p = findViewById(R.id.p);
    if (PractUtils.isEmpty(p) || PractUtils.isEmpty(t)){
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage("Заповніть всі поля")
                .setTitle("Не заповнені поля");
        AlertDialog dialog = builder.create();
        dialog.show();
```

```
    } else {
        double tValue = Double.parseDouble(t.getText().toString());
        double pValue = Double.parseDouble(p.getText().toString());
        double k = (293 * pValue) / ((273 + tValue) * 760);
        Intent intent = new Intent(this, AnalysisNext.class);
        intent.putExtra(AnalysisNext.DEV_NAME, devName);
        intent.putExtra(AnalysisNext.K, k);
        intent.putExtra(AnalysisNext.STATE, 1);
        startActivity(intent);
    }
}
```

Даний метод перевіряє чи були заповнені всі поля. Якщо поля не заповнені виводить повідомлення про це. Якщо всі поля заповнені обраховує коефіцієнт k та переходить на наступну сторінку.

Наступна сторінка керується контролером AnalysisNext. Інтерфейс користувача виконаний в візуальному редакторі Android Studio (Рисунок 3.11).

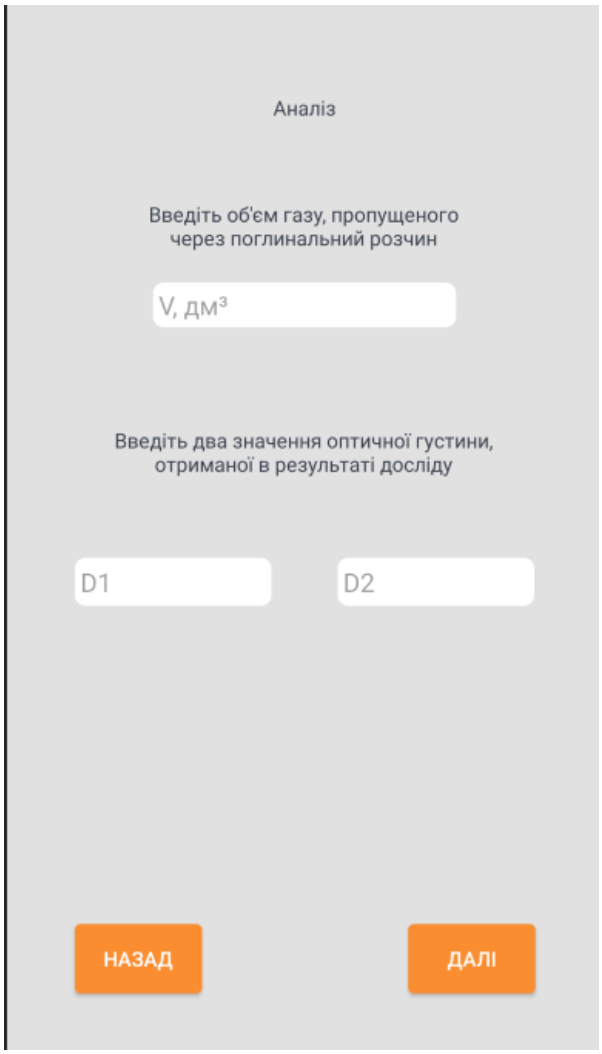

Рисунок 3.11 – Інтерфейс сторінки AnalysisNext

Даний етап користувач проходить двічі. Це необхідно для перевірки збіжності результату.

Користувач заповнює всі поля після чого натискає кнопку Далі після чого запускає подію onAnalysisNextNextClick.

```
public void onAnalysisNextNextClick (View view) {
    EditText v = findViewById(R.id.analysisV);
    EditText d1 = findViewById(R.id.analysisD1);
    EditText d2 = findViewById(R.id.analysisD2);
    if (PractUtils.isEmpty(v) || PractUtils.isEmpty(d1) ||
PractUtils.isEmpty(d2)){
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage("Заповніть всі поля")
                .setTitle("Не заповнені поля");
```

```
        AlertDialog dialog = builder.create();
        dialog.show();
    } else {
        double d1Value = Double.parseDouble(d1.getText().toString());
        double d2Value = Double.parseDouble(d2.getText().toString());
        double vValue = Double.parseDouble(v.getText().toString());
        if (d2Value > d1Value * 0.2){
            AlertDialog.Builder builder = new AlertDialog.Builder(this);
            builder.setMessage("Повторіть дослід з меншим об'ємом газу")
                    .setTitle("d2 > d1 * 0.2");
            AlertDialog dialog = builder.create();
            dialog.show();
            v.getText().clear();
            d1.getText().clear();
            d2.getText().clear();
        } else {
            double dAv;
            if (d2Value < d1Value * 0.05)
                dAv = d1Value;
            else
                dAv = d1Value + d2Value;
            Aprox aprox = new Aprox(dots);
            double x = aprox.getX(dAv);
            double c = x / (vValue * k * 1000);
            if (state == 1) {
                Intent intent = new Intent(this, AnalysisNext.class);
                intent.putExtra(AnalysisNext.DEV_NAME, devName);
                intent.putExtra(AnalysisNext.K, k);
                intent.putExtra(AnalysisNext.STATE, 2);
                intent.putExtra(AnalysisNext.C, c);
                startActivity(intent);
            } else {
                double convergence = Math.abs(c - c1);
                if (convergence > Double.parseDouble("2E-4")) {
                    AlertDialog.Builder builder = new AlertDialog.Builder(this);
                    builder.setMessage(convergence + " > " +
Double.parseDouble("2E-4"))
                            .setTitle("Стабільності не досягнено")
                            .setPositiveButton("Повернутись до початку аналізу",
(dialog, id) -> {
                                Intent intent = new Intent(AnalysisNext.this,
StartGrad.class);
```

```
                                intent.putExtra(StartGrad.DEV_NAME, devName);
                                intent.putExtra(StartGrad.STATE, 2);
                                startActivity(intent);
                            })
                            .setNegativeButton("Повернутись до головної
сторінки", (dialog, id) -> {
                                Intent intent = new Intent(AnalysisNext.this,
MainPage.class);
                                intent.putExtra(MainPage.DEV_NAME, devName);
                                startActivity(intent);
                            });

                    AlertDialog dialog = builder.create();
                    dialog.show();
                } else {
                    double[] cArr = new double[]{c1, c, (c1 + c) / 2};
                    Intent intent = new Intent(this, AnalysisResult.class);
                    intent.putExtra(AnalysisResult.DEV_NAME, devName);
                    intent.putExtra(AnalysisResult.C_ARR, cArr);
                    intent.putExtra(AnalysisResult.CONVERGENCE, convergence);
                    startActivity(intent);
                }
            }
        }
    }
}
```

Даний метод перевіряє чи заповнені всі поля та виводить повідомлення якщо деяке поле не заповнене. Після цього перевіряє що D2 < D1 * 0.2. Якщо ця умова не відповідає дійсності виводить повідомлення користувачу з проханням ввести нові дані. Якщо дані задовольняють даній умові, то на їх основі з апроксимованої прямої градуювальної характеристики знаходять масу сірководню у випробуваному розчині, після чого знаходять концентрацію.

Якщо користувач вводить ввів дані для другої ітерації досліду, то крім дій наведених вище метод також перевіряє збіжність та виводить повідомлення, якщо збіжності не досягнено. Якщо збіжність задовольняє похибці користувач переходить на сторінку з результатом.

Сторінка з результатом керується контролером AnalysisResult. Інтерфейс користувача виконаний в візуальному редакторі Android Studio (Рисунок 3.12).

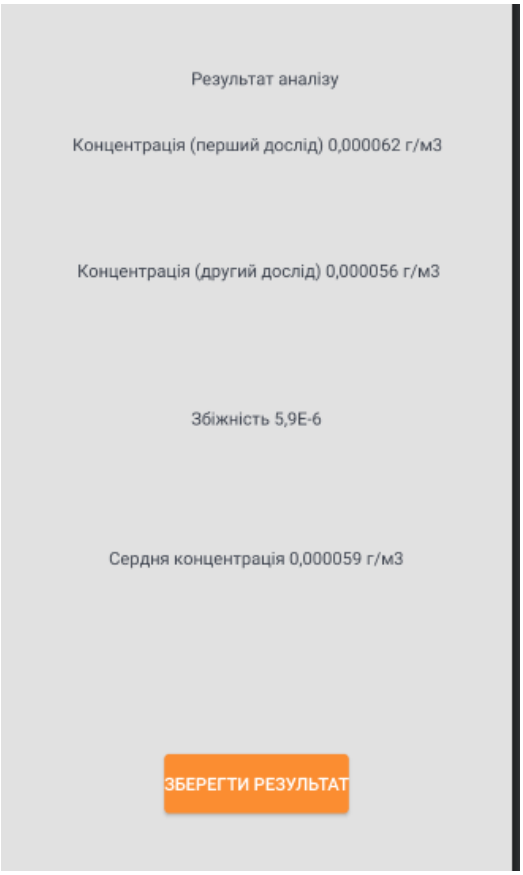

Рисунок 3.12 – Інтерфейс сторінки результату аналізу

При переході на сторінку запускається подія onCreate, що виводить результати досліду.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_analysis_result);
    Intent intent = getIntent();
    devName = intent.getExtras().getString(DEV_NAME);
    cArr = intent.getExtras().getDoubleArray(C_ARR);
    double convergence = intent.getExtras().getDouble(CONVERGENCE);
    TextView firstRes = findViewById(R.id.firstRes);
    TextView secondRes = findViewById(R.id.secondRes);
    TextView convergenceView = findViewById(R.id.convergence);
    TextView avRes = findViewById(R.id.avRes);
    firstRes.setText("Концентрація сірководню в природному газі (перший дослід)"
+ decimalFormat.format(cArr[0]) + "г/м3");
```

```
    secondRes.setText("Концентрація сірководню в природному газі (другий
дослід)" + decimalFormat.format(cArr[1]) + "г/м3");
    convergenceView.setText("Збіжність " +
decimalFormatExp.format(convergence));
    avRes.setText("Сердня концентрація сірководню в природному газі " +
decimalFormat.format(cArr[2]) + " г/м3");
}
```

Далі користувач натискає кнопку Зберегти Результат та запускає подію

```
public void onSaveResClick(View view){
    Thread t1 = new Thread(() -> {DAOProj.insertResult(cArr[2],devName);});
    t1.start();
    try {
        t1.join();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    Intent intent = new Intent(this,MainPage.class);
    intent.putExtra(MainPage.DEV_NAME, devName);
    startActivity(intent);
}
```

Даний метод зберігає результат до бази даних за допомогою методу insertResult класу DAOProj після чого користувач повертається на головну сторінку.

