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.

---

1. **Fitness Tracker Application**
    a. **Task 1: Login to the application**
        i. For this task, I opened the Fitness Tracker application on my Android phone. On opening the application, I was able to understand the different elements on the screen by tapping on them.
        ii. Editable elements were clearly stated - for example, on clicking the username input field, Talkback said "Editing username. Edit box. Double tap to edit text. Double tap and hold to long press…".
        iii. On double tapping the username field, I was able to understand that the keyboard showed up based on the feedback("Showing English supporting keyboard") from Talkback.
        iv. On entering a wrong username/password combination, Talkback reads out the alert shown saying that the login credentials are incorrect.
        v. On entering the correct credentials, I was able to login into the application. However, I wasn't able to understand that I had successfully logged in unless I tapped somewhere on the screen in the 'today' view.
    b. **Task 2: Delete an exercise from the exercise view**
        i. I wasn't able to delete an exercise from the exercise view. When I tapped on the delete icon, the whole row containing both the edit and delete icons were selected and no action was taken.
2. **Gmail**
    a. **Task 1: Read all the unread emails**

[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

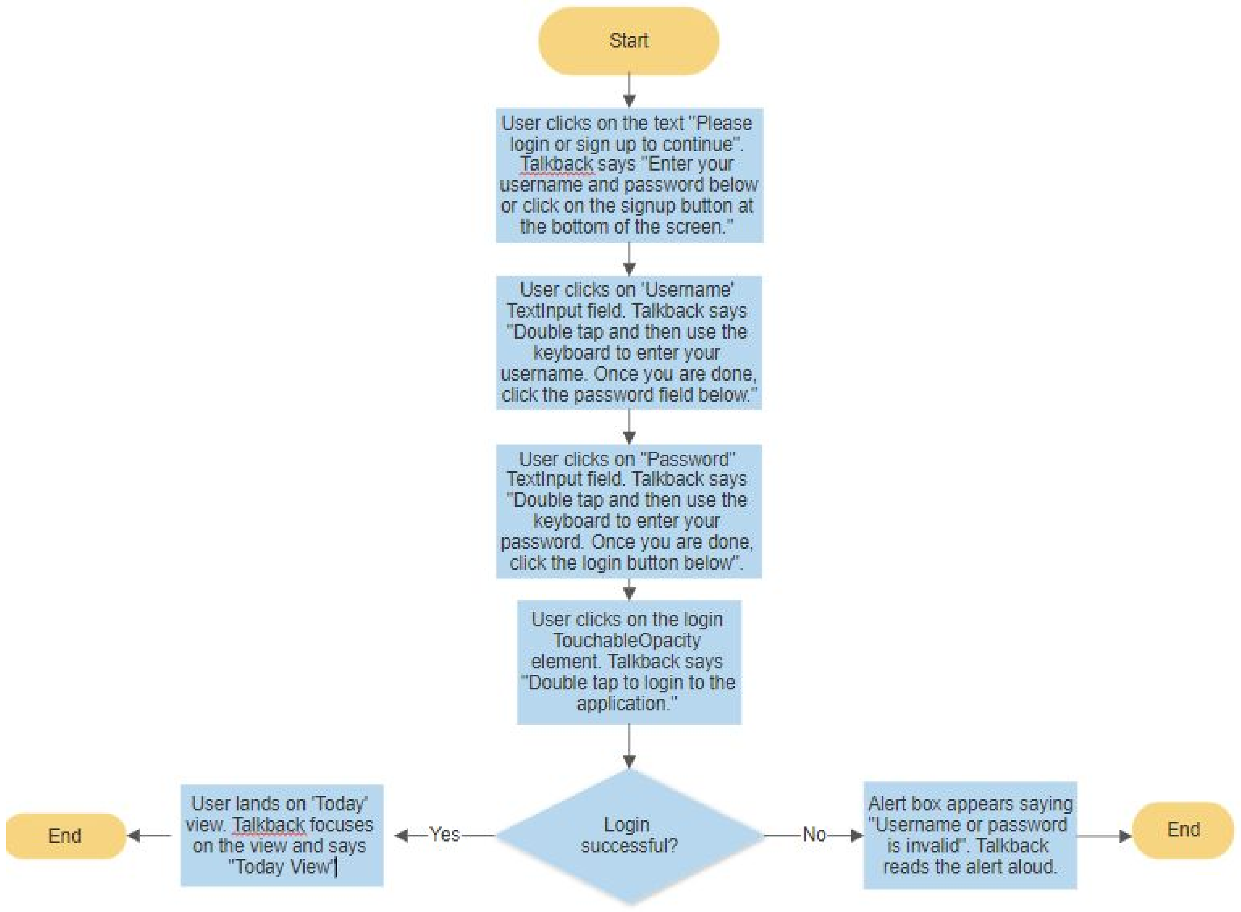
On opening Gmail, the focus was on the search bar. On clicking the elements below, Talkback started reading out the status of the email(read/unread), the sender's name, the subject, and a few first lines of the email. I was able to scroll through the list by using a swipe up/down gesture using two fingers.

