RehabTrack

Technical Documentation

Tool for Disability Rehabilitation

Authors:  
Sean Fowers

Zhendong Su (Chris)

9/10/2023

# Table of Contents

[**Table of Contents 1**](#_u5f0g0yim710)

[**Setting Up the RehabTrack React Native Project 5**](#_b0qie0g0ph77)

[Prerequisites 5](#_fyneem30mrcq)

[1. Clone the Project Repository 5](#_vaojt4wbhxkt)

[2. Node.js and npm (or Yarn) 5](#_dnv5i9nzx788)

[3. Install Expo CLI 6](#_3va8dyshig87)

[4. Install Dependencies 6](#_jq9so5k1y5ip)

[5. Start the Development Server 6](#_apsqujduslah)

[6. Testing on Mobile Devices 7](#_5gscoudw6p3i)

[Expo Go App (Recommended) 7](#_e7ied0vq4jin)

[Emulators 7](#_uki8ezd7kkqt)

[7. Code Editing 7](#_ug437aaxaj9d)

[Conclusion 7](#_w7no97fh6sbx)

[**Navigation Documentation 8**](#_5t66hqlup51q)

[Introduction 8](#_wz7q43z35alt)

[Navigation Structure 8](#_z0ccxo5g1m42)

[Navigation Container 8](#_j3dz4up615mh)

[Navigator Stack 8](#_vvvmhedk9syy)

[Screen Definitions 9](#_puaxqfj6ot9o)

[Screens 9](#_69ayp3c6qx7q)

[1. Home Screen 9](#_ysxyic6f8vcp)

[2. New Session Screen 9](#_jsmxkj90pb0o)

[3. Patient Records Screen 9](#_o2o2p2eruovl)

[4. Settings Screen 10](#_69c9lcildm84)

[5. Repetitions Screen 10](#_bvfoghq6a1d)

[6. Patient Screen 10](#_dkvkuh2szcss)

[7. Session History Screen 10](#_vseuz3829qxr)

[8. Session Screen 10](#_evdk23jttbl0)

[9. Graphing Screen 11](#_on3m01endvpb)

[10. Data Export Screen 11](#_8yr7f47hiv8g)

[Conclusion 11](#_lgpa72i1m7cn)

[**HomeScreen Component Documentation 11**](#_belc4yvzu5a6)

[Introduction 12](#_y9v0yen0tamh)

[Component Structure 12](#_y2jshj3db4iv)

[Navigation Buttons 13](#_b5shnviwfnv1)

[1. New Session 13](#_rpfkg0qrcxb4)

[2. Patient Records 13](#_uvxcm34sned6)

[3. Data Export 13](#_erunt55g762q)

[4. Settings 13](#_i6sbgf2ebric)

[StatusBar 13](#_ds7q99smcndj)

[Styles 14](#_k6bmd3vfebxk)

[Conclusion 14](#_6g6nvrvh0agj)

[**New Session Screen 15**](#_9zhdn2js2cx1)

[Overview 15](#_jka50hvkrdlm)

[Dependencies 15](#_gln5sh258t8b)

[Components 15](#_g26tof3rqg11)

[NewSessionScreen Component 15](#_a6kje0nqumnx)

[Styling 15](#_8n0h2ftt2yh4)

[Usage 16](#_4su1mlgepbvf)

[Notes 16](#_a38xgt3hulvj)

[**Patient Records Screen 17**](#_wzvtnjmxbi1m)

[Overview 17](#_fr0mq97ynnrk)

[Dependencies 17](#_mc4iy9ttzw9)

[Components 17](#_kxwssb3ojzgm)

[PatientRecordsScreen Component 17](#_oirkjvn7dew5)

[Styling 17](#_1ew5tka89o5v)

[Usage 18](#_fkydg8yhejb1)

[Notes 18](#_sskt8fmo7wtb)

[**Settings Screen 19**](#_3zqkexzf8zu0)

[Overview 19](#_s2wzcef433kc)

[Dependencies 19](#_tbzra1jl4h5m)

[Components 19](#_5tuvldsacre4)

[SettingsScreen Component 19](#_19k334cryprr)

[Styling 19](#_bazw7ynj83qc)

[Usage 19](#_6dl7c6dnu1aj)

[Notes 20](#_qofvvp3s2twl)

[**Repetitions Screen 21**](#_60faznvc3lrn)

[Overview 21](#_kvae9s73vetf)

[Dependencies 21](#_vksqm3yk1uwt)

[Components 21](#_2djqrpc3s0pf)