4 ТЕСТУВАННЯ РОБОТИ ДОДАТКУ

Після запуску додатку користувач потрапляє на сторінку авторизації (Рисунок 4.1).

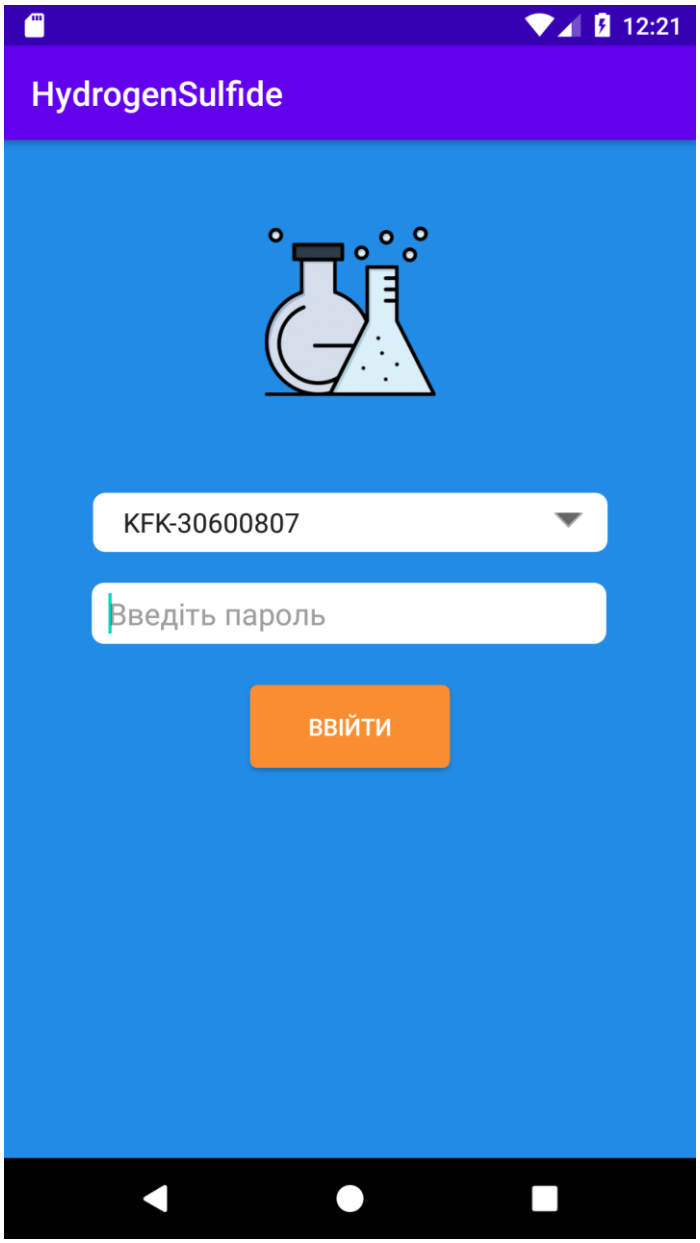

Рисунок 4.1 – Сторінка авторизації

Введемо невірний пароль, після чого отримаємо повідомлення про те, що пароль – невірний (Рисунок 4.2).

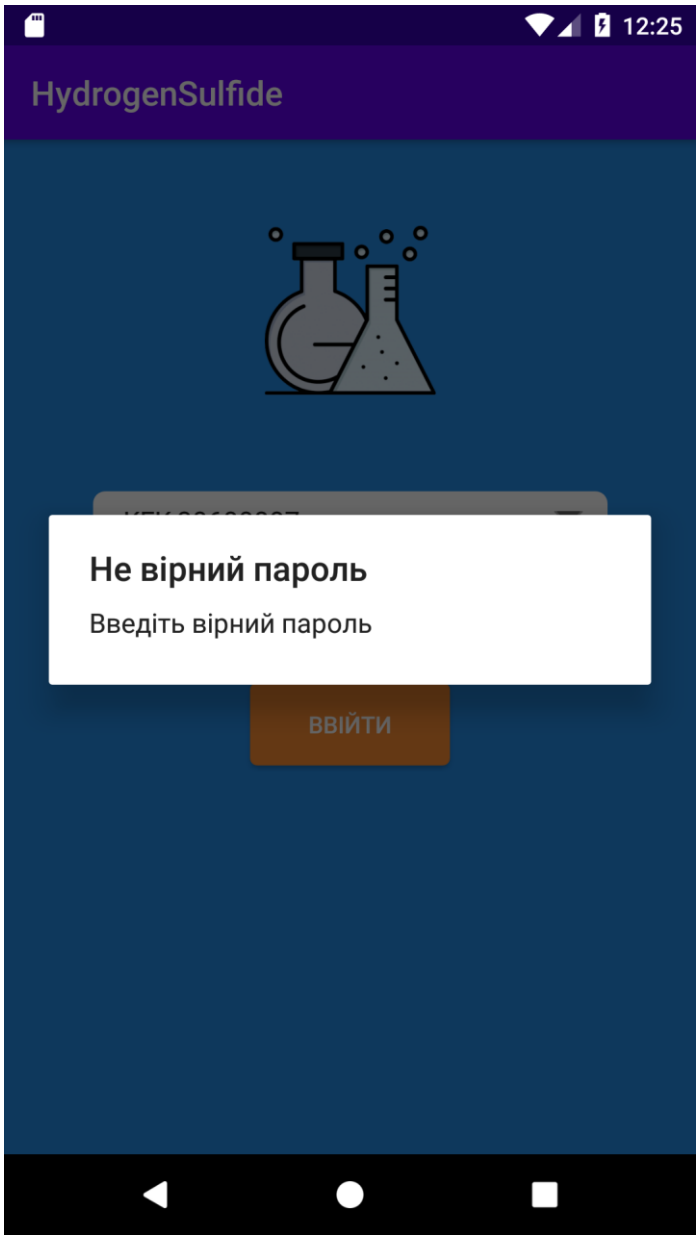

Рисунок 4.2 – Невірний пароль

Введемо вірний пароль, та натиснемо кнопку Ввійти. Після цього потрапляємо на головну сторінку додатку (Рисунок 4.3).

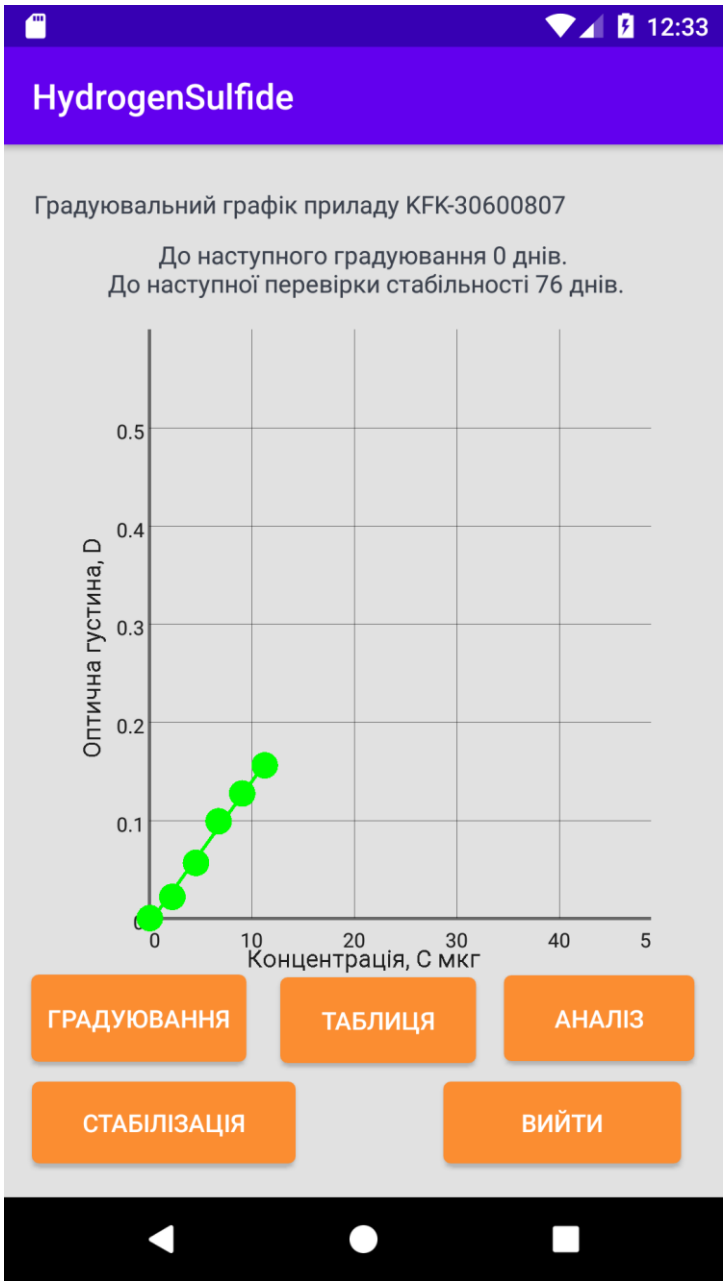

Рисунок 4.3 – Головна сторінка додатку

Для проведення тестування змінив дату останнього градуювання в базі, тому на сторінці додатку показано, що до наступного градуювання залишилось 0 днів. При спробі розпочати процедуру аналізу буде виведено повідомлення про те, що необхідно провести процедуру градуювання (Рисунок 4.4).

