Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційні систем та технологій

Лабораторна робота №3
із дисципліни «Розробка мобільних застосувань під Android»
Тема: «Дослідження способів збереження даних»

Виконав:
Студент групи ІА-24
Боднар А.Д.

Перевірив:
Орленко С. П.

Київ-2025

Мета: дослідити способи збереження даних (база даних, файлова система, тощо) та отримати практичні навички щодо використання сховищ даних.

Хід роботи

17	Боднар Антон Дмитрович	ІА-24

Варіант - 17

17.	Вікно містить поле введення питання/завдання, дві групи опцій (складність – три рівні, тип – теоретичне або практичне), тобто радіо-батони) та кнопку «ОК». Вивести інформацію щодо вибору при натисканні на кнопку «ОК» у деяке текстове поле.

БД я обрав Room

```
@Database(entities = [Question::class], version = 1)
abstract class AppDatabase : RoomDatabase() {
    abstract fun questionDao(): QuestionDao

    companion object {
        @Volatile
        private var INSTANCE: AppDatabase? = null

        fun getDatabase(context: Context): AppDatabase {
            return INSTANCE ?: synchronized(this) {
                val instance = Room.databaseBuilder(
                    context.applicationContext,
                    AppDatabase::class.java,
                    "questions_db"
                ).build()
                INSTANCE = instance
                instance
            }
```

```
        }
    }
}
```

Клас AppDatabase, котрий визначає базу даних Room для зберігання сутності **Question** з версією 1 та містить метод для отримання єдиного екземпляра бази даних через синглтон.

```
@Entity(tableName = "questions")
data class Question(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val question: String,
    val complexity: String,
    val type: String
)
```

Клас Question є сутністю бази даних для таблиці "questions", яка містить поля: id (первинний ключ, що автоматично генерується), question (текст питання), complexity (складність) і type (тип питання).

```
if (question.isBlank() || difficulty == "null" || questionType
== "null") {
    // Якщо одне з полів порожнє
    Toast.makeText(context, "Заповніть всі поля!",
Toast.LENGTH_SHORT).show()
} else {
    // Формуємо результат
    val resultText = "Питання: $question\nСкладність:
$difficulty\nТип питання: $questionType"
    (activity as
MainActivity).resultFragment.updateResult(resultText)
```

```
    // Додаємо в БД
    val db = AppDatabase.getDatabase(requireContext())
    val dao = db.questionDao()

    viewLifecycleOwner.lifecycleScope.launch(Dispatchers.IO) {
        dao.insert(Question(question = question, complexity =
difficulty, type = questionType))

        withContext(Dispatchers.Main) {
            Toast.makeText(context, "Питання збережено!",
Toast.LENGTH_SHORT).show()
        }
    }
}
```

Тут в класі InputFragment я змінюю перевірку(випадково побачив що .isNull не працю на радіокнопки а що працює тільки == “null”); та також додаю додавання в таблицю бд записа.

```
@Dao
interface QuestionDao {
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun insert(question: Question)

    @Query("SELECT * FROM questions")
    fun getAll(): List<Question>
```

```
    @Query("DELETE FROM questions")
    fun clearAll()
}
```

Цей інтерфейс QuestionDao є DAO для таблиці "questions", який містить методи для вставки питання, отримання всіх питань та очищення таблиці.

```
private lateinit var viewDatabaseButton: Button

viewDatabaseButton =
binding.findViewById(R.id.viewDatabaseButton)
```

```
viewDatabaseButton.setOnClickListener {
    try {
        val intent = Intent(activity,
ViewDatabaseActivity::class.java)
        startActivity(intent)
    } catch (e: Exception) {
        Toast.makeText(context, "Помилка запуску: ${e.message}",
Toast.LENGTH_LONG).show()
    }
}
```

В клас ResultFragment додаю підключення до кнопки та функціонал у вигляді переходу на іншу діяльність.

```
class ViewDatabaseActivity : AppCompatActivity() {

    private lateinit var textViewDatabase: TextView
    private lateinit var buttonBack: Button
    private lateinit var buttonClear: Button

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_view_database)

        Log.d("DEBUG", "ViewDatabaseActivity створено")

        textViewDatabase = findViewById(R.id.textViewDatabase)
        buttonBack = findViewById(R.id.buttonBack)
        buttonClear = findViewById(R.id.buttonClear)

        buttonBack.setOnClickListener {
            finish()
        }

        buttonClear.setOnClickListener {
            clearDatabase()
        }

        loadDatabase()
    }
```

```
    private fun loadDatabase() {
        try {
            val db = AppDatabase.getDatabase(this)
            val dao = db.questionDao()

            lifecycleScope.launch(Dispatchers.IO) {
                val questions = dao.getAll()

                withContext(Dispatchers.Main) {
                    textViewDatabase.text = if
(questions.isEmpty()) {
                        "База даних порожня"
                    } else {
                        questions.joinToString("\n") {
                            "ID: ${it.id}\nПитання:
${it.question}\nСкладність: ${it.complexity}\nТип:
${it.type}\n"
                        }
                    }
                }
            }
        } catch (e: Exception) {
            Log.e("ERROR", "Помилка у loadDatabase:
${e.message}")
            Toast.makeText(this, "Помилка завантаження БД",
Toast.LENGTH_LONG).show()
        }
```

```
    }



    private fun clearDatabase() {
        val db = AppDatabase.getDatabase(this)
        val dao = db.questionDao()

        lifecycleScope.launch(Dispatchers.IO) {
            dao.clearAll()

            withContext(Dispatchers.Main) {
                Toast.makeText(this@ViewDatabaseActivity, "База даних очищена", Toast.LENGTH_SHORT).show()
                loadDatabase() // Оновлюємо екран після очищення
            }
        }
    }
}
```

