Darshan Chauhan (1276346)

EXERCISE 2: YOUR REACT NATIVE MESSAGES DIRECTORY

CS5450 MOBILE PROGRAMMING

DR. SABAH MOHAMMED

DEPARTMENT OF COMPUTER SCIENCE

Github Link - https://github.com/DarshanChauhan22/Exercise2

Overview

- The Message Directory App is a React Native application designed to organize messages into distinct "directories" or categories. Each directory (e.g., "Vin", "Brain", "Tezz") has its own icon, color, and a list of associated messages. Users can view messages within a directory, send new messages to other directories (or to the same directory as a note-to-self), and delete existing messages. The app demonstrates several key React Native concepts, including navigation, state management with React Context, dynamic styling, and platform-specific UI adjustments.

Features

Directory Listing:

- Displays a grid of available message directories, each with a unique icon and color.
- Responsive grid layout: 2 columns on mobile, 3 columns on web. Dynamically sized icons for better visual appeal across screen sizes.

Message Viewing:

- Tapping a directory opens a screen showing all messages associated with it. Messages are sorted by timestamp.
  Distinguishes between messages sent from the current directory ("my messages") and messages received by or sent to the current directory from others.
- Displays sender and recipient names (if applicable).
  Color-coded message bubbles for "my messages" match the directory color.

- Send Messages:
  Users can compose and send new messages from the current directory to any other directory (including itself).
- A picker allows selection of the recipient directory.
  Input validation (prevents sending empty messages or messages without a recipient).

Delete Messages:

- Users can delete individual messages with a confirmation prompt.
- State Management: Uses React Context (MessageDataContext) to manage directory and message data globally, ensuring instant UI updates when data changes (e.g., new message sent, message deleted).

Navigation:

- Uses React Navigation (Stack Navigator) to move between the directory list and individual message screens.
- Screen headers are dynamically styled (title and background color) based on the current directory.

Platform Awareness:

- Adjusts the number of columns in the directory grid for web vs. mobile. Uses KeyboardAvoidingView with platform-specific behavior for a better text input experience.
- Safe area views for proper layout on devices with notches/islands.

Technologies Used:

- React Native: Core framework for building cross-platform mobile (and web) applications.
- React: JavaScript library for building user interfaces.
- React Navigation: For handling navigation between screens.
  @react-navigation/native
  @react-navigation/stack
  @react-navigation/elements (for useHeaderHeight)
- React Native SafeArea Context: For handling safe area insets on different devices.
- React Native Picker: For the recipient selection dropdown.
  @react-native-picker/picker
- JavaScript : Programming language.

Project Structure (App.js)

- The entire application logic is currently contained within MessageDirectoryApp/App.js.
- MessageDirectoryApp/
  App.js Main application component, screens, context, and logic
  assets/
  i1.png Icon for 'Vin' directory
  i2.png Icon for 'Brain' directory
  i3.png Icon for 'Tezz' directory
  i4.png Icon for 'Sam' directory
  i5.png Icon for 'Jason' directory
  i6.png Icon for 'Latty' directory
- Key Components and Logic within App.js:

1. MessageDataContext (React Context):
    - Created using createContext(). Provides messages, directories, addMessageToList, and deleteMessageFromList to consuming components.

2. rawInitialDirectoriesData (Data):
    - An array of objects defining the initial state of directories and their sample messages.
    - Includes id, name, icon (local image require), color, and an array of messages.

3. Sizing Constants (for DirectoryScreen):
    - Calculations for numColumns, baseItemWidth, and iconCircleDiameter to make the directory grid responsive.

4. DirectoryScreen (Component):
    - Consumes MessageDataContext to get the list of directories. Renders directories in a FlatList with a platform-dependent number of columns.
    - Each item is a TouchableOpacity that navigates to the MessagesScreen for that directory. Displays directory icon (dynamically sized) and name.

5. MessagesScreen (Component):
    - Receives directoryId, directoryName, and color as route parameters.
    - Consumes MessageDataContext to get all messages, directories (for recipient/sender names), addMessageToList, and deleteMessageFromList functions.