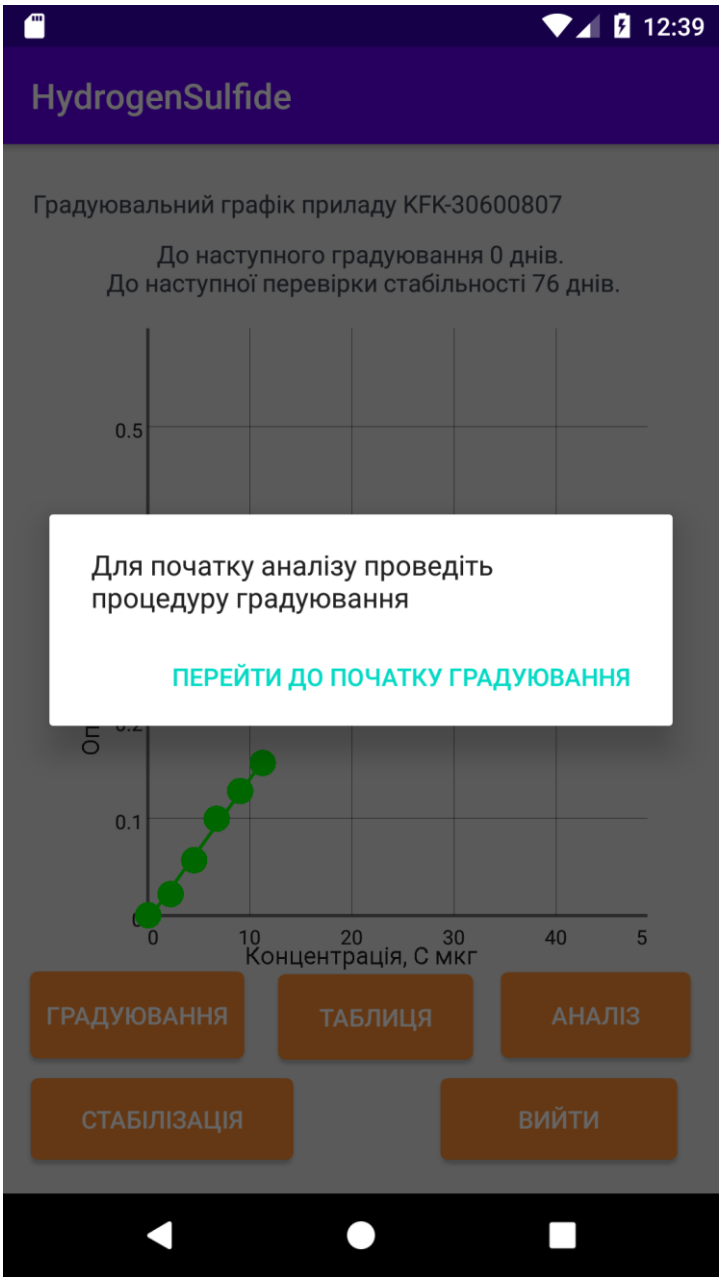

Рисунок 4.4 – Дані градуювання не актуальні

Проведемо процедуру градуювання. Для цього натиснемо перейти до початку градуювання, або натиснемо кнопку Градуювання на головній сторінці додатку, після чого потрапляємо на першу сторінку процедури проведення градуювання. Заповнимо її даними (Рисунок 4.5).

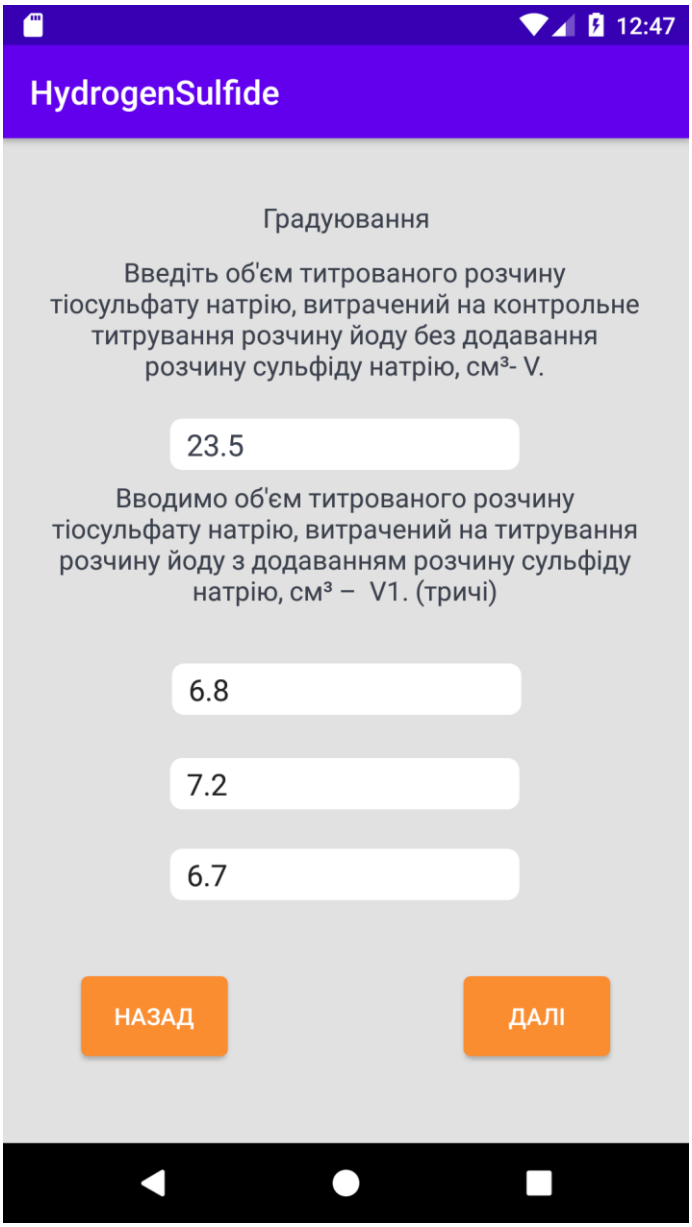

Рисунок 4.5 – Перша сторінка процедури градуювання

Далі потрапляємо на сторінку заповнення градуювальної таблиці (Рисунок 4.6).

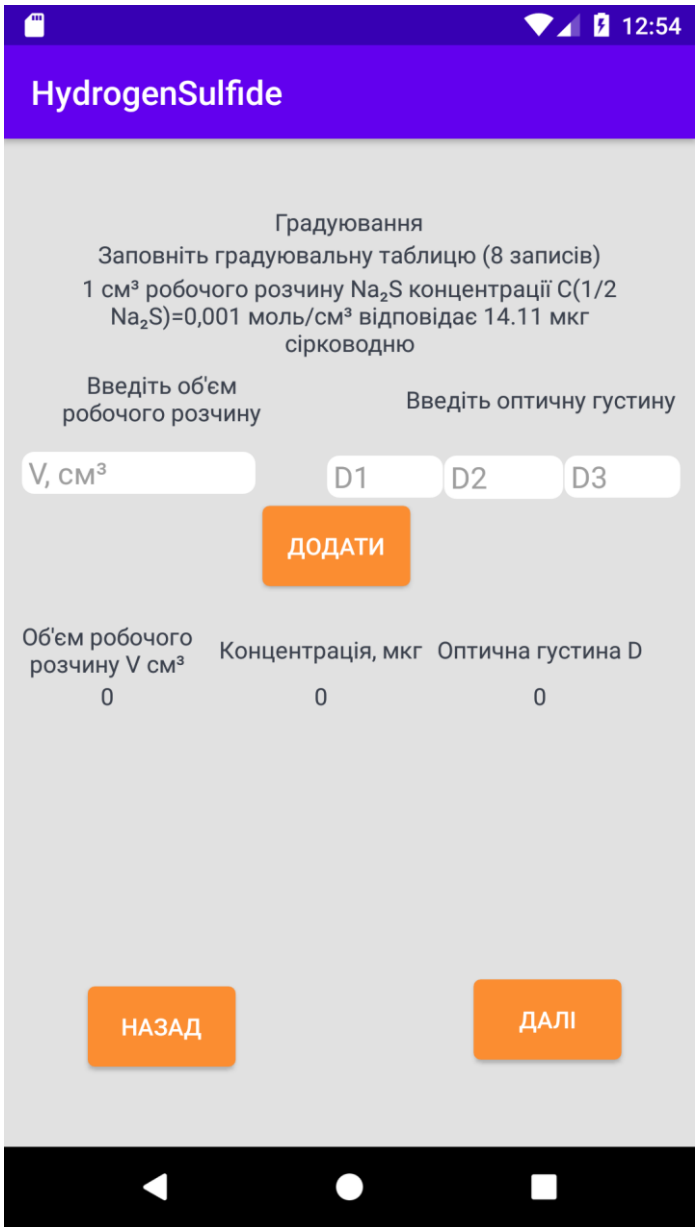

Рисунок 4.6 – сторінка заповнення градуювальної характеристики.

Після спроби натиснути далі виведеться повідомлення про те, що потрібно заповнити щонайменше 8 записів (Рисунок 4.7).

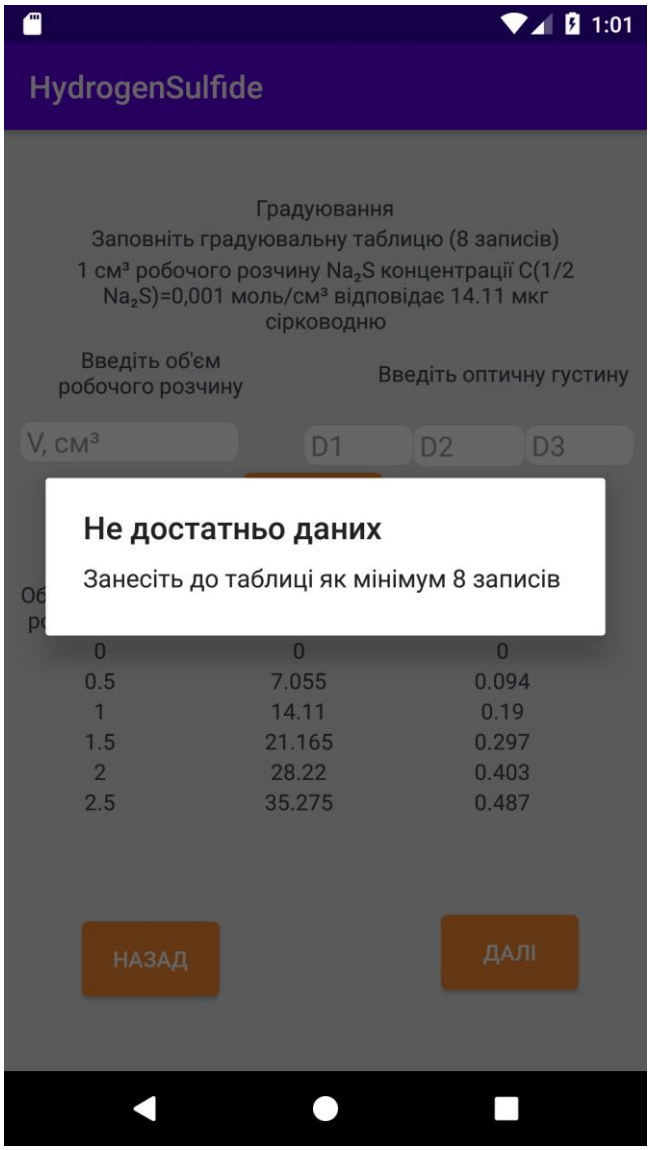

Рисунок 4.7 – Градуювальна таблиця, недостатньо данних

Заповнимо таблицю даними та натиснемо кнопку Далі (Рисунок 4.8).

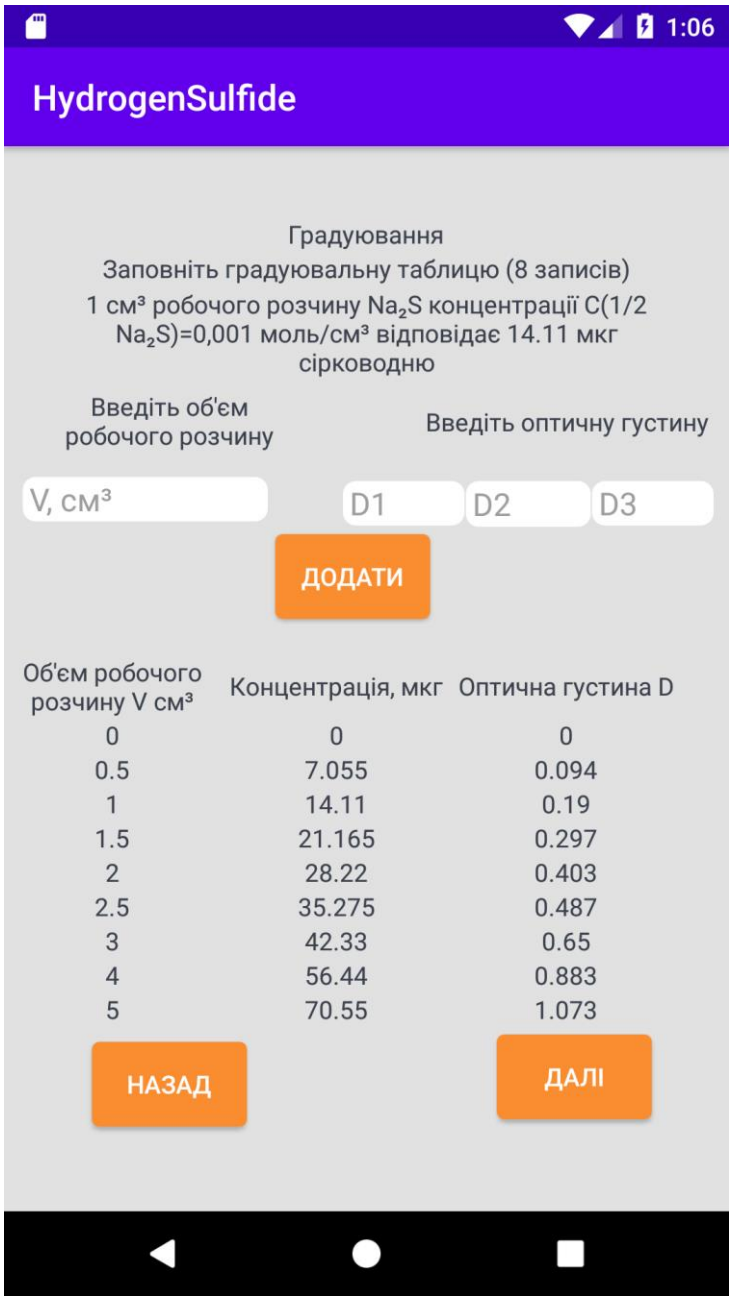

Рисунок 4.8 – Градуювальна таблиця

Потрапляємо на сторінку результатів градуювання, де видно градуювальний графік (Рисунок 4.9).

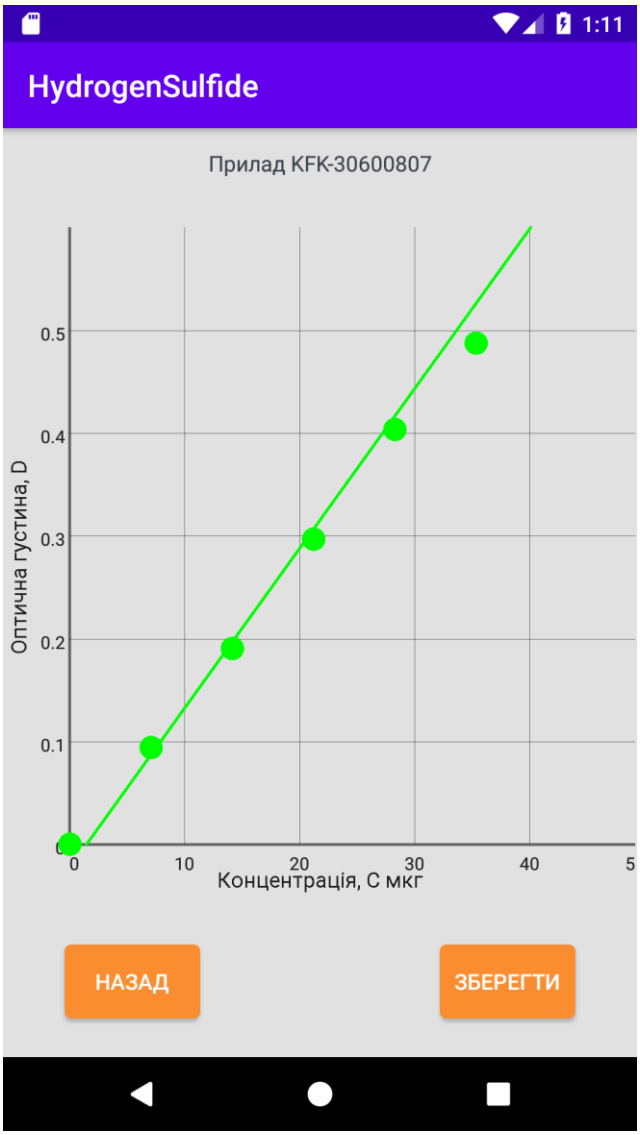

Рисунок 4.9 – Результати градуювання

Натискаємо зберегти та потрапляємо на головну сторінку додатку. Змінилась кількість днів до наступного градуювання, та кількість днів до наступної перевірки стабільності градуювальної характеристики, оскільки поточна перевірка стала не валідною (Рисунок 4.10).

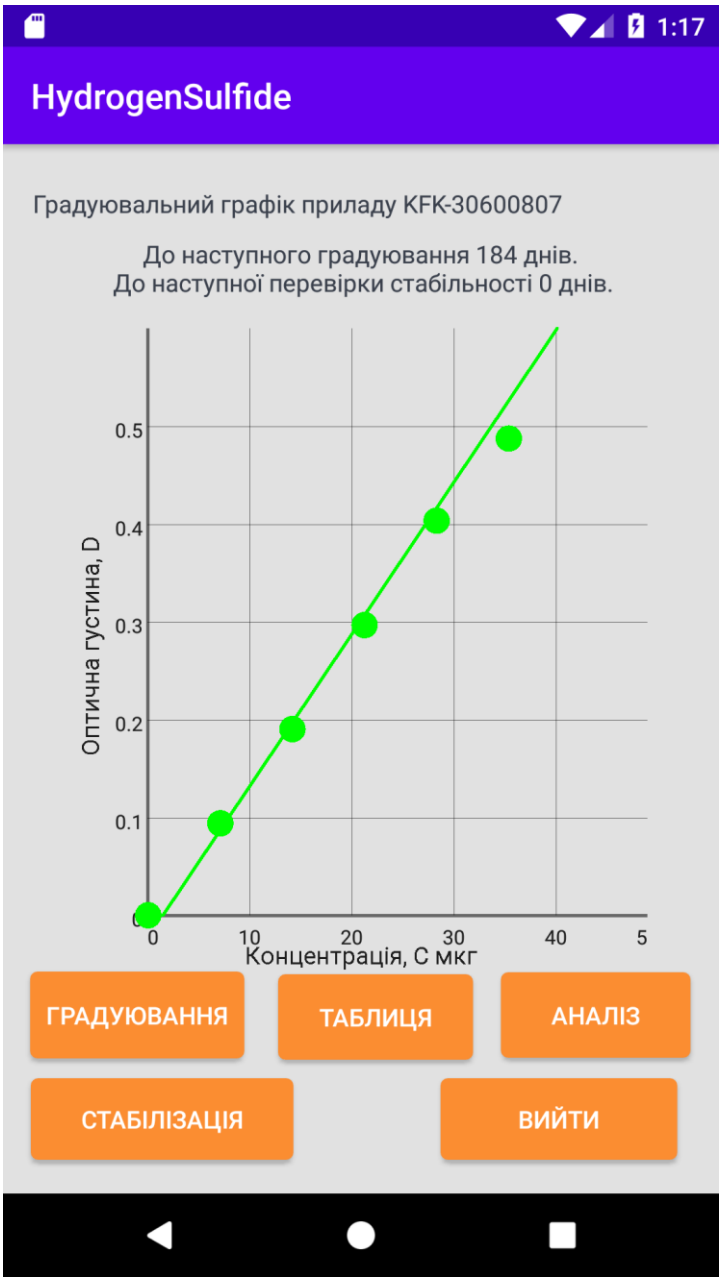

Рисунок 4.10 – Головна сторінка після градуювання

При спробі провести аналіз виведеться повідомлення про те, що необхідно пройти процедуру перевірки стабільності.

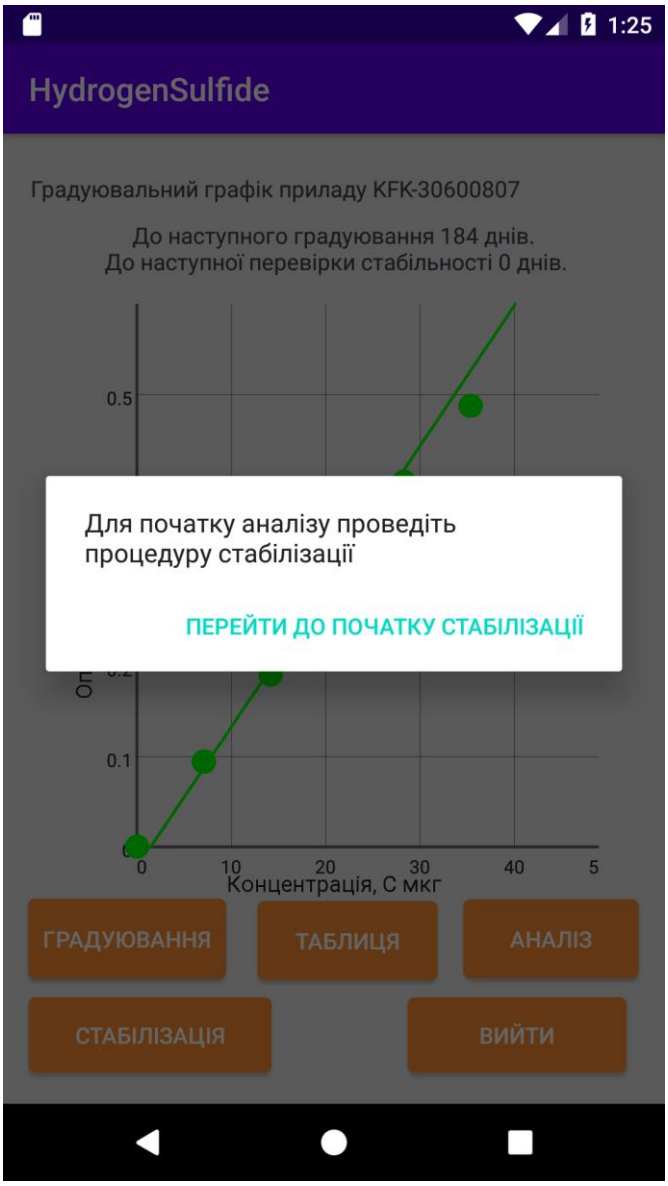

Рисунок 4.11 – Дані стабілізації не актуальні

Перейдемо до процедури перевірки стабільності натиснувши кнопку в повідомленні, або кнопку стабілізація на головній сторінці. Після цього потрапляємо на першу сторінку та заповнюємо її даними.