[RepetitionsScreen Component 21](#_okgmqxqs8ju9)

[Styling 21](#_sy6wj75r7fgt)

[Usage 22](#_8v21tcwo0wa)

[Notes 22](#_radbfonssodd)

[**Patient Screen 23**](#_dpfe8td5xk2)

[Overview 23](#_mpkpa5wwp8bo)

[Dependencies 23](#_pfk9kdrztnm)

[Components 23](#_y5488xxuomr8)

[PatientScreen Component 23](#_951wx373tg46)

[Styling 23](#_iwbxkeiuud8t)

[Usage 24](#_jgkqyyq6cx2i)

[Notes 24](#_v3pg7jwfzpy3)

[**Session History Screen 25**](#_9tqyl8igclkl)

[Overview 25](#_v3uquv2mp8jj)

[Dependencies 25](#_nijncocwe8xe)

[Components 25](#_ii6cbipxx8w4)

[SessionHistoryScreen Component 25](#_z4ti8ka9ge19)

[Styling 25](#_kwwryg9poenq)

[Usage 25](#_gmfa5g6zr2um)

[Notes 26](#_f7dvuqymfcur)

[**Session Screen 27**](#_ujpbd8o651l4)

[Overview 27](#_8tttijb798tb)

[Dependencies 27](#_ayr3yk1700ak)

[Components 27](#_mift9tezgz1g)

[SessionScreen Component 27](#_bv91bg4enkef)

[Styling 27](#_rs00w691jh6d)

[Usage 27](#_s0n4f8puu42w)

[Notes 28](#_8y6qqf4rzoms)

[**Graphing Screen 29**](#_qujd60j9awbp)

[Overview 29](#_4p8vrg1sm7oy)

[Dependencies 29](#_4xum986gtr9v)

[Components 29](#_tynfpma6x87)

[GraphingScreen Component 29](#_nefrb93vmcuw)

[Styling 29](#_54ibflg5g6vy)

[Usage 30](#_4yt2x43zvrau)

[Notes 30](#_w0l4gnlhsh7t)

[**Data Export Screen 31**](#_7rpolckrsye2)

[Overview 31](#_uqxhdrm4ivxo)

[Dependencies 31](#_encb25qrawnp)

[Components 31](#_7zfe82ehp5rq)

[DataExportScreen Component 31](#_8y09pkajnvk0)

[Styling 32](#_ioguqtokhm19)

[Usage 32](#_oogu4n2yindm)

[Notes 32](#_s1or3lsbsxnm)

[**Patient Data Structures 33**](#_5dd1eaul3lc)

[Overview 33](#_n1kiwol88mf)

[Classes 33](#_qemyvzbjryqi)

[PatientData Class 33](#_ls8j2gd852i)

[Constructor 33](#_mm2snd1vc41w)

[Methods 33](#_k7ioprrwri77)

[Patient Class 33](#_lh6gu8uc8pf0)

[Constructor 33](#_w12anba634j3)

[Methods 34](#_xytf8ooymegm)

[Session Class 34](#_alb9k64obfmo)

[Constructor 34](#_el7hsc4bx4l)

[Methods 34](#_wqg466p02muc)

[Exercise Class 34](#_f7xqjt61lcy8)

[Constructor 35](#_ze0mh148cgm1)

[Usage 35](#_mvren1ba8lkz)

[**Other Resources 36**](#_syyeh47lza71)

### Project Git Repository:

<https://github.com/SFowers/RehabTrack>

# Setting Up the RehabTrack React Native Project

Welcome to the RehabTrack React Native project setup guide. In this guide, we will walk you through the steps to set up your development environment for the RehabTrack project using Expo.

## Prerequisites

Before getting started, make sure you have the following prerequisites installed on your machine:

* **Operating System:** macOS, Windows, or Linux (Choose the one that's appropriate for your system)
* **Node.js:** Installed globally on your system
* **npm or Yarn:** Package manager
* **Git:** Version control system
* **Code Editor:** Visual Studio Code or your preferred code editor

## 1. Clone the Project Repository

First, you need to clone the RehabTrack project repository from GitHub to your local machine. Open your terminal or command prompt and run the following command:

git clone https://github.com/SFowers/RehabTrack.git

This will download the project's source code to your computer.

