HW2 – Storage Management

Various approaches to storage management on your platform of choice (react-native) and their pros and cons-

1. **AsyncStorage**: a simple, asynchronous, unencrypted, persistent, key-value storage system that is global to the app.
Pros –
- Simple and easy to use, especially for small amounts of data like user preferences or settings.
- Built-in with React Native, requiring no additional libraries.
- Supports key-value pairs for data storage.

Cons-
- Limited storage capacity.
- Data is not encrypted, so not suitable for sensitive information.
- Not ideal for large or complex data structures.

2. **SQLite Databases:** For more complex data storage needs, React Native provides a bridge to interact with SQLite databases, allowing CRUD (Create, Read, Update, Delete) operations on structured data.
Pros:
- Offers persistent storage on the device, meaning data survives app restarts.
- Stores data in a structured format (tables with rows and columns).
- More efficient for larger datasets compared to AsyncStorage.

Cons:
- Requires additional libraries like react-native-sqlite-storage.
- Setting up and querying databases involves more complexity compared to AsyncStorage.

3. **Realm (NoSQL):** Realm is a mobile database that is suitable for use in React Native applications. It's a fast, easy-to-use alternative to SQLite and AsyncStorage. Realm provides an object-oriented API for working with data, making it easier to work with complex data models. It also offers features like data synchronization, encryption, and cross-platform support.
Pros:
- Flexible and efficient NoSQL database with a focus on mobile development.
- Offers object-oriented data modeling, making it easier to map your app's data structures.
- Supports offline functionality with local data synchronization.

Cons:
- Requires integrating the realm library.
- Might have a steeper learning curve compared to simpler options.

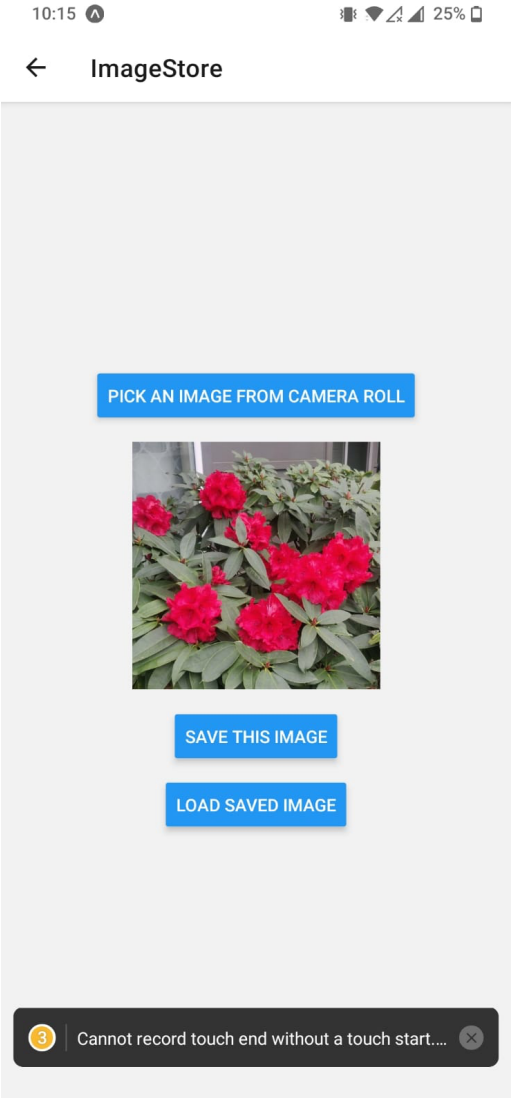
4. **Firebase Realtime Database / Firestore**: Firebase provides cloud-based solutions for data storage, including the Realtime Database and Firestore.
Pros:
- Cloud-based storage solution, allowing data access from multiple devices.
- Offers various features like authentication, real-time data synchronization, and more.
- Scalable solution for handling large datasets and user bases.

Cons:
- Requires setting up a Firebase project and managing backend services.
- Introduces network dependency for data access (though caching is available).

