Elijah Harris

# Databases

## Original Code Overview

The original implementation of the Mass Tracker App handled database operations directly in the DashboardActivity class. This approach not only violated the principle of separation of concerns but also introduced redundancy, making the code harder to maintain and prone to errors. Critical database operations such as fetching, adding, and updating data were scattered across the application without any centralization, which created a tightly coupled and fragile system.

Additionally, the database schema lacked indexes and constraints, which limited its performance and data integrity. Furthermore, there was no functionality for database backups or saving data externally, making the application less reliable and less user-friendly.

## Enhancement Objectives

The main goals of the database and repository enhancements were:

1. **Centralize Database Access:** Introduce a repository layer (MassRepository) to abstract and centralize all database operations.Enhance modularity: Create reusable components for business logic and data access to facilitate future development and collaboration.
2. **Improve Performance:** Add indexes and constraints to the database schema for faster queries and enhanced data integrity.
3. **Add Backup Functionality:** Implement a mechanism for saving and restoring the database, ensuring data reliability and user confidence.
4. **Enhance Documentation:** Add comprehensive comments to describe the methods and database schema updates to improve collaboration and maintainability.

# Key Enhancements

## Introduction of MassRepository Class

* **Original Issue:** Database operations were intermingled with UI and business logic in activities, leading to tightly coupled code.
* **Enhancement:** I introduced the MassRepository class, which serves as a centralized layer for all database interactions. This class encapsulates methods for fetching, adding, updating, and deleting records, abstracting the database logic away from the UI.
* **Impact:** The repository pattern promotes modularity and reusability, allowing the UI and business logic to remain independent of database implementation details. It also simplifies unit testing by enabling mock repositories.

## Schema Enhancements

* **Original Issue:** The database schema lacked indexes and constraints, which are essential for performance optimization and ensuring data integrity.
* **Enhancement:**
  + Added indexes on frequently queried columns, such as date and user\_id\_fk, to speed up lookups.
  + Introduced a UNIQUE constraint on the date column within the context of a specific user to prevent duplicate logs.
  + Enforced NOT NULL constraints on critical columns to avoid null value errors.
* **Impact:** These changes improved query performance and ensured that data integrity violations, such as duplicate or incomplete entries, are caught at the database level.

## Backup and Restore Functionality

* **Original Issue:** The application lacked any mechanism to back up or restore the database, risking data loss in cases of crashes or reinstallation.
* **Enhancement:**
  + Implemented a method to export the database as a .db file for backup purposes.
  + Added functionality to import and restore the database from a saved file.
* **Impact:** This feature enhances user confidence in the application by ensuring that their data is not permanently lost in the event of a failure. It also aligns the app with industry standards for data reliability.

## Alignment with Course Outcomes

1. **Employing Strategies for Collaboration**
   * Centralizing database logic in the repository layer enables clearer collaboration by creating a single source of truth for data operations.
   * Enhanced documentation fosters a collaborative environment, making the codebase easier for a diverse audience of developers to understand and contribute to.
2. **Designing Professional-Quality Communications**
   * The restructured database and repository codebase, combined with detailed documentation, provides clear, technically sound communication to both developers and stakeholders.
   * The backup functionality and schema enhancements directly communicate the application's commitment to reliability and professionalism.
3. **Designing and Evaluating Computing Solutions**
   * Adding indexes and constraints to the database schema demonstrates an understanding of performance optimization and data integrity principles.
   * The decision to use the repository pattern reflects thoughtful trade-offs, balancing simplicity, scalability, and modularity.
4. **Demonstrating Techniques and Tools**
   * The repository pattern and enhanced schema highlight the use of well-founded techniques and tools to solve real-world problems in data management.
   * The backup and restore functionality demonstrates the implementation of innovative features that deliver tangible value to users.
5. **Developing a Security Mindset**
   * Enforcing constraints such as NOT NULL and UNIQUE reduces the risk of data corruption, which aligns with the goal of minimizing vulnerabilities.
   * Centralizing database operations in the repository provides a single location to validate inputs and outputs, ensuring consistent security practices.

## Reflection on the Process

Enhancing the database and introducing the MassRepository class deepened my understanding of modular design principles and efficient data management techniques. The process of updating the schema with constraints and indexes required careful consideration to ensure backward compatibility and minimal disruption to existing functionality.

The most challenging aspect was designing the backup and restore functionality in a way that is both user-friendly and robust. This feature required a deep dive into Android file storage APIs and careful handling of file permissions.

Through these enhancements, I gained valuable experience in optimizing database performance, ensuring data integrity, and implementing features that align with user expectations. These changes significantly improved the scalability, maintainability, and reliability of the Mass Tracker App, aligning it with industry standards for modern software systems.