## 2. Node.js and npm (or Yarn)

* Download and install Node.js from the official website: [Node.js Downloads](https://nodejs.org/).
* Verify the installation by running the following commands in your terminal:

node -v

npm -v

or if using Yarn:

yarn -v

## 3. Install Expo CLI

Expo is a powerful toolchain and library for building React Native applications. To work with the RehabTrack project, you'll need Expo CLI. Install Expo CLI globally using npm or Yarn by running one of the following commands:

npm install -g expo-cli

or

yarn global add expo-cli

## 4. Install Dependencies

Open a terminal at the projects file directory, or navigate to the project directory in the terminal with:

cd RehabTrack

Install project dependencies using Yarn or npm:

yarn install

or

npm install

This will download and install all the required libraries and packages.

## 5. Start the Development Server

You are now ready to start the development server. Run the following command:

npx expo start

Expo will open a web interface in your default web browser. This interface allows you to run the app on different platforms, including Android, iOS, and web.

## 

## 6. Testing on Mobile Devices

### Expo Go App (Recommended)

For easy testing on mobile devices, we recommend using the Expo Go app. You can install Expo Go from the app store on your Android or iOS device:

* [Expo Go on Google Play](https://play.google.com/store/apps/details?id=host.exp.exponent&pcampaignid=web_share)
* [Expo Go on the App Store](https://apps.apple.com/us/app/expo-go/id982107779)

Once installed, use the Expo Go app to scan the QR code displayed in your computer's terminal (from npx expo start). This will load the RehabTrack app onto your mobile device.

### Emulators

If you prefer to use emulators for testing, you can set up Android emulators using Android Studio for Windows and Linux, or Xcode for macOS.

* [Install Android Studio](https://developer.android.com/studio)
* [Install Xcode (macOS only)](https://developer.apple.com/xcode/)

Follow the respective installation guides for Android Studio and Xcode. Once installed, launch your preferred emulator from the Android Studio AVD Manager (for Android) or Xcode (for iOS).

## 7. Code Editing

We recommend using Visual Studio Code or your preferred code editor for working on the project. Install relevant extensions for React Native development in your code editor to enhance your workflow.

## Conclusion

You have successfully set up the RehabTrack React Native project on your development environment. Now, you can start contributing to the project, making improvements, and testing it on your mobile device or emulator. Enjoy your development journey!

# Navigation Documentation

## Introduction

This technical documentation provides an overview of the navigation structure and screens used in a React Native application. The application uses the [React Navigation](https://reactnavigation.org/) library to manage navigation between different screens. It includes various screens for managing patient records, sessions, settings, data export, and more.

## Navigation Structure

### Navigation Container

The main navigation structure is wrapped in a NavigationContainer component from @react-navigation/native. This container is responsible for managing the navigation state and rendering the appropriate screens based on the current route.

jsx

<NavigationContainer>

{/\* Navigator and screen definitions go here \*/}

</NavigationContainer>

### Navigator Stack

Within the NavigationContainer, a createNativeStackNavigator is used to create a stack-based navigation structure. The stack navigator manages a stack of screens, allowing users to navigate back and forth between them.

jsx

const Stack = createNativeStackNavigator();

### Screen Definitions

Each screen in the application is defined using the Stack.Screen component provided by the stack navigator. Screens are associated with components and can have custom options, such as screen titles.

jsx

<Stack.Screen  
 name="Home"  
 component={HomeScreen}  
 options={{ title: "RehabTrack" }}  
/>

## Screens

### 1. Home Screen

* **Route Name:** "Home"
* **Component:** HomeScreen
* **Options:** Title set to "RehabTrack"

The home screen serves as the main entry point for the application. It provides an overview of available actions and navigation links to other parts of the app.

### 2. New Session Screen

* **Route Name:** "New Session"
* **Component:** NewSessionScreen
* **Options:** Title set to "Create New Session"

This screen allows users to create a new session for a patient. It provides input fields for session details and exercise repetitions.

### 3. Patient Records Screen

* **Route Name:** "Patient Records"
* **Component:** PatientRecordsScreen
* **Options:** Title set to "Patient Records"

The patient records screen displays a list of patients and allows users to view and manage patient information.

### 4. Settings Screen

* **Route Name:** "Settings"
* **Component:** SettingsScreen
* **Options:** Title set to "Settings"

The settings screen provides options for configuring the application, including clearing data and adding dummy patient data.

### 5. Repetitions Screen

* **Route Name:** "Repetitions"
* **Component:** RepetitionsScreen
* **Options:** Title set to "Edit Repetitions"

This screen allows users to edit the repetitions for a specific exercise session.