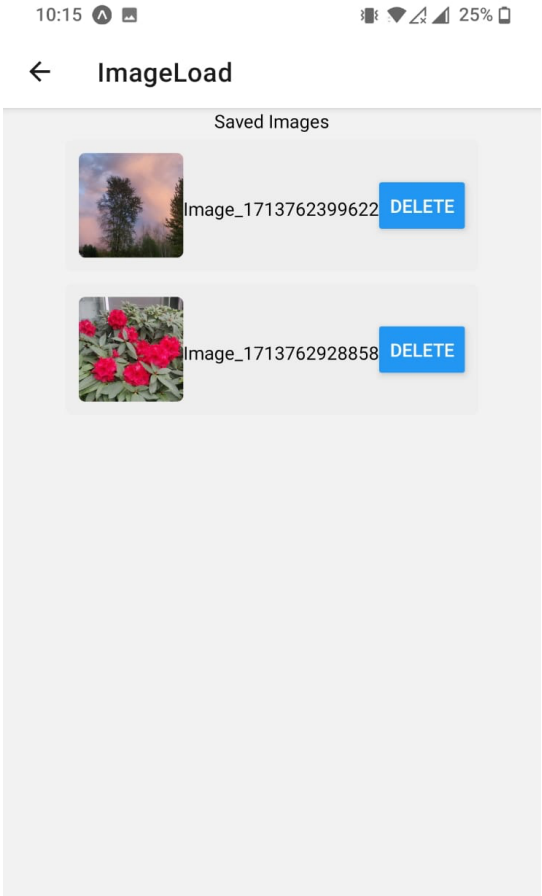
- May incur costs for extensive use depending on your chosen Firebase plan.

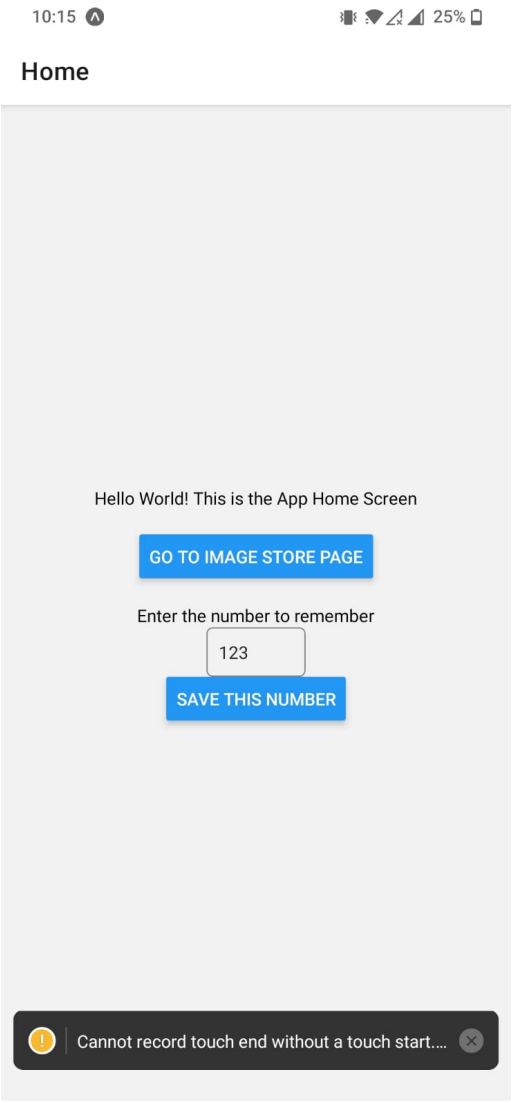
For my project (AI powered story teller application), a combination of Async storage and database storage can be used. Async storage can be used to store user data and preferences, like the font and background they select for reading the stories. Database or cloud storage needs to be used to store the stories themselves as stories with pictures will occupy more space, and storing them in. shared database also allows for sharing them amongst various users.

For HW2 part 2 – implementation of store and load media item in app, async storage has been used.



-> The first button opens the user's gallery for user to pick an image to load. 'Save this image' saves the user's image in the app.
-> 'Load saved image' button opens up the next page showing all the images stored by user in the app.





The numeric input box acts as a user preference, for example to store their mobile number, age etc. in app. ‘Save this number’ saves it within the app so when user reopens the app the saved number will be loaded.