CS-571 Building User Interfaces, Fall 2021, Professor Yuhang Zhao

React Native 2 *β* (3 Points)

Accessible Design

In this assignment, you will build on your React Native 2 α assignment to explore accessibility features and assistive technologies of mobile platforms. Specifically, you will integrate React Native accessibility features into your fitness tracking app to support screen reader use.

Part 1—Discovery, Planning, & Specifying: In this part, you will discover the screen reader assistive technology features of your mobile device, plan how you might support two tasks in your fitness app using these features, and develop specifications to implement these features into your RN components.

Part 2—Implementation: This part will involve implementing the specifications developed in the previous part as well as ensuring that other components do not distract a user with visual impairments.

Part 3—Testing & Demonstration: In this part, you will demonstrate the two tasks you are supporting with your implementation and capture your demonstration in the form of a narrated screen recording.

Submission Details

GitHub Classroom Starter Code

React Native 2 β will build on your implementation of React Native 2 α. You should copy your code from your React 2 α project to the React 2 β repository linked above, as that will be your starter code. When you commit/push, ensure that you are committing/pushing to the react-native2-beta repository, not react-native2-alpha. To complete the assignment, you will need to submit:

1. A completed version of this document as PDF to Canvas;
2. Your repository name and latest commit hash from GitHub Classroom, e.g. `react-native2-beta-osori, 91ddcb6`
3. A video recording of you demonstrating in MP4 format the intended use of accessibility features in your app, saved in your Google Drive folder and shared through a link (instructions) (as video files can be too large for Canvas to handle).

NOTE: *If you were unable to complete one or more problems in React Native 2* ***α*** *Assignment and your code is not fully functional, you can still complete this assignment and get full credit. Remember, in β assignments, you will be judged on the design aspects of your application. For example, if you were unable to implement the screen that shows activities, you can hard-code a few activity details in card components and apply accessibility design to them.*

Part 1: Discovery, Planning, & Specifying (1.4 Point)

In this part, you will engage in discovery of the screen reader assistive technology in your mobile platform of choice, prepare tasks for supporting accessibility in your application, and design the experience for a user with visual impairment across three steps.

Step 1. Discovery of Accessibility Features (0.3 Point). In this step, you will explore the accessibility features of the mobile device platform in which you have been testing your React Native projects. Your testing environment can be an iOS or Android device using the Expo app or an iOS or Android emulator on the computer. By enabling VoiceOver in iOS[1] (Settings → Accessibility → VoiceOver) and TalkBack in Android[2] (Settings → Accessibility → TalkBack) or accessibility testing tools in your emulator (e.g., Accessibility Inspector in Xcode), you will assess how screen readers work across two applications:

1. The latest version of your React Native fitness tracking application
2. Another application of your choice that you frequently use

Complete or attempt to complete two common tasks in both applications with the screen reader on and report below your observations. Specifically, describe what tasks you performed or attempted to in each application and how the applications supported the task.

<task-descriptions-and-observations>

Task 1: Add an item

RN Fitness App: This task includes adding an exercise. I assume that the user is logged in and on the exercise tab, and I am using VoiceOver on iOS. Tapping on the "Add Exercise" button triggers VoiceOver to say "Add Exercise button". Then, double tapping to add an exercise, the first thing VoiceOver says when the modal is displayed is "Exercise Details", which is the title of the modal. Then, for each form entry, tapping it triggers VoiceOver to say "<Placeholder> text field, double tap to edit". For the datepicker, VoiceOver says "Datepicker 11 slash 20 slash 21, double tap to expand". Once the user is in the datepicker, VoiceOver immediately says "Month November 2021, button adjustable, double tap to change month and year, swipe up and down with one finger to adjust the value". Tapping anywhere outside the datepicker triggers VoiceOver to say "Dismiss pop-up, double tap to dismiss pop-up window". For the timepicker, VoiceOver defaultly says "16 o'clock, picker item adjustable, 16 of 24, swipe up or down with one finger to adjust the value". This is similar to the datepicker, which I notice that both of them have appropriate support for accessibility since they are iOS's default date and time pickers. When the activity is added, the user can read from the alert which is narrated by VoiceOver.

