David Wang

CSE 120

**Final Source Code**

(I was unable to find our original App.js with 4.2’s code but I uploaded the changes I contributed to it in the word doc “firstdraftcode4.2”)

Our final App.js is attached in the submitted zip folder. Of that file, the source code I contributed:

I split the code up into the HomeScreen, ProfileScreen, CalendarScreen, MapScreen, and RootStack classes.

**HomeScreen class:**

In the HomeScreen class, I coded the button functionality with “onPress” statements and linked each button to its respective place in the RootStack class for the navigation route.

I coded the original buttons and added the “this.props.navigation.navigate(…)” statements to each button to navigate to their respective classes. Neil changed the style of the buttons with the “color = statements.”

Jason added the “Check-In” button and state checker functionality with “onPress={() => **this**.setState(state => ({checkedIn: 1}))}” and also helped add the ImageBackground and view style statements from our original UI to incorporate it into each screen.

**ProfileScreen class:**

I coded the ProfileScreen class and linked it to RootStack, adding the “onPress={() => **this**.props.navigation.push(…)}” statements to navigate back and forth between the ProfileScreen and the HomeScreen.

**CalendarScreen class:**

Tristan coded a Calendar.js file, and I transfered the code to the CalendarScreen class and linked it to the RootStack and navigation route

**StatsScreen class:**

I coded the basic functionality for the Details screen class as a dummy page for adding UI onto later, and I coded the onPress={() => **this**.props.navigation.push(…)}” statements to navigate back and forth between the StatsScreen and HomeScreen. I added the StatsScreen to RootStack to get the navigation route set.

**MapScreen class:**

Neil coded a Map.js file, and I incorporated his code into the MapScreen class and linked it to the RootStack and Navigation route

**Const RootStack:**

I coded the format for the various Screen classes and formatted the RootStack class in order to set up navigation functionality between the screens.

In the

**class** HomeScreen **extends** React.Component {

**constructor**(props) {

**super**(props);

**this**.state = { checkedIn: 0, };

}

render() {

**if**(**this**.state.checkedIn == 0) {

**return**(

<ImageBackground style={styles.container}>

<View style={styles.top}>

<Image source={logo} style={styles.logo} />

</View>

<View style={styles.button}>

<Button

title = "Check-In"

color = 'black'

onPress={() => **this**.setState(state => ({checkedIn: 1}))}

/>

</View>

<View style={styles.button}>

<Button

title = "Event Calendar"

color = 'black'

onPress={() => **this**.props.navigation.navigate('Calendar')}

/>

</View>

<View style={styles.button}>

<Button

title = "Statistics"

color = 'black'

onPress={() => **this**.props.navigation.navigate('Statistics')}

/>

</View>

<View style={styles.button}>

<Button

title = "Profile"

color = 'black'

onPress={() => **this**.props.navigation.navigate('Profile')}

/>

</View>

<View style={styles.button}>

<Button

title = "Track Runner"

color = 'black'

onPress={() => **this**.props.navigation.navigate('Map')}

/>

</View>

<View style={styles.button}>

<Button

title = "Log Out"

color = 'black'

onPress={() => **this**.props.navigation.navigate('Login')}

/>

</View>

</ImageBackground>

)

} **else** {

**return**(

<ImageBackground style={styles.container}>

<View style={styles.top}>

<Image source={logo} style={styles.logo} />

</View>

<View style={styles.button}>

<Button

title = "Check-Out"

color = 'black'

onPress={() => **this**.setState(state => ({checkedIn: 0}))}

/>

</View>

<View style={styles.button}>

<Button

title = "Event Calendar"

color = 'black'

onPress={() => **this**.props.navigation.navigate('Calendar')}

/>

</View>

<View style={styles.button}>

<Button

title = "Statistics"

color = 'black'

onPress={() => **this**.props.navigation.navigate('Statistics')}

/>

</View>

<View style={styles.button}>

<Button

title = "Profile"

color = 'black'

onPress={() => **this**.props.navigation.navigate('Profile')}

/>

</View>

<View style={styles.button}>

<Button

title = "Track Runner"

color = 'black'

onPress={() => **this**.props.navigation.navigate('Map')}

/>

</View>

<View style={styles.button}>

<Button

title = "Log Out"

color = 'black'

onPress={() => **this**.props.navigation.navigate('Login')}

/>

</View>

</ImageBackground>

)