- State: Manages input text, keyboardVisible status, and selectedRecipientId.
- Message Display: Uses useMemo to efficiently filter and sort messages relevant to the current directoryId.
- Renders messages using FlatList. Styles messages differently based on whether they are "sent by me" or "sent by others".
- Shows sender/recipient names and timestamp. Includes a delete button for each message.
- Input Area:
  KeyboardAvoidingView for better UX when the keyboard is open. A Picker component to select the recipient directory.
- A TextInput for composing messages. A "Send" TouchableOpacity that calls addMessageToList.
- Functionality:
  onSend(): Creates a new message object and adds it to the context.
  handleDeleteMessage(): Shows an Alert confirmation and then calls deleteMessageFromList from the context.

6. App (Main Component):
   - State: directoriesData: Derived from rawInitialDirectoriesData (currently just names, icons, colors).
   - messagesData: Initialized by flattening rawInitialDirectoriesData.messages and adding senderId, recipientId, and timestamp. Sorted by timestamp.
   - Context Provider: Wraps the navigation structure with MessageDataContext.Provider, passing down the state and updater functions (addMessageToList, deleteMessageFromList).
   - Navigation Setup: Uses NavigationContainer and createStackNavigator.

- Defines two screens: Directories and Messages.
  Sets global header styles and dynamic header styles for the MessagesScreen (title and color based on the directory).
- State Updaters: addMessageToList(newMessage): Appends a new message to messagesData and re-sorts.
  deleteMessageFromList(messageId): Filters out the message with the given ID from messagesData.

7. styles (StyleSheet):
   - A comprehensive StyleSheet.create object containing styles for all components, promoting organization and reusability.

How to Run

1. Prerequisites:
   Node.js and npm
   Expo CLI (npm install -g expo-cli) or a physical device.

2. Clone the repository and navigate to the project directory.

3. Install dependencies:
   npm install

4. Ensure assets are in place:
   Make sure the assets folder with i1.png through i6.png is in the MessageDirectoryApp root .

5. Start the development server:
   For Expo projects:
   expo start

Then scan the QR code with the Expo Go app on your device, or press a for Android emulator / i for iOS simulator.
For React Native CLI projects:
npx react-native start
In a new terminal:
npx react-native run-android
or
npx react-native run-ios

Code Highlights & Key Concepts:

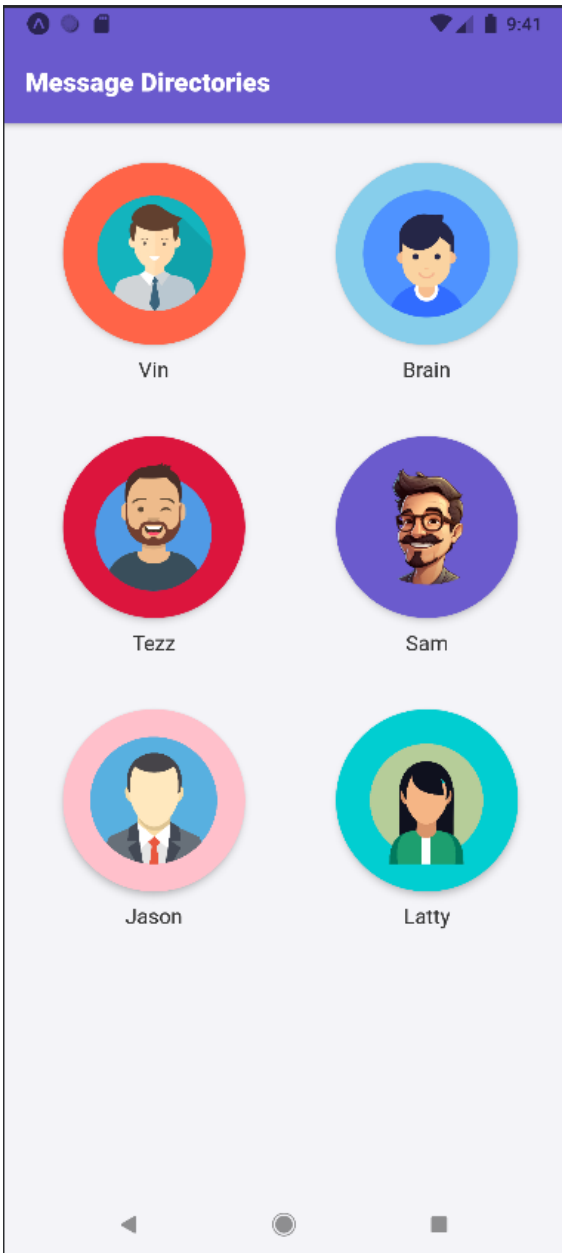
Responsive Design:
Dimensions.get('window').width is used to calculate item sizes in DirectoryScreen.
Platform.OS checks adapt UI elements (e.g., numColumns, KeyboardAvoidingView behavior, Picker height).