### 6. Patient Screen

* **Route Name:** "Patient"
* **Component:** PatientScreen
* **Options:** Title set to "Patient"

The patient screen displays detailed information about a specific patient, including session history and exercises.

### 7. Session History Screen

* **Route Name:** "Session History"
* **Component:** SessionHistoryScreen
* **Options:** Title set to "Session History"

The session history screen displays a history of sessions for a selected patient, including exercise details.

### 8. Session Screen

* **Route Name:** "Session"
* **Component:** SessionScreen
* **Options:** Title set to "Session"

The session screen provides detailed information about a specific exercise session, including repetitions and exercise details.

### 9. Graphing Screen

* **Route Name:** "Graphing"
* **Component:** GraphingScreen
* **Options:** Title set to "Graphing Screen"

The graphing screen allows users to visualize and analyze exercise data through graphs and charts.

### 10. Data Export Screen

* **Route Name:** "Data Export"
* **Component:** DataExportScreen
* **Options:** Title set to "Data Export"

The data export screen enables users to export exercise data for further analysis or sharing.

## Conclusion

This technical documentation outlines the navigation structure and screens of the React Native application. The use of React Navigation simplifies the process of navigating between screens, making it easier to manage the flow of the application. Each screen serves a specific purpose in managing patient records and exercise data.

# HomeScreen Component Documentation

## Introduction

This technical documentation provides an overview of the HomeScreen component in a React Native application. The HomeScreen serves as the main entry point of the application and provides navigation buttons to access different features and screens.

## Component Structure

The HomeScreen component is designed to be a user-friendly interface for easy navigation to various parts of the application. It includes several navigation buttons, each associated with a specific action or screen.

jsx

import React from 'react';  
import { StatusBar } from 'expo-status-bar';  
import { TouchableOpacity, Text, View } from 'react-native';  
import Icon from 'react-native-vector-icons/AntDesign';  
import { styles } from '../stylesheet/Style';

export default function HomeScreen({ navigation }) {

return (  
 <View style={styles.container}>  
 <StatusBar style="auto" />  
 {/\* Navigation Buttons \*/}  
 {/\* New Session \*/}  
 {/\* Patient Records \*/}  
 {/\* Data Export \*/}  
 {/\* Settings \*/}  
 </View>  
 );  
}

## Navigation Buttons

The HomeScreen component features several navigation buttons, each represented by a TouchableOpacity element. These buttons allow users to navigate to different screens and access specific features of the application. Here are the navigation buttons:

### 1. New Session

* **Icon:** Plus sign icon
* **Label:** "New / Edit Session"
* **Action:** When pressed, it navigates the user to the "New Session" screen using navigation.navigate("New Session").

### 2. Patient Records

* **Icon:** Open folder icon
* **Label:** "Patient Records"
* **Action:** When pressed, it navigates the user to the "Patient Records" screen using navigation.navigate("Patient Records").

### 3. Data Export

* **Icon:** Export icon
* **Label:** "Data Export"
* **Action:** When pressed, it navigates the user to the "Data Export" screen using navigation.navigate("Data Export").

### 4. Settings

* **Icon:** Settings icon
* **Label:** "Settings"
* **Action:** When pressed, it navigates the user to the "Settings" screen using navigation.navigate("Settings").

## StatusBar

At the top of the HomeScreen, a StatusBar component is used to manage the status bar's appearance. The StatusBar component ensures that the status bar content appears correctly based on the user's device settings.

jsx

<StatusBar style="auto" />

## Styles

The styling for the HomeScreen component is defined using the styles imported from an external stylesheet. These styles control the layout, appearance, and spacing of the navigation buttons and other elements on the screen.

## Conclusion

The HomeScreen component provides an intuitive and accessible user interface for users to navigate the application's various features and screens. It serves as the entry point for the app, helping users quickly access the functionalities they need. The navigation buttons are visually represented by icons and labels, making it easy for users to understand their purpose and take action.

# 

# New Session Screen

## Overview

The New Session Screen is a component of a mobile application that facilitates the creation and management of new patient sessions. It allows users to select a patient, initiate a new session, and add exercises to the session. The screen also provides functionalities for editing and saving sessions, with data persistence achieved through AsyncStorage.

