

MAD QUIZ 2

Submitted by:

Zil-e-huma 21-SE-17

Submitted to:

Engr. Kanwal Yousaf

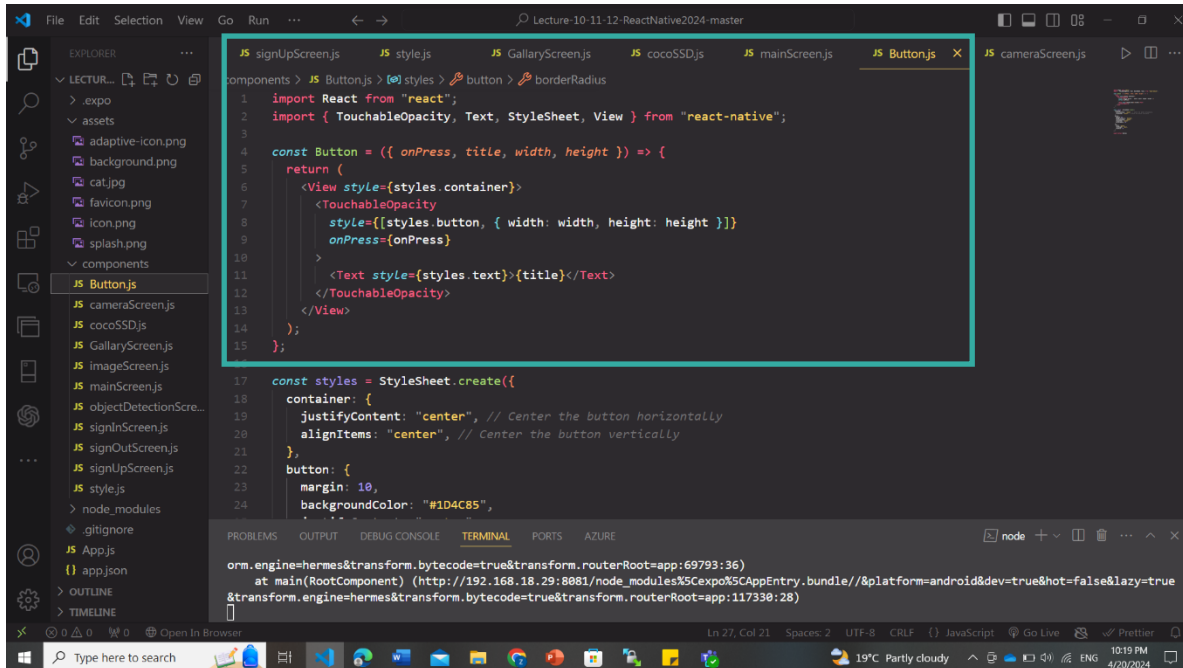
APRIL 21



Question #1: Enhancing App UI and Integrating AI Model

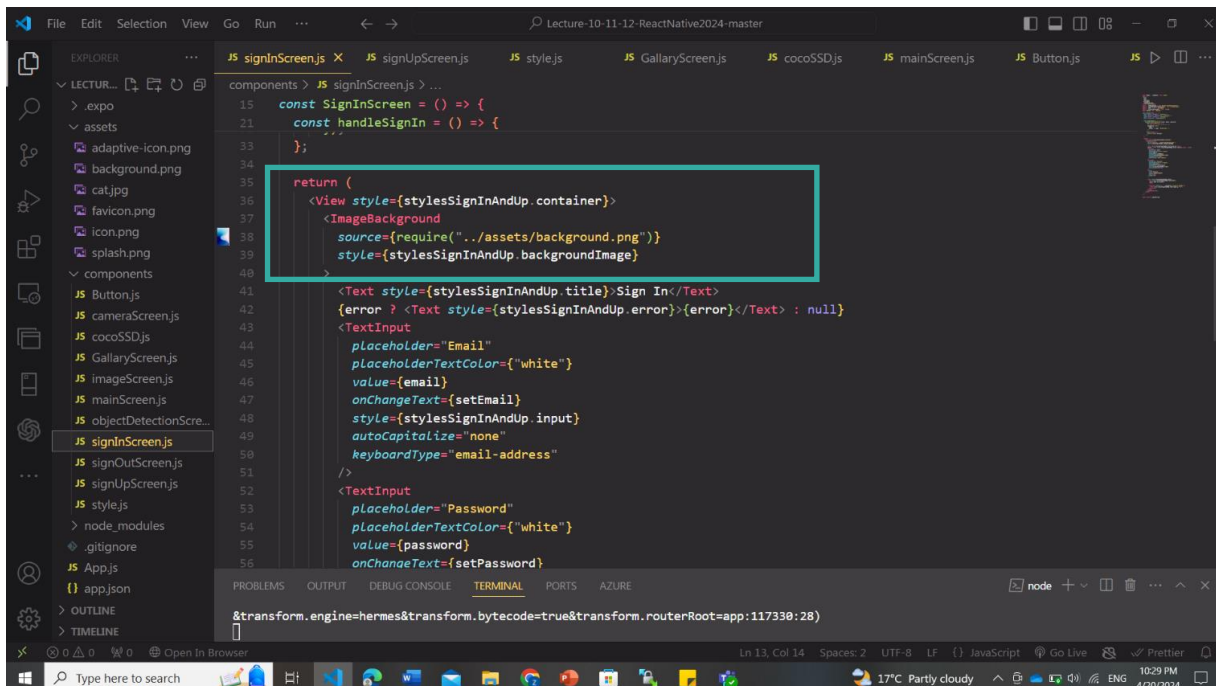
1,2,3. Code Reusable, UI Enhancement, Menu Integration:

Added a button component to make code reusable, rather than styling buttons for each screen again and again.