Apple Reminder: This task includes adding a reminder, which is a similar task to adding an exercise in the fitness app. On the bottom toolbar of the reminder app, tapping on the "+ New Reminder" triggers VoiceOver to say "New reminder button". Once the new reminder editor is shown, VoiceOver says "Text field, is editing, title, character mode, insertion point start". When tapping on the notes field, VoiceOver says "Notes, text field, double tap to edit". Once the user is editing, VoiceOver tells the user where the

[1] You can learn more about iOS VoiceOver here: https://www.apple.com/accessibility/iphone/vision/
[2] You can learn more about Android TalkBack here: https://support.google.com/accessibility/android/answer/6283677?hl=en

insertion point is at. When toggling on the button to add a date, the default date picker shows up and VoiceOver immediately says "More options shown". This is something the fitness app is not doing, which is informing the user of new items coming up on the screen. Here, the date and time pickers are the same as in the fitness app since they are all default iOS date and time pickers. When the reminder is added, there is a short sound played, but no message from VoiceOver.

Task 2: Log in

RN Fitness App: This task includes the user logging into the fitness app. When the log in view is first shown, VoiceOver reads out "Fitness Tracker Welcome, title". Tapping on the username field triggers VoiceOver to say "Username, text field, double tap to edit". Tapping on the password field triggers VoiceOver to say "Password, secure text field, double tap to edit". This is important because the user is informed that the password field is secure. However, I notice that the VoiceOver will also read out each character the user types in, which can be secure if someone around the user has some malicious intents. Once the user is logging in, there is no helpful message from VoiceOver. The first thing the user hears is "Logout" since the log out button is on the header.

