Topic	THEMES	
Class Description	**In this class, the student will be building and then integrating the light theme into the app, so that the user can choose between the themes they prefer.**	
Class	C87	
Class time	45 mins	
Goal	• Creating the light theme. • Integrating themes into the app so that users can choose between both themes.	

Resources Required	• Teacher Resources
	  ○ Visual Studio Code Editor
	  ○ laptop with internet connectivity
	  ○ earphones with mic
	  ○ notebook and pen
	
	• Student Resources
	  ○ Visual Studio Code Editor
	  ○ laptop with internet connectivity
	  ○ earphones with mic
	  ○ notebook and pen	

Class structure	Warm-Up Teacher-led Activity Student-led Activity Wrap-Up	5 mins 15 mins 20 mins 5 mins

WARM-UP SESSION - 5 mins

Teacher starts slideshow from slides 1 to 12
Refer to speaker notes and follow the instructions on each slide.

Activity details	Solution/Guidelines

Run the presentation from slide 1 to slide 4. **The following are the warm-up session deliverables:** • Revise the previous lesson. • Themes. • Warm-Up Quiz Session.	**ESR**: Varied Response. Click on the slide show tab and present the slides.
QnA Session	
Question	**Answer**

What does the following piece of code do?
 	
``
let customFonts = {
'Bubblegum-Sans': require('../assets/fonts/BubblegumSans-Regular.ttf'),
};
``
A. We are setting the font.
B. We are downloading the fonts.
C. We are importing the fonts from the ttf file.
D. None of the above.
	C

What does the firebase.auth().signOut() function do? A. It allows the user to login. B. It allows the user to logout of the app. C. It allows the user to register to the app. D. It allows the user to go to the next screen.	**B**
Activity details	**Solution/Guidelines**
Run the presentation from slide 5 to slide 12 to set the problem statement. **The following are the warm-up session deliverables:** • Introduce the concepts of Teacher-Led Activity.	Narrate the slides by using hand gestures and voice modulation methods to bring in more student interest.

Teacher ends slideshow		
TEACHER-LED ACTIVITY 15 mins		
Teacher Initiates Screen Share		
<u>CHALLENGE</u> • **Building the light theme for the profile screen.**		
Step 2: Teacher-led Activity (15 min)	Now that we have successfully added the code that updates the theme which the user prefers, between light and dark, we can use this value from the DB in multiple screens to display their preferred theme. But the first thing that we want to do is to go to our firebase database and change one of our user's theme to light for testing as shown below - ***Note** - If the student and/or teacher is using the snack editor for these classes, please refer to the support document in [Teacher Activity 4]*	

	Great! Now let's start with the Profile Screen. Why the Profile Screen, can you guess?	**ESR:** Because we are fetching the data of users on that screen.
	That's right! Now the idea to implement different themes is that we will create multiple styling and apply the styles based on what the user prefers. We have a state called **light_theme** in our Profile Screen, which is **true** if the user has preferred a light theme and **false** if the user has preferred the dark theme. If we apply an **if-else** condition to this state and apply different styles based on it, we can actually style our app differently.	

	Let’s take a look at an example -	

If we change our first view from this in **Profile.js**.

```
<View style={styles.container}>
```

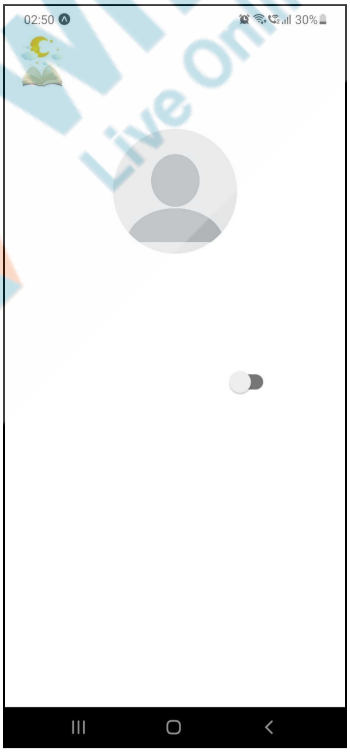
To this:

```
<View style={this.state.light_theme ? styles.containerLight : styles.container}>
```

