CS-639 Building User Interfaces, Fall 2020, Professor Mutlu

React Native 2 β (3 Points)

Accessible Design

Early Due Date (+1 Point of Extra Credit) Tuesday, November 17th @ 11:59 PM CDT

Regular Due Date Sunday, November 22nd @ 11:59 PM CDT

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-ctnelson1997, 2b0ef83"
3. A video recording of you demonstrating in MP4 format the intended use of your prototype, saved in your Google Drive folder and shared through a link (instructions) (as video files can be too large for Canvas to handle).

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.

Tasks: login to the account and check and edit profile.

I use an app for preparing for GRE to be the app that I frequently use. I found that the screen reader perform similarly on both apps when I am performing the tasks.

- For both tasks:
  - The screen reader will read out the placeholder value text input box and tell me how to input texts. It will also tell me that I am entering words to the text input box and that I am entering words at the end of the input box (and tell me about where I am entering in when I change the enter point)
  - The screen reader will read out the name of the button and tell me that it is a button
  - When entering texts for the normal text input box, the screen reader will read out the letter that I pressed on and again after I type 2 time (I can press on anywhere for two time to enter after I selected on a letter)
  - When I click on texts (such as welcome texts), the screen reader will read out the text)
  - It will tell me that it is an alert when the alert shows up.
  - When I scroll down the pages, the reader will tell me which page I am in (e.g., in a Scrollview, tell me "page 2 of 2" when I scroll down)
- Log In:

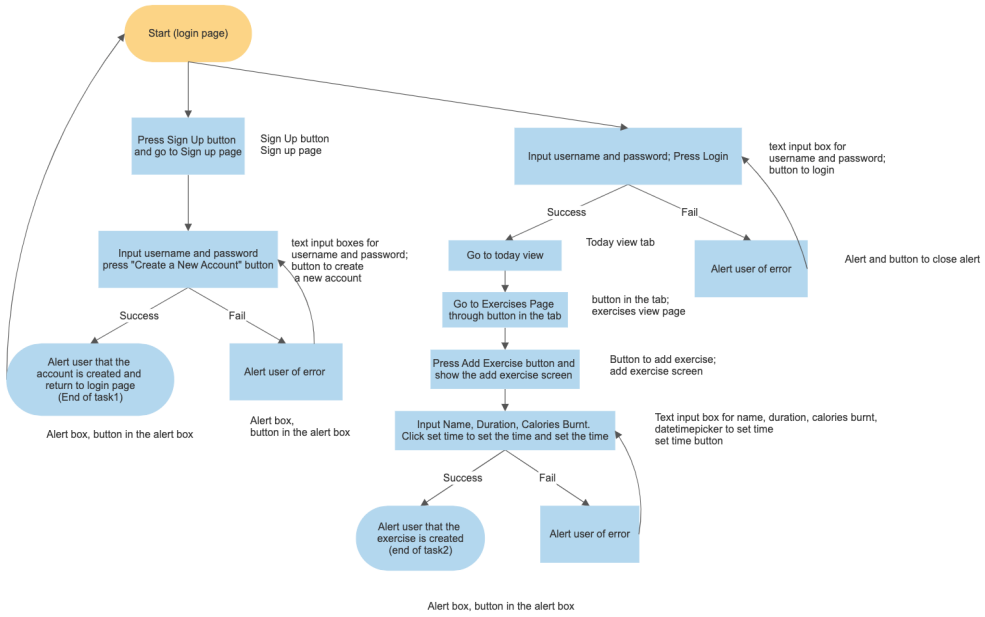
[1] You can learn more about iOS VoiceOver here: https://www.apple.com/accessibility/iphone/vision/
[2] You can learn more about Android TalkBack here: https://support.google.com/accessibility/android/answer/6283677?hl=en

- When entering texts for the text input box for password (where the app hide the user input), the screen reader will read out the letter that I pressed on and again after I type 2 time, but with reading out a word starting (e.g., alpha for a) with the letter that I enter which is different from normal text input

- Check and edit profile
  - When I am in the profile page and press on the profile button again in the tabs, the screen reader will tell me that the profile tab is selected. It will also tell me that the "place" of the profile button in the tab ("4 of 4" when the button is on the left most, "1 of 4" when the button is on the right most, when there are 4 buttons in the tab)
  - When I pressed on the icon in the profile page, the screen reader will tell me the meaning of the icon (But it does nothing when I type on the icon in the log in page of my fitness ap, might be due to that I use icons from different packages) It is not sure whether reading out the icon would help the user to perform tasks or be distracting.

Step 2. Planning for Accessible Design (0.5 Point). In this step, you will choose <u>two</u> 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 (using your NetID login) or free versions of other tools, such as LucidChart or Creatly.

Tasks: sign up for an account and add an exercise for the current day



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 accessbility actions should be supported, and so 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.

- Enable the accessibility features buttons, text input boxes, date time picker, some texts, alert box.
- Modify the Alert to let the reader directly reads out the message instead of reading "Alert."
- Texts: do not need to specify as it is set to be accessible by default and reader will read out the texts. Not disable all as some texts let users know where they are or are some instruction. (disable useless texts such as let's get to work)
- Button:
  - Do not apply accessibilityHint directly but create an alert when user perform magicTap (IOS) to only provide hints when they needed as there most of the button usages can be simply explained by label. Those buttons like Nevermind might be confusing, but it does not worth telling the hint every time and could be distracting as majority of the users can understand its meaning.
  - Use accessibilityLabel to tell the user about the name of the button (e.g., login button), add accessibility actions for magic tap to tell the user the function of the button if the label cannot express the usage of the button (e.g., no need for login button cause the usage is clear, might need hint for nevermind to cancel signing up or add exercises to specify that users are canceling entering info)
- Text input box:
  - Use accessibilityLabel to tell the user about the name of the text Input box (e.g., username text input), use accessibilityHint to tell the user the about what to input if it is not clear by the label (e.g., no need for username text input as it is clear that the text input is used for entering username with label, might need for calories burnt, that specified that user is entering in kcal)
  - Use accessibilityHint here for specification of units (e.g., kilocalories for calories burnt). Not use accessibility action as the specification for unit is short and it would be fine to inform the user about the unit every time (not distracting).
- Date time picker: has default accessibility properties to read what user has selected.
- Icons: disable accessibility to not distract users and icon name or meanings has no effects on performing users the info

- Accessbility actions: enable magicTap (ios) to show user more detailed info with showing an alert. This is useful as it will not be invoked with normal usage and can show user instructions or info if user need it.

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 theReact 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.

Ios iPhone8, react_native2_beta-YurenSUN, 436fc80

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.

Videos in the links below can be accessed with wisc email

Sign up:

https://drive.google.com/file/d/1dsSv7gsYK04afQZtccc4dSXLNGeMSSTm/view?usp=sharing

Add exercise:

https://drive.google.com/file/d/1A0MYUa8Gt8tuSq2jUKVZnrhnB2C778L3/view?usp=sharing

My phone is not able to capture my voice when I record the screen. I add my narration after I record the screen so that there are some conflicts with voiceover and my voice, but I tried my best to narrate without conflicts.