## Dependencies

* React
* React Native
* AsyncStorage from '@react-native-async-storage/async-storage'
* DropDownPicker from 'react-native-dropdown-picker'
* uuidv4 from 'uuid'
* RepetitionsScreen component
* PatientData, Patient, Session, Exercise structures from '../patientdata/patientDataStructures'

## Components

### NewSessionScreen Component

This component is responsible for creating and managing new sessions for patients. It incorporates various functionalities including:

* **Patient Selection:** Utilizes DropDownPicker to select a patient from a list.
* **Session Management:** Allows the creation of new sessions and the selection of existing ones.
* **Exercise Addition:** Enables the addition of exercises to a selected session.
* **Data Persistence:** Uses AsyncStorage to save and load patient data.
* **Modal Utilization:** Employs modals for session and exercise management.

### Styling

The component employs styles defined in an external stylesheet (styles.js) to ensure a consistent and user-friendly interface.

## Usage

The New Session Screen is used for adding new sessions to an existing patient, and for adding exercise to that specific session. Navigate to the New Session Screen from the Home Screen.

## Notes

* Ensure that all dependencies are installed and properly configured.
* The component can be customized to fit specific design and functional requirements.
* Patient data is stored and retrieved from AsyncStorage, ensure proper handling and privacy considerations.

# 

# Patient Records Screen

## Overview

The Patient Records Screen is a part of a mobile application that allows users to manage patient data. This screen provides functionality to view a list of patients, add new patients, and navigate to individual patient profiles.

## Dependencies

* React
* React Native
* React Navigation
* React Native Vector Icons
* React Native Dropdown Picker
* React Native AsyncStorage

## Components

### PatientRecordsScreen Component

* This component represents the main screen for managing patient records.
* It uses various state variables to manage the screen's state, including patient data, modal visibility, and selected patient information.
* The component consists of the following key features:
  + **Loading Patient Data**: It loads patient data from AsyncStorage when the component is mounted using the loadPatientData function.
  + **Adding New Patients**: Users can add new patients by clicking the "New Patient" button, which opens a modal with an input field for the patient's name. The patient data is then saved to AsyncStorage.
  + **Patient List**: A list of patients is displayed using a FlatList. Each patient item is tappable and leads to their respective profiles.
  + **Dropdown Picker**: Patients can be selected from a dropdown list that is searchable and navigates to the selected patient's profile.

### Styling

* The component utilizes styles defined in an external stylesheet (styles.js) to maintain a consistent and visually appealing UI.
* The styles include button styles, text styles, modal styles, and more.

## Usage

The Patient Records Screen is used for listing all current patients and adding new patients to the system. Navigate to the Patient Records Screen from the Home Screen.

## Notes

* Ensure that you have the required dependencies installed and set up in your React Native project.
* Customize the component and styles to match your application's design and functionality.
* Handle any additional navigation and data management requirements based on your application's needs.

# 

# Settings Screen

## Overview

The Settings Screen is a component within a healthcare mobile application that provides users with the ability to manage exercises and patient data. It allows users to add, view, and delete exercises, clear all data, and add dummy patient data for testing purposes.

## Dependencies

* React
* React Native
* React Native AsyncStorage

## Components

### SettingsScreen Component

This component is responsible for rendering and managing the exercise and patient data settings. It provides interactive features for a customized data management experience.

Key Features:

* **Adding Exercises:** Allows users to add new exercises to the list stored in AsyncStorage.
* **Viewing Exercises:** Displays a list of added exercises, allowing users to view and manage them easily.
* **Deleting Exercises:** Users can delete specific exercises from the list.
* **Clearing Data:** Provides an option to clear all data from AsyncStorage for a fresh start.
* **Adding Dummy Data:** Users can add dummy patient data to AsyncStorage for testing and development purposes.

### Styling

The styling is managed through an external stylesheet, ensuring a consistent and user-friendly interface. It caters to various UI elements, including buttons, text inputs, and list items.

## Usage

The Settings Screen is used for adding ‘dummy data’ to the system (premade data), clearing all data currently in storage, and for adding/removing exercising from the exercise list saved in storage. Navigate to the Settings Screen from the Home Screen.

## Notes

* Ensure all dependencies are installed and configured in the React Native project.
* The components and styles can be customized to align with the application’s design and functional requirements.
* The SettingsScreen is equipped with alert dialogs to confirm actions like clearing data and adding dummy data, ensuring that users are aware of the implications of their actions.

Code Structure

The SettingsScreen component is structured to handle various functionalities, including adding and removing exercises, clearing the exercise list, and managing AsyncStorage data. It uses state hooks to manage the exercises list and new exercise input. AsyncStorage is used to persist the exercises list and patient data.

Future Enhancements

* Implementing user authentication to secure patient data.
* Enhancing the UI/UX for a more intuitive user experience.
* Adding more options for data management and customization.