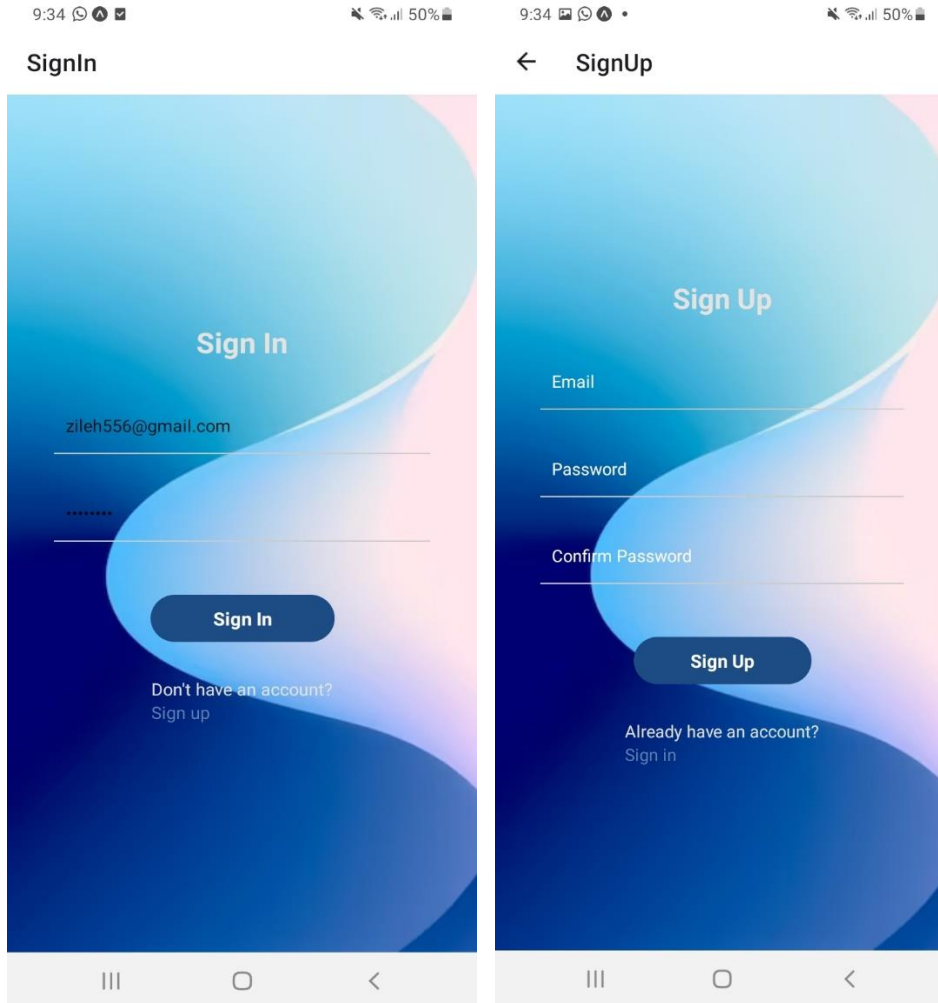


```
1 import React from "react";
2 import { TouchableOpacity, Text, StyleSheet, View } from "react-native";
3
4 const Button = ({ onPress, title, width, height }) => {
5   return (
6     <View style={styles.container}>
7       <TouchableOpacity
8         style={[styles.button, { width: width, height: height }]}
9         onPress={onPress}>
10         <Text style={styles.text}>{title}</Text>
11       </TouchableOpacity>
12     </View>
13   );
14 }
15
16
17 const styles = StyleSheet.create({
18   container: {
19     justifyContent: "center", // Center the button horizontally
20     alignItems: "center", // Center the button vertically
21   },
22   button: {
23     margin: 10,
24     backgroundColor: "#1D4C85",
25   },
26   text: {
27     color: "white",
28     font-weight: "bold",
29   },
30 });
```

Added background image in signup and sign in screens to make it visually appealing



```
15 const SignInScreen = () => {
16   const handleSignIn = () => {
17     // ...
18   };
19
20   return (
21     <View style={stylesSignInAndUp.container}>
22       <ImageBackground
23         source={require("../assets/background.png")}
24         style={stylesSignInAndUp.backgroundImage}>
25         <Text style={stylesSignInAndUp.title}>Sign In</Text>
26         {error ? <Text style={stylesSignInAndUp.error}>{error}</Text> : null}
27         <TextInput
28           placeholder="Email"
29           placeholderTextColor="white"
30           value={email}
31           onChangeText={setEmail}
32           style={stylesSignInAndUp.input}
33           autoCapitalize="none"
34           keyboardType="email-address"
35         />
36         <TextInput
37           placeholder="Password"
38           placeholderTextColor="white"
39           value={password}
40           onChangeText={setPassword}
41         />
42       </ImageBackground>
43     </View>
44   );
45 }
```



Bottom tab navigation is used to seamlessly navigate between screens. (Main.js)

```

1  import React from "react";
2  import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";
3  import { useNavigation } from "@react-navigation/native";
4  import { Feather } from "@expo/vector-icons";
5  import CameraScreen from "../cameraScreen";
6  import GalleryScreen from "../galleryScreen";
7  import ImageClassifier from "../objectDetectionScreen";
8  import CocoSSD from "../cocoSSD";
9  import { auth } from "../firebase";
10
11 const Tab = createBottomTabNavigator();
12
13 const MainScreen = () => {
14   const navigation = useNavigation();
15
16   const HandleSignOut = () => {
17     auth
18       .signOut()
19       .then(() => {
20         navigation.reset({
21           index: 0,
22           routes: [{ name: "SignIn" }],
23         });
24       });
25   };
26 }
  
```

Terminal output:

```

orm.engine=hermes&transform.bytecode=true&transform.routerRoot=app:69793:36)
at main(RootComponent) (http://192.168.18.29:8081/node_modules%5Cexpo%5CAppEntry.bundle//&platform=android&dev=true&hot=false&lazy=true
&transform.engine=hermes&transform.bytecode=true&transform.routerRoot=app:117330:28)
  
```

File Edit Selection View Go Run ... Lecture-10-11-12-ReactNative2024-master

EXPLORER

components > JS mainScreen.js > MainScreen > <function> > tabBarIcon

```
13 const MainScreen = () => {
14   const HandleSignOut = () => {
24   }
25   .catch((error) => {
26     console.log(error);
27   });
28 };
29 return (
30   <Tab.Navigator
31     screenOptions={({ route }) => ({
32       tabBarIcon: ({ color, size, focused }) => {
33         let iconName;
34         if (route.name === "Capture") {
35           iconName = "camera";
36         } else if (route.name === "Gallery") {
37           iconName = "image";
38         } else if (route.name === "Classifier") {
39           iconName = "aperture";
40         } else if (route.name === "ObjectDetection") {
41           iconName = "search";
42         } else if (route.name === "SignOut") {
43           iconName = "log-out";
44         }
45       }
46     })
47   >
48     <Tab.Screen name="Capture" component={CameraScreen} />
49     <Tab.Screen name="Gallery" component={GalleryScreen} />
50     <Tab.Screen name="Classifier" component={ImageClassifier} />
51     <Tab.Screen name="ObjectDetection" component={CocoSSD} />
52     <Tab.Screen
53       name="SignOut"
54       component={() => null}
55       listeners={({ navigation }) => ({
56         tabPress: (e) => {
57           e.preventDefault();
58           HandleSignOut();
59         }
60       })}
61     />
62   </Tab.Navigator>
63 );
64 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

```
node + v ...
orm.engine=hermes&transform.bytecode=true&transform.routerRoot=app:69793:36)
at main(RootComponent) (http://192.168.18.29:8881/node_modules%SCexpo%SCAppEntry.bundle//&platform=android&dev=true&hot=false&lazy=true
&transform.engine=hermes&transform.bytecode=true&transform.routerRoot=app:117330:28)
```

Ln 44, Col 12 Spaces: 2 UTF-8 CRLF {} JavaScript Go Live Prettier

Type here to search

File Edit Selection View Go Run ... Lecture-10-11-12-ReactNative2024-master

EXPLORER

components > JS mainScreen.js > MainScreen > <function> > tabBarIcon

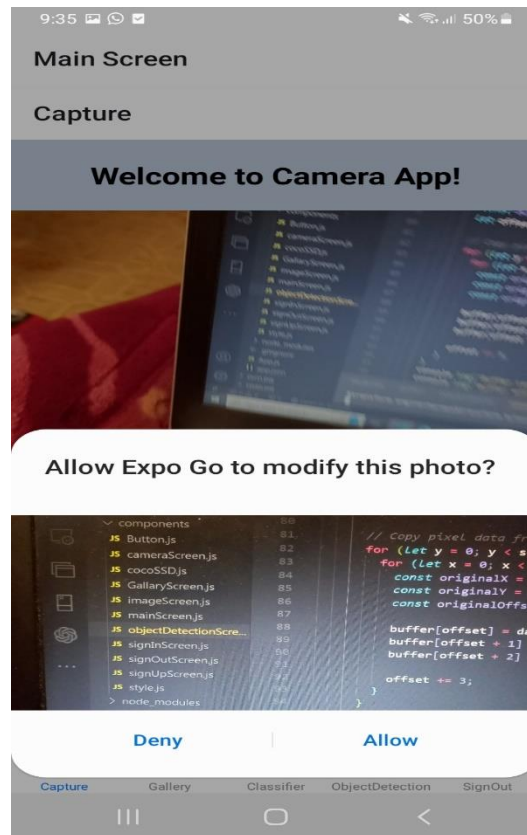
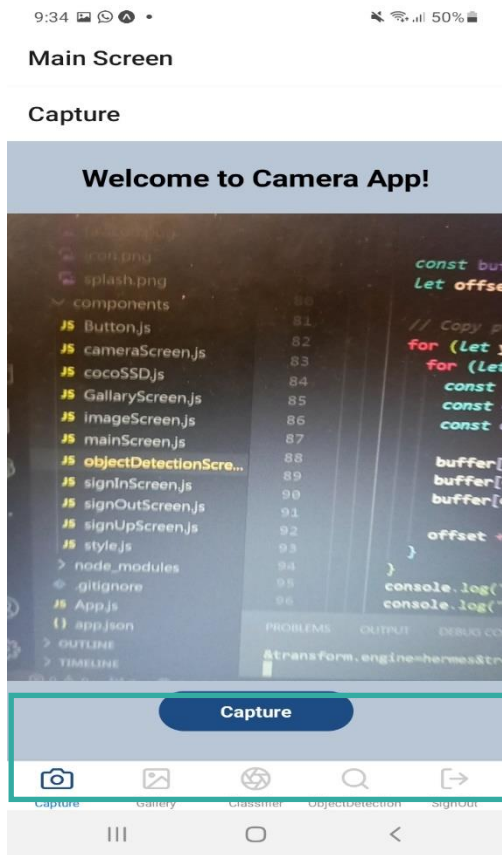
```
31 const MainScreen = () => {
32   screenOptions={({ route }) => ({
33     tabBarIcon: ({ color, size, focused }) => {
47       return <Feather name={iconName} size={size} color={iconColor} />;
48     },
49   })
50 }
51 <Tab.Navigator
52   screenOptions={({ route }) => ({
53     tabBarIcon: ({ color, size, focused }) => {
54       return <Feather name={iconName} size={size} color={iconColor} />;
55     },
56   })
57 >
58   <Tab.Screen name="Capture" component={CameraScreen} />
59   <Tab.Screen name="Gallery" component={GalleryScreen} />
60   <Tab.Screen name="Classifier" component={ImageClassifier} />
61   <Tab.Screen name="ObjectDetection" component={CocoSSD} />
62   <Tab.Screen
63     name="SignOut"
64     component={() => null}
65     listeners={({ navigation }) => ({
66       tabPress: (e) => {
67         e.preventDefault();
68         HandleSignOut();
69       }
70     })}
71   />
72 </Tab.Navigator>
73 );
74 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