Клас **ViewDatabaseActivity** відповідає за перегляд та очищення бази даних питань. У методі **onCreate()** ініціалізуються елементи інтерфейсу, такі як **TextView** для відображення даних з бази та кнопки для повернення до попереднього екрану й очищення бази даних. Метод **loadDatabase()** завантажує всі питання з бази даних у фоновому потоці і відображає їх на екрані. Якщо база порожня, на екрані з’являється повідомлення "База даних порожня". Метод **clearDatabase()** очищує всі дані в базі, після чого оновлюється екран із порожнім списком питань. Обидва методи використовують корути для виконання операцій з базою даних у фоновому потоці.

```
<Button
   android:id="@+id/viewDatabaseButton"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:text="Переглянути" />
```

Додаю в result_fragment.xml кнопку для перенаправлення на іншу діяльність

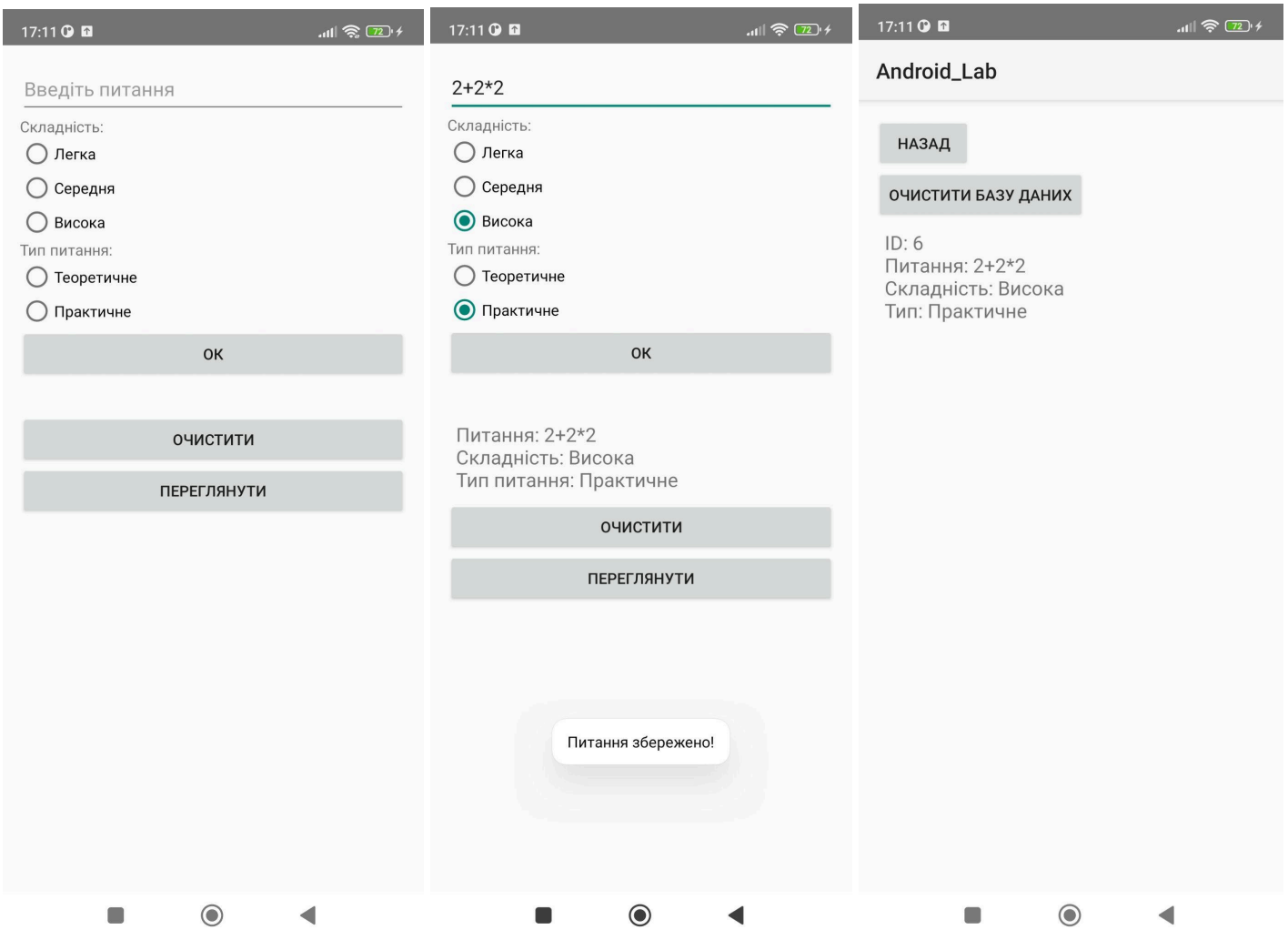
```
<Button
   android:id="@+id/buttonBack"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Назад" />


<Button
   android:id="@+id/buttonClear"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Очистити базу даних" />


<TextView
   android:id="@+id/textViewDatabase"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
```