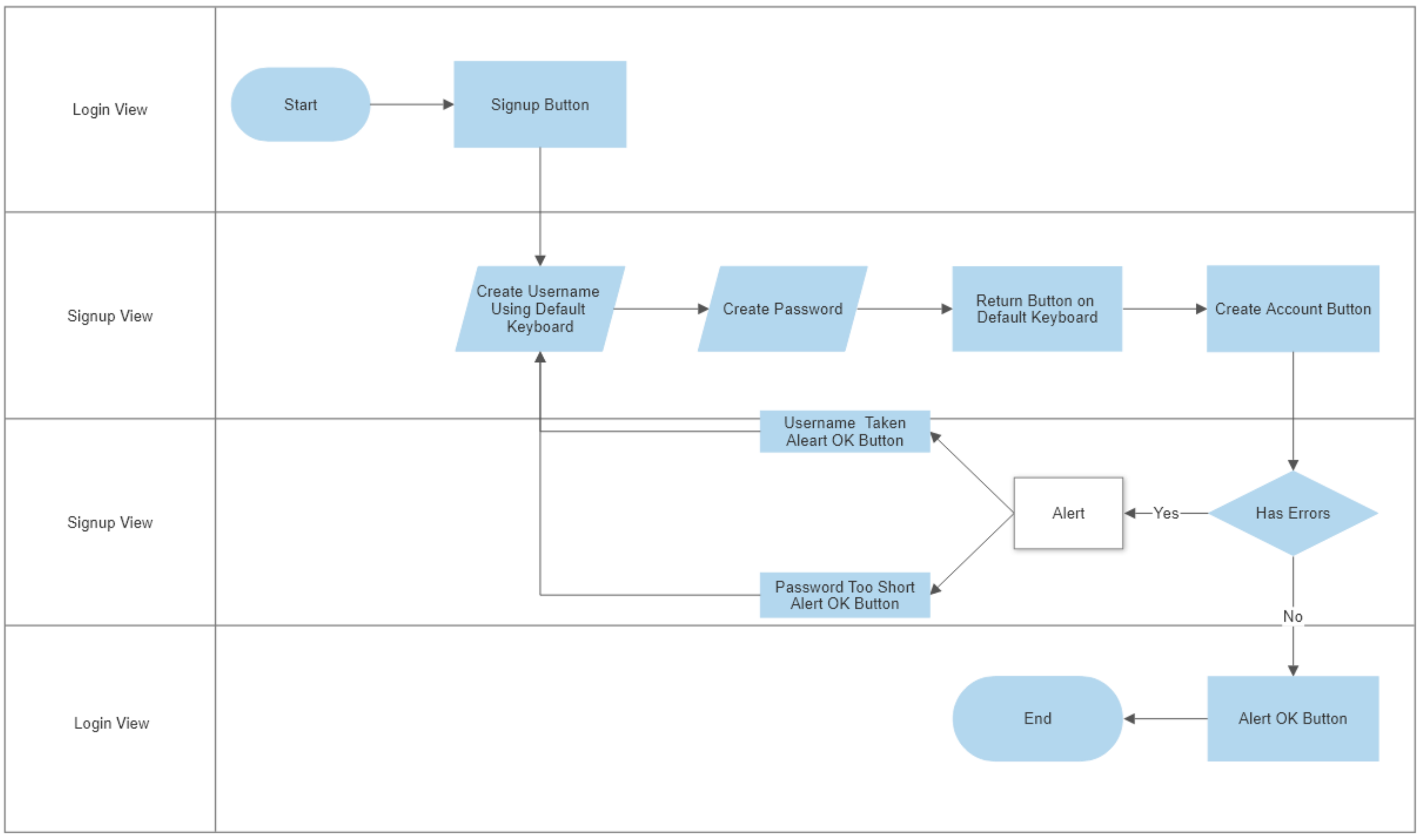
Global Protect: The task includes the user logging in and connecting to the VPN, which is a similar task as logging in on the fitness app. Tapping on the round button in the middle triggers VoiceOver to say "button border, disconnected, button". This information is important because it informs the user of the current status of the VPN: disconnected. Double tapping the button gives the message "Button border connecting". When the login screen shows up, the first thing heard is "Back" because the back button is on the header. When entering user credentials, tapping on the password field only gives the message "password". Only tapping on it triggers VoiceOver to say "Secure text field, is editing, character mode, insert point start". After entering the credentials, double tapping the sign in button leads the user to the MFA-Duo sign in view. The MFA-Duo passcode is another secure text field. After entering 1 and the user is successfully authenticated through the Duo app, VoiceOver reads out the round button as "button border, connected, button".

Step 2. Planning for Accessible Design (0.5 Point). In this step, you will choose two tasks supported by your React Native 1 α deliverable (e.g., sign up for an account) or your React Native 2 α deliverable (e.g., add an account for the current day) and map out how you expect users with blindness or severe visual impairments to interact with them given what you learned in Step 1. You can repeat the task you specified in Step 1. Specifically, you will create flowcharts of what components the user must interact with and in what order to perform the task. This activity will help you choose the right groupings for your React Native elements in order to define the accessibility features you will need. To generate flowcharts, you can use SmartDraw[3] or free versions of other tools, such as LucidChart or Creatly.