}

}

}

**class** ProfileScreen **extends** React.Component {

render() {

**return** (

<View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>

<Text>Details Screen</Text>

<Button

title="Profile Page"

onPress={() => **this**.props.navigation.push('Profile')}

/>

<Button

title="Go to Home"

onPress={() => **this**.props.navigation.navigate('Home')}

/>

<Button

title="Return"

onPress={() => **this**.props.navigation.goBack()}

/>

</View>

);

}

}

**class** StatsScreen **extends** React.Component {

render() {

**return** (

<View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>

<Text>User Statistics</Text>

<Button

title="Reload Screen"

onPress={() => **this**.props.navigation.push('Statistics')}

/>

<Button

title="Go to Home"

onPress={() => **this**.props.navigation.navigate('Home')}

/>

<Button

title="Return"

onPress={() => **this**.props.navigation.goBack()}

/>

</View>

);

}

}

**class** CalendarScreen **extends** React.Component {

render() {

**return** (

<View>

<CalendarList

// Callback which gets executed when visible months change in scroll view. Default = undefined

onVisibleMonthsChange={(months) => { console.log('now these months are visible', months); }}

// Max amount of months allowed to scroll to the past. Default = 50

pastScrollRange={50}

// Max amount of months allowed to scroll to the future. Default = 50

futureScrollRange={50}

// Enable or disable scrolling of calendar list

scrollEnabled={**true**}

// Enable or disable vertical scroll indicator. Default = false

showScrollIndicator={**true**}

//...calendarParams

// Initially visible month. Default = Date()

// Minimum date that can be selected, dates before minDate will be grayed out. Default = undefined

minDate={'2019-01-01'}

// Maximum date that can be selected, dates after maxDate will be grayed out. Default = undefined

maxDate={'2019-12-30'}

// Handler which gets executed on day press. Default = undefined

onDayPress={(day) => { console.log('selected day', day) }}

// Handler which gets executed on day long press. Default = undefined

onDayLongPress={(day) => { console.log('selected day', day) }}

// Month format in calendar title. Formatting values: http://arshaw.com/xdate/#Formatting

monthFormat={'MMMM yyyy'}

// Handler which gets executed when visible month changes in calendar. Default = undefined

onMonthChange={(month) => { console.log('month changed', month) }}

// Hide month navigation arrows. Default = false

hideArrows={**true**}

// Replace default arrows with custom ones (direction can be 'left' or 'right')

renderArrow={(direction) => (<Arrow />)}

// Do not show days of other months in month page. Default = false

hideExtraDays={**false**}

// If hideArrows=false and hideExtraDays=false do not switch month when tapping on greyed out

// day from another month that is visible in calendar page. Default = false

disableMonthChange={**true**}

// If firstDay=1 week starts from Monday. Note that dayNames and dayNamesShort should still start from Sunday.

firstDay={0}

// Hide day names. Default = false

hideDayNames={**false**}

// Show week numbers to the left. Default = false

showWeekNumbers={**false**}

// Handler which gets executed when press arrow icon left. It receive a callback can go back month

onPressArrowLeft={substractMonth => substractMonth()}

// Handler which gets executed when press arrow icon left. It receive a callback can go next month

onPressArrowRight={addMonth => addMonth()}

/>

</View>

);

}

}

**let** {height, width} = Dimensions.**get**('window');

**class** MapScreen **extends** React.Component {

render(){

**return**(

<View style={styles.mapContainer}>

<MapView

style={styles.map}

initialRegion={{

latitude: 37.364216,

longitude: -120.425419,

latitudeDelta: 0.0922,

longitudeDelta: 0.0421,

}}

showsUserLocation = {**true**}

>

<MapView.Marker

coordinate={{

latitude: 37.364216,

longitude: -120.425419,

}}

/>

</MapView>

</View>

)

}

}

**const** RootStack = createStackNavigator({

Login: LoginScreen,

Home: HomeScreen,

Calendar: CalendarScreen,

Statistics: StatsScreen,

Profile: ProfileScreen,

Map: MapScreen,

},

{

headerMode: 'none',

}

);