b. **Task 2: Compose a new email**

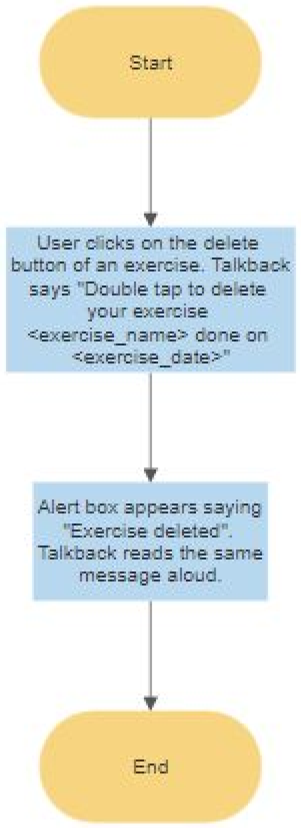
On double tapping the 'compose' button, the compose email screen opened and Talkback focused on the 'To' input field. I was able to type in the recipient's address by double tapping and then using the keyboard. I was able to type in the subject and email body similarly. Finally, I was able to send the email by double-tapping the send button. I think the major difficulty in accomplishing this task was finding the location of the 'send' 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 (using your NetID login) or free versions of other tools, such as LucidChart or Creatly.

Task 1: Log in to the application:


Start
User clicks on the text "Please login or sign up to continue". Talkback says "Enter your username and password below or click on the signup button at the bottom of the screen."
User clicks on 'Username' TextInput field. Talkback says "Double tap and then use the keyboard to enter your username. Once you are done, click the password field below."
User clicks on "Password" TextInput field. Talkback says "Double tap and then use the keyboard to enter your password. Once you are done, click the login button below".
User clicks on the login TouchableOpacity element. Talkback says "Double tap to login to the application."
Login successful?
Yes
User lands on 'Today' view. Talkback focuses on the view and says "Today View"
End
No
Alert box appears saying "Username or password is invalid". Talkback reads the alert aloud.
End

Task 2: Delete an exercise


Start
User clicks on the delete button of an exercise. Talkback says "Double tap to delete your exercise <exercise_name> done on <exercise_date>"
Alert box appears saying "Exercise deleted". Talkback reads the same message aloud.
End

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, 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.

Task 1: Log in to the application

1. ‘LoginView’ component:
    a. ‘Welcome’ Text: Since users with low or no vision do not know what the screen contains, I am using the below property and hint to let them know what to do.
        i. accessible = {True}
        ii. accessibilityHint = “Enter your username and password below or click on the signup button at the bottom of the screen.”
    b. ‘Username’ TextInput:

        i. accessible={True}
        ii. accessibilityLabel="Username input field"
        iii. accessibilityHint = “Double tap and then use the keyboard to enter your username. Once you are done, click the password field below.”
    c. ‘Password’ TextInput:
        i. accessible = {true}
        ii. accessibilityLabel="Password input field"
        iii. accessibilityHint = “Double tap and then use the keyboard to enter your password. Once you are done, click the login button below.”
    d. ‘Login’ Touchable Opacity:
        i. accessibilityHint = “Double tap to login to the application.”
    e. ‘TodayView’ component:
        i. Navigation listener - On screen focus, trigger ‘typeViewFocused’ event so that users know they have successfully logged into the application and are on the ‘today view’
        ii. ScrollView : Since focus moves to the ScrollView element, adding this hint helps the user understand that they are on the ‘today view’:
            1. accessible = {true}
            2. accessibilityHint = "Today View"

Task 2: Delete an exercise

1. ‘Delete’ MaterialIcons icon
    a. accessible={true}
    b. accessibilityLabel ="Delete exercise"
    c. accessibilityHint={"Double tap to delete your exercise <exercise_name> done on <exercise_date>"

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.

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.

Video recording:

https://drive.google.com/drive/folders/1LVX-k-t6UBY_S7hDU84DYdVK9qJXyzs2?usp=sharing