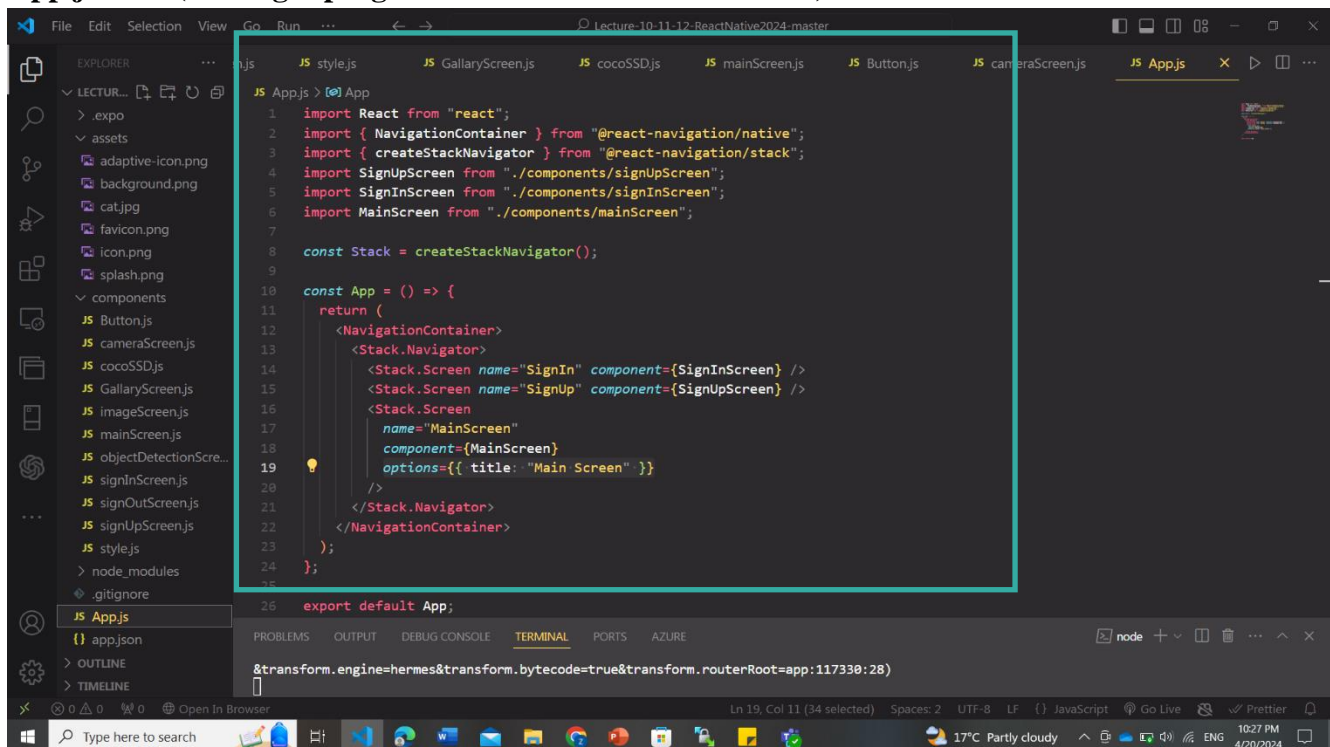
```
node + v ...
orm.engine=hermes&transform.bytecode=true&transform.routerRoot=app:69793:36)
at main(RootComponent) (http://192.168.18.29:8881/node_modules%SCexpo%SCAppEntry.bundle//&platform=android&dev=true&hot=false&lazy=true
&transform.engine=hermes&transform.bytecode=true&transform.routerRoot=app:117330:28)
```

Ln 44, Col 12 Spaces: 2 UTF-8 CRLF {} JavaScript Go Live Prettier

Type here to search



App.js code (how signup sign in and the main screen linked):



4. AI model Integration:

Enhanced version of already present AI.

