**Updated Database Design Document**

**1. Database Technology Choice**

**Chosen Technology:** Dexie.js (IndexedDB)

**Justification:**

* Local Storage: Dexie.js is built on top of IndexedDB, which allows for efficient local data storage, making it an ideal choice for applications that need offline access and fast retrieval without relying on an external server.
* Flexibility: Dexie.js provides a straightforward API for querying and managing data structures, simplifying development and reducing overhead.
* Scalability: While Dexie.js is used for local storage, future scalability plans may involve integrating with a cloud database (such as Firebase or AWS DynamoDB) for user data syncing and backup.

**2. Updated Data Structures**

The revised data model includes three tables: Users, Calculations, and Favorites. Additionally, I have incorporated metadata fields and features that were previously considered as future enhancements.

Table 1: Users

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| id | INTEGER | Primary key, auto-incremented identifier |
| username | TEXT | Unique username for authentication |
| email | TEXT | User's email address |
| password\_hash | TEXT | Hashed password for secure login |
| created\_at | TEXT | Timestamp of user registration |

**Purpose:** The Users table stores user account information, enabling user-specific data management. This table is crucial for allowing multiple users to maintain their own calculation history and preferences.

**Usage Scenario:**

* When a user signs up, their information is stored in this table.
* The app checks this table for user authentication during login.

Table 2: Calculations

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| id | INTEGER | Primary key, auto-incremented identifier |
| user\_id | INTEGER | Foreign key linking to Users table |
| type | TEXT | Type of calculation (for example, "Ohm's Law") |
| voltage | TEXT | Voltage value entered by the user |
| current | TEXT | Current value entered by the user |
| resistance | TEXT | Resistance value entered by the user |
| result | TEXT | Result of the calculation |
| calculated\_at | TEXT | Timestamp of the calculation |
| is\_deleted | BOOLEAN | Flag for soft deletion |

**Purpose:** The Calculations table logs user-generated calculations, allowing users to track and manage their calculation history. The is\_deleted flag supports a soft delete feature, enabling users to "delete" entries without permanently removing them.

**Usage Scenario:**

* When a user performs a calculation, it is saved in this table.
* The app fetches user-specific history based on the user\_id to display past calculations.

Table 3: Favorites

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| id | INTEGER | Primary key, auto-incremented identifier |
| user\_id | INTEGER | Foreign key linking to Users table |
| calculation\_id | INTEGER | Foreign key linking to Calculations table |
| favorited\_at | TEXT | Timestamp when the calculation was favorited |

**Purpose:** The Favorites table allows users to mark specific calculations as favorites, providing quick access to frequently referenced entries.

**Usage Scenario:**

* Users can mark a calculation as a favorite, and it will be added to this table.
* The app displays a list of favorited calculations on a separate page for easy access.

**3. Stretch Features**

If time permits or as part of future updates, consider adding the following enhancements:

* Analytics Table: Track user activity (such as most frequently used calculations) to provide insights and app suggestions.
* Data Export: Allow users to export their calculation history as a CSV or PDF file.
* Cloud Sync: Integrate with a cloud database for syncing user data across multiple devices.

**4. Normalization and Optimization**

The database design follows a normalized approach, separating user data, calculations, and favorites to avoid redundancy. Indexed columns (user\_id in Calculations and Favorites) will improve query performance, especially when fetching user-specific data.

**5. Integration with UI and Service Layer**

The data structures are closely integrated with the user interface and service endpoints:

* User Account Page: Interacts with the Users table for signup, login, and profile management.
* History Page: Fetches data from the Calculations table, displaying user-specific calculation history.
* Favorites Page: Retrieves data from the Favorites table for quick access to marked calculations.

**Example Data**

Users Table Example:

| **id** | **username** | **email** | **password\_hash** | **created\_at** |
| --- | --- | --- | --- | --- |
| 1 | jsmith | cfreeman@email.com | [hashed\_pw] | 2024-11-09T12:00:00 |

Calculations Table Example:

| **id** | **user\_id** | **type** | **voltage** | **current** | **resistance** | **result** | **calculated\_at** | **is\_deleted** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | Ohm's Law | 120 | 10 | 12 | Voltage (V) = 120 V | 2024-11-09T12:05:00 | false |

Favorites Table Example:

| **id** | **user\_id** | **calculation\_id** | **favorited\_at** |
| --- | --- | --- | --- |
| 1 | 1 | 1 | 2024-11-09T12:10:00 |

**Turn-In Comments**

* Changes Made: Expanded the database to include user accounts, favorites, and enhanced the MVP scope based on teacher feedback. Added new tables and optimized the existing structure.
* No Changes: None, as the document was significantly updated to align with the project expectations.