This documentation provides a comprehensive overview of the SettingsScreen component, ensuring developers have the necessary information to integrate and customize it within their React Native applications effectively.

# 

# Repetitions Screen

## Overview

The Repetitions Screen is a component within a healthcare mobile application that allows users to manage and track the repetitions of exercises for patients. It provides a user-friendly interface for adding, editing, and saving exercise repetitions, enhancing the user's experience in monitoring patient progress.

## Dependencies

* React
* React Native
* Expo StatusBar
* React Native Vector Icons
* React Native Dropdown Picker
* React Native AsyncStorage
* External Stylesheet

## Components

### RepetitionsScreen Component

This component is designed to be a modal that pops up for the user to add or edit the repetitions of selected exercises. It provides options to increment, decrement, or directly input the number of repetitions and select the exercise from a dropdown list.

**Key Features:**

* **Loading Exercise List:** Retrieves the list of exercises from AsyncStorage to populate the dropdown picker.
* **Exercise Selection:** Utilizes a dropdown picker for selecting exercises.
* **Repetitions Counter:** Allows users to increment, decrement, or input the number of repetitions.
* **Save and Discard Options:** Offers options to save the changes or discard and close the modal.

### Styling

The styling is managed through an external stylesheet to ensure a consistent and user-friendly interface across various elements of the component, including buttons, text inputs, and the dropdown picker.

## Usage

The Repetitions screen is used to assign values to the exercise data, assigning an exercise and adjusting the amount of repetitions. It is used on the New Session Screen.  
To integrate the RepetitionsScreen on other screens, ensure all dependencies are installed, import the RepetitionsScreen component, and use it as a modal or a standalone screen as needed.

import RepetitionsScreen from './RepetitionsScreen';

// Use as a modal or standalone screen

<RepetitionsScreen visible={modalVisible} onClose={closeModal} onSave={saveData} exercise={selectedExercise} />

## Notes

* Ensure all dependencies are installed and configured in the React Native project.
* The component can be customized to align with the application’s design and functional requirements.
* The RepetitionsScreen is designed to be used as a modal but can be adapted for other use cases.
* The Repetitions Screen is used as a modal on the New Session Screen.

# 

# Patient Screen

## Overview

The Patient Screen is a component within a healthcare mobile application that provides users with detailed information and options for a specific patient. It allows users to view session history, progress graphs, and offers the functionality to delete a patient from the database.

## Dependencies

* React
* React Native
* Expo Status Bar
* React Navigation Native
* React Native Vector Icons
* React Native Async Storage

## Components

### PatientScreen Component

This component is responsible for rendering and managing the patient's detailed view. It provides interactive features for a customized patient data management experience.

**Key Features:**

* **Display Patient Name:** Retrieves and displays the patient's name from the route parameters.
* **Session History Navigation:** Navigates to the session history screen for the selected patient.
* **Progress Graph Navigation:** Navigates to the graphing screen to visualize the patient's progress.
* **Delete Patient:** Allows the user to delete the patient's data from AsyncStorage.
* **Return to Home:** Navigates back to the home screen.

### Styling

The styling is managed through an external stylesheet, ensuring a consistent and user-friendly interface. It caters to various UI elements, including buttons, text, and icons.

## Usage

The Patient Screen is used for displaying navigation to patient specific screens. The user can delete the current patient on this screen, navigate to the Session History Screen or to the Graphing Screen, passing patientName as a parameter. Navigate to the Patient Screen from the Patient Records Screen, passing patientName as a parameter.

## Notes

Ensure all dependencies are installed and configured in the React Native project. The components and styles can be customized to align with the application’s design and functional requirements. Additional features and functionalities can be added to enhance the patient data management and user interaction experience.

# 

# Session History Screen

## Overview

The Session History Screen is a component within a healthcare mobile application that displays the history of sessions for a selected patient. It retrieves and lists all sessions, sorted by date and time, and allows users to navigate to a detailed view of each session.

## Dependencies

* React
* React Native
* Expo Status Bar
* React Navigation
* React Native Vector Icons
* React Native AsyncStorage
* Custom Stylesheet

## Components

### SessionHistoryScreen Component

This component is tasked with displaying the session history for a specific patient. It retrieves the patient's session data from AsyncStorage and presents it in a scrollable list, where each session can be selected to view detailed information.

Key Features:

* **Load Sessions:** Retrieves and displays the patient's sessions from AsyncStorage.
* **Sort Sessions:** Organizes sessions by date and time in descending order.
* **Navigation:** Allows users to navigate to a detailed session screen.
* **Format Date and Time:** Converts session date and time into a readable format.