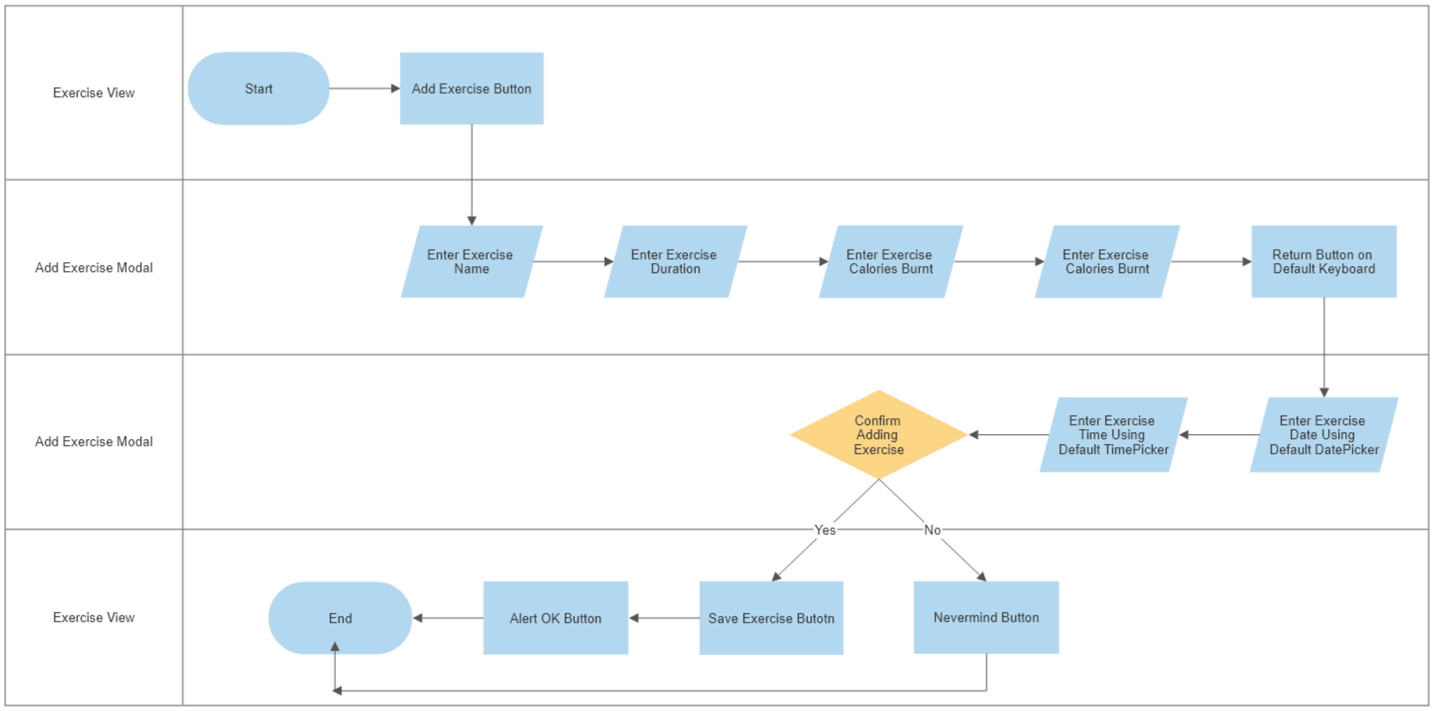
<task-flowcharts>

[3] You can get the school license key for SmartDraw from the Campus Software Library. Follow their instructions to activate the online version, or download the Windows version of SmartDraw.

Task 1: Sign up for an account


Login View
Start
Signup Button
Signup View
Create Username Using Default Keyboard
Create Password
Return Button on Default Keyboard
Create Account Button
Signup View
Username Taken Aleart OK Button
Alert
Yes
Has Errors
Password Too Short Alert OK Button
No
Login View
End
Alert OK Button

Task 2: Add an activity



Step 3. Specifying Accessibility Features (0.6 Point). This step will involve determining how to specify accessibility features for the components you included in your flowcharts in Step 2. For each component, you will write out how you will enable accessibility features using React Native Accessibility (review the accessibility properties), such as where accessibility features should be enabled, what labels and hints should be provided, what accessibility actions should be supported, on. It is important to put yourselves in the shoes of a user with visual impairments and consider how you would like to support user navigation and interaction, what the labels should say exactly so that they accurately and effectively communicate the functionality of each component.

