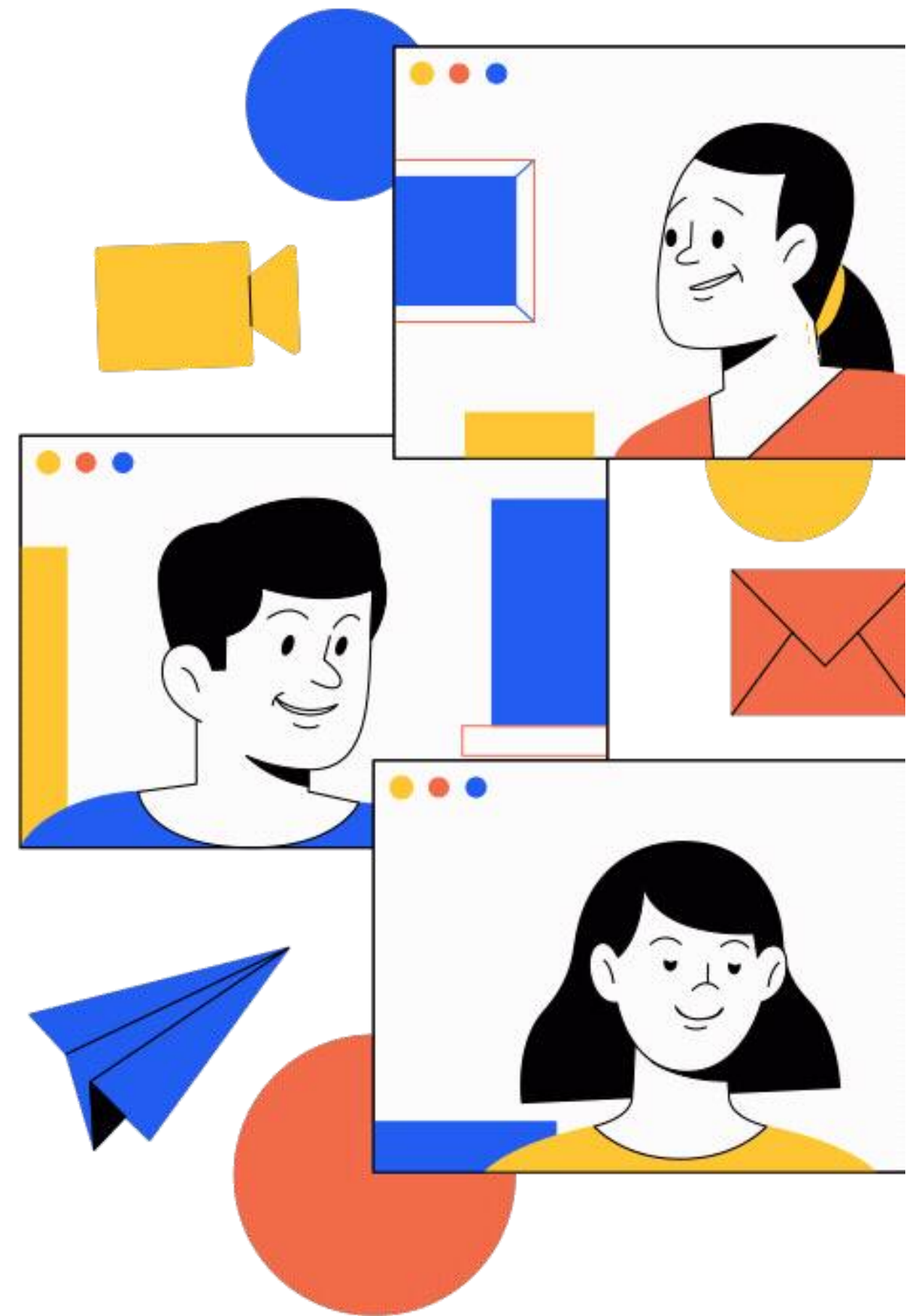


# Week 12 - React Native

No more URLs?

