Amos Roland

Southern New Hampshire University

CS-499 Computer Science Capstone

Professor Akhil Gudivada

30 November 2025

CS 499 Module 5-2 Milestone Four: Enhancement Three: Databases

Artifact Description

The artifact I selected for the databases category is my mobile Weight Tracking App, originally created in CS 360 Mobile Architecture and Programming. This Android Studio project allows users to register, log in, set weight goals, and log their daily progress. The original version stored all data using a basic SQLite database with simple CRUD operations and a limited structure.

For this milestone, I significantly enhanced the artifact to meet modern database, security, and mobile application standards. These enhancements demonstrate my ability to securely manage user information, protect sensitive data, and prepare the application for scalable cloud integration.

Why This Artifact Is in My ePortfolio

I selected this artifact because it highlights my ability to design, secure, and improve a mobile application that relies heavily on database operations. Database management is critical to information security and software engineering, which aligns with my professional goals. The

enhancements completed for this milestone demonstrate my skills in secure storage, structured database design, and user experience improvements.

Below are the enhancements completed for this module:

1. AES Encryption for Sensitive Data
    - I implemented AES encryption for weight entries and goal values before storage. This protects user health data and demonstrates secure handling of sensitive information.

```
46
47      // Simple AES settings for this course project
48      // In a real app these would not be hardcoded
        2 usages
49      private static final String ENCRYPTION_KEY = "A1b2C3d4E5f6G7h8"; // 16 chars
        2 usages
50      private static final String ENCRYPTION_IV  = "1H2g3J4k5L6m7N8";  // 16 chars
51
```

```
108     // ------------------------------------------------------------------------
109     // Encryption helpers
110     // ------------------------------------------------------------------------
111
        3 usages
112     private String encrypt(String plainText) {
113         if (plainText == null) return null;
114         try {
115             IvParameterSpec iv = new IvParameterSpec(ENCRYPTION_IV.getBytes());
116             SecretKeySpec keySpec = new SecretKeySpec(ENCRYPTION_KEY.getBytes(), algorithm: "AES");
117             Cipher cipher = Cipher.getInstance( transformation: "AES/CBC/PKCS5Padding");
118             cipher.init(Cipher.ENCRYPT_MODE, keySpec, iv);
119             byte[] encrypted = cipher.doFinal(plainText.getBytes());
120             return Base64.encodeToString(encrypted, Base64.NO_WRAP);
121         } catch (Exception e) {
122             // If encryption fails, fall back to plain text to avoid crashes
123             return plainText;
124         }
125     }
126
        5 usages
127     private String decrypt(String cipherText) {
128         if (cipherText == null) return null;
129         try {
130             IvParameterSpec iv = new IvParameterSpec(ENCRYPTION_IV.getBytes());
131             SecretKeySpec keySpec = new SecretKeySpec(ENCRYPTION_KEY.getBytes(), algorithm: "AES");
132             Cipher cipher = Cipher.getInstance( transformation: "AES/CBC/PKCS5Padding");
133             cipher.init(Cipher.DECRYPT_MODE, keySpec, iv);
134             byte[] decoded = Base64.decode(cipherText, Base64.NO_WRAP);
135             byte[] original = cipher.doFinal(decoded);
136             return new String(original);
137         } catch (Exception e) {
138             // If decryption fails, just return the raw string
139             return cipherText;
140         }
141     }
142
```

Figure 1: AES encryption and decryption methods inside DatabaseHelper

2. Secure Hashed Password Storage

    - I replaced plaintext password storage with SHA-256 hashing, which prevents exposure of user credentials and aligns with industry security practices.

```
143     // ------------------------------------------------------------------------
144     // Password hashing
145     // ------------------------------------------------------------------------
146
        2 usages
147     private String hashPassword(String plainPassword) {
148         if (plainPassword == null) {
149             return null;
150         }
151         try {
152             MessageDigest digest = MessageDigest.getInstance( algorithm: "SHA-256");
153             byte[] hashedBytes = digest.digest(plainPassword.getBytes());
154             StringBuilder sb = new StringBuilder();
155             for (byte b : hashedBytes) {
156                 String hex = Integer.toHexString( i: 0xff & b);
157                 if (hex.length() == 1) {
158                     sb.append('0');
159                 }
160                 sb.append(hex);
161             }
162             return sb.toString();
163         } catch (NoSuchAlgorithmException e) {
164             // Fallback to plain text if hashing fails to avoid crashes
165             return plainPassword;
166         }
167     }
```

Figure 2: Password hashing method used during registration

3. Material Design UI Upgrade

    - I updated the layouts using Material Design principles, improving accessibility, color contrast, spacing, and overall usability.