9:53 48%

48%

Main Screen

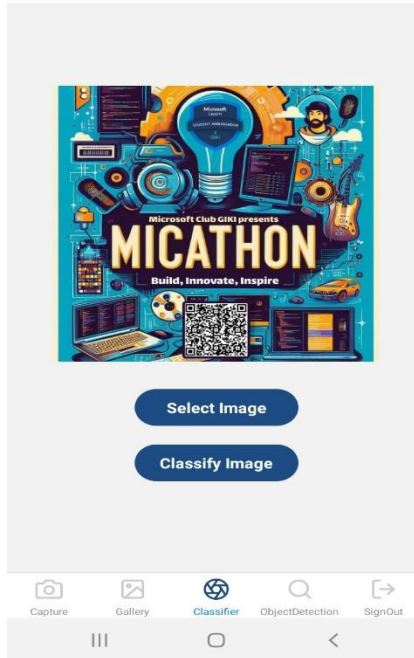
Classifier



9:51 48%

Main Screen

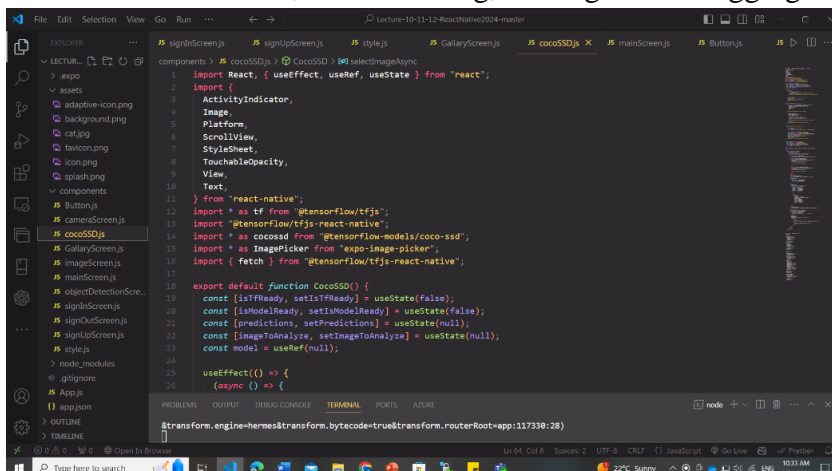
Classifier

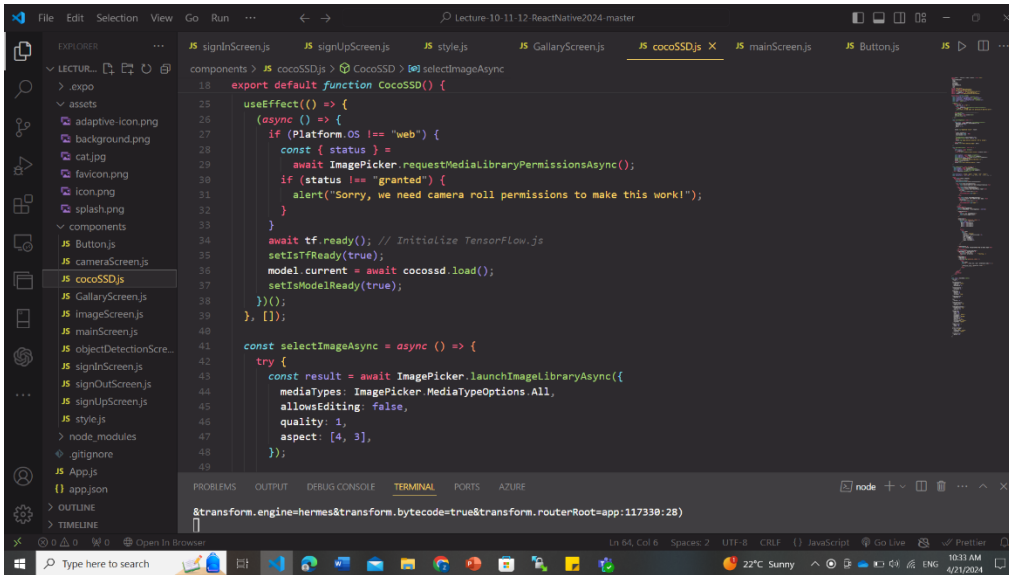


A new AI model:

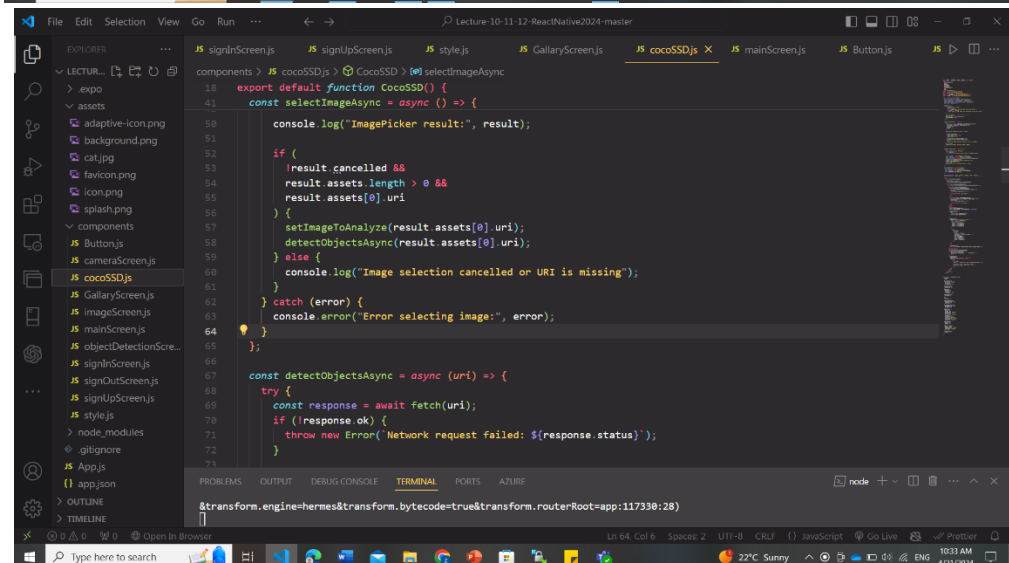
I implemented coco-ssd a pre-trained tensor flow object detection AI model.

This React Native component uses TensorFlow.js and the COCO-SSD model to detect objects in images. It initializes TensorFlow and the COCO-SSD model, then allows users to select an image from their device using Expo's Image Picker. After an image is selected, the code converts it into a tensor and uses the model to detect objects, returning a list of predictions with object classes and confidence scores. The component displays the selected image, drawing bounding boxes around detected objects, and lists predictions with object classes and scores. It includes status indicators for TensorFlow and COCO-SSD readiness, error handling, and logs for debugging.