Dynamic Styling:

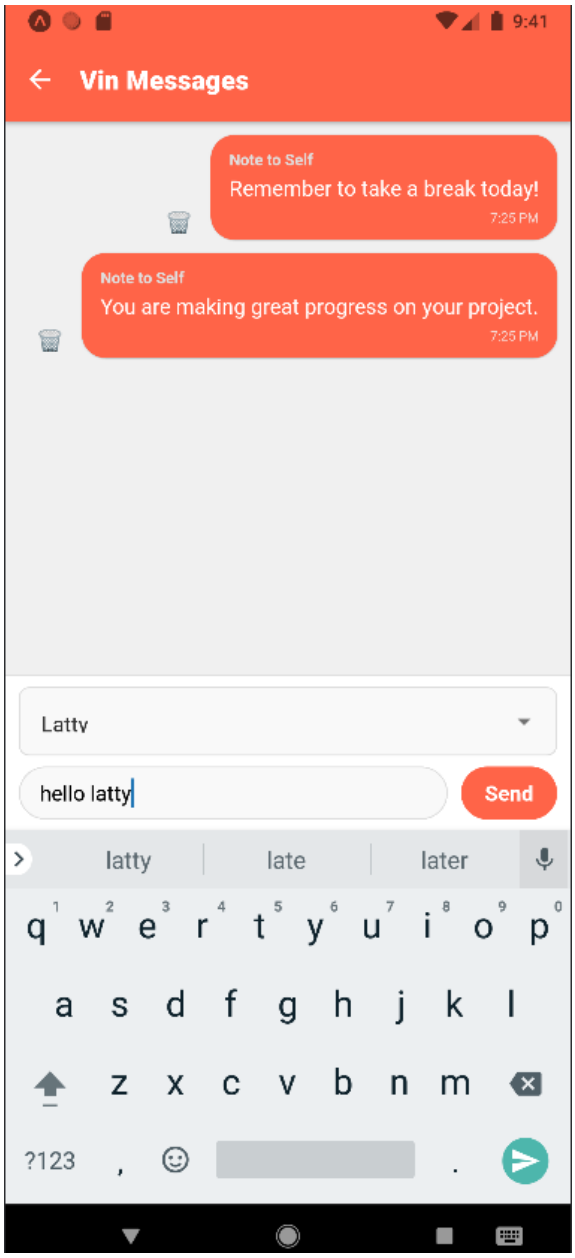
Header colors in MessagesScreen and message bubble colors are determined by the directory's color prop.

User Experience:
KeyboardAvoidingView improves usability of the text input on mobile.
SafeAreaView and useSafeAreaInsets ensure content is not obscured by system UI elements.
Confirmation dialogs for destructive actions (deleting messages).
Disabled state for the send button when input is empty or no recipient is selected.