**Ahmed Fouad Lotfy**

React Session Lead



# Agenda



## What we'll cover in this session

- What is Navigation?
- Navigation Setup
- How to Create Navigation?
- Planning Mobile Navigation
- Live Demo

# What is Navigation?

- **The concept of navigation in mobile apps:**

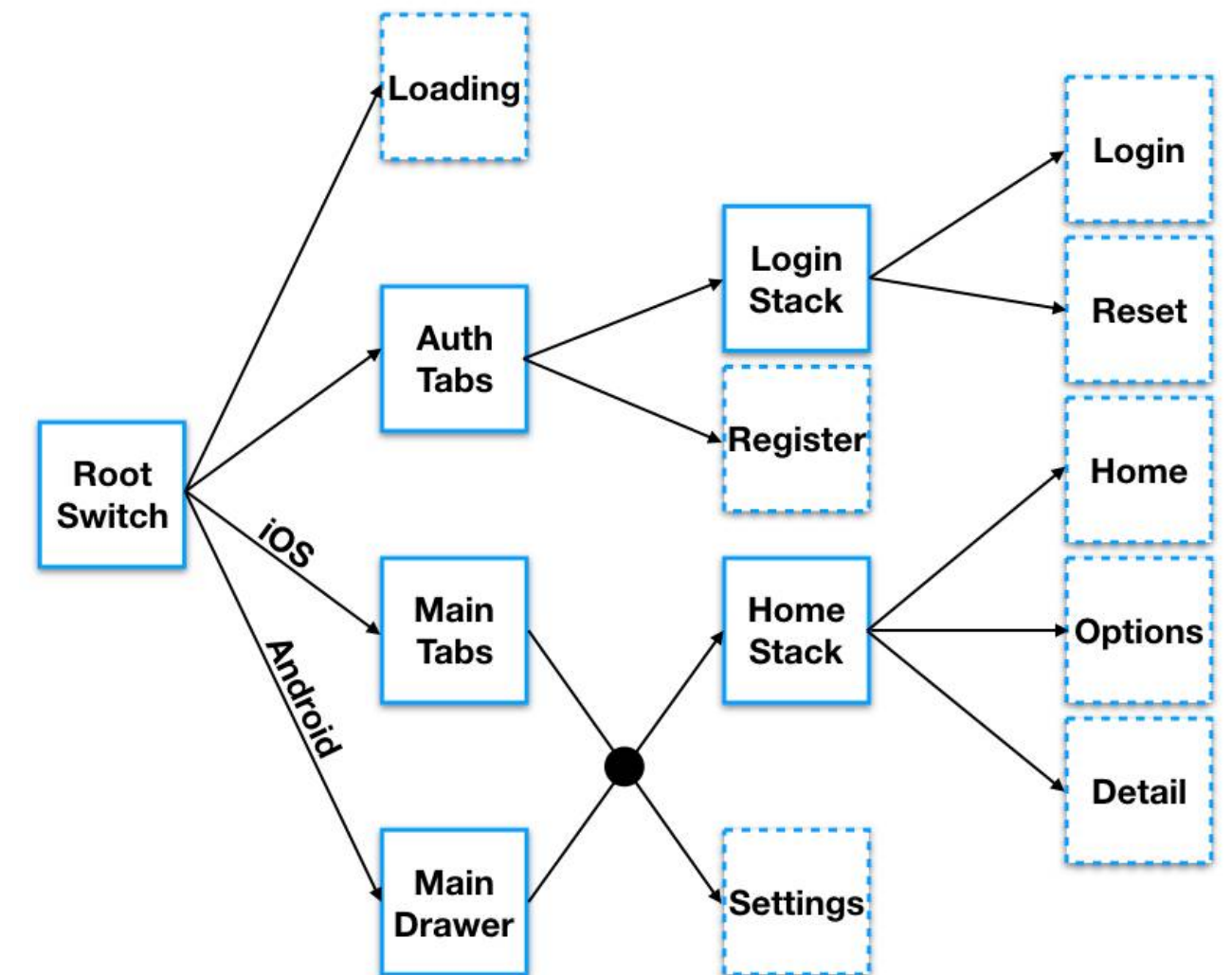
*Why do users need to navigate between screens?*

- **How navigation helps users move between screens:**

*Each screen should have a distinct Name.*

- **Keep navigation simple:**

*Ensure users can easily move between screens, this is done by navigation library by creating navigation map.*



# WEB



- **Navigation Method:** URL-based, using links, sidebars, tabs.
- **Tools:** React-router
- **Inputs:** Mouse clicks, hovers, keyboard shortcuts.
- **Design :** Focus on responsiveness for various screen sizes.

