ANKARA BILIM UNIVERSITY

FINAL ASSIGNMENT – CENG443

Student Name: Taha Uğur Toyran

Student ID: 220201055

Course Instructor: Prof. Dr. Önder Eyecioğlu

Course Code: CENG443

Date:12.01.2025

1. EXECUTIVE SUMMARY

This project presents the design and implementation of a **Conference Registration & Verification System** developed for the Android platform using **modern Android development principles**. The main objective of the application is to provide a **robust, crash-resistant, and user-friendly solution** for managing participant registration and verification processes during a conference.

The system integrates **hardware interaction (camera usage)** and **local data persistence (Room database)** while strictly following a **clean architectural separation** based on the MVVM pattern. Special attention was given to **data validation, error handling, and lifecycle awareness**, ensuring that the application remains stable under all user interactions, including invalid inputs and configuration changes.

2. ARCHITECTURAL DESIGN – MVVM PATTERN

To fully satisfy the **Architecture & Code Quality (25 points)** requirement, the application was designed using the **Model–View–ViewModel (MVVM)** architectural pattern. This approach improves maintainability, testability, and separation of concerns.

Model Layer

The Model layer defines the core data structure of the application.

- The `Participant` class represents a single conference participant and is annotated as a **Room Entity**.
- It contains essential fields such as `userId`, `fullName`, `registrationType`, and `photoPath`.
- Data consistency is enforced at the database level using a `@PrimaryKey` annotation on `userId`.

ViewModel Layer

The ViewModel layer acts as an intermediary between the UI and the data source.

- The `ParticipantViewModel` survives configuration changes such as screen rotations.
- It exposes participant data to the UI using **LiveData / Flow**, ensuring the UI reacts automatically to database updates.
- Business logic is handled here instead of the Activity, keeping UI classes lightweight and focused only on rendering.

Repository Layer

The Repository serves as a **single source of truth** for all data operations.

- It abstracts the Room database from the rest of the application.
- All insert, query, and validation operations are delegated through the repository.
- This design allows future extension (e.g., remote API integration) without changing the UI or ViewModel layers.

3. MODULE A – PARTICIPANT REGISTRATION (INPUT)

Module A is responsible for collecting and validating participant data during registration.

Data Integrity & Error Prevention

- Each participant is identified using a **unique integer User ID**.
- Before inserting new data, a **pre-insertion check** is performed to verify whether the entered ID already exists in the database.
- If a duplicate ID is detected, the insertion is safely aborted and the user is notified, preventing runtime crashes and database conflicts.

Input Validation

- The **Full Name** field is protected using an `InputFilter`.
- Numeric characters are actively blocked during typing, ensuring that only valid alphabetic input is accepted.
- This proactive validation approach improves data quality and user experience.

Hardware Integration (Camera)

- The application uses an **Implicit Intent** to launch the device's native camera application.
- A high-resolution image is captured and stored in **internal storage**.
- Instead of storing the bitmap directly in the database, only the **file path / URI** is persisted, ensuring optimal memory usage and database performance.

External Linking

- A **Browser Intent** is implemented to redirect users to the official **Ankara Science University website**.
- This fulfills the conference information requirement and demonstrates safe external intent handling.

4. MODULE B – PARTICIPANT VERIFICATION (OUTPUT)

Module B functions as the **verification desk logic** of the system and focuses on data retrieval and dynamic UI feedback.

Dynamic UI Feedback

- When a participant ID is entered, the system queries the Room database.
- Based on the retrieved `registrationType`, the background color of the verification screen is updated programmatically:
  - **Type 1 (Full Registration):** Green
  - **Type 2 (Student Registration):** Blue
  - **Type 3 (No Registration):** Orange
- This visual feedback allows instant interpretation without additional user actions.

Search & Error Handling

- If the entered ID does not exist in the database:
  - The background color changes to **Red**
  - A clear and descriptive error message is displayed
- The system handles all invalid states gracefully and **never crashes**, fully satisfying the error handling requirements.

5. DATABASE IMPLEMENTATION – ROOM

The application uses **Room Persistence Library** to manage local data storage reliably and efficiently.

Entity

- The `Participant` class is annotated with `@Entity`.
- `userId` is defined as the `@PrimaryKey`, guaranteeing uniqueness at the database level.

DAO

- The `ParticipantDao` interface defines all database operations.
- All functions are declared as `suspend`, enabling execution within coroutines.
- This ensures that database operations are performed **off the main thread**, preventing UI freezes.

Database Instance

- The `AppDatabase` class implements the **Singleton pattern**.
- This prevents multiple database instances from being created and optimizes resource usage across the application lifecycle.

6. CONCLUSION

The developed application successfully meets and exceeds all requirements defined in the grading rubric:

- **Module A – Registration (25 pts):** Fully implemented with validation, camera integration, and external linking
- **Module B – Verification (25 pts):** Dynamic UI updates, database querying, and safe error handling
- **MVVM Architecture (25 pts):** Strict separation of concerns and lifecycle-aware components
- **Room Database (15 pts):** Proper entity, DAO, and singleton database design

- **Error Handling (10 pts):** All invalid inputs and edge cases handled gracefully without crashes

In conclusion, this project demonstrates a solid understanding of **Android architecture, persistence, lifecycle management, and user-centered error handling**, resulting in a stable and professional-grade mobile application.