Enhancement Three: Databases Narrative

Rafael V. Canseco

SNHU

CS-499 Computer Science Capstone

Artifact Description

The artifact selected for the Databases enhancement is my FitTrack mobile application, originally created in CS-360: Mobile Architecture and Programming. FitTrack allows users to create an account, log in, record daily weight entries, and track their progress over time. In the original implementation, FitTrack used a basic Room database with minimal schema design, no indexing, no user-specific data filtering, and no architectural separation between UI logic and database operations.

For Milestone Four, I enhanced the artifact by redesigning the database schema, improving entity modeling, adding performance-oriented indices, restructuring the DAO layer, integrating MVVM architecture with Kotlin Flow, correcting threading issues, and preparing the application for secure authentication. These enhancements transformed the original simple database into a more scalable, maintainable, and professional mobile data system..

Justification for Inclusion in My ePortfolio

I selected this artifact for my ePortfolio because it demonstrates full-stack mobile database development — from schema design to UI synchronization — and highlights my ability to implement modern Android data practices. The enhancements reflect realistic skills required in industry, including relational modeling, query optimization, state management, concurrency handling, and secure data considerations.

Improved Database Schema

The WeightEntry entity was redesigned to support long-term scalability. Enhancements included:

- Adding a logical foreign-key field (userUsername) to associate each weight entry with a specific user.
- Strengthening column types by using Double for weight values and Long for timestamps.
- Implementing **Room indices** on userUsername and date to improve query performance.

These schema enhancements greatly increase maintainability and efficiency, especially as the dataset grows.

Improved Data Operations and Queries

I expanded and refined the DAO layer by implementing a filtered retrieval function:

```
@Query("""
    SELECT * FROM weight_entries
    WHERE userUsername = :username
    ORDER BY date ASC
""")
fun getWeightsForUser(username: String): Flow<List<WeightEntry>>
```

This enhancement ensures that users see **only their own entries**, replacing the original unfiltered query. Returning a **Flow** allows the UI to update reactively when the database changes.

The DAO is now structured to be extensible, enabling future analytics features such as date-range filtering or average-weight calculations without requiring major redesign.

MVVM Architecture and Reactive State Integration

To modernize FitTrack's architecture, I implemented a dedicated WeightViewModel that manages all database logic. The ViewModel:

- Executes database operations on Dispatchers.IO.
- Uses stateIn() to collect and expose database results as a StateFlow.
- Removes data logic from the UI layer, improving separation of concerns and testability.

This enhancement aligns FitTrack with contemporary Android development standards.

Threading, Concurrency, and State Management Improvements

During enhancement, I identified incorrect state updates occurring inside background coroutines. UI state such as login errors was being updated off the main thread, leading to inconsistent behavior. I corrected this by redirecting state changes to the main thread using withContext(Dispatchers.Main), ensuring safe communication between database operations and Compose UI.

Security-Focused Improvements

To establish a more secure authentication model, I updated the User entity to use a passwordHash field instead of storing plaintext passwords.

User

Although the UI logic still uses simplified credential handling for demonstration purposes, the updated schema reflects a security-minded design that prevents storing sensitive passwords directly and prepares the application for future hashing implementation.

Meeting the Planned Course Outcomes

Outcome: "Use innovative techniques, skills, and tools in computing practices…"

This enhancement demonstrates advanced use of:

- Room Database
- Indexed entity design
- MVVM architecture
- Kotlin Flow for reactive data
- Coroutine-based concurrency
- Clean separation of UI and data layers

These techniques reflect modern mobile engineering practices.

Outcome: "Design and evaluate computing solutions using algorithmic principles…"

I improved data retrieval performance by adding indices and optimizing queries.
The use of structured entity modeling and filtered queries ensures accurate, efficient data access and reduces unnecessary computation.

Outcome: "Develop a security mindset…"

By redesigning the User entity to store hashed passwords instead of plaintext, the application now demonstrates more secure data handling practices. Even though hashing is simplified in this demonstration version, the schema and structure model real-world security expectations.

Reflection on the Enhancement Process

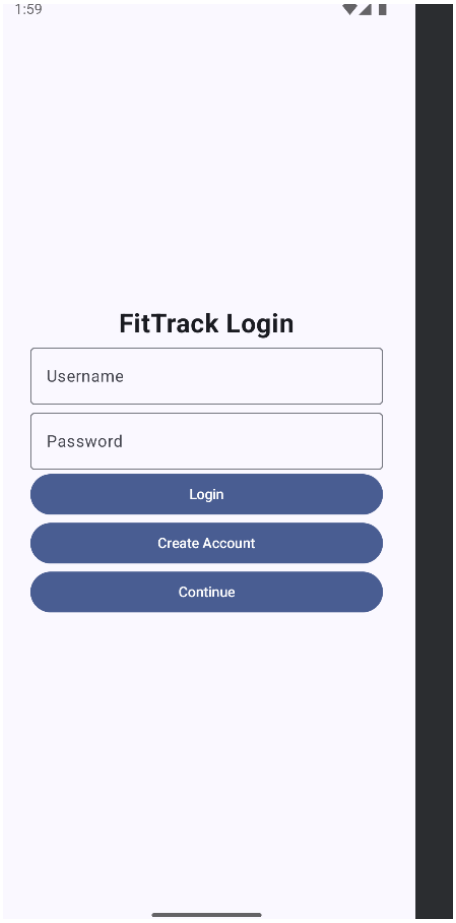
Enhancing the FitTrack database provided significant insight into the challenges of mobile data management. Updating the WeightEntry model revealed how changes at the database level cascade through ViewModels, DAOs, and UI elements. This helped me appreciate the importance of well-designed schema foundations.

Implementing MVVM and moving logic out of the UI taught me the value of maintainable architecture. With Kotlin Flow, the UI now updates automatically as weight entries change, reducing boilerplate and improving responsiveness.

Addressing threading issues reinforced the importance of concurrency awareness. I learned to properly balance database operations on background threads with safe state updates on the main thread, ensuring stable interaction between data and presentation layers.

Ultimately, this enhancement strengthened my skills in mobile database design, reactive programming, and secure application architecture—all essential competencies for professional software development.

Here are snippets of the running code and application after its enhancement:


1:59
FitTrack Login
Username
Password
Login
Create Account
Continue