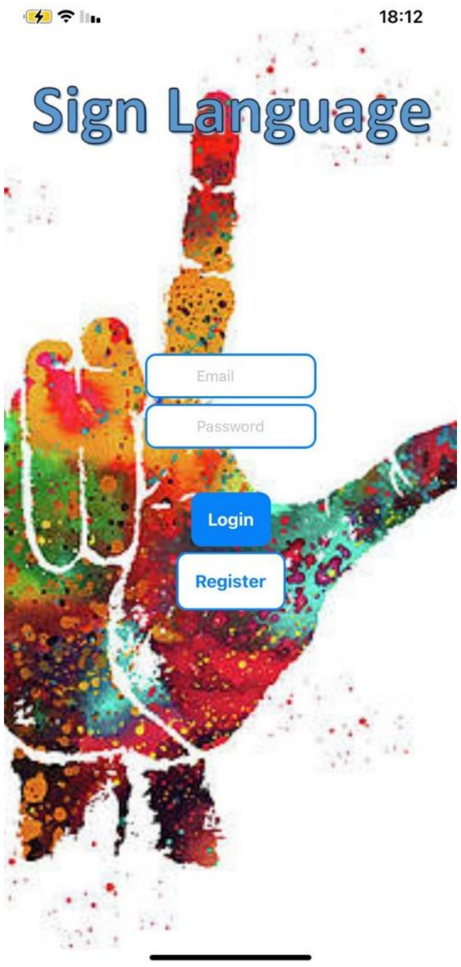
Sign-Language-Detection-React Native App

Github: https://github.com/matannagar/Sign-Language-Detection-ReactNative.git

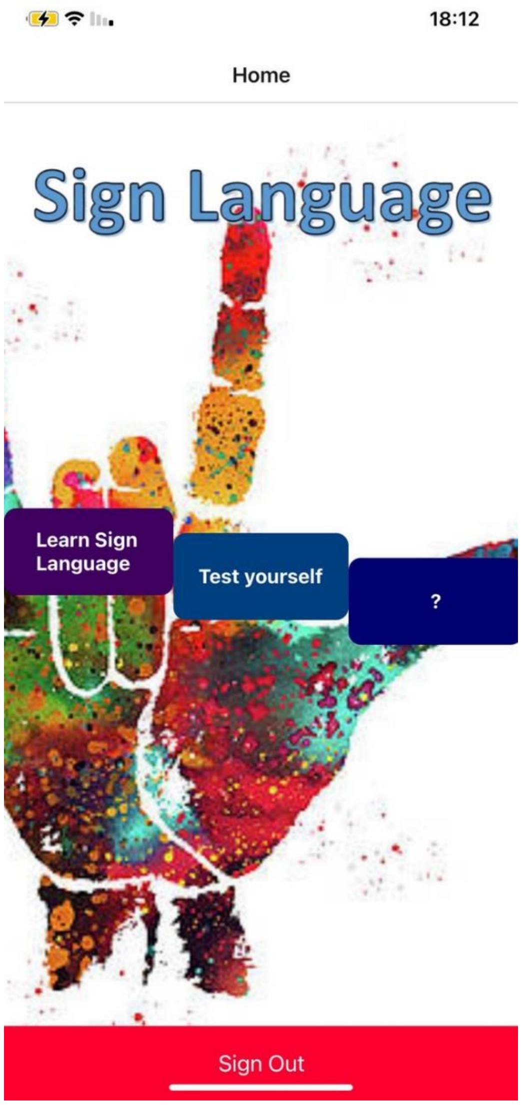
Contributors: Matan Ben-Nagar, Orya Spiegel, Asahel Cohen

Screens:

- ### Login page:
  The first page that opens when opening the app.
  Through this page you can login to your profile or register for a new one. The login and registration are done using firebase.



- ### Payment Page:
  A page where the user can add payment information and become a premium user.