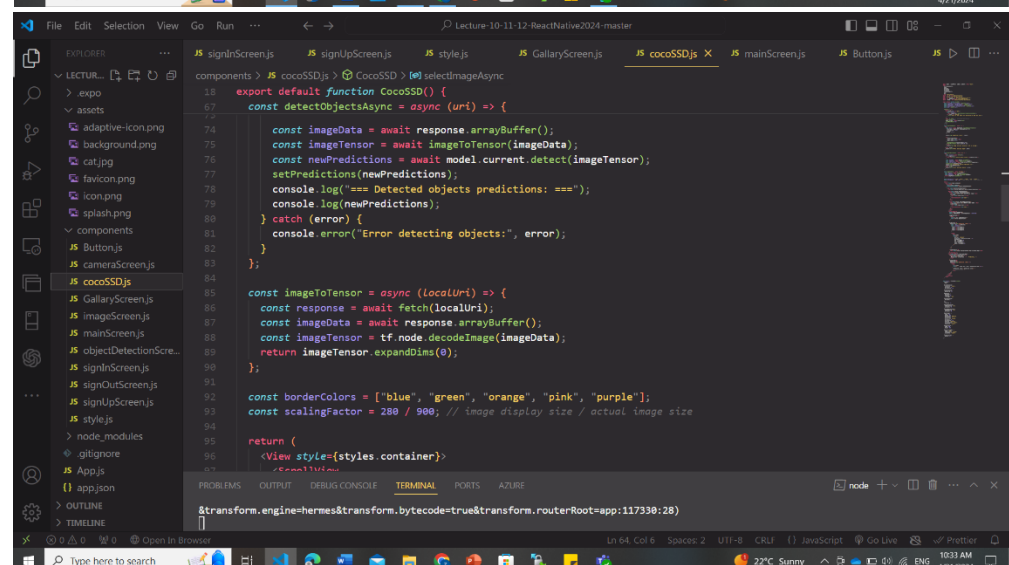




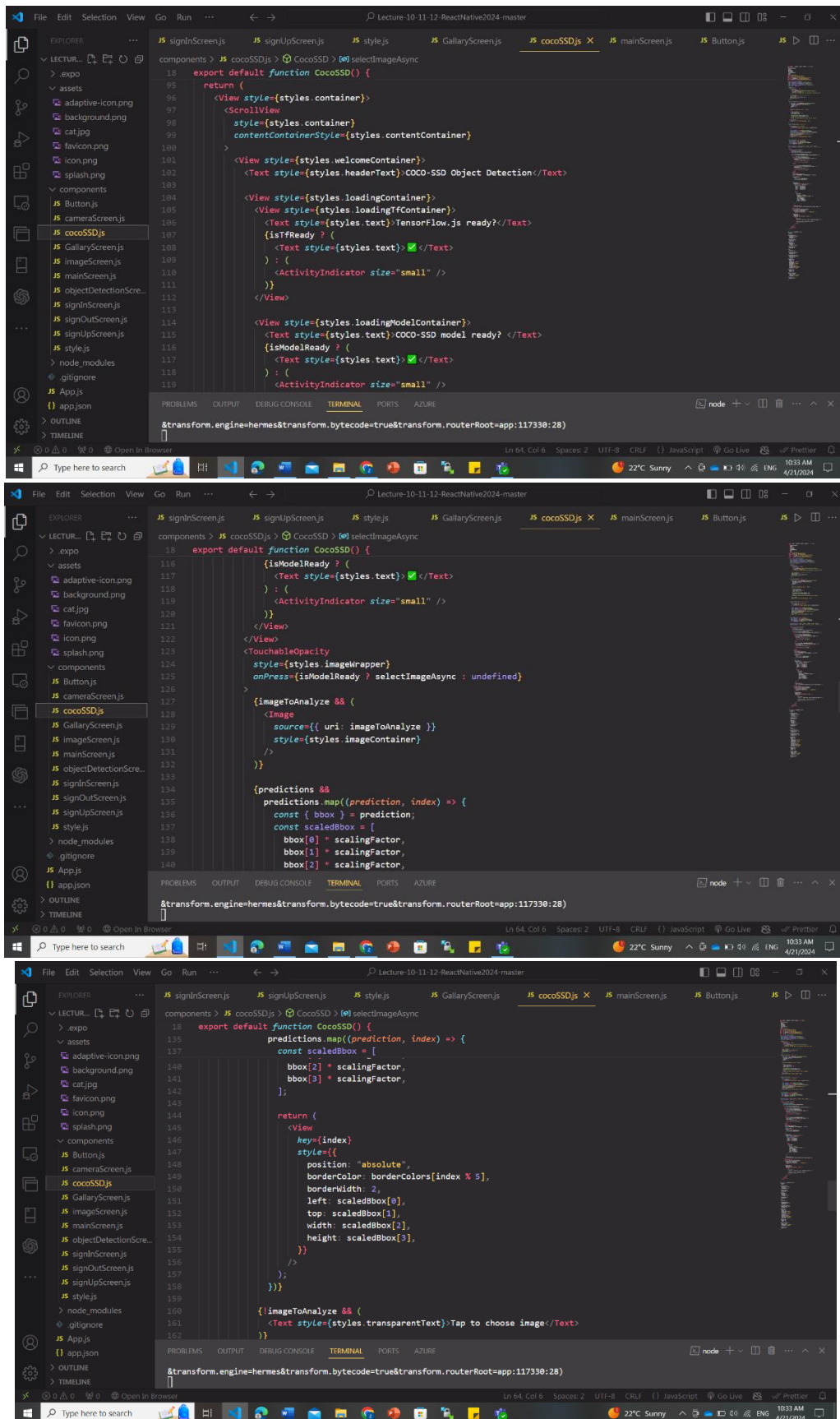
```
components > JS cocoSSD.js > CocoSSD > selectImageAsync
18 export default function CocoSSD() {
25   useEffect(() => {
26     (async () => {
27       if (Platform.OS !== "web") {
28         const { status } =
29           await ImagePicker.requestMediaLibraryPermissionsAsync();
30         if (status !== "granted") {
31           alert("Sorry, we need camera roll permissions to make this work!");
32         }
33       }
34       await tf.ready(); // Initialize TensorFlow.js
35       setIsTfReady(true);
36       model.current = await cocoSSD.load();
37       setIsModelReady(true);
38     })();
39   }, []);
40
41   const selectImageAsync = async () => {
42     try {
43       const result = await ImagePicker.launchImageLibraryAsync({
44         mediaTypes: ImagePicker.MediaTypeOptions.All,
45         allowsEditing: false,
46         quality: 1,
47         aspect: [4, 3],
48       });
49     } catch (error) {
50       console.error("Image selection cancelled or URI is missing");
51     }
52   };
53
54   const detectObjectsAsync = async (uri) => {
55     try {
56       const response = await fetch(uri);
57       if (!response.ok) {
58         throw new Error("Network request failed: ${response.status}");
59       }
60     } catch (error) {
61       console.error("Error detecting objects:", error);
62     }
63   };
64
65   const imageToTensor = async (localUri) => {
66     const response = await fetch(localUri);
67     const imageData = await response.arrayBuffer();
68     const imageTensor = tf.node.decodeImage(imageData);
69     return imageTensor.expandDims(0);
70   };
71
72   const borderColors = ["blue", "green", "orange", "pink", "purple"];
73   const scalingFactor = 280 / 960; // image display size / actual image size
74
75   return (
76     <View style={styles.container}>
77       <Text>CocoSSD</Text>
78       <ImagePicker result={result} uri={uri} />
79       <Text>Detect Objects</Text>
80       <Text>Image to Tensor</Text>
81     </View>
82   );
83 }
```



```
components > JS cocoSSD.js > CocoSSD > selectImageAsync
41 const selectImageAsync = async () => {
50   console.log("ImagePicker result:", result);
51
52   if (
53     !result.cancelled &&
54     result.assets.length > 0 &&
55     result.assets[0].uri
56   ) {
57     setImageToAnalyze(result.assets[0].uri);
58     detectObjectsAsync(result.assets[0].uri);
59   } else {
60     console.log("Image selection cancelled or URI is missing");
61   }
62 } catch (error) {
63   console.error("Error selecting image:", error);
64 }
65
66 const detectObjectsAsync = async (uri) => {
67   try {
68     const response = await fetch(uri);
69     if (!response.ok) {