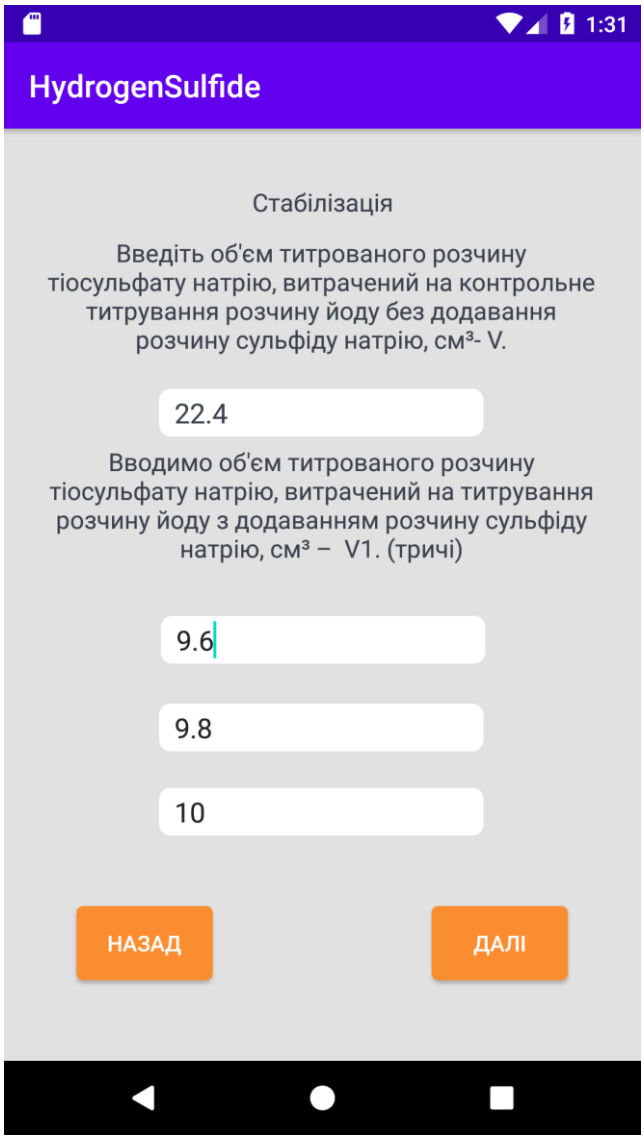

Рисунок 4.12 – Перша сторінка перевірки стабільності

Далі потрапляємо на сторінку заповнення градуювальної таблиці (Рисунок 4.13).

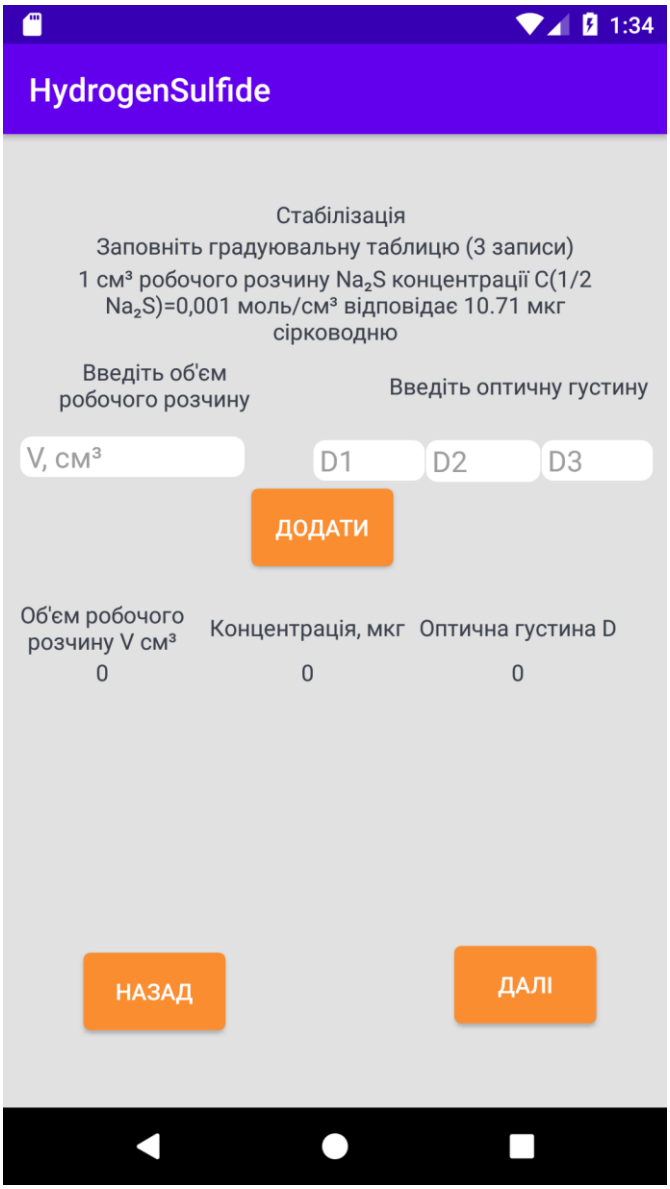

Рисунок 4.13 – Заповнення градуювальної таблиці для перевірки стабільності

При спробі натиснути кнопку далі виведеться повідомлення про те, що даних недостатньо (Рисунок 4.14).

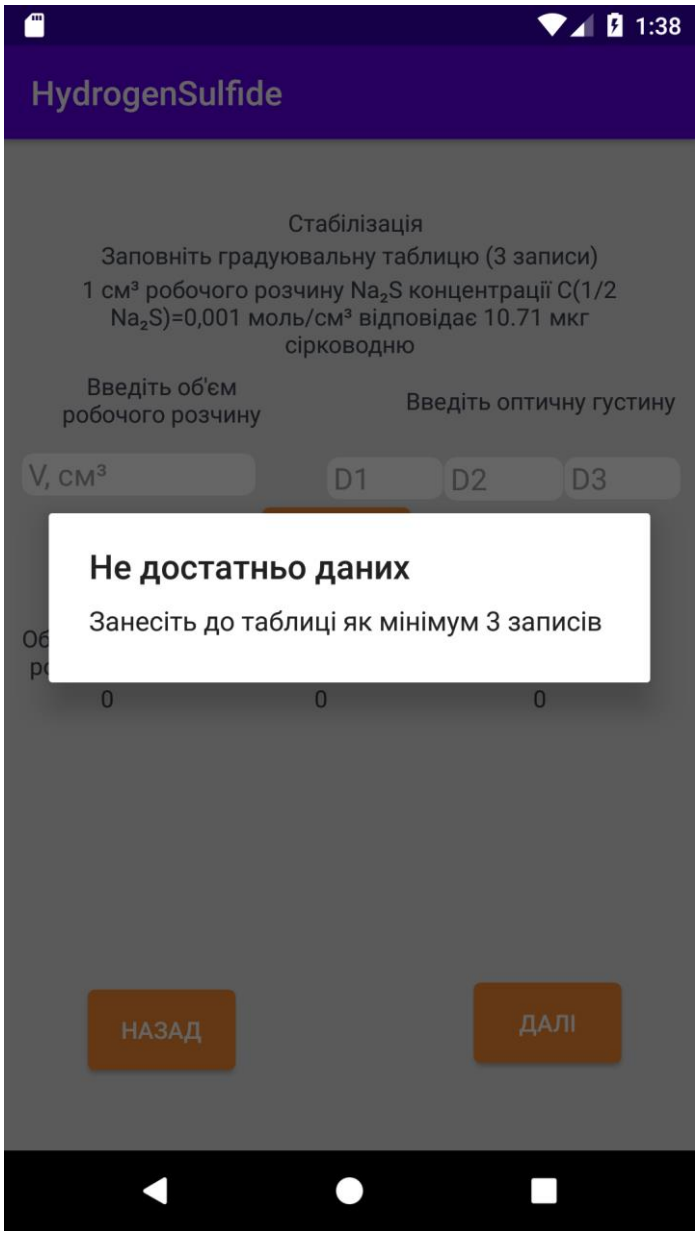

Рисунок 4.14 – Перевірка стабільності, недостатньо даних

Заповнимо таблицю даними (Рисунок 4.15), натиснемо кнопку Далі та отримаємо.