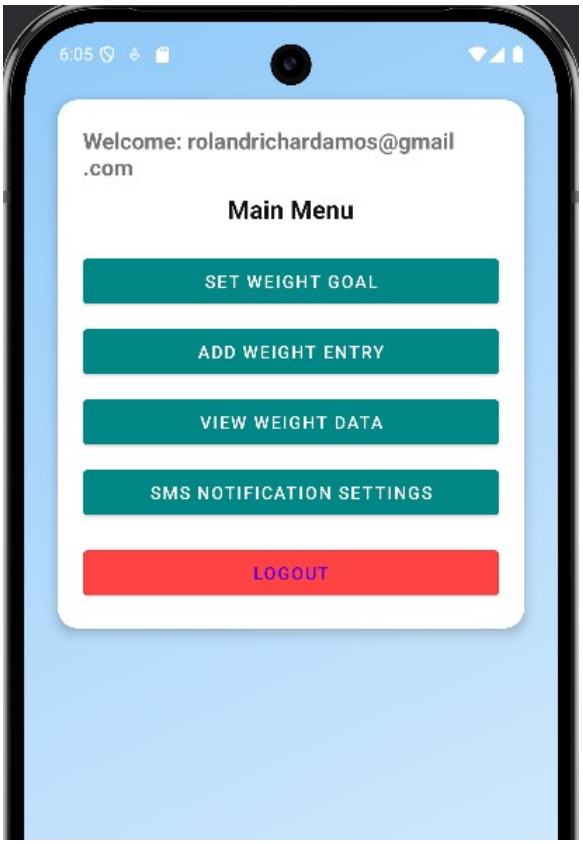

Figure 3: Main Menu structure

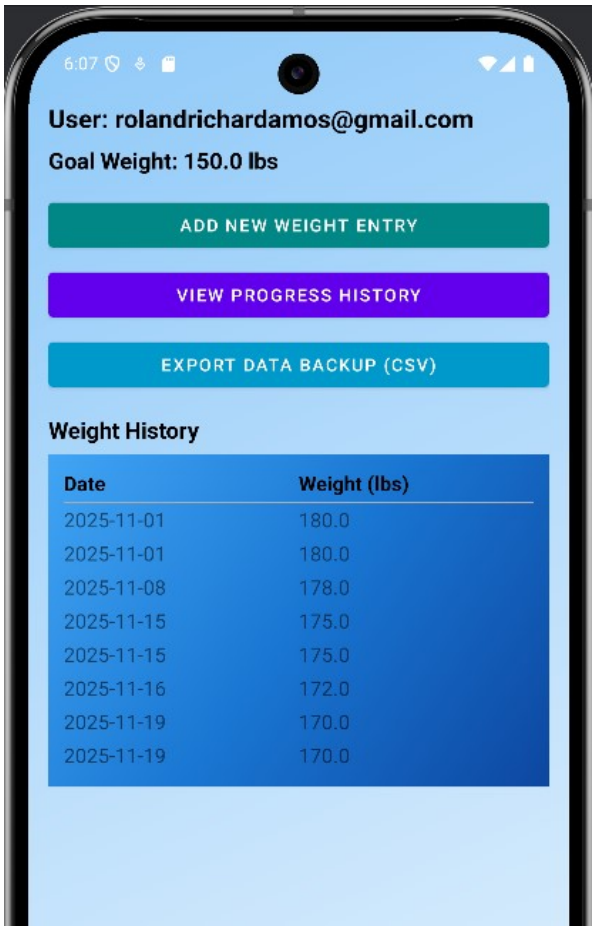

Figure 4: Enhanced progress table with edit and delete dialog

4. Structured Database Layer for Future Cloud Sync

Even though the module did not require implementing Firebase cloud sync, I prepared the architecture so that cloud integration can be added later without rewriting the application. This includes:

- Clean separation of data logic
- Model class for weight history
- Export-to-CSV feature

- Unified encryption and decryption methods

5. Enhanced Database Functions

   I added multiple new database features:

   - Weight history retrieval
   - Date-range query support
   - CSV export for backup
   - More efficient SQL queries
   - Clean and organized CRUD operations

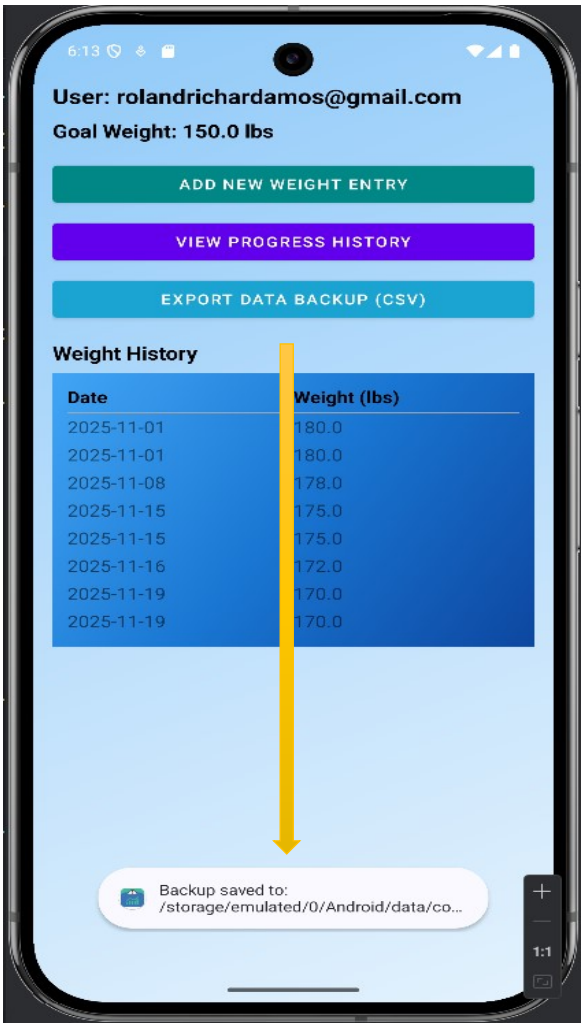

Figure 5: Exported CSV file and success message in app

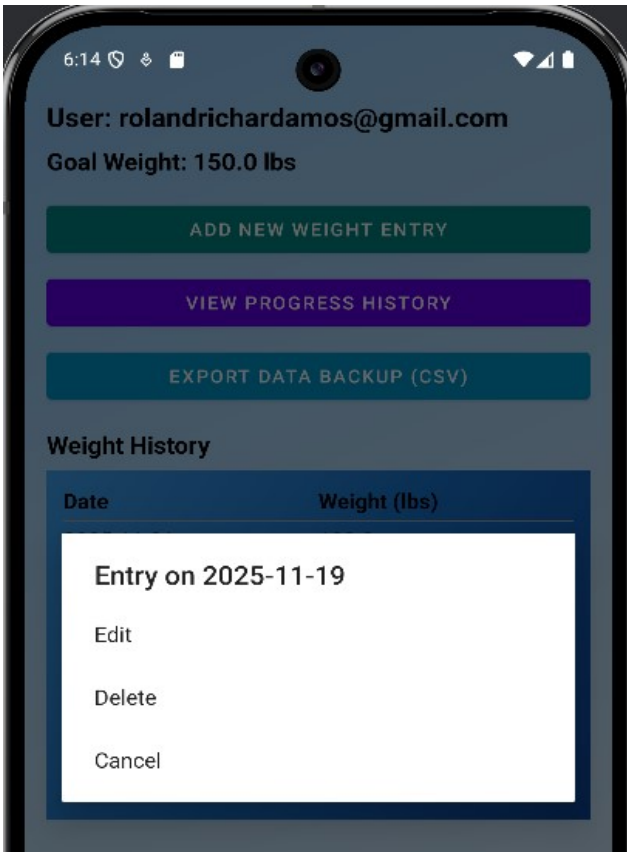

Figure 6: SQLite table structure and CRUD functions

6. Weight Progress Line Chart (New Addition)

    - I added a new visual feature using MPAndroidChart. The chart decrypts stored weight values, sorts them by date, and plots them on a line graph so users can visually track their progress over time.