# Mobile

- **Navigation Method:** Gesture-based, stack navigation (swipes, taps).
- **Tools:** React Navigation
- **Inputs:** Touch gestures like swiping, tapping, long press.
- **Design:** Simplified, stacked layouts.
- **Platform-Specific:** Follows Material Design (Android) and HIG (iOS).

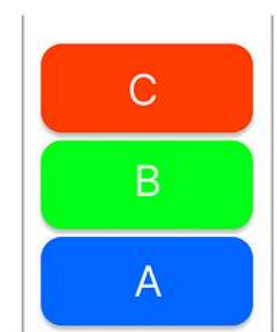
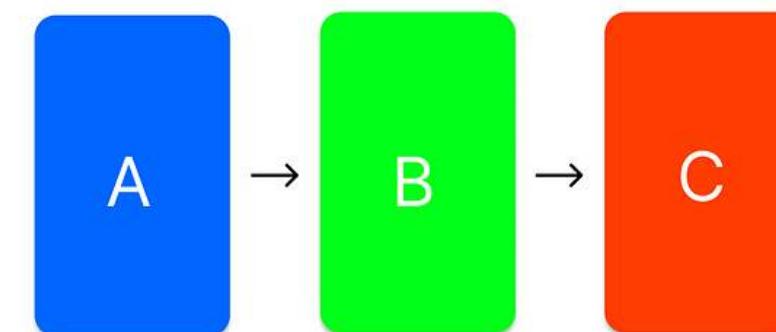
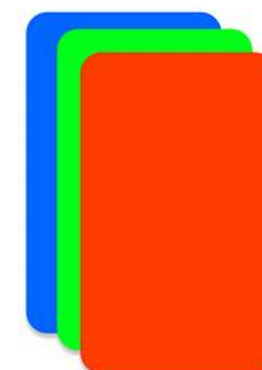
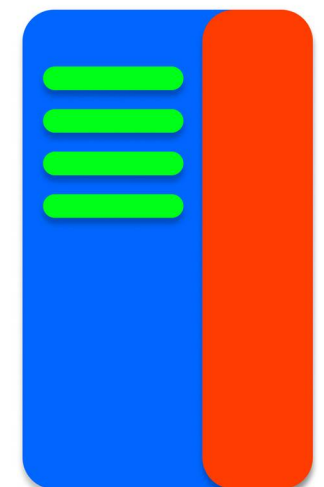
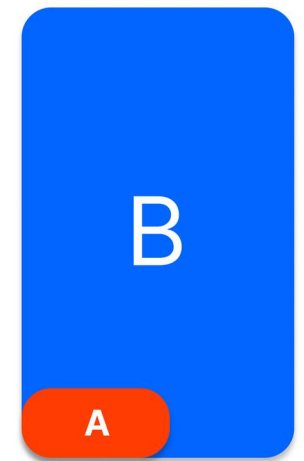
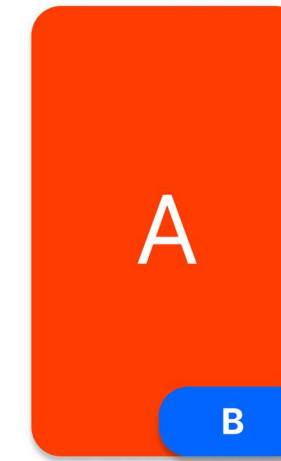
# Why use a navigation library?

- **Key benefits of using react-navigation:**

- *Easy to use*
- *Components built for iOS and Android*
- *Completely customizable*

- **Key features:**

*Stack, Tab, Drawer Navigators.*



# Installing React Navigation

- **Commands for installing react-navigation and dependencies:**

```
npm install @react-navigation/native  
npm install react-native-screens react-native-safe-area-context
```

# Creating a Basic Stack Navigator

1. **Import Library**: import createStackNavigator from @react-navigation/native-stack library.
2. **Create**: Create an instance from createNativeStackNavigator()
3. **Combine**: Combine your screens under Stack Navigator tag using unique names.
4. **Containerization**: Add a container to wrap your navigator.