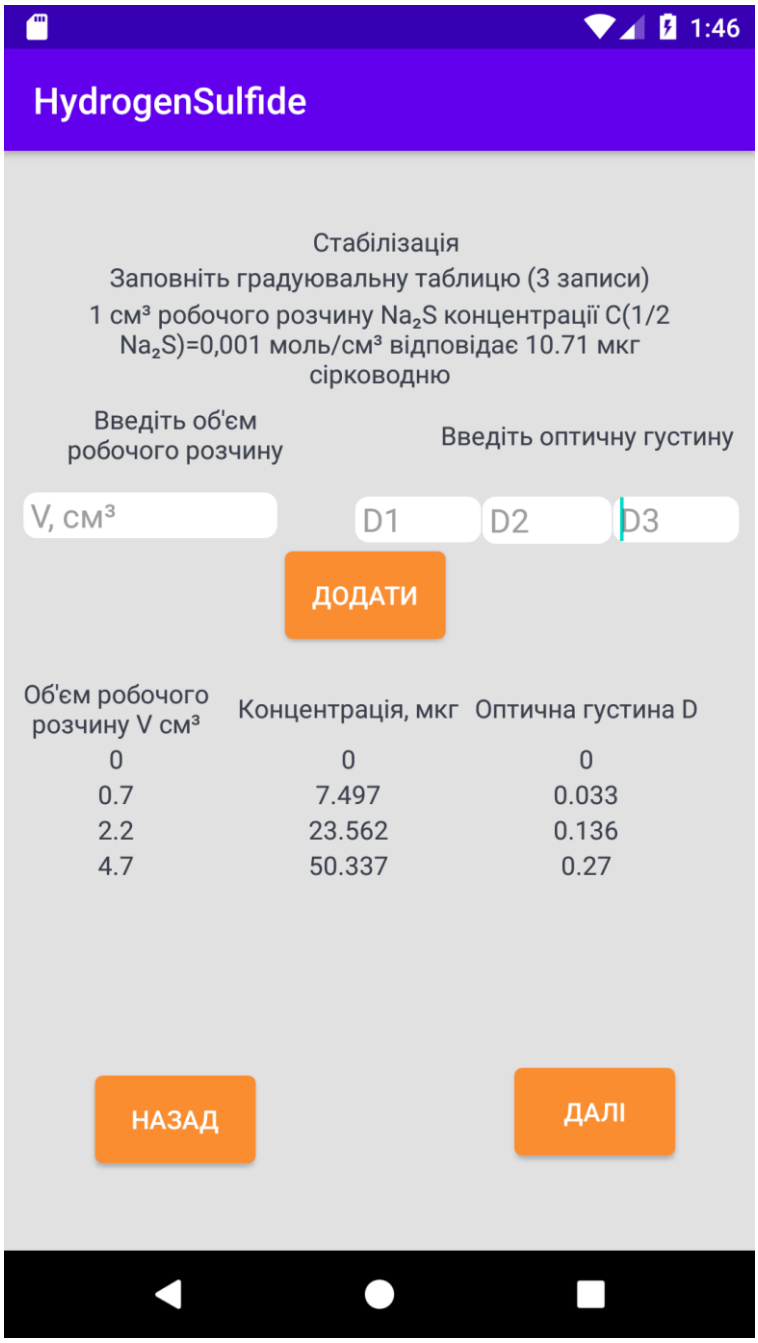

Рисунок 4.15 – Градуювальна таблиця

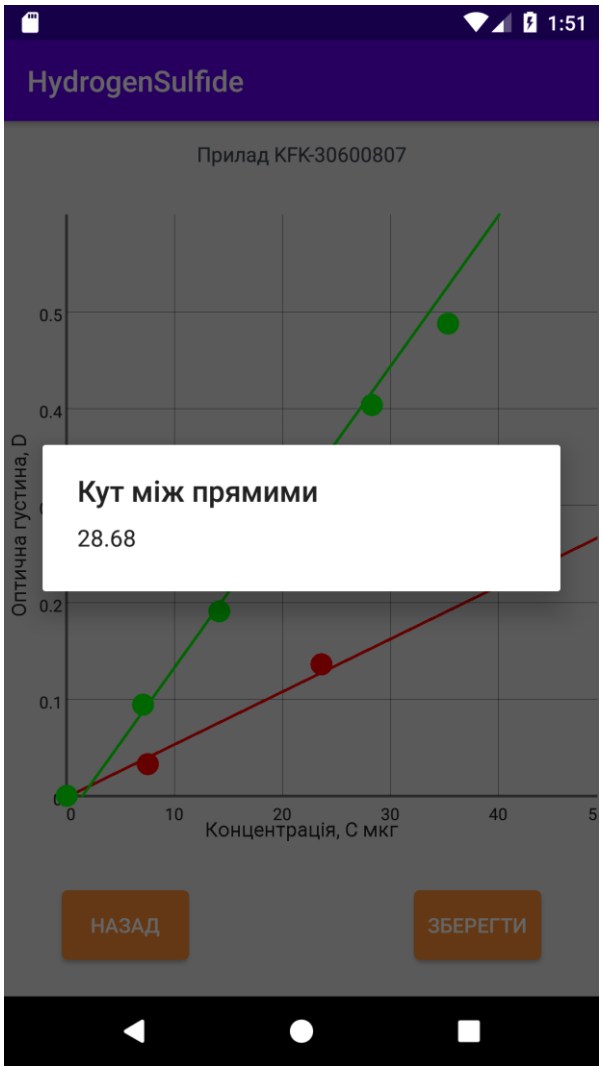

Рисунок 4.16 – Результат перевірки стабільності

Оскільки кут між прямими більше 15 градусів при спробі зберегти результат виведеться повідомлення, в якому на попросять провести заново перевірку стабільності або процедуру градуювання (Рисунок 4.17).

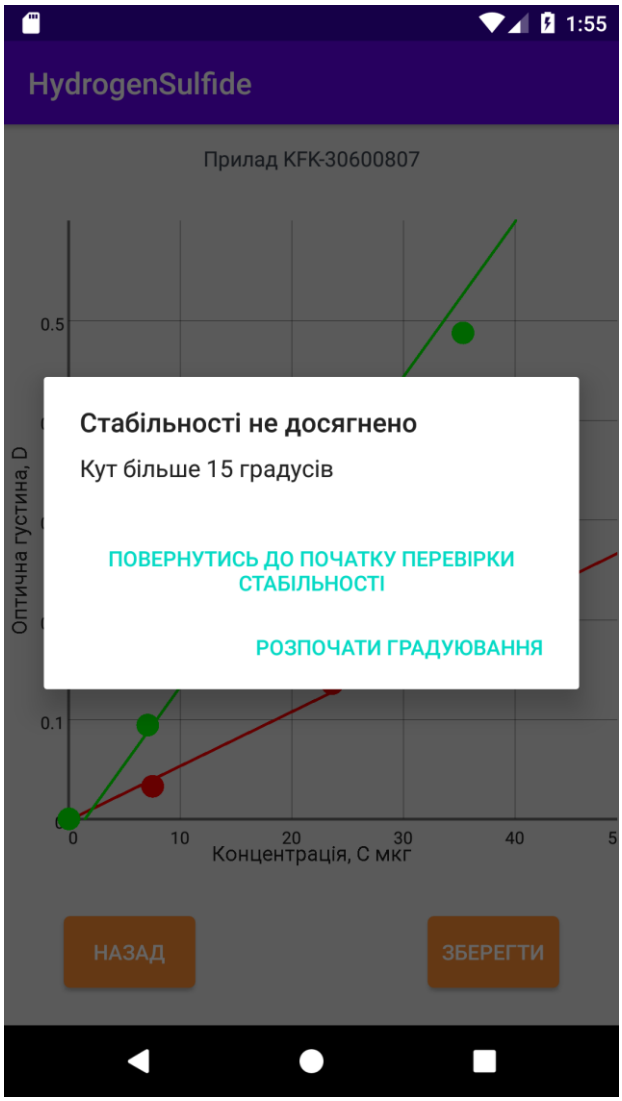

Рисунок 4.17 – Повідомлення про невдалу стабілізацію

Проведемо процедуру з даними при яких кут буде менше 15 та отримаємо результат (Рисунок 4.18).

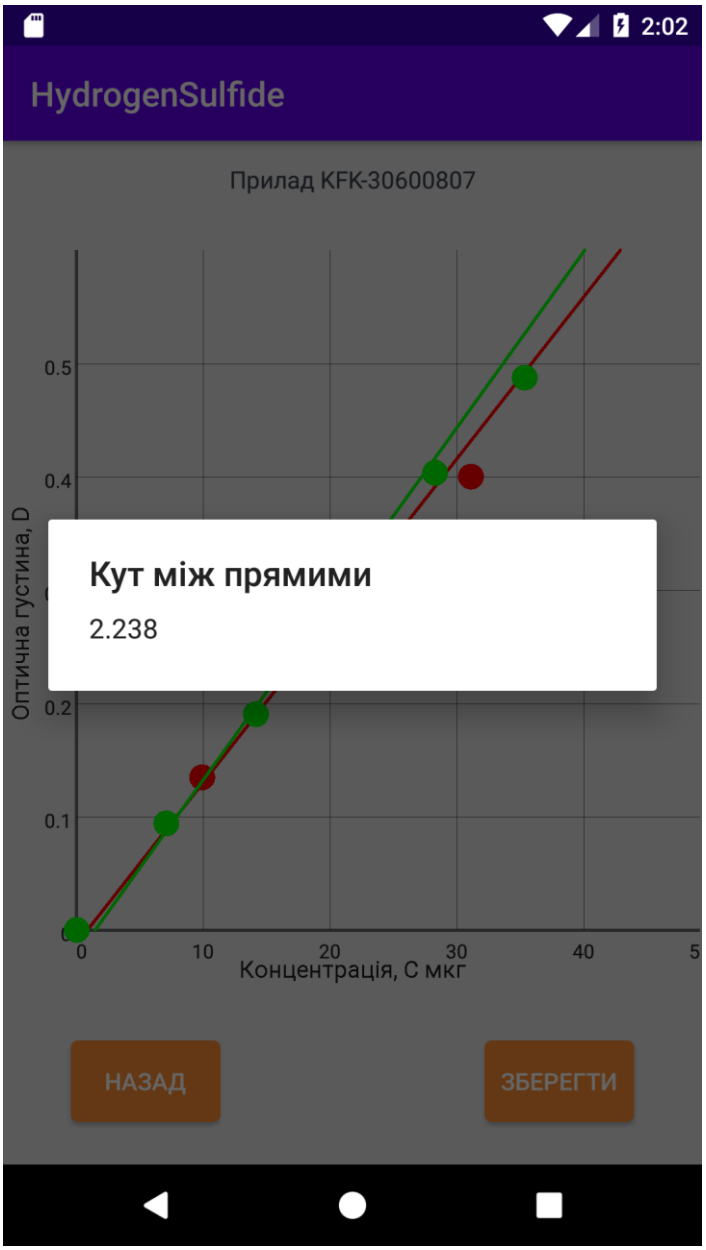

Рисунок 4.18 – Валідний результат перевірки стабільності.

Після чого натискаємо зберегти та переходим на головну сторінку з оновленими даними про перевірку стабільності (Рисунок 4.19).