And add the styles for **containerLight -**

```
containerLight: {
    flex: 1,
    backgroundColor: "white"
},
```

Our app would look like this -



	This way, since the user has selected the **light theme**, we changed our app's background to **white**. We did it just with the help of simple conditions in our styling. We can do this for other elements too. ***Note**: Teacher to open code from the previous class and starts modifying it. The code is also provided - [Teacher Activity 1](#).* Our code would become - *The teacher changes the code in **Profile.js.*** ***Note:** The changes are highlighted for reference purposes.*	*The student observes the changes.*

```
return (
                <View style={this.state.light_theme ? styles.containerLight :
styles.container}>
                    <SafeAreaView style={styles.droidSafeArea} />
                    <View style={styles.appTitle}>
                        <View style={styles.appIcon}>
                            <Image source={require("../assets/logo.png")} style={{ width:
60, height: 60, resizeMode: 'contain', marginLeft: 10 }}></Image>
                        </View>
                        <View style={styles.appTitleTextContainer}>
                            <Text style={this.state.light_theme ?
styles.appTitleTextLight : styles.appTitleText}>
                                Storytelling App
                            </Text>
                        </View>
                    </View>
                    <View style={styles.screenContainer}>
```

```
                        <View style={styles.profileImageContainer}>
                            <Image source={{ uri: this.state.profile_image }}
style={styles.profileImage}></Image>
                        </View>
                        <View style={styles.nameContainer}>
                            <Text style={this.state.light_theme ? styles.nameTextLight :
styles.nameText}>{this.state.name}</Text>
                        </View>
                        <View style={styles.themeContainer}>
                            <View style={styles.themeTextContainer}>
                                <Text style={this.state.light_theme ?
styles.themeTextLight : styles.themeText}>Dark Theme</Text>
                            </View>
                            <View style={styles.switchContainer}>
                                <Switch
                                    style={{ transform: [{ scaleX: 1.3 }, { scaleY: 1.3
}] }}
                                    trackColor={{ false: "#767577", true:
this.state.light_theme ? "#eee" : "white" }}
                                    thumbColor={this.state.isEnabled ? "#ee8249" :
"#f4f3f4"}
                                    ios_backgroundColor="#3e3e3e"
                                    onValueChange={() => this.toggleSwitch()}
                                    value={this.state.isEnabled}
                                />
                            </View>
                        </View>
                    </View>
                </View>
            )
```

We have just added conditions to styles where applicable.

Our styles would be as shown below -

```
const styles = StyleSheet.create({
   container: {
       flex: 1,
       backgroundColor: "#15193c"
   },
```

```
  containerLight: {
      flex: 1,
      backgroundColor: "white"
  },
  droidSafeArea: {
      marginTop: Platform.OS === "android" ? StatusBar.currentHeight : 0
  },
  appTitle: {
      flex: 0.07,
      flexDirection: "row",
      flexWrap: "wrap",
      padding: 5,
  },
  appIcon: {
      flex: 0.3
  },
  appTitleTextContainer: {
      justifyContent: "center",
      alignItems: "center"
  },
  appTitleText: {
      color: "white",
      fontSize: 28,
      fontFamily: "Bubblegum-Sans",
      paddingLeft: 20
  },
  appTitleTextLight: {
      color: "black",
      fontSize: 28,
      fontFamily: "Bubblegum-Sans",
      paddingLeft: 20
  },
  screenContainer: {
      flex: 0.85
  },
  profileImageContainer: {
      flex: 0.3,
      marginTop: 50,
      alignItems: "center"
  },
```

```
profileImage: {
    width: 150,
    height: 150,
    borderRadius: 150 / 2,

},
nameContainer: {
    flex: 0.1,
    alignItems: "center"
},
nameText: {
    color: "white",
    fontSize: 40,
    fontFamily: "Bubblegum-Sans",
},
nameTextLight: {
    color: "black",
    fontSize: 40,
    fontFamily: "Bubblegum-Sans",
},
themeContainer: {
    flexDirection: "row",
    justifyContent: "center",
    paddingTop: 80
},
themeTextContainer: {
    alignItems: "center",
    flex: 0.5
},
themeText: {
    color: "white",
    fontSize: 30,
    fontFamily: "Bubblegum-Sans",
},
themeTextLight: {
    color: "black",
    fontSize: 30,
    fontFamily: "Bubblegum-Sans",
},
switchContainer: {
```

```
        justifyContent: "center",
        alignItems: "center"
    }
});
```