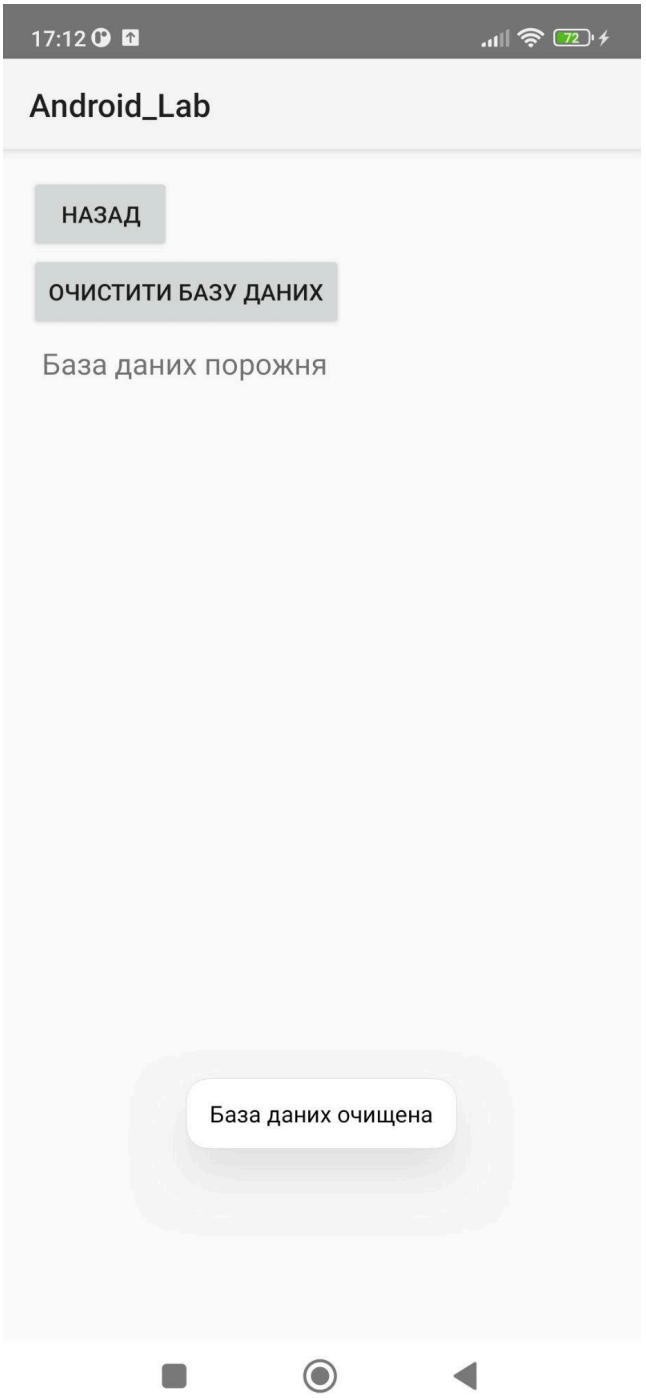
```
    android:text="Завантаження..."
    android:textSize="18sp"
    android:padding="8dp" />
```

Код activity_view_database.xml в якому дві кнопки: назад, котра повертає на першу діяльність та кнопка очищення таблиці; та ще в кінці лейбл в якому або написано що бд порожня або показуються записи бд.

Результати:

При тестуванні використовувався власний телефон Xiaomi Redmi Note 11





1. Оновлений вигляд першої діяльності
2. Демонстрація додавання запису в таблицю
3. Демонстрація вигляду другої діяльності коли є записи в таблиці
4. Демонстрація вигляду другої діяльності при очищенні таблиці та коли немає записів в таблиці

Більш повна демонстрація роботи застосунку буде у відео котре буде разом з звітом.