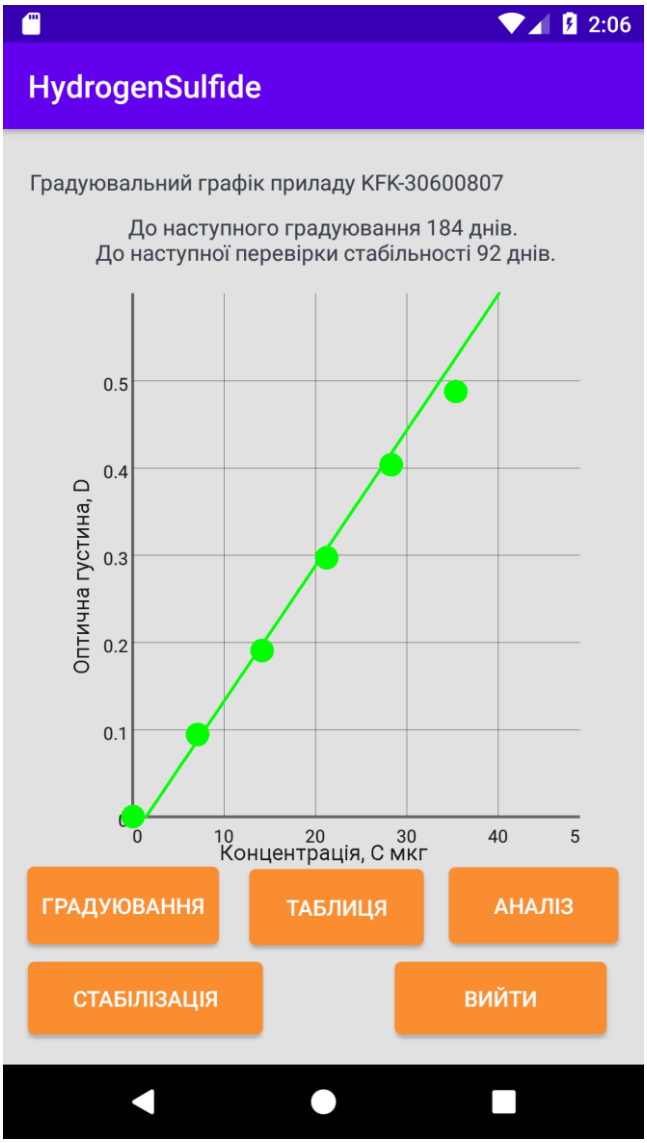

Рисунок 4.19 – Головна сторінка

Для початку процедури аналізу натиснемо кнопку Аналіз, після чого потрапляємо на першу сторінку аналізу. Заповнимо її даними (Рисунок 4.20).

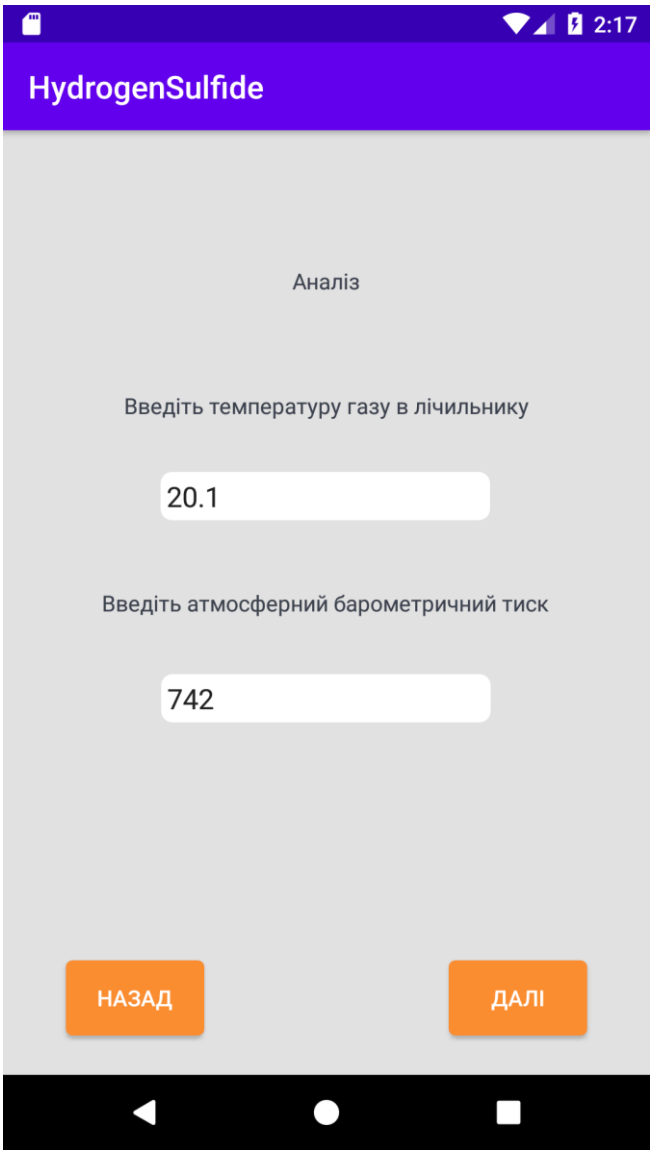

Рисунок 4.20 – Перша сторінка аналізу

Далі необхідно заповнити дані про дві точки. Потрапляємо на другу сторінку процедури аналізу та заповнюємо даними (Рисунок 4.21), після чого знов потрапляємо на цю сторінку (Рисунок 4.22). Якщо оптична густина D2 більше 20% D1 виводиться повідомлення про це (Рисунок 4.23).

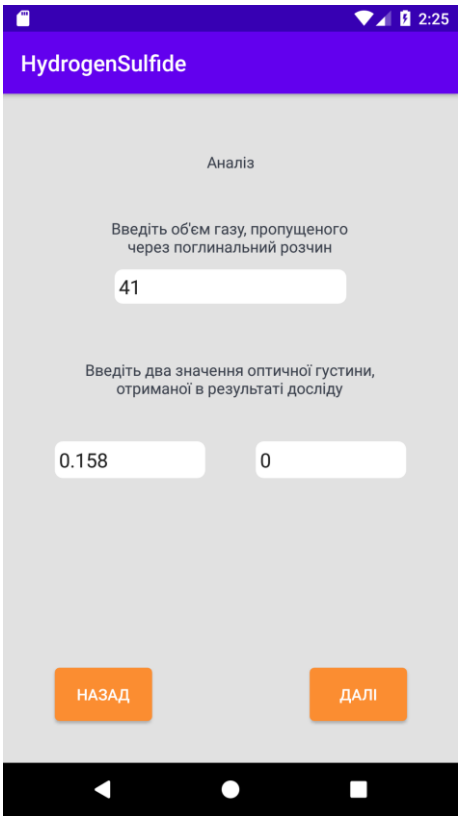

Рисунок 4.21 – Перша точка аналізу

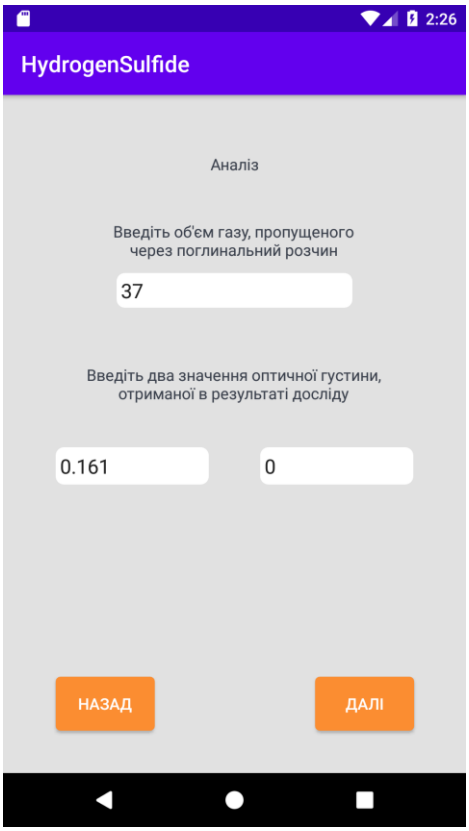

Рисунок 4.22 – Друга точка аналізу

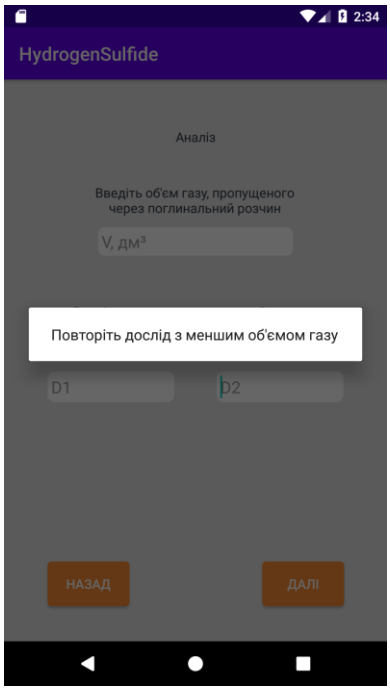

Рисунок 4.23 – Повідомлення про помилку

Далі потрапляємо на сторінку з результатами аналізу, після чого натискаємо Зберегти результат та повертаємось на головну сторінку (Рисунок 4.24).

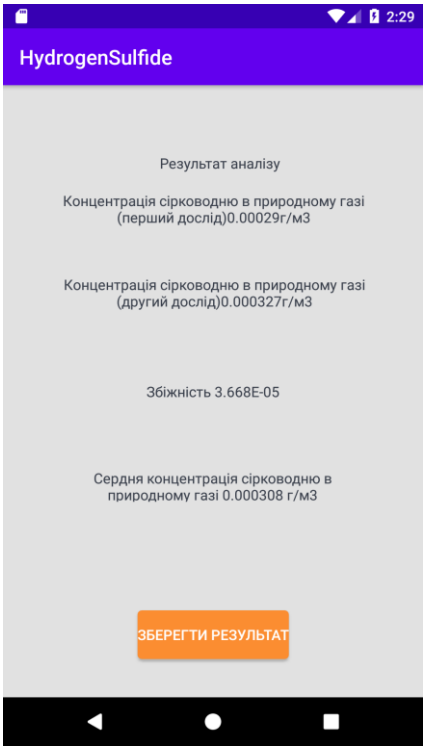

Рисунок 4.24 – Результати аналізу

ВИСНОВКИ

Під час виконання завдання випускної роботи вдалося спроектувати та розробити інформаційну систему для автоматизації обчислювального процесу хіміко-лабораторних досліджень концентрації сірководню в природному газі. Реалізація системи складається з додатку, розробленого для операційної системи Android на мові програмування Java, та реляційної бази даних, яка використовує діалект мови запитів MySQL.

Інформаційна система допомагає проводити процедуру градуювання, перевірки градуювальної характеристики а також проводити аналіз вмісту сірководню в природному газі згідно нормативних документів ГОСТ 22387.2-97, НДТОВ 04-014:2019, зберігати результати цих операцій до бази даних.

В подальшому систему можна розвивати додавши до неї можливість проводити аналіз вмісту меркаптанової сірки в природному газі, оскільки цей аналіз майже повністю аналогічний аналізу вмісту сірководню, а також розробити API для створення звітів за даними аналізами.

СПИСОК ЛІТЕРАТУРИ

19. Лисікова К.О. Апроксимація функцій методом найменших квадратів / С.О. Цибульник, К.О. Лисікова // Вісник інженерної академії України. – Київ, 2017. – № 1. – С. 106-110. (фахове видання)

20. Брановицька С.В. «Обчислювальна математика та програмування» / С.В. Брановицька, Р.Б. Медведєв, Ю.А. Фіалков – К.: Політехніка, 2004. — 248 с.