### Styling

The styling is managed through an external stylesheet, ensuring a consistent and user-friendly interface across various UI elements, including buttons, text, and the scrollable list.

## Usage

The Session History Screen is used for displaying a list of sessions for a specified patient. It is navigated to from the Patient Screen, passing patientName as a parameter.

## Notes

* Ensure all dependencies are installed and configured in the React Native project.
* The component and styles can be customized to align with the application’s design and functional requirements.
* Additional features and functionalities can be added to enhance the user experience and data presentation.

# 

# Session Screen

## Overview

The Session Screen is a component within a healthcare mobile application that allows users to view detailed information about a specific session of a patient. It displays the exercises performed, repetitions, and provides options to delete the session or navigate back to the home screen.

## Dependencies

* React
* React Native
* Expo-status-bar
* React-native-vector-icons/AntDesign
* @react-navigation/native
* @react-native-async-storage/async-storage

## Components

### SessionScreen Component

This component is responsible for displaying the detailed information of a specific session, including the exercises and repetitions. It also provides the functionality to delete the session.

Key Features:

* **Load Exercise Data:** Retrieves exercise data from AsyncStorage based on the selected patient and session.
* **Delete Session:** Allows users to delete a specific session from the patient’s record.
* **Navigation:** Provides navigation options to return to the home screen.

### Styling

The styling is managed through an external stylesheet, ensuring a consistent and user-friendly interface across various UI elements.

## Usage

The SessionScreen is used to display detailed information about a specific session of a patient. It is navigated to from the SessionHistoryScreen, passing the patient's name and session date/time as parameters.

## Notes

* Ensure all dependencies are installed and configured in the React Native project.
* The component can be customized to align with the application’s design and functional requirements.
* Error handling and alerts are implemented to inform users of any issues during the deletion of a session.

# 

# Graphing Screen

## Overview

The Graphing Screen is a component within a healthcare mobile application that provides users with graphical representations of patient exercise data. It allows users to visualize the data in bar charts, filter the data by date range, and select specific exercises for detailed insights.

## Dependencies

* React
* React Native
* React Native AsyncStorage
* React Native Gifted Charts
* React Native Community DateTimePicker
* React Native Vector Icons

## Components

### GraphingScreen Component

This component is responsible for rendering and managing the patient's exercise data in graphical form. It provides interactive features for a customized data viewing experience.

Key Features:

* **Loading Patient Data:** Retrieves patient data from AsyncStorage and processes it for graphical representation.
* **Date Range Filtering:** Allows users to filter the displayed data by selecting a specific date range.
* **Exercise Selection:** Users can select specific exercises to view their respective data charts.
* **Bar Chart Visualization:** Utilizes the 'react-native-gifted-charts' library to render the exercise data in bar charts.

### Styling

The styling is managed through an external stylesheet, ensuring a consistent and user-friendly interface. It caters to various UI elements, including buttons, text, and charts.

## Usage

The Graphing Screen is used for Graphing the Data of a specified patient. The data is displayed as a bar graph, and sorted by exercise and by date range. Navigate to the Graphing Screen through the Patient Screen, sending patientName as a parameter.

## Notes

* Ensure all dependencies are installed and configured in the React Native project.
* The components and styles can be customized to align with the application’s design and functional requirements.
* Additional features and functionalities can be added to enhance data visualization and user interaction.

# 

# Data Export Screen

## Overview

The Data Export Screen is a component of a mobile application designed for healthcare professionals to manage and export patient data. It offers various data export options, including exporting all data, specific patient data, or data within a selected date range.

## Dependencies

* React
* React Native
* Expo
* Expo File System
* Expo Sharing
* React Native Vector Icons
* React Native Dropdown Picker
* React Native Community DateTimePicker
* PapaParse
* React Native AsyncStorage

## Components

### DataExportScreen Component

This component is the core of the data export functionality. It utilizes multiple state variables and functions to manage and facilitate the data export process.

Key Features:

* **Loading Patient Data:** Utilizes the useEffect hook to load patient data from AsyncStorage when the component mounts.
* **Exporting All Data:** A button that triggers the exportAllData function, exporting all patient data to a CSV file.
* **Exporting Specific Patient Data:** Incorporates a dropdown picker for selecting a specific patient whose data is then exported by the exportPatientData function.
* **Exporting Data in a Date Range:** Allows users to select a date range and exports the patient data within that range using the exportRangeData function.
* **Date Pickers:** Utilizes DateTimePicker components for users to easily select start and end dates for the range export.

