🗃️ Offline-First Data Sync and Logout Behavior

Offline-First Architecture: Data Saving and Sync Queue

- All note changes (create, edit, delete) are performed locally.
- Each such change is added to a synchronization queue (sync queue).
- The sync queue is stored separately from the main notes table and must include a user ID to associate each record with a specific user.
- The user ID is an internal unique identifier that is generated after authentication (e.g., using a UUID) and used to track sync records per user.
- This ID is not a username or email, in order to avoid storing any personal data. It allows the app to:
  - preserve local changes even after logout,
  - and correctly restore them after the user logs back into the same device.

☁️ Syncing with the Server

- Sync occurs:
  - **Automatically** (based on the interval set in Settings)
  - **Manually** (via the “Sync Data” button)
- WorkManager performs background synchronization by processing all records from the sync_queue table associated with the current user (userId). The type of operation (CREATE, UPDATE, DELETE) is determined from the operation field and used to perform the corresponding API request.
- After a successful sync the relevant entries are removed from the sync_queue.

📦 sync_queue Structure (example)

```
data class SyncRecord(
    val id: UUID,
    val userId: String,
    val noteId: String,
    val operation: SyncOperation, // CREATE, UPDATE, DELETE
    val payload: String, // JSON representation of the note
    val timestamp: Long
)
```

Sync Behavior

- Synchronization can be triggered in two ways:
  - **Automatic sync**, based on the configured interval in Settings (15 min, 30 min, 1 hour)
  - **Manual sync**, via the “Sync Data” action in the Settings screen
- When sync is triggered (manually or via WorkManager), the app:
  - Reads all pending changes from the sync queue
  - Sends them to the API
  - Removes or updates records upon successful sync
  - Retries failed syncs at the next interval or login
- The sync system must support:
  - Reliable background execution
  - Automatic resume after device reboot
  - Retry logic on failure

Logout Behavior (Online Only)

- Before initiating logout, the app checks if the sync queue contains any unsynced changes.
- If it does, show the dialog:
  - “You have unsynced changes. What would you like to do before logging out?” [Sync now] [Log out without syncing]

➤ If the user selects Sync now:

- Perform a full two-way sync with the backend
- If sync succeeds:
  - Proceed with logout
  - Clear the local notes database
  - Clear session tokens
  - Navigate to the login screen
  - Sync queue **should be cleared**
- If sync fails, show another dialog:
  - “An error occurred during sync. You can try again or log out without syncing. Your changes will remain saved locally.” [Cancel] [Log out without syncing]
  - **Cancel**: Close the dialog, do not log out
  - **Log out without syncing**: Proceed as described below

➤ If the user selects Log out without syncing:

- Immediately log out
- Clear local notes database and session data
- **Sync queue remains — all pending records are retained with the internal userId**

Post-login Sync Recovery

- After login, if the sync queue contains records for the current internal userId:
  - The app automatically syncs them in the background
  - Upon successful sync, the corresponding records are removed from the queue

Offline Restriction

- Logout is not allowed when the device is offline
- In that case, show the message: “You need an internet connection to log out.”
- This prevents the loss of pending unsynced data.