70       throw new Error("Network request failed: ${response.status}");
71     }
72   } catch (error) {
73     console.error("Error detecting objects:", error);
74   }
75 }
76
77 const imageToTensor = async (localUri) => {
78   const response = await fetch(localUri);
79   const imageData = await response.arrayBuffer();
80   const imageTensor = tf.node.decodeImage(imageData);
81   return imageTensor.expandDims(0);
82 }
83
84 const borderColors = ["blue", "green", "orange", "pink", "purple"];
85 const scalingFactor = 280 / 960; // image display size / actual image size
86
87 return (
88   <View style={styles.container}>
89     <Text>CocoSSD</Text>
90     <ImagePicker result={result} uri={uri} />
91     <Text>Detect Objects</Text>
92     <Text>Image to Tensor</Text>
93   </View>
94 );
95 }
```



```
components > JS cocoSSD.js > CocoSSD > selectImageAsync
18 export default function CocoSSD() {
25   useEffect(() => {
26     (async () => {
27       if (Platform.OS !== "web") {
28         const { status } =
29           await ImagePicker.requestMediaLibraryPermissionsAsync();
30         if (status !== "granted") {
31           alert("Sorry, we need camera roll permissions to make this work!");
32         }
33       }
34       await tf.ready(); // Initialize TensorFlow.js
35       setIsTfReady(true);
36       model.current = await cocoSSD.load();
37       setIsModelReady(true);
38     })();
39   }, []);
40
41   const selectImageAsync = async () => {
42     try {
43       const result = await ImagePicker.launchImageLibraryAsync({
44         mediaTypes: ImagePicker.MediaTypeOptions.All,
45         allowsEditing: false,
46         quality: 1,
47         aspect: [4, 3],
48       });
49     } catch (error) {
50       console.error("Image selection cancelled or URI is missing");
51     }
52   };
53
54   const detectObjectsAsync = async (uri) => {
55     try {
56       const response = await fetch(uri);
57       if (!response.ok) {
58         throw new Error("Network request failed: ${response.status}");
59       }
60     } catch (error) {
61       console.error("Error detecting objects:", error);
62     }
63   };
64
65   const imageToTensor = async (localUri) => {
66     const response = await fetch(localUri);
67     const imageData = await response.arrayBuffer();
68     const imageTensor = tf.node.decodeImage(imageData);
69     return imageTensor.expandDims(0);
70   };
71
72   const borderColors = ["blue", "green", "orange", "pink", "purple"];
73   const scalingFactor = 280 / 960; // image display size / actual image size
74
75   return (
76     <View style={styles.container}>
77       <Text>CocoSSD</Text>
78       <ImagePicker result={result} uri={uri} />
79       <Text>Detect Objects</Text>
80       <Text>Image to Tensor</Text>
81     </View>
82   );
83 }
```