21. Дюбуа, Поль MySQL; М.: Вильямс; Издание 2-е - Москва, 2010. - 185 с.

22. Griffiths, Dawn & Griffiths, David (2017) Head First Android Development: A Brain-Friendly Guide. Sebastopol, California: O'Reilly Media, Inc.

23. Шилдт, Герберт Java. Полное руководство, 10-е изд. — СПб.: ООО “Альфакнига”; 2018. - 1488 с.: ил. — Парал. тит. англ. ISBN 978-5-6040043-6-4

24. Beerbohm, Max (2020) Xamarin: Xamarin for beginners, Building Your First Mobile App with C# .NET and Xamarin - 3nd Edition. Independently published [in English].

25. Cinar, Onur (2013) Pro Android C++ with the NDK. New York City, NY: aPress [in English].

26. Griffiths, Dawn & Griffiths, David (2017) Head First Android Development: A Brain-Friendly Guide. Sebastopol, CA: O'Reilly Media, Inc [in English].

27. Leiva, Antonio (2016) Kotlin for Android Developers: Learn Kotlin the easy way while developing an Android App. Scotts Valley, CA: CreateSpace Independent Publishing Platform [in English].

28. Coronel, Carlos & Morris, Steven (2018) Database Systems Design, Implementation, & Management, Loose-Leaf Version. Boston, MA: CENGAGE Learning [in English].

29. Bell, Charles (2018) Introducing InnoDB Cluster: Learning the MySQL High Availability Stack. New York City, NY: aPress [in English].

30. Kreibich, Jay A. (2010) Using SQLite. Sebastopol, CA: O'Reilly Media, Inc [in English].

31. Табунщик Г. В. Проектування та моделювання програмного забезпечення сучасних інформаційних систем / Г. В. Табунщик, Т. І. Каплієнко, О.А. Петрова − Запоріжжя : Дике Поле, 2016. – 250 с.

32. Ларман, Крэг Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итеративную разработку / Крэг Ларман. - Москва: Гостехиздат, 2017. - 736 с.

33. Анісімов А.В. Інформаційні системи та бази даних: Навчальний посібник для студентів факультету комп’ютерних наук та кібернетики. / А.В. Анісімов, П.П. Кулябко − Київ. – 2017. – 110 с.

34. Android Studio Development [Електронний ресурс] // Android Platform & Tools Survey. – 2017. – Режим доступу до ресурсу: https://developer.android.com/.

35. Варфел Т. Прототипирование. Практическое руководство. - М.: Ман, Иванов и Фербер, 2013. - 240 с.

36. Гринберг С., Карпендейл Ш., Маркардт Н., Бакстон Б. UX-дизайн. Идея – эскиз – воплощение. – СПб.: Издательство "Питер", 2014. - 272 с.

37. Мюллер, Р.Дж. Базы данных и UML. Проектирование / Р.Дж. Мюллер. - М.: ЛОРИ, 2017. - 420 с.

ДОДАТОК А

(обов’язковий)

Сценарій створення таблиць бази даних

```
CREATE TABLE `device` (
  `devName` varchar(40) NOT NULL,
  `password` varchar(40) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 ROW_FORMAT=COMPACT;

INSERT INTO `device` (`devName`, `password`) VALUES
('KFK-30600807', '1111'),
('V-1000V1218073002', '2222'),
('(M)KFK-30600807', 'M1111'),
('(M)V-1000V1218073002', 'M2222');

CREATE TABLE `grad` (
  `grad_id` int NOT NULL,
  `devName` varchar(40) NOT NULL,
  `gradDate` date NOT NULL,
  `V` decimal(5,2) NOT NULL,
  `V1` decimal(5,2) NOT NULL,
  `V2` decimal(5,2) NOT NULL,
  `V3` decimal(5,2) NOT NULL,
  `C` decimal(7,4) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `grad` (`grad_id`, `devName`, `gradDate`, `V`, `V1`, `V2`, `V3`, `C`) VALUES
```

```
(1, 'KFK-30600807', '2020-04-24', '24.30', '10.80', '11.60', '11.00', '11.1920'),

(2, 'V-1000V1218073002', '2020-06-24', '24.20', '8.40', '8.00', '8.00', '13.6570'),

(3, '(M)KFK-30600807', '2021-05-03', '24.40', '24.00', '23.60', '23.30', '49.0670'),

(4, '(M)V-1000V1218073002', '2021-05-03', '24.40', '24.00', '23.60', '23.30', '49.0670');


CREATE TABLE `grad_table` (
  `grad_t_id` int NOT NULL,
  `grad_id` int NOT NULL,
  `V` decimal(6,3) NOT NULL,
  `C` decimal(6,3) NOT NULL,
  `D1` decimal(5,4) NOT NULL,
  `D2` decimal(5,4) NOT NULL,
  `D3` decimal(5,4) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 ROW_FORMAT=COMPACT;


INSERT INTO `grad_table` (`grad_t_id`, `grad_id`, `V`, `C`, `D1`, `D2`, `D3`) VALUES
(1, 1, '0.000', '0.000', '0.0000', '0.0000', '0.0000'),
(2, 1, '0.200', '2.200', '0.0220', '0.0210', '0.0220'),
(3, 1, '0.400', '4.500', '0.0570', '0.0560', '0.0560'),
(4, 1, '0.600', '6.700', '0.0980', '0.0990', '0.0990'),
(5, 1, '0.800', '9.000', '0.1270', '0.1270', '0.1270'),
(6, 1, '1.000', '11.200', '0.1560', '0.1550', '0.1560'),
(7, 2, '0.000', '0.000', '0.0000', '0.0000', '0.0000'),
(8, 2, '0.500', '6.800', '0.0700', '0.0740', '0.0720'),
(9, 2, '1.000', '13.700', '0.1400', '0.1490', '0.1490'),
```

```
(10, 2, '1.500', '20.500', '0.2740', '0.2730', '0.2700'),
(11, 2, '2.000', '27.300', '0.3310', '0.3300', '0.3340'),
(12, 2, '2.500', '34.100', '0.4230', '0.4280', '0.4140'),
(37, 3, '0.000', '0.000', '0.0000', '0.0000', '0.0000'),
(38, 3, '0.500', '24.500', '0.0700', '0.0710', '0.0710'),
(39, 3, '1.000', '49.100', '0.1010', '0.1000', '0.1020'),
(40, 3, '1.500', '73.600', '0.3980', '0.3970', '0.3970'),
(41, 3, '2.000', '98.100', '0.4680', '0.4700', '0.4700'),
(42, 3, '2.500', '122.700', '0.5630', '0.5630', '0.5620'),
(43, 3, '3.000', '147.200', '0.6810', '0.6820', '0.6830'),
(44, 3, '4.000', '196.300', '0.8880', '0.8910', '0.8900'),
(45, 3, '5.000', '245.300', '1.0990', '1.0980', '1.0980'),
(46, 4, '0.000', '0.000', '0.0000', '0.0000', '0.0000'),
(47, 4, '0.500', '24.500', '0.0700', '0.0710', '0.0710'),
(48, 4, '1.000', '49.100', '0.1010', '0.1000', '0.1020'),
(49, 4, '1.500', '73.600', '0.3980', '0.3970', '0.3970'),
(50, 4, '2.000', '98.100', '0.4680', '0.4700', '0.4700'),
(51, 4, '2.500', '122.700', '0.5630', '0.5630', '0.5620'),
(52, 4, '3.000', '147.200', '0.6810', '0.6820', '0.6830'),
(53, 4, '4.000', '196.300', '0.8880', '0.8910', '0.8900'),
(54, 4, '5.000', '245.300', '1.0990', '1.0980', '1.0980');


CREATE TABLE `results` (
  `res_id` int NOT NULL,
  `resDate` date NOT NULL,
  `res` decimal(10,6) NOT NULL,
  `devName` varchar(40) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
INSERT INTO `results` (`res_id`, `resDate`, `res`, `devName`) VALUES
(1, '2020-08-04', '0.000068', 'KFK-30600807'),
(2, '2021-05-03', '0.000267', 'KFK-30600807');

CREATE TABLE `stabilization` (
  `stab_id` int NOT NULL,
  `devName` varchar(40) CHARACTER SET latin1 COLLATE latin1_swedish_ci NOT NULL,
  `stab_date` date NOT NULL,
  `valid` tinyint(1) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `stabilization` (`stab_id`, `devName`, `stab_date`, `valid`) VALUES
(1, 'V-1000V1218073002', '2021-04-16', 1),
(2, 'KFK-30600807', '2021-05-03', 1),
(3, '(M)KFK-30600807', '2021-04-16', 0),
(4, '(M)V-1000V1218073002', '2021-04-16', 0);

ALTER TABLE `device`
  ADD PRIMARY KEY (`devName`),
  ADD UNIQUE KEY `uniquePass` (`password`);

ALTER TABLE `grad`
  ADD PRIMARY KEY (`grad_id`),
  ADD KEY `devName` (`devName`);

ALTER TABLE `grad_table`
  ADD PRIMARY KEY (`grad_t_id`),
```

```
  ADD KEY `grad_id_fk` (`grad_id`);


ALTER TABLE `results`
  ADD PRIMARY KEY (`res_id`),
  ADD KEY `devNameIndex` (`devName`);


ALTER TABLE `stabilization`
  ADD PRIMARY KEY (`stab_id`),
  ADD KEY `devName` (`devName`);


ALTER TABLE `grad`
  MODIFY `grad_id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=8;


ALTER TABLE `grad_table`
  MODIFY `grad_t_id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=55;
ALTER TABLE `results`
  MODIFY `res_id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;
ALTER TABLE `stabilization`
  MODIFY `stab_id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;
ALTER TABLE `grad`
  ADD CONSTRAINT `grad_ibfk_1` FOREIGN KEY (`devName`) REFERENCES `device`
(`devName`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `stab_ibfk_1` FOREIGN KEY (`devName`) REFERENCES `device`
(`devName`) ON DELETE CASCADE ON UPDATE CASCADE;
ALTER TABLE `grad_table`
  ADD CONSTRAINT `grad_table_ibfk_1` FOREIGN KEY (`grad_id`) REFERENCES
`grad` (`grad_id`) ON DELETE CASCADE ON UPDATE CASCADE;
ALTER TABLE `results`
  ADD CONSTRAINT `results_ibfk_1` FOREIGN KEY (`devName`) REFERENCES
`device` (`devName`) ON UPDATE CASCADE;
```

```
ALTER TABLE `stabilization`

  ADD CONSTRAINT `stabilization_ibfk_1` FOREIGN KEY (`devName`) REFERENCES
`device` (`devName`) ON DELETE RESTRICT ON UPDATE RESTRICT;

COMMIT;
```