# Creating a Basic Stack Navigator



```
import { createStackNavigator } from '@react-navigation/stack';

const Stack = createStackNavigator();

function MyStack() {
  return (
    <Stack.Navigator>
      <Stack.Screen name="Home" component={Home} />
      <Stack.Screen name="Notifications" component={Notifications} />
      <Stack.Screen name="Profile" component={Profile} />
      <Stack.Screen name="Settings" component={Settings} />
    </Stack.Navigator>
  );
}
```



# Creating a Basic Stack Navigator



```
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';

const Stack = createStackNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator>{/* ... */</Stack.Navigator>
    </NavigationContainer>
  );
}
```

# Linking Screens in Navigation

1. ***useNavigation***: `useNavigation` is a hook which gives access to navigation object. `useNavigation()` returns the navigation prop of the screen it's inside.
2. ***useLinkTo***: The `useLinkTo` hook lets us navigate to a screen using a path instead of a screen name based on the linking options. It returns a function that receives the path to navigate to.
3. ***useRoute***: `useRoute` is a hook which gives access to route object. `useRoute()` returns the route prop of the screen it's inside.

# Linking Screens in Navigation

```
import * as React from 'react';
import { Button } from 'react-native';
import { useNavigation } from '@react-navigation/native';

function MyBackButton() {
  const navigation = useNavigation();

  return (
    <Button
      title="Back"
      onPress={() => {
        navigation.goBack();
      }}
    />
  );
}
```

# Linking Screens in Navigation



```
import { useLinkTo } from '@react-navigation/native';

function Home() {
  const linkTo = useLinkTo();

  return (
    <Button onPress={() => linkTo('/profile/jane')}>
      Go to Jane's profile
    </Button>
  );
}
```

# Navigation Best Practices

1. Organize screens in a way that logically connects similar features.

**Example:** Group settings-related screens together, and product-related screens together.

**Best Practice:** Use nested navigators to keep a clean structure. For example, placing user profile screens in a stack within a tab navigator.



# Navigation Best Practices



```
<Tab.Navigator>
  <Tab.Screen name="Home" component={HomeScreen} />
  <Tab.Screen name="ProfileStack" component={ProfileStackScreen} />
</Tab.Navigator>

function ProfileStackScreen() {
  return (
    <Stack.Navigator>
      <Stack.Screen name="Profile" component={ProfileScreen} />
      <Stack.Screen name="Settings" component={SettingsScreen} />
    </Stack.Navigator>
  );
}
```



# Navigation Best Practices

2. Maintain Back Navigation for a smooth User Flow:

- Ensure users can easily navigate back through the app.
- Use `navigation.goBack()` effectively to handle smooth transitions.



```
function DetailsScreen({ navigation }) {  
  return (  
    <View>  
      <Text>Details Screen</Text>  
      <Button title="Go Back" onPress={() => navigation.goBack()} />  
    </View>  
  );  
}
```

# Customization

- Customize headers in navigation by using the options property of your stack screens.
- Customize elements such as title, background color, and text style.



```
<Stack.Screen
  name="Overview"
  component={OverviewScreen}
  options={{
    title: 'Overview',
    headerStyle: { backgroundColor: '#f4511e' },
    headerTintColor: '#fff',
    headerTitleStyle: { fontWeight: 'bold' }
  }}
/>
```

# Navigation Types

- 1. **Tab Navigation:** General-purpose wrapper for touchable elements, allowing custom styling.
- 2. **Drawer Navigation:** Predefined button component with basic styling.
- 3. **Stack Navigation:** More flexible touch handling, suitable for custom interactions.