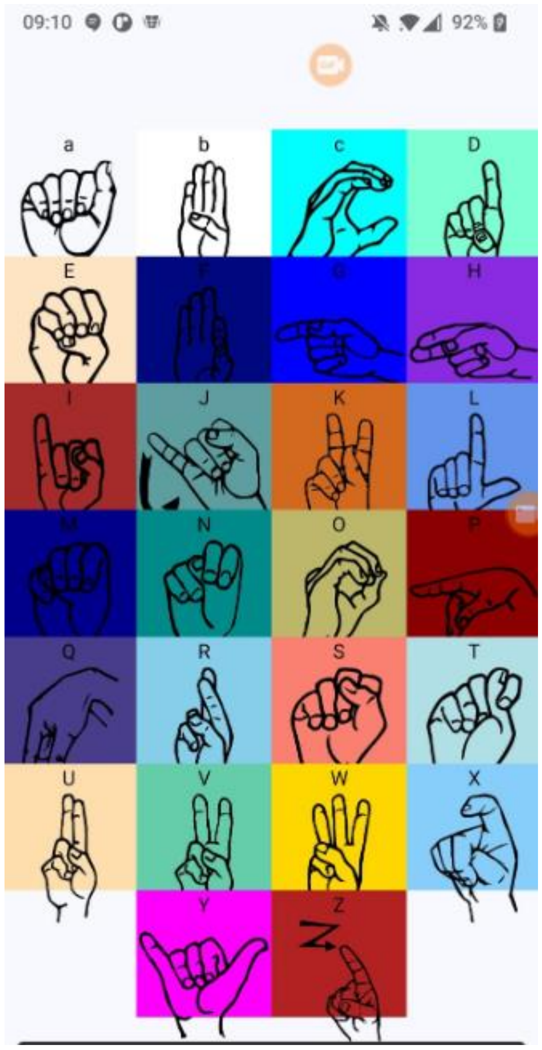


- ### Home page:
  The home page has three different options that take you to three of the following screens:
  1) Learn Sign-Language
  2) Test yourself on letters using live video
  3) Test yourself using live video, to write full sentence by signing letter by letter (premium users)

- 1) Learn Sign-Language Page:
  The page has all the Alphabet written on it. By pressing a specific letter, the image of how to preform the letter in sign-language pops up. The user can press the letter again to make the image disappear.

- 2) Test yourself on letters using live video Page:
  future screen

- 3) Test yourself using live video, to write full sentence by signing letter by letter (premium users) Page:
  future screen



Moving around the app:

When opening the app, the first page that is opened is the "Login page". After signing up or registering the user is moved automatically to the "Home page".
At the "Home page" the user has four options:

1) return to "Login page" by clicking "Sign-out" button.
2) move to "Learn Sign-Language" page by clicking "Learn sign-language" button.
3) move to "Test yourself on letters" using live video, by clicking "Test yourself" button.
4) move to "Test yourself on sentences" using live video, by clicking "Test yourself - sentence" button. (For premium users) If user is not premium, then the app will refer the user to "Payment Page".

To return to "Home page" from pages 2-4, simply press the "back" button.

components:

- Image:
  A React component for displaying different types of images, including network images, static resources, temporary local images, and images from local disk, such as the camera roll.
- ImageBackground:
  A common feature request from developers familiar with the web is background-image. To handle this use case, you can use the <ImageBackground> component, which has the same props as <Image>, and add whatever children to it you would like to layer on top of it.
  - Image & ImageBackground are used in all pages to design a nicer page, and in order to learn from the Images.
- View:
  The most fundamental component for building a UI, View is a container that supports layout with flexbox, style, some touch

handling, and accessibility controls. View maps directly to the native view equivalent on whatever platform React Native is running on.

View is designed to be nested inside other views and can have 0 to many children of any type.

- KeyboardAvoidingView:

  It is a component to solve the common problem of views that need to move out of the way of the virtual keyboard. It can automatically adjust either its height, position, or bottom padding based on the keyboard height.

-we used this every time we needed the keyboard. Moves the screen so we can see the full keyboard without hiding the page.

- Text:

  A React component for displaying text. Text supports nesting, styling, and touch handling.
- TextInput:

  A foundational component for inputting text into the app via a keyboard. Props provide configurability for several features, such as auto-correction, auto-capitalization, placeholder text, and different keyboard types, such as a numeric keypad.
- TouchableOpacity:

  A wrapper for making views respond properly to touches. On press down, the opacity of the wrapped view is decreased, dimming it.
- Button:

  A basic button component that should render nicely on any platform. Supports a minimal level of customization.

- Button & TouchableOpacity are used in all pages to allow user's to move from page to page, and display requested sources. TouchableOpacity is used so we can style the "button" more.

API:

- Alert:

  Launches an alert dialog with the specified title and message.

Optionally provide a list of buttons. Tapping any button will fire the respective onPress callback and dismiss the alert. By default, the only button will be an 'OK' button.

This is an API that works both on Android and iOS and can show static alerts. Alert that prompts the user to enter some information is available on iOS only.

- StyleSheet:

  A Style prop, that allows to edit Images, layouts, text, and the view.