(COP3650) Mobile Application Development – Final Project Guidelines

Final Project Guidelines for Mobile Application Course

Students will build a complete mobile application using an **Expo-managed workflow**, incorporating key features such as data handling, real-time communication, and responsive navigation.

Core Requirements

1. **React Native Core & Hooks**
   - Use **React hooks** such as useState, useEffect, and useContext.
   - Build custom hooks for reusable logic, like server communication or form handling.
   - Make use of **Expo** APIs to simplify development (e.g., for permissions and media access).
2. **Navigation with React Navigation Native**
   - Set up **React Navigation Native** to manage nested navigation, including stack and tab navigators.
   - Use a **drawer navigator** if the application design requires it.
   - Structure navigation for easy access and smooth transitions between screens.
3. **File Uploads & Downloads**
   - Enable file uploads and downloads using **expo-document-picker** and **expo-file-system**.
   - Allow users to select files and view download progress or status within the app.
4. **Server Communication**
   - Fetch data from a remote server using the **Fetch API** or **axios**.
   - Manage loading and error states gracefully to improve user experience.
   - Integrate with **React Query** (optional) for handling frequent server requests and caching.
5. **Real-Time Data with Socket.IO**
   - Implement **real-time communication** using **Socket.IO** to support features like notifications or chat.
   - Configure real-time updates so data changes are displayed instantly without manual refresh.

6. **Form Validation with react-hook-form**
    - Use **react-hook-form** to build and manage forms, applying validation on input fields.
    - Integrate validation messages for real-time feedback on user input errors.
    - Customize form fields for easy entry and accessibility.
7. **Local Data Storage with SQLite**
    - Use **expo-sqlite** to store and manage local data, enabling offline access.
    - Allow users to save data locally and synchronize with the server when online.
    - Implement CRUD functionality for full control over stored data.
8. **Push Notifications with Expo Notifications**
    - Configure push notifications using **expo-notifications** for both foreground and background events.
    - Set permissions, handle user interactions, and ensure notifications are customizable in settings.
9. **File and Picture Handling**
    - Integrate **expo-image-picker** to enable image selection or capture with the device's camera.
    - Include options for basic editing (e.g., cropping) before upload.
    - Manage storage and display of images using **expo-file-system**.

Optional Features

- **Data Syncing and Conflict Handling**
    - Sync data between local SQLite storage and the server, handling conflicts (e.g., timestamp-based resolution).
- **Animations**
    - Use **React Native Reanimated** or **Animated API** for custom animations to enhance the UI.
    - Include feedback animations, screen transitions, or loading indicators where appropriate.
- **Error Handling & Logging**
    - Implement centralized error handling and logging, optionally using **Sentry** for tracking errors in production.

Evaluation Criteria

- **Code Quality**: Organization, modularity, and adherence to Expo and React Native practices.
- **Functionality**: Completeness of required features, including real-time updates and form validation.
- **UI/UX**: Quality of design, effective navigation, and responsiveness.
- **Data Handling**: Efficient data storage, caching, and synchronization.
- **Real-Time & Notifications**: Proper implementation of Socket.IO for real-time communication and seamless notifications.

These guidelines ensure a well-rounded, professional mobile application using Expo, Socket.IO, and efficient tools to manage navigation, real-time data, and data storage.