Feature	Tab Navigation	Drawer Navigation	Stack Navigation
<i>Use Case</i>	Main sections or categories of an app	Apps with many sections or settings	Hierarchical or linear screen navigation
<i>UX Flow</i>	Hierarchical or linear screen navigation	Hidden navigation options in a side menu	Navigate through screens one by one
<i>Visibility</i>	Always visible at the bottom	Hidden until swiped or tapped	Only one screen visible at a time
<i>Customization</i>	Customizable icons and labels for each tab	Custom drawer items and icons	Custom transitions between screens
<i>Ideal for</i>	Apps with 3-5 sections (e.g., social media)	Apps with many sections (e.g., e-commerce)	Tasks with sequential steps (e.g., settings)

# Tab Navigator



```
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';

const Tab = createBottomTabNavigator();

function App() {
  return (
    <NavigationContainer>
      <Tab.Navigator>
        <Tab.Screen name="Home" component={HomeScreen} />
        <Tab.Screen name="Settings" component={SettingsScreen} />
      </Tab.Navigator>
    </NavigationContainer>
  );
}
```


# Tab Navigator Customization



```
<Tab.Screen
  name="Home"
  component={HomeScreen}
  options={{
    tabBarIcon: () => (<Icon name="home" size={24} />),
    tabBarLabel: 'Home'
  }}
/>
```



# Drawer Navigator



```
import { createDrawerNavigator } from '@react-navigation/drawer';
import { NavigationContainer } from '@react-navigation/native';

const Drawer = createDrawerNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Drawer.Navigator initialRouteName="Home">
        <Drawer.Screen name="Home" component={HomeScreen} />
        <Drawer.Screen name="Notifications" component={NotificationsScreen} />
      </Drawer.Navigator>
    </NavigationContainer>
  );
}
```