<per-component-accessibility-specifications>

Task 1: Sign up for an account

Under Login View:

- FitnessTracker Text
    - accessible={true}
    - accessibilityLabel="Fitness Tracker Header"
    - accessibilityHint="You can either log in using the text fields or sign up on the bottom of the page"
- Login Button
    - accessible={true}
    - accessibilityLabel="Login button"
- Sign up Button
    - title="Sign up" (instead of signup because VoiceOver reads out signup weirdly)
    - accessible={true}
    - accessibilityLabel="Sign up button"

Under Signup View:

- General View:
    - onAccessibilityEscape={() => { this.backToLogin(); }}
- FitnessTracker Text
    - ref={(el) => {this.viewRef = el; }} (*)
    - style={styles.bigText}
    - accessible={true}
    - accessibilityLabel="Fitness Tracker Header"
    - accessibilityHint="You can sign up on this page"
- Username TextInput
    - accessible={true}
    - accessibilityHint="Username should be at least 5 characters"
- Password TextInput
    - accessible={true}

    - accessibilityHint="Password should be at least 5 characters"
- Create Account Button
    - accessible={true}
    - accessibilityHint="Double tap to create account"
- Nevermind Button
    - accessible={true}
    - accessibilityHint="Double tap to exit sign up page"

Note (*): This is used for changing the default focus of VoiceOver when navigating through the stacks. When the user navigates from Login View to Signup View, VoiceOver defaultly puts the focus on the nearest location of the user's last double tap location, which is the Nevermind button. However, we want the user to start from the top. Also, When the user finishes creating username and password and double taps the return button on the keyboard, VoiceOver defaultly puts focus on the top exit button, which is not user friendly. I changed the focus to be on the button.

Task 2: Add an activity

Under Exercise View:

- Running man Icon
    - accessible={true}
    - accessibilityLabel="Running man icon"
- Exercises Text
    - accessible={true}
    - accessibilityLabel="Exercises page, title"
    - accessibilityHint="You can view and add exercises on this page"
- Add Exercise Button
    - accessible={true}
    - accessibilityHint="Double tap to add an exercise"

Under Add Exercise Modal:

- Exercise Detail Text
    - accessible={true}
    - accessibilityLabel="Exercises Details, modal title"
    - accessibilityHint="You can add an exercise in this modal, exit on bottom of modal"
- Exercise Name Text
    - accessible={true}
    - accessibilityLabel="Exercise name, form entry title"
- Exercise Duration Text
    - accessible={true}
    - accessibilityLabel="Duration in minutes, form entry title"
- Exercise Calories Burnt Text
    - accessible={true}
    - accessibilityLabel="Calories burnt in kilocalorie, form entry title"
- Set Date Button

        - accessible={true}
        - accessibilityLabel="Set date, button, double tap and swipe to right twice to access the date picker"
    - Set Time Button
        - accessible={true}
        - accessibilityLabel="Set time, button, double tap and swipe to right once to access the time picker"

Part 2: Implementation (0.8 Point)

The outcome of Steps 2 and 3 in Part 1 provides you with exact specifications for implementing the accessibility features into your code of the React Native application. In addition to carrying out these specifications in your code, you will also have to disable accessibility features for components that do not support your tasks and might be distractions for users with visual impairments. The deliverable of this part of the assignment is the code you will submit into GitHub Classroom.

react-native2-beta-YushunChen, 59e7ad5, iOS

Part 3: Testing & Demonstration (0.8 Point)

In this part of the assignment, you will perform the tasks you chose in Step 2 of Part 1 with the screen reader on and capture a video of your demonstration. You can use the iOS in-built screen recorder, one of the various screen recording options on the Android, or another device (e.g., your friend's phone, or a tablet computer) to record yourself demonstrating the tasks. Save this recording into your Google Drive, set permissions such that the video is viewable for anyone with the link, and include the link in your submission on Canvas. You can save two separate video files for the two tasks, or a single video file that demonstrates the tasks back to back. In addition to capturing your screen, screen recorders can also capture your voice, and you will be asked to provide narration along with your demonstration.

Task1 Demo Google Drive Link:

https://drive.google.com/file/d/1jpj2ZsHlVpHPYQU6MJ6rnOdy78v-araT/view?usp=sharing

Task1 Demo Backup YouTube Link:

https://youtu.be/ffxxzRuNOPo

Task2 Demo Google Drive Link:

https://drive.google.com/file/d/1FA1uEP6go4CKlAWivdmvUzeDv1o8gNpk/view?usp=sharing

Task2 Demo Backup YouTube Link:

https://youtu.be/BkbOvtotHTo