	And that's all we had to do for having two themes in our app. If we wanted to have more than two, we could have separated the stylesheet into a different file and imported the style file we wanted based on the user's preferred theme. Let's implement the same changes for the light theme in the Feed Screen now. Remember that we also would have to fetch the user to get their preferred theme in Feed Screen as well.	

Teacher Stops Screen Share

Teacher starts slideshow for slide 13 to 15

Refer to speaker notes and follow the instructions on each slide.

STUDENT-LED ACTIVITY 20 mins

- **Ask the student to press the ESC key to come back to the panel.**
- **Guide the student to start screen share.**
- **Teacher gets into fullscreen.**

ACTIVITY

- **The student builds the light theme for the Feed Screen**

Step 3: Student-Led Activity (20 mins)	Please refer to *Student Activity 1* to clone the boilerplate code. Don't forget to add the **config.js** with your credentials in it. You will also have to update the OAuth IDs in the login screen.	*The student refers to Student Activity 1, clones the repo and adds config.js.*

We will first import our **firebase** module -

```
import firebase from "firebase";
```

Then change this line -

```
const BottomTabNavigator = () => {
```

To this line -

```
export default class BottomTabNavigator extends Component {
```

(Don't forget to import **{Component}** from "**react**" above).

```
import React, {Component} from 'react';
```

We will then add our **Constructor** and **componentDidMount()** functions -

```
constructor(props) {
    super(props);
    this.state = {
        light_theme: true
    };
}

componentDidMount() {
    let theme;
    firebase
        .database()
        .ref("/users/" + firebase.auth().currentUser.uid)
        .on("value", function (snapshot) {
            theme = snapshot.val().current_theme
        })
    this.setState({ light_theme: theme === "light" ? true : false })
}
```

Then wrap our **return()** function inside a **render()** function -

```
render() {
    return (
        <Tab.Navigator
            labeled={false}
            barStyle={styles.bottomTabStyle}
            screenOptions={({ route }) => ({
                tabBarIcon: ({ focused, color, size }) => {
                    let iconName;
                    if (route.name === 'Feed') {
                        iconName = focused
                            ? 'home'
                            : 'home-outline';
                    } else if (route.name === 'Create Story') {
                        iconName = focused ? 'add-circle' : 'add-circle-outline';
                    }
                    return <Ionicons name={iconName} size={30} color={color} style={{ w
                },
            })}
            activeColor={'#ee8249'}
            inactiveColor={'gray'}
        >
            <Tab.Screen name="Feed" component={Feed} />
            <Tab.Screen name="Create Story" component={CreateStory} />
        </Tab.Navigator>
    );
}
```

Now, we will add our styling conditions to the **barStyle** attribute -

```
barStyle={this.state.light_theme ? styles.bottomTabStyleLight : styles.bottomTabStyle}
```

And add styles for it as shown below -

```
bottomTabStyleLight: {
    backgroundColor: "#eaeaea",
    height: "8%",
    borderTopLeftRadius: 30,
    borderTopRightRadius: 30,
    overflow: 'hidden',
    position: 'absolute'
}
```

Remove the **export** statement at the last, since we are already exporting our newly converted class component above.