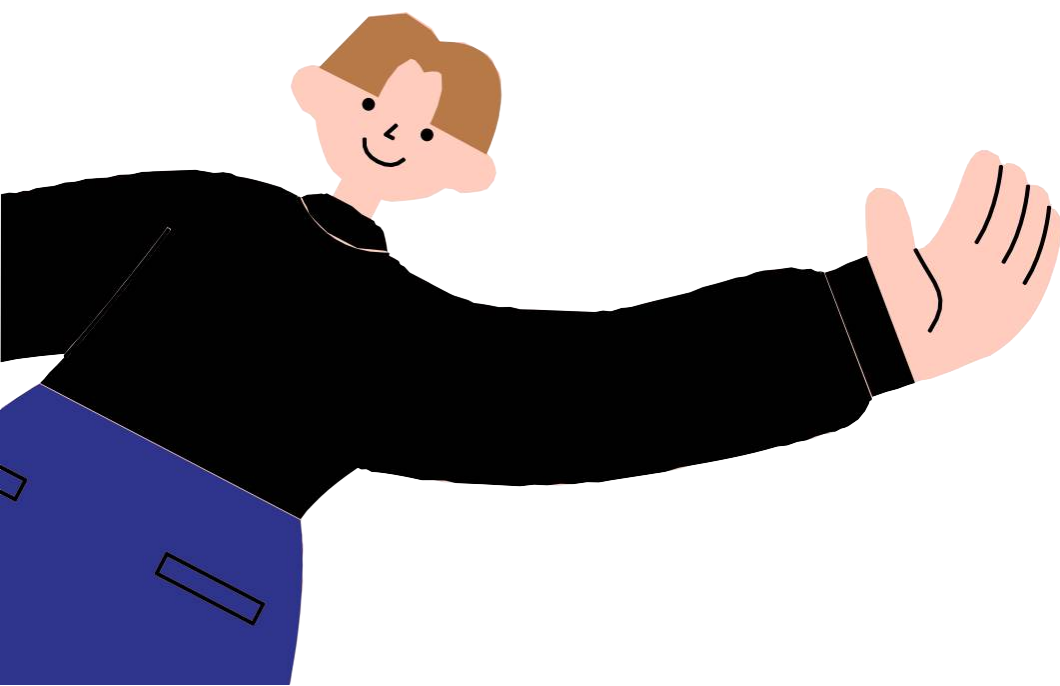


**Demo  
Time**



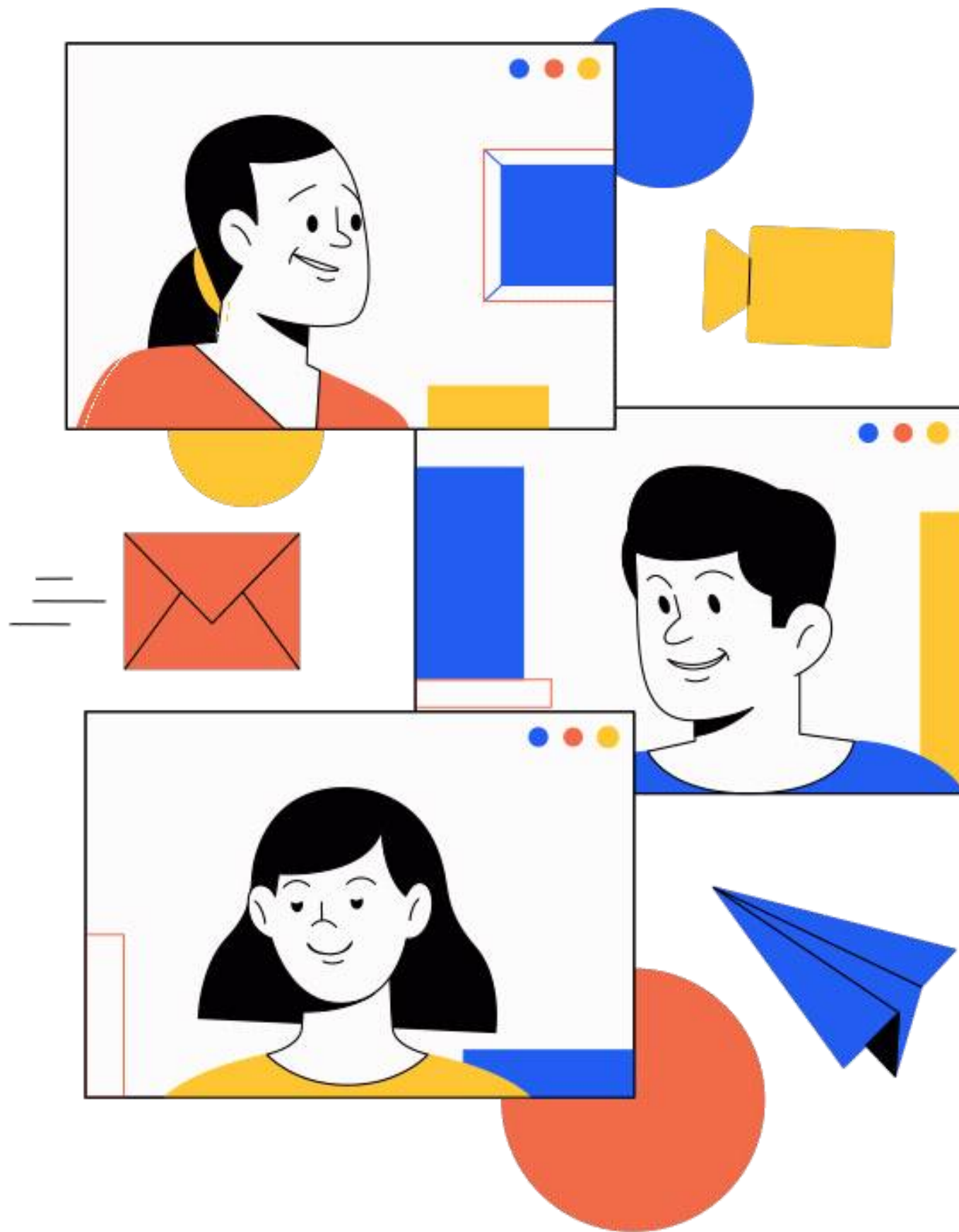


**Any Questions?**



**Next Session:**  
**React Native**  
**with Redux**





# Thank you for attending!

Feel free to email at [a.lotfy@fci-cu.edu.eg](mailto:a.lotfy@fci-cu.edu.eg) or reach me at circle anytime for any questions or clarifications!



Follow me on Github [@ahmeddx Fouad](https://github.com/ahmeddx Fouad)  
code and slides are found at this [github repo](#)