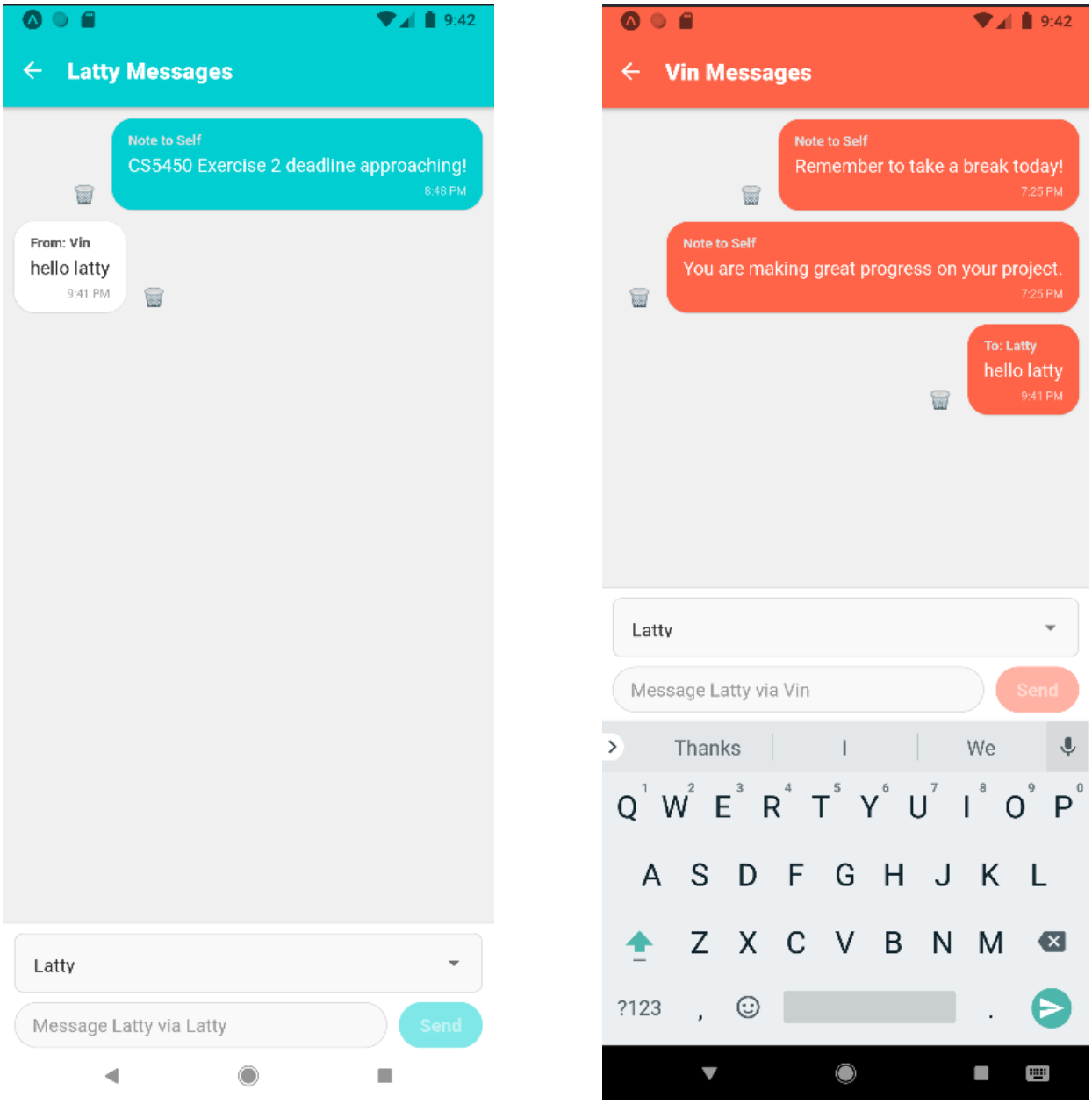
Mobile Output:



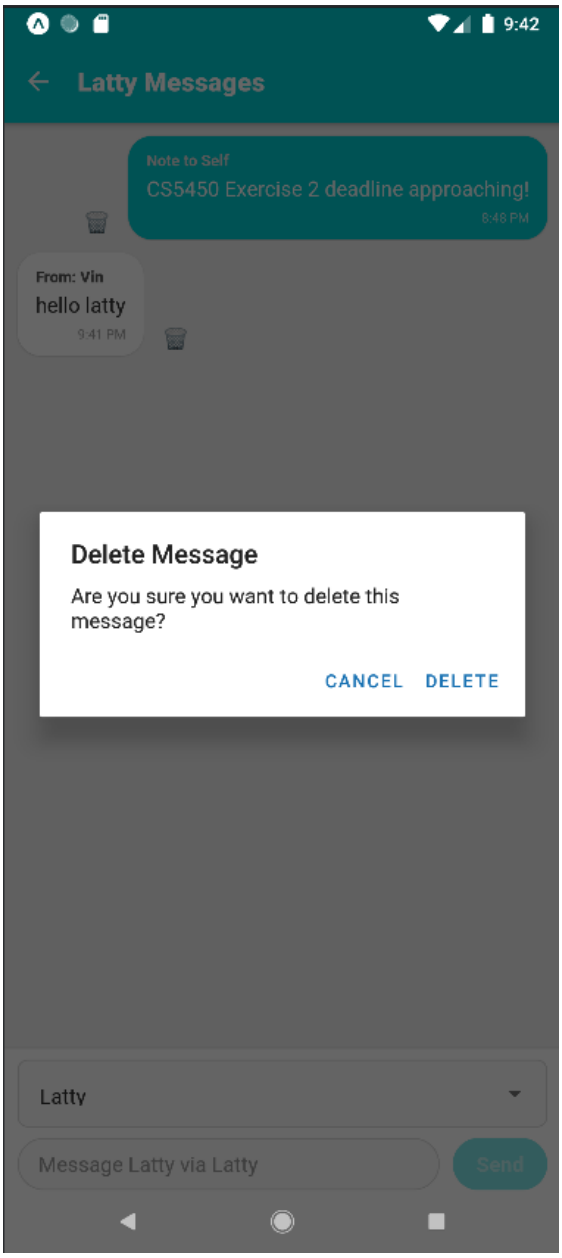
Home Page



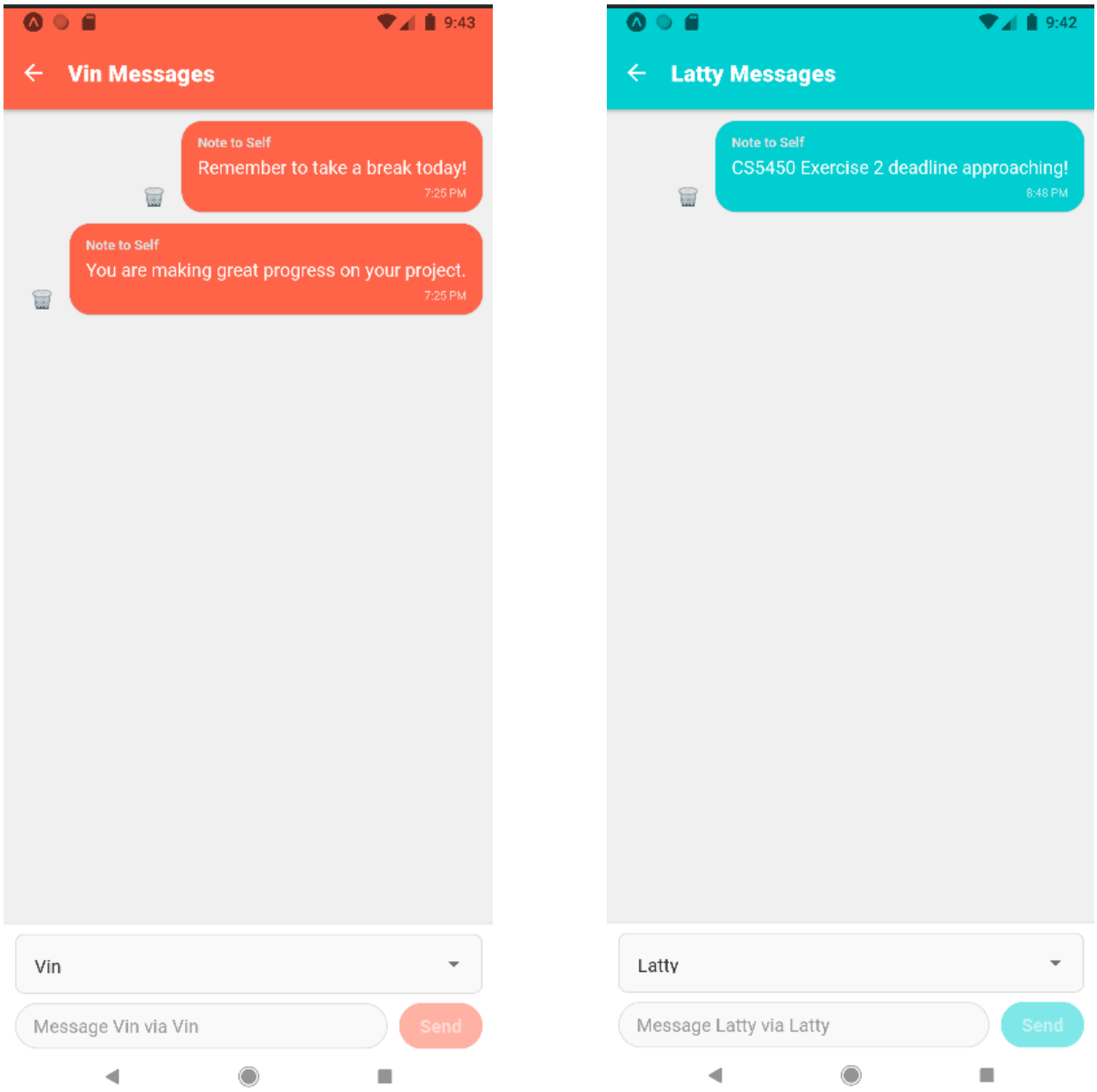
Massage Screen



Massage to each other

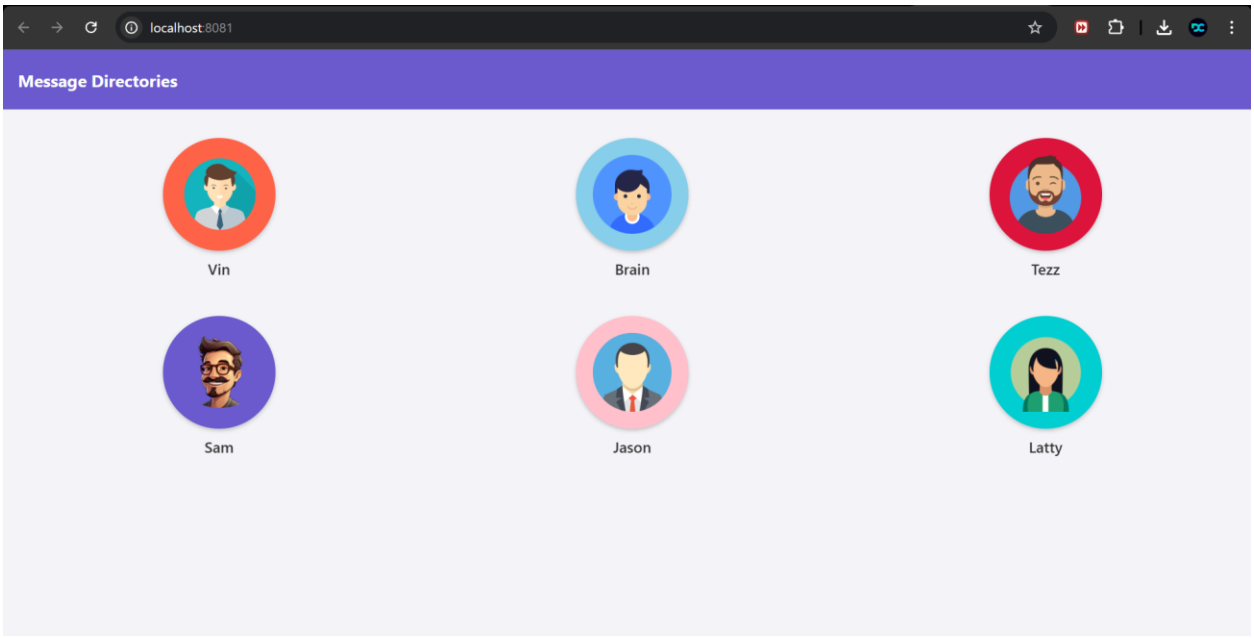


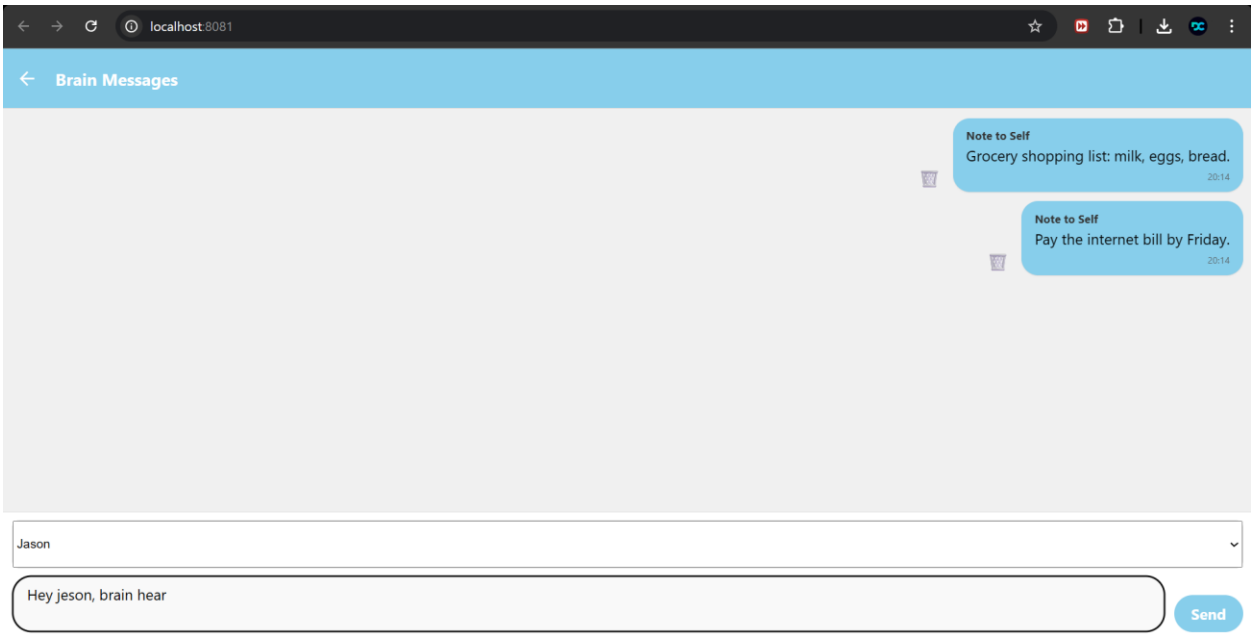
Delete massage

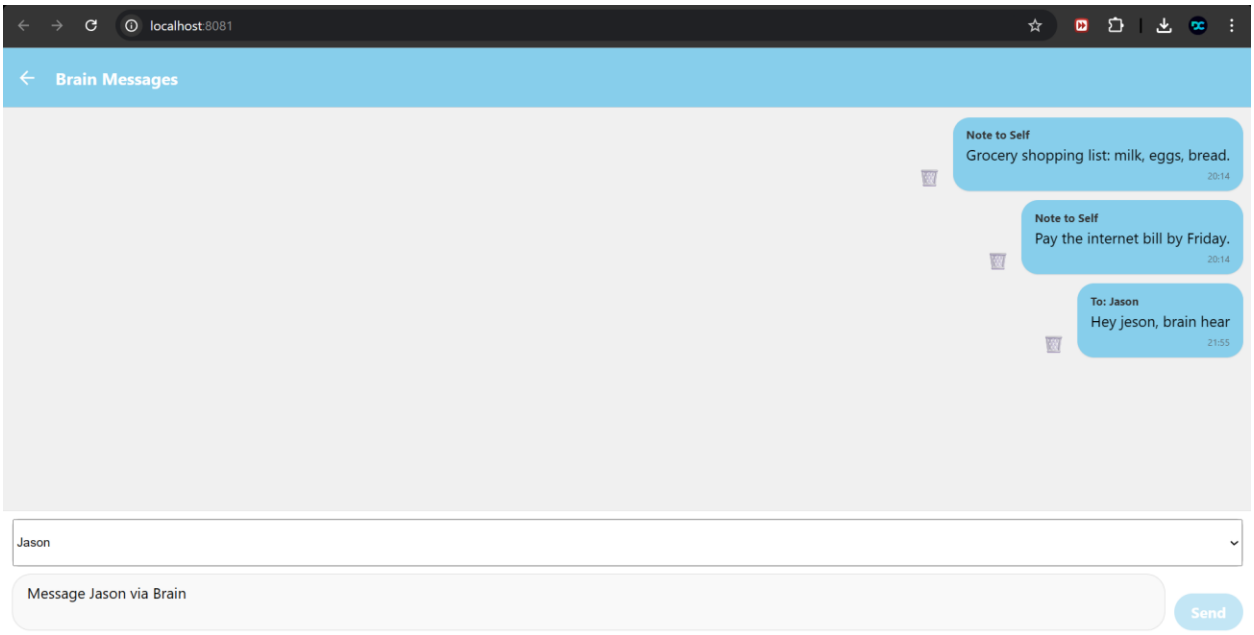


