FPT Education
FPT UNIVERSITY

Assignemen 1: NoteApp

1. PROJECT OVERVIEW

Section	Content
Project Name	**NoteApp**
Objective	Build a note management application with features: CRUD (Create, Read, Update, Delete) notes, search notes by keyword, categorize notes by category, store data locally, display notes using FlatList, support infinite scroll and pull-to-refresh, and add animations (using react-native-reanimated) to enhance user experience. Manage state using Redux Toolkit and Context API.
Target Users	Users who want to manage personal notes, categorize them, with quick search functionality and a smooth interface.
Core Technologies	React Native + Expo, styled-components, Formik + Yup, AsyncStorage, FlatList, Redux Toolkit, Context API, react-native-reanimated.

2. MAIN FEATURES

- **Note Management**: Create, edit, delete notes with fields for title, content, and category.
- **Note Search**: Search notes by keyword (title or content).
- **Note Categorization**: Filter notes by category (e.g., Work, Personal, Ideas).
- **Local Storage**: Store notes using AsyncStorage.
- **List Display**: Use FlatList with infinite scroll and pull-to-refresh.
- **Animations**: Add transition effects when adding/deleting notes using react-native-reanimated.
- **State Management**: Use Redux Toolkit for global state and Context API for local state (e.g., current category).
- **Interface**: Responsive and visually appealing with styled-components and Flexbox.

3. PROJECT DIRECTORY STRUCTURE

```
NoteApp/
├── App.js                      # Main application file
├── assets/                     # Static assets (images, fonts, etc.)
├── components/                 # Reusable components
│   ├── NoteItem.js             # Component to display a single note
│   ├── NoteForm.js             # Component for note input form
│   ├── SearchBar.js            # Component for search bar
│   └── CategoryFilter.js       # Component for category filtering
├── contexts/                   # Context API for managing categories
│   └── CategoryContext.js
├── redux/                      # Redux configuration
│   ├── actions.js
│   ├── reducers.js
│   └── store.js
├── screens/                    # Main screens
│   ├── HomeScreen.js           # Main screen displaying the note list
│   └── EditNoteScreen.js       # Screen for editing a note
├── services/                   # Services for handling logic
│   └── storageService.js
├── styles/                     # Theme and style configuration
│   └── theme.js
├── package.json                # Dependency configuration
```

4. SUMMARY OF IMPLEMENTATION STEPS

1. Initialize the project with Expo.
2. Install required libraries (styled-components, Formik, Yup, AsyncStorage, Redux Toolkit, react-native-reanimated).
3. Build the basic UI with styled-components and Flexbox.
4. Create a form for adding/editing notes with Formik + Yup for validation.
5. Implement a search bar and category filter.
6. Store and retrieve note data with AsyncStorage.
7. Manage state with Redux Toolkit (for notes) and Context API (for categories).
8. Display the note list with FlatList, supporting infinite scroll and pull-to-refresh.
9. Add animations for adding/deleting notes using react-native-reanimated.
10. Write test cases and test on LDPlayer/Expo Go.

5. TEST CASES

No.	Test Description	Expected Outcome
1	Add a new note with title, content, and category	The note is added to the list, displayed in the correct category, and saved to AsyncStorage.
2	Search notes by keyword	The list only shows notes with titles or content matching the keyword.
3	Delete a note with animation	The note is removed from the list, the deletion animation (fade out) is smooth, and data is updated in AsyncStorage.