Data Backup Application Report

Submitted By:
Asees Khuran (MT23024)
Devkul Sahu (MT23116)
Mazhar Sayed (MT23124)

Introduction:

The application is designed to help users automatically back up files that haven't been accessed within a specific time frame. It scans the device's Downloads directory for files that haven't been accessed in the last month (adjusted to one minute for testing) and uploads them to Firebase Storage. This helps users free up space and secure their data without manual intervention. The app respects device constraints such as battery levels and network availability, ensuring that backups are performed without disrupting the user's device usage. It dynamically requests the necessary permissions to access files and provides a user-friendly interface to monitor the progress of these operations.

Key Features:

1. Backup Old Files: Identifies and backs up files that have not been accessed within a defined time frame.

2. Firebase Integration: Uses Firebase Storage for storing backed-up files and Firebase Database to track file metadata.

3. Progress Display: Shows a progress bar to indicate the current progress of backup operations.

4. Dynamic Permissions Handling: Requests and checks necessary permissions at runtime to ensure smooth operation.

5. Device Status Checks: Verifies that the device has sufficient battery and network connectivity before proceeding with backup tasks.

6. Sanitization of File Paths: Ensures that filenames are sanitized to meet Firebase path requirements, avoiding common pitfalls like invalid characters.

Components

MainActivity: It is the entry point of the application. It hosts the primary user interface that users interact with when they open the app.

Functionality:

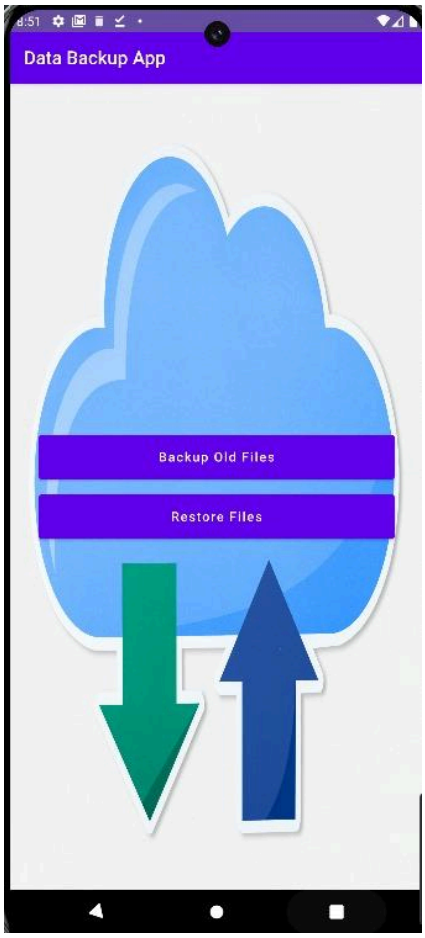
- Displays options to navigate to BackupActivity or RestoreActivity.
- Provides a visual interface for initiating the backup and restore processes.
- MainActivity uses the Android lifecycle to manage the initiation of background tasks and ensures the user can navigate through different functionalities efficiently.

BackupActivity: The core UI component of the app, handling user interactions, triggering backup operations, and displaying progress. It manages the backup process using state holders to reflect UI changes dynamically based on ongoing operations.

RestoreActivity is responsible for the user interface and functionality related to restoring files from Firebase Storage back to the device.

Functionality:

- Allows users to view and select backed-up files stored in Firebase.
- Initiates the download and restoration of selected files.
- This activity provides feedback on the restoration progress and ensures files are correctly restored to their original locations.



FileEntity: Represents the metadata of a file, including its name, path, and backup status. This class is crucial for tracking which files need to be processed and their current state in the backup process.

FileRepository: Manages interactions with the local database to store and retrieve file metadata, helping track backed-up files and those pending backup.

FirebaseDBHelper: Provides methods for interacting with Firebase, including uploading files to Firebase Storage and updating Firebase Database with file metadata, facilitating cloud storage operations.

DeviceStatusUtils: Utility class to check device-specific constraints like battery level and network availability, ensuring that backups are done without interrupting the user's normal device usage.

Operations

1. Permission Checks: At startup, the app checks for necessary storage permissions. If not granted, it requests them from the user.

2. File Scanning: Scans the Downloads directory for files that have not been accessed within the specified time frame, preparing them for backup.

3. Backup Execution: For each file identified, the app checks device conditions (battery and network), sanitizes the file name to avoid Firebase path errors, and then uploads the file to Firebase Storage while updating both the local and Firebase databases.

4. Progress Monitoring: During the backup process, progress is updated in real-time on the UI, giving users clear feedback on the operation's status.

5. Error Handling: The app provides robust error handling throughout the backup process, catching exceptions, and providing feedback via toast messages for issues like permission denials, file I/O errors, and Firebase upload problems.

Key Technologies and Practices

- Kotlin:
    - Usage: Primary programming language.
    - Benefits: Concise syntax, null safety, and interoperability with Java. Coroutines facilitate efficient background task management.
- Android Jetpack Compose:
    - Usage: UI development.
    - Benefits: Declarative approach accelerates UI development, creating responsive and dynamic UIs with less code.
- Firebase Storage:
    - Usage: Cloud storage for backed-up files.
    - Benefits: Powerful, simple, and cost-effective object storage service, facilitating easy storage and retrieval of user-generated content.
- Firebase Realtime Database:
    - Usage: Store and manage metadata in real-time.

    - Benefits: Flexible and scalable database, supports real-time syncing across all clients, even when offline.
- Coroutines & Kotlin Flow:
    - Usage: Asynchronous task management and real-time data flow.
    - Benefits: Simplifies asynchronous programming, supports reactive patterns, and enhances the development of a robust data layer.
- Material Design Components:
    - Usage: Designing UI components.
    - Benefits: Comprehensive suite of UI tools ensuring consistency and professional aesthetics.
- Data Binding & View Binding:
    - Usage: Interaction between app logic and UI elements.
    - Benefits: Minimizes boilerplate code, improves code clarity, and reduces bugs.
- Android Permissions (Runtime Permissions):
    - Usage: Managing access to sensitive permissions.
    - Benefits: Enhances app security by requesting permissions at runtime, providing more control to the user.
- Local Storage Access (File I/O):
    - Usage: Reading from and writing to local storage.
    - Benefits: Essential for scanning and backing up files, efficiently managing file operations.
- Device Status Utils (Battery and Network Checks):
    - Usage: Checking device conditions before operations.
    - Benefits: Ensures optimal operation conditions, avoiding battery drain and failed uploads.
- Android WorkManager (Planned):
    - Usage: Managing deferrable, guaranteed background work.
    - Benefits: Efficient scheduling of background tasks, considering device constraints.
- Dependency Injection (Potential with Hilt/Dagger, Planned):
    - Usage: Managing dependencies for easier testing.
    - Benefits: Simplifies dependency wiring, enhances testability and modularity.

References:

- Android Developer Documentation: https://developer.android.com/docs
- Retrofit Documentation: https://square.github.io/retrofit/
- Firebase Documentation: https://firebase.google.com/docs