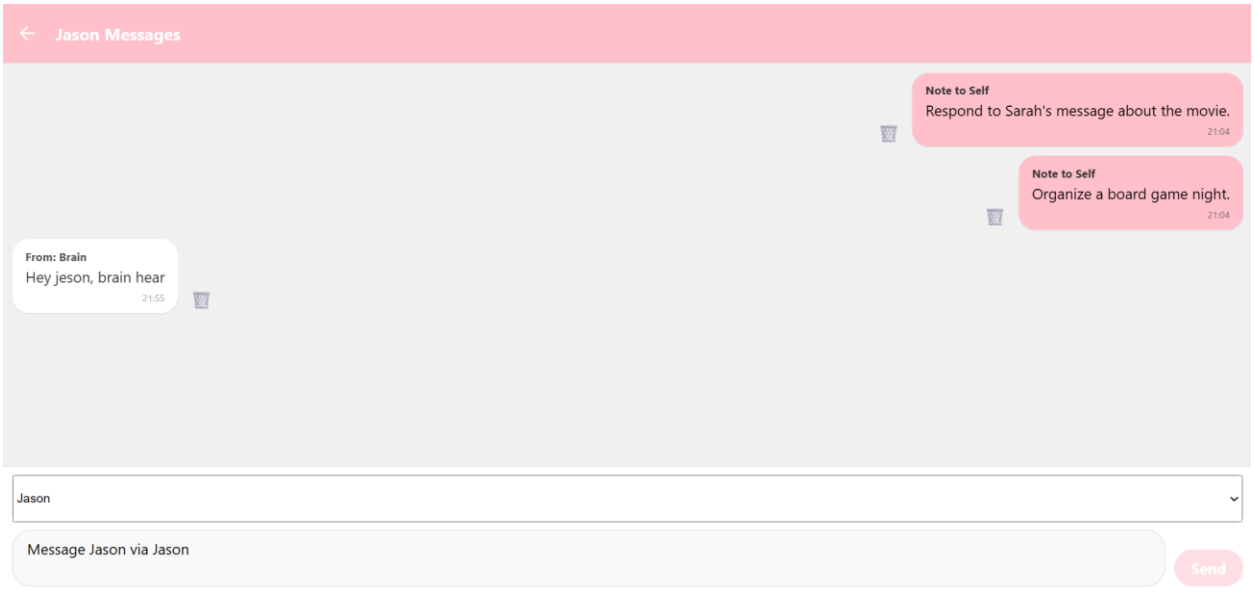
Massage Deleted from both chat

Desktop Output:






localhost:8081
Brain Messages
Note to Self
Grocery shopping list: milk, eggs, bread.
20:14
Note to Self
Pay the internet bill by Friday.
20:14
To: Jason
Hey jeson, brain hear
21:55
Jason
Message Jason via Brain
Send


Jason Messages
Note to Self
Respond to Sarah's message about the movie.
21:04
Note to Self
Organize a board game night.
21:04
From: Brain
Hey jeson, brain hear
21:55
Jason
Message Jason via Jason
Send

Conclusion:

The Message Directory App shows how to make a React Native application that sorts messages into clearly marked directories. It efficiently highlights main development features such as react context for handling state, easy screen navigation with React Navigation, a responsive design working on both web and mobile and styling that makes the app more interesting to users.