CS 499 - Artifact Enhancement 3 Database Focus

Conor Steward

conor.steward@snhu.edu

10/6/24

The Appointment Now application has recently undergone enhancements focused on single sign-on, logging out, historical look-up, and PDF saving. This was done for this class, CS 499, but the application was originally created in CS 360. Appointment Now is intended as an appointment scheduling application for healthcare facilities. Before this enhancement the app could handle adding new appointments, creating a new account, secure login/logout, appointment history lookup, SMS permissions, and appointment editing/deleting capabilities. Recent rounds of enhancement have added PDF saving, DB triggers/views, and optimized data retrieval with a friendly user interface. This application was ideal for these enhancements as it already had a database structure that would accommodate alterations to vastly improve the functionality of the application.

These recent changes demonstrate strong capabilities in managing local storage with SQLite, handling CRUD operations for both User and Event tables. Implemented features like foreign key constraints, cascading actions, and triggers to ensure data integrity showcases an understanding of relational database principles. The integration of views and indexing highlights knowledge of optimizing database queries. By refactoring the DatabaseHelper, it is ensured that proper resource management is taken care of, like closing the database and cursor instances. This enhanced the stability of the app, preventing memory leaks and crashes. The addition of foreign key constraints and database triggers improved data integrity and consistency, ensuring that "orphan" records are not left when users or events are deleted.

Implementation of the AddEventDialogFragment demonstrates proficiency in creating user-friendly interfaces. Implementing input validation, date picking, and PDF file selection enriches the user experience. Skills in handling Android fragment lifecycle events are shown, such as managing communication between the fragment and the parent activity using

OnSaveListener, ensuring that the event list is updated appropriately after saving. Handling of input validation and use of interactive dialogs show attention to detail in UX design.

Improvements include a refactoring of the save functionality to ensure that the event creation and updates are properly handled, including database interaction and passing the correct data back to the activity. Adding PDF upload functionality and integrating it into the event creation flow demonstrates a feature enhancement, improving the app's capabilities.

The communication between the EventDisplayActivity and the AddEventDialogFragment demonstrates an ability to pass data between Android components, which is crucial for maintaining state between activities and fragments. SharedPreferences has been utilized to maintain user session data, showing an understanding of persistent storage in Android. Refactoring the event update logic ensures that the edit event process correctly updates the database and the UI. This enhancement ensures that the event list refreshes after any modification. Implementation of proper error handling and logging in multiple places demonstrates an understanding of how to prevent crashes and improve app robustness. Improved logging in the database interaction methods makes for easier debugging and simpler identification of failed operations, enhancing the maintainability of the code.

Knowledge of Android's lifecycle management is shown by implementing proper resource management techniques, like closing cursors and database instances in the onDestroy method. By implementing the onDestroy method in activities and fragments, the resource management and lifecycle handling in the application was improved, reducing memory leaks and ensuring smoother performance.

Use of fragments like AddEventDialogFragment shows an understanding of how to break down UI into reusable components, making it more modular and maintainable. By refactoring the save button functionality within the AddEventDialogFragment, ensuring that data is passed back to the parent activity, and handling PDF uploads and event editing, the functionality and robustness of the fragment have been improved.

- Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision-making in the field of computer science.
    - By integrating modular components like AddEventDialogFragment, the code now facilitates better collaboration among team members. Different developers can work on specific fragments or features independently while maintaining consistency across the app, which sticks closely to industry standard best practices.
- Design, develop, and deliver professional-quality oral, written, and visual communications that are coherent, technically sound, and appropriately adapted to specific audiences and contexts.
    - The use of detailed, clear, and well-structured code comments, along with appropriate logging and error messages, improves communication within the codebase. These changes make the codebase more accessible to other developers or maintainers, ensuring technical soundness.
- Design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution while managing the trade-offs involved in design choices.
    - By refactoring the event save and update functionality a key problem in the event creation/editing process has been solved. The changes improved the way data is passed between fragments and activities, applying sound algorithmic principles.

- Demonstrate an ability to use well-founded and innovative techniques, skills, and tools in computing practices for the purpose of implementing computer solutions that deliver value and accomplish industry-specific goals.
    - The use of SQLite for local data storage, handling PDF uploads, and integrating user-friendly UI components demonstrates an application of innovative techniques and tools in Android development.
- Develop a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources.
    - The use of SharedPreferences to securely manage user sessions and handle permissions reflects a consideration for data privacy and security. The refactoring to close database connections enhances the security of the app by reducing potential vulnerabilities. Input validation ensures that user-provided data like patient names and appointment details are properly sanitized, mitigating potential injection attacks or other adversarial exploits.

Update plans were altered in that new tables were not needed to store PDF's instead, existing tables were altered to hold the PDF's. Saving is handled through the use of a URI (Uniform Resource Identifier). This means that when a user selects a PDF file, its URI is stored in the SQLite database, but the file itself is not saved directly in the database.

The myriad challenges I faced in implementing all of the above alterations were significant in my journey as a software developer. I gained a deeper understanding of designing SQLite database schemas to accommodate evolving requirements. I learned how to implement foreign key constraints and design views, indices, and triggers to maintain data integrity and optimize retrieval. I became more proficient at designing features that enhance the user experience, such as allowing users to upload PDFs as part of event creation or modification. This added functionality required careful thought to balance usability with the underlying technical complexity. Understanding how to streamline the UI while incorporating new features was a valuable lesson in balancing technical capabilities with an intuitive interface. Handling file storage in Android is a nuanced task. While it's easy to allow users to select files, storing and accessing these files across different activities required using content URIs. I learned how to manage file permissions and handle content URIs appropriately to allow users to view and interact with uploaded files. The challenge here was ensuring that the selected PDFs are stored as URIs in the database while ensuring that users can easily access and view them through the app. Working with Android's ActivityResultLauncher for selecting files and DatePickerDialog for picking dates required understanding how Android's newer APIs integrate with fragments and activities. This was an important learning experience, as older mechanisms like onActivityResult() are being deprecated. Effective logging with Logcat and error handling, especially when dealing with database operations, is crucial to ensure the app works smoothly. I learned how to implement robust logging to catch potential errors related to database access or file handling, which was instrumental in troubleshooting and resolving issues. Working on refactoring existing features to meet new requirements reinforced the importance of maintaining modular and reusable code. For example, the use of fragments for event creation and editing made the code more reusable, but I had to be mindful of managing state across fragments and activities.