    This enhancement shows:

    - Real use of database query results
    - Decryption before rendering
    - Transformation of structured data into meaningful visuals
    - Professional data visualization skills

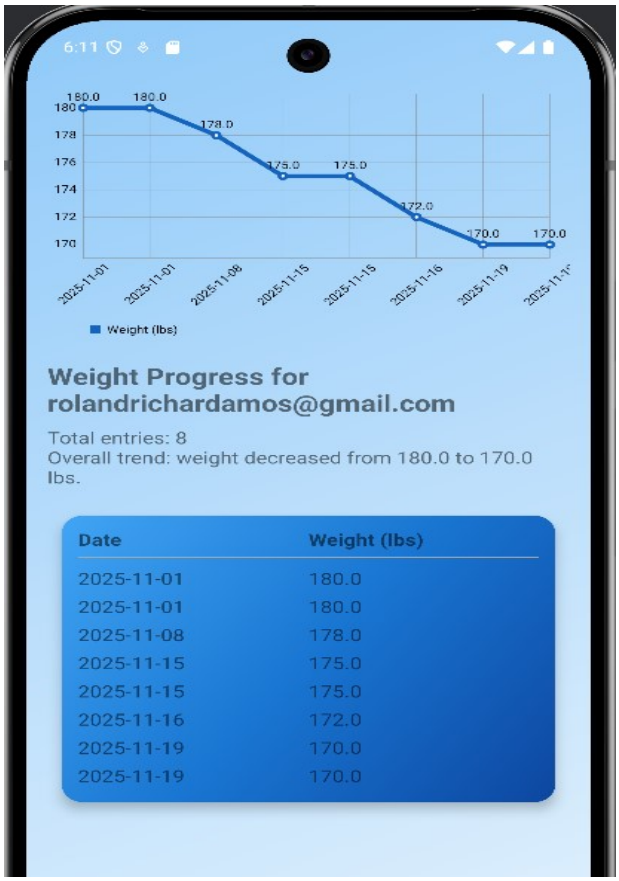

Figure 7: Graph showing weight progression with blue trend line

Alignment to Course Outcomes

In Module One, I planned to target Outcome Four and Outcome Five, and the completed enhancement fulfills both.

- Outcome Four: Innovative Techniques, Skills, and Tools. I demonstrated this outcome through:
  - AES encryption implementation
  - SHA-256 password hashing
  - MPAndroidChart integration
  - Structured SQLite data architecture

- Export-to-CSV backup system
- Material Design UI improvements

These demonstrate my ability to use modern tools and secure design practices to upgrade an existing system.

- Outcome Five – Security Mindset. I met this outcome by:
  - Encrypting sensitive weight and goal data
  - Hashing all user passwords
  - Validating all user input
  - Preventing exposed data access
  - Designing database queries that avoid injection risks

These enhancements show that I approach mobile development with a strong security foundation.

Reflection on the Enhancement Process

Enhancing this artifact helped me deepen my understanding of secure database design, encrypted data storage, and scalable architecture in mobile applications. I learned how encryption affects database queries and how to structure an application so that both local and cloud databases can be supported in the future.

The biggest challenge was ensuring that encrypted values remained usable for analytics and graph visualization. I had to carefully decrypt values and map them into the MPAndroidChart framework without breaking the database workflow.

I also learned how important UI clarity is when presenting database-driven content. Adding the progress line chart helped me improve my data visualization skills and showed how users benefit from graphical feedback.

Through this enhancement, I strengthened my skills in mobile development, database design, and secure data handling. This artifact now reflects my readiness to develop secure applications in real-world environments and demonstrates the database expertise required in professional computer science work.

References:

Design for Safety. (n.d.). *Android Developers*. Retrieved November 27, 2025, from https://developer.android.com/quality/privacy-and-security

Firebase Realtime Database. (n.d.). Firebase. Retrieved November 28, 2025, from https://firebase.google.com/docs/database

PhilJay. (n.d.). *MPAndroidChart documentation.* GitHub. https://github.com/PhilJay/MPAndroidChart

What is cloud security? (n.d.). Amazon Web Services, Inc. Retrieved November 28, 2025, from https://aws.amazon.com/security/