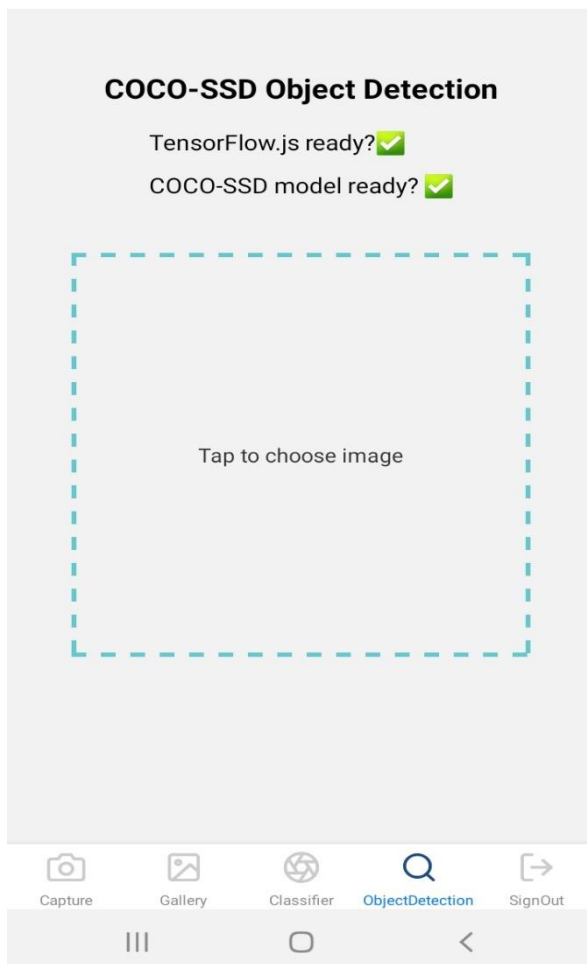



```
components > JS cocoSSD.js > selectImageAsync
18 export default function CocoSSD() {
19   predictions.map((prediction, index) => {
20     })
21   }
22   {imageToAnalyze && (
23     <Text style={styles.transparentText}>Tap to choose image:</Text>
24   )}
25   </TouchableOpacity>
26   <View style={styles.predictionWrapper}>
27     {isModelReady && imageToAnalyze && (
28       <Text style={styles.text}>
29         Predictions: {predictions ? "" : "Predicting..."}
30       </Text>
31     )}
32   )}
33   {isModelReady &&
34     predictions &&
35     predictions.map((prediction, index) => {
36       return (
37         <Text
38           key={index}
39           style={{...styles.text, color: borderColors[index % 5]}}
40         >
41           {prediction.class}: {prediction.score}{" "}
42           {/" prediction.bbox "/}
43         </Text>
44       )
45     })
46   }
47 }
```

9:53 48%

Main Screen

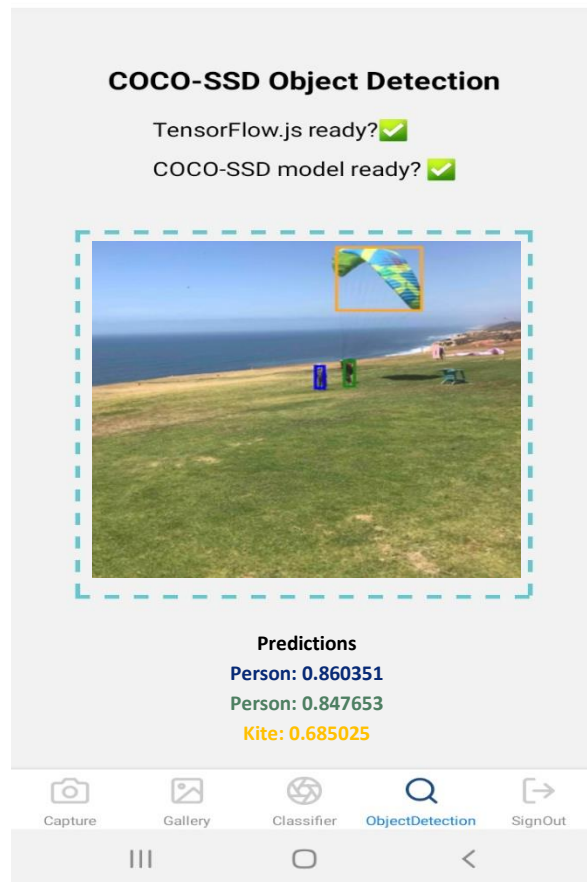
ObjectDetection



9:53 48%

Main Screen

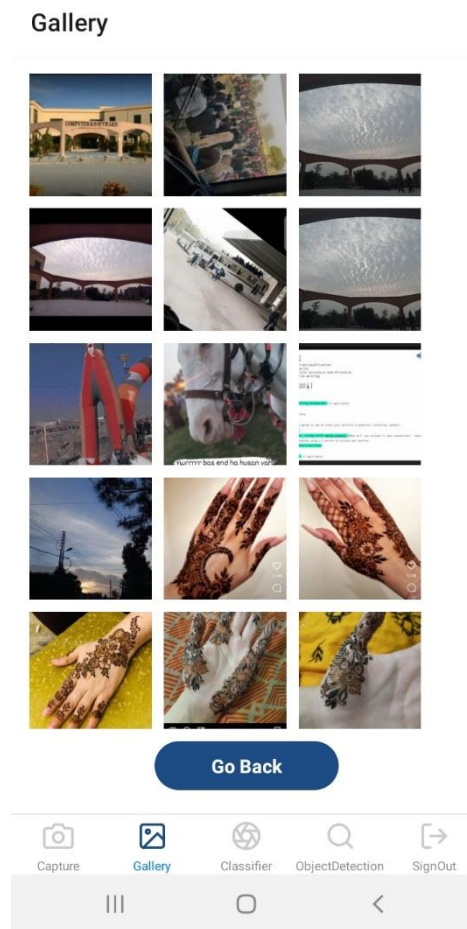
ObjectDetection



Gallery Screen:



Main Screen



Description:

This project is an application that begins by requiring users to sign up or sign in to access its features. After successful authentication, users can navigate through several screens using a bottom tab interface. The key screens include a camera screen for capturing and saving photos, a gallery screen to view saved images, an object classification screen using a TensorFlow-based AI model, and an object detection screen utilizing the COCO-SSD TensorFlow model. The app also includes a sign-out button to allow users to log out when needed. The design ensures a seamless user experience with intuitive navigation and integration of AI-powered image processing capabilities.

GitHub Repository Link:

https://github.com/Zil-e-huma01/MAD-Quiz02_AI-model-and-firebase-storage