### Styling

The component employs styles imported from an external stylesheet to ensure a consistent and user-friendly interface. The styles cater to various UI elements, including buttons, text, dropdown pickers, and date pickers.

## Usage

The Data Export Screen is used for exporting Patient Data from the save system. Users are able to export all data, export data between a date range, or export a specified patients data. Navigate to the Data Export Screen from the Home Screen.

## Notes

* Ensure all dependencies are installed and configured in the React Native project.
* The component and styles can be customized to align with the application’s design and functional requirements.
* Additional error handling and data validation can be added to enhance the robustness of the data export features.
* The exported data is saved as a CSV file and can be shared or stored as per the application’s requirements.

# Patient Data Structures

## Overview

This documentation outlines a set of JavaScript classes used for managing patient rehabilitation data. These data structures provide a hierarchical representation of patients, their sessions, exercises performed during sessions, and associated video data. The data structures facilitate the organization and storage of patient rehabilitation records.

## Classes

### PatientData Class

* **Description:** The PatientData class serves as a container for storing patient records. It maintains an array of Patient instances, allowing you to manage and organize multiple patient records.

#### **Constructor**

* **Method:** constructor()
  + **Description:** Initializes an empty patients array.
  + **Parameters:** None

#### **Methods**

* **Method:** addPatient(patient)
  + **Description:** Adds a Patient instance to the patients array.
  + **Parameters:**
    - patient (instance of Patient): The patient record to add.

### Patient Class

* **Description:** The Patient class represents an individual patient's rehabilitation record. It contains patient-specific information, including the patient's name, a unique identifier (id), and an array of Session instances associated with the patient.

#### **Constructor**

* **Method:** constructor(patientName)
  + **Description:** Initializes a Patient instance with a given patientName. It generates a unique identifier (id) for the patient and initializes an empty sessions array.
  + **Parameters:**
    - patientName (string): The name of the patient.

#### **Methods**

* **Method:** addSession(sessionDateTime)
  + **Description:** Creates a new Session instance and adds it to the sessions array for the patient. It returns the created Session instance.
  + **Parameters:**
    - sessionDateTime (string or Date): The date and time of the session.

### Session Class

* **Description:** The Session class represents an individual rehabilitation session. It contains session-specific data, such as a unique identifier (id), the date and time of the session (sessionDateTime), an array of Exercise instances performed during the session, and an array of associated video data (videos).

#### **Constructor**

* **Method:** constructor(sessionDateTime)
  + **Description:** Initializes a Session instance with a given sessionDateTime. It generates a unique identifier (id) for the session and initializes empty arrays for exercises and videos.
  + **Parameters:**
    - sessionDateTime (string or Date): The date and time of the session.

#### **Methods**

* **Method:** addExercise(exerciseName, repetitions)
  + **Description:** Creates a new Exercise instance with the specified exercise name and repetitions and adds it to the exercises array for the session.
  + **Parameters:**
    - exerciseName (string): The name of the exercise.
    - repetitions (number): The number of repetitions performed for the exercise.

### Exercise Class

* **Description:** The Exercise class represents an individual exercise performed during a rehabilitation session. It contains exercise-specific data, including the exercise name (exerciseName), the number of repetitions (repetitions), and a unique identifier (id).

#### **Constructor**

* **Method:** constructor(exerciseName, repetitions)
  + **Description:** Initializes an Exercise instance with the specified exercise name and repetitions. It generates a unique identifier (id) for the exercise.
  + **Parameters:**
    - exerciseName (string): The name of the exercise.
    - repetitions (number): The number of repetitions performed for the exercise.

## Usage

These patient data structures can be used to create and organize rehabilitation records for multiple patients. You can add patients, sessions, exercises, and associated data to maintain a comprehensive record of rehabilitation progress. These structures offer flexibility and organization, making it easier to manage and analyze patient data.

# Other Resources

Projects Git Repository:  
<https://github.com/SFowers/RehabTrack>

Official React Native setup guide:  
<https://reactnative.dev/docs/environment-setup>

Building an APK through expo for internal use:

<https://docs.expo.dev/build-reference/apk/>

AntDesign App Icons used in this project:  
<https://oblador.github.io/react-native-vector-icons/>

Gifted Charts (Graphing Screen) usage:  
<https://gifted-charts.web.app/>

React Native Async Storage (Saving and Loading throughout the project):

<https://github.com/react-native-async-storage/async-storage>