Therefore, this -

```
76
77    export default BottomTabNavigator
```

Becomes this -

```
76
77
```

:P

	Now that these changes are done, our app looks like -	

	Similarly, if we make changes to the **CreateStory** screen and the **StoryScreen** as well, our app will have 2 different themes! The steps to add lighter theme to these screens is similar, therefore we have added it on your behalf so we can move further to more exciting things in the next class! ***Teacher refers to Teacher Activity 2 to clone it and test the output***	***Student refers to Student Activity 2 to clone it and test the output***

	Awesome! Now navigate through the app! Try to change the themes and notice your theme getting changed now?	*The student tries to change the theme and see*
	Next class, we will use firebase to add stories and get stories, instead of using temporary data. We will also run to check if there are any bugs in our App and debug those.	
Teacher Guides Student to Stop Screen Share		
WRAP-UP SESSION - 5 mins		
FEEDBACK • **Appreciate the student for their class** • **Get them to play around with different ideas**		
Teacher can show slideshow from slides 16 to 25 Refer to speaker notes and follow the instructions on each slide.		
Run the presentation from slide 16 to slide 25. **Following are the wrap-up session deliverables:** • **Explain the facts and trivias** • **Next class challenge** • **Project for the day** • **Additional Activity**		Guide the student to develop the project and share it with us.
QnA Session		
Question		**Answer**

'TabNavigator' is a functional component. What does it mean? A. This means it is a function that can not have a constructor() or render() method. B. This means it is a function that can ONLY have a constructor but not the render() methods. C. This means it is a function that can ONLY have a render method and not the constructor() method. D. This means it is a function that can have any method in it.	**A**
What does the following snippet of code do? `barStyle={this.state.light_theme ? styles.bottomTabStyleLight : styles.bottomTabStyle}` A. Adds the styling to the barStyle attribute based on the theme. B. It is used for toggling between the themes. C. Changing themes based on the user preference. D. None of the above	**A**
Why do we remove the statement where we export BottomTabNavigator component? A. Because we don't need it anymore. B. Since we are already exporting our newly converted class component. C. Because it is no longer a function, hence it can't be exported. D. None of the above.	**B**
End the quiz panel	
Amazing work today! You get a "hats-off". Alright. See you in the next class.	*Make sure you have given at least 2 Hats Off during the class for:* Creatively Solved Activities +10

		Great Question +10 Strong Concentration +10

Project Overview

Spectagram Stage -7

Goal of the Project:

In Class 87, we built and integrated the light theme into the app to let the user choose between light and dark themes.

In this project, you will practice the concepts learned in the class to allow users of the Spectagram App to change the theme of their app.

*This is a continuation project of 81-86, please make sure to finish that before attempting this one.

Story:

Jenny is a photographer. She wants to share pictures taken by her with others, at the same time she wants to create a space for others to share their talent too. She decided to create a social media app for her and all upcoming talents. She has asked for your help to create an App.

Guide Jenny to give an option to change the theme of the app from a dark to a light theme.

Teacher ends slideshow

Teacher Clicks	✖ End Class	
ADDITIONAL ACTIVITY		
Additional Activities	*Encourage the student to write reflection notes in their reflection journal using Markdown.* Use these as guiding questions: ● What happened today? ○ Describe what happened. ○ The code I wrote. ● How did I feel after the class? ● What have I learned about programming and developing games? ● What aspects of the class helped me? What did I find difficult?	*The student uses the Markdown editor to write their reflections in a reflection journal.*

Links:

Activity	Activity Name	Links
Teacher Activity 1	Previous Class Code	https://github.com/pro-whitehatjr/ST-86-Solution
Teacher Activity 2	Reference Code	https://github.com/pro-whitehatjr/ST-87-Solution
Teacher Activity 3	Teacher Aid	https://drive.google.com/file/d/1WA1BQff4dmgv5BInU3f_imk4vlpvAyMa/view?usp=sharing
Teacher Activity 4	Snack Support Document	https://docs.google.com/document/d/11vq49uJQCfdxaUUzOoY7A65aau

		0kZqNMFhObZH-e71Y/edit?usp=sharing
Student Activity 1	Boilerplate Code	https://github.com/pro-whitehatjr/Story-Telling-App-87-S
Student Activity 2	Reference Code	https://github.com/pro-whitehatjr/ST-87-Solution
Teacher Reference visual aid link	Visual aid link	https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/PRO_V3_C87_LITE_withcues.html
Teacher Reference In-class quiz	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/71dd75be-edec-4928-b4bd-5d2a71280653.pdf