For my database category artifact, I chose to further enhance my Android application, "Inventory Pro". It had a core limitation of storing every item in a single, flat table in the SQLite database. This design proved to be non-scalable and quite inefficient. As a user's inventory grew, the application presented a monolithic, unsorted list, making it nearly impossible to organize, categorize, or efficiently locate specific items. This fundamental design flaw made the application impractical for any serious use and presented a perfect opportunity to demonstrate my skills in relational database design and data-driven application development.

The primary goal of this enhancement was to re-architect the application's data layer to support a hierarchical, folder-based organization system, transforming it from a simple list into a structured management tool. This was achieved by fundamentally redesigning the SQLite database schema. I introduced a new Folders table and established a one-to-many relationship with the existing Items table by adding a folder\_id foreign key. This enhancement showcases several key database and software engineering skills. First, it demonstrates relational database design, as I modeled a real-world requirement (categorization) into a normalized schema with primary and foreign keys. Second, it highlights my ability to implement advanced SQL and data manipulation by developing full CRUD (Create, Read, Update, Delete) operations for the new Folders table and updating the item-related queries to manage the new relationship. A major component of this was enforcing data integrity. For instance, the logic now prevents the deletion of a folder if it still contains items, protecting the user from accidental data loss. Finally, the project required full-stack integration, as the database changes required a complete overhaul of the user experience, from the backend queries to the frontend UI.

This enhancement successfully meets the course outcomes I planned, particularly Outcome 4 (Use Well-Founded Techniques). By replacing a flat-file structure with a proper relational model, I implemented a standard, well-founded database technique that delivered immense value in usability and organization. The project also aligns with Outcome 3 (Design and Evaluate Solutions), as the new architecture is a far superior solution to the problem of data organization, directly addressing the trade-offs of the original, simplistic design. I also achieved Outcome 1 (Employ strategies for building collaborative environments) during this artifact enhancement. The refactoring process forced a thoughtful reorganization of the entire application into a consistent, logical project structure. By separating concerns into distinct components the codebase became highly modular. This modularity, combined with properly documented code that explains the purpose and logic of each function, significantly reduces the barrier to entry for new team members. A future developer can now quickly become familiarized with the project, understand the data flow, and contribute effectively without needing to navigate a monolithic, complex system.

The process of implementing this enhancement was quite a learning experience for me. I again initially underestimated the scope of the changes required. I learned that modifying the database was not an isolated task, it required creating an entirely new FolderContentsActivity to display items within a folder, registering this new activity in the AndroidManifest.xml, and developing a new FolderAdapter for the main screen's RecyclerView. The original DashboardActivity had to be refactored to manage folders instead of items, and the AddEditItemActivity required significant modification to include the folder selection logic.

I faced several technical challenges that strengthened my practical skills. The most significant was implementing a database migration strategy. To ensure existing users wouldn't lose their data upon updating, I used the onUpgrade method in my DatabaseHelper to programmatically create a default "General" folder and then execute a SQL query to migrate all existing items into it. Another challenge involved debugging the application's state management, particularly an Activity lifecycle bug where a NullPointerException occurred because the DatabaseHelper was being used before it was initialized. This reinforced the importance of understanding the precise order of operations in Android's onCreate method. Lastly, managing the dynamic UI in the "Add/Edit Item" screen, where the folder selection controls could overlap, required me to restructure the ConstraintLayout using a FrameLayout as a stable container. This taught me how to better build robust and adaptive user interfaces.

Here are some screenshots from my enhancement:

