TEXT EDITOR APPLICATION

By

S. ASHLY P JOY

(Register No:111923MC02005)

K. PRIYANKA

(Register No: 111923MC02038)

S. SHAROON

(Register No:111923MC02047)

SC. VAISHALI

(Register No:111923MC02055)

S. YUVASHRI

(Register No:111923MC02060)

MASTER OF COMPUTER APPLICATIONS

S. A. ENGINEERING COLLEGE

(An Autonomous

Institution)

CHENNAI-600-077

APRIL-2024

INTRODUCTION

The **Text Editor App** is a mobile-based text editing application built using React Native. Designed for flexibility and ease of use, the app enables users to create, style, and save text with various formatting options. Whether you're taking notes, drafting content, or working on creative writing, this app offers essential text editing features that enhance your writing experience. Users can format text by applying **bold**, **italic**, and **underline** styles, making the content more expressive and visually distinct. The app also allows users to switch between different **font families** (normal and serif) and customize the **text color** with options like red, blue, yellow, and green. One of the standout features of the app is its use of **AsyncStorage**, which ensures that your text is automatically saved locally on the device. This means you can close the app and return to your work without losing any progress, offering a seamless experience across sessions. The app's **real-time text preview** shows changes as they are made, providing instant feedback on how the text will appear. With its clean interface and essential functionality, this text editor is an ideal tool for users looking for a convenient and customizable text creation experience on their mobile devices.

TECHNOLOGY USED

Visual Studio Code (VS Code):

- **VS Code** is an open-source code editor developed by Microsoft, providing support for various programming languages and frameworks. It offers a range of features such as syntax highlighting, code completion, and integrated Git version control.
- It was used for writing and managing the React Native project code, using extensions that facilitate easier development and debugging.

Android Studio:

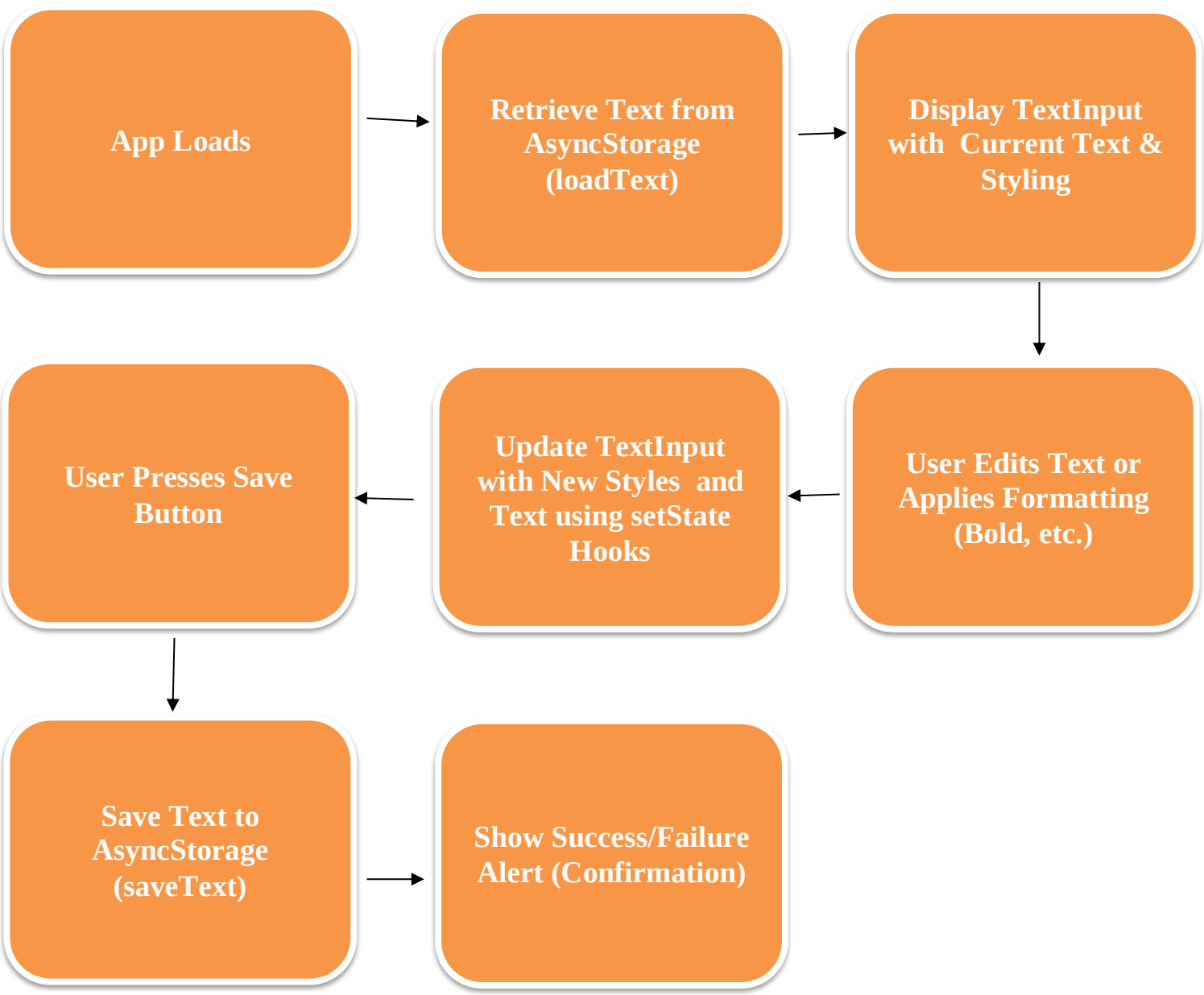
- **Android Studio** is the official Integrated Development Environment (IDE) for Android development. It provides advanced tools for app debugging, performance optimization, and device emulation.
- Android Studio was used for testing the app on Android devices to ensure its compatibility and functionality.

WORK FLOW

APP WORKFLOW

- **User Input:** The user types in the Text Input field.
- **Formatting:** Users apply formatting options like Bold, Italic, Underline, and change text color.
- **Saving:** The user can save their text, which will be stored using AsyncStorage (local storage).
- **Re-Loading:** Upon reopening the app, the saved text is loaded back for further editing.
- **Retrieve Text from AsyncStorage**: If text is found, it updates the text state.
- **Text Input Interaction**: Users can type text or apply styling (bold, italic, underline, color, font).

Diagrammatic Representation Of WorkFlow



CODE

```
import React, { useState, useEffect } from 'react';
import { StyleSheet, View, TextInput, Button, Alert, Text, TouchableOpacity } from 'react-native';
import AsyncStorage from '@react-native-async-storage/async-storage';

const App = () => {
  const [text, setText] = useState('');
  const [isBold, setIsBold] = useState(false);
  const [isItalic, setIsItalic] = useState(false);
  const [isUnderlined, setIsUnderlined] = useState(false);
  const [fontFamily, setFontFamily] = useState('normal');
  const [textColor, setTextColor] = useState('#000'); // Default color: black

  // Load saved text on component mount
  useEffect(() => {
    const loadText = async () => {
      const savedText = await AsyncStorage.getItem('text');
      if (savedText) {
        setText(savedText);
      }
    };
    loadText();
  }, []);

  // Save text to local storage
  const saveText = async () => {
    try {
      await AsyncStorage.setItem('text', text);
      Alert.alert('Saved!', 'Your text has been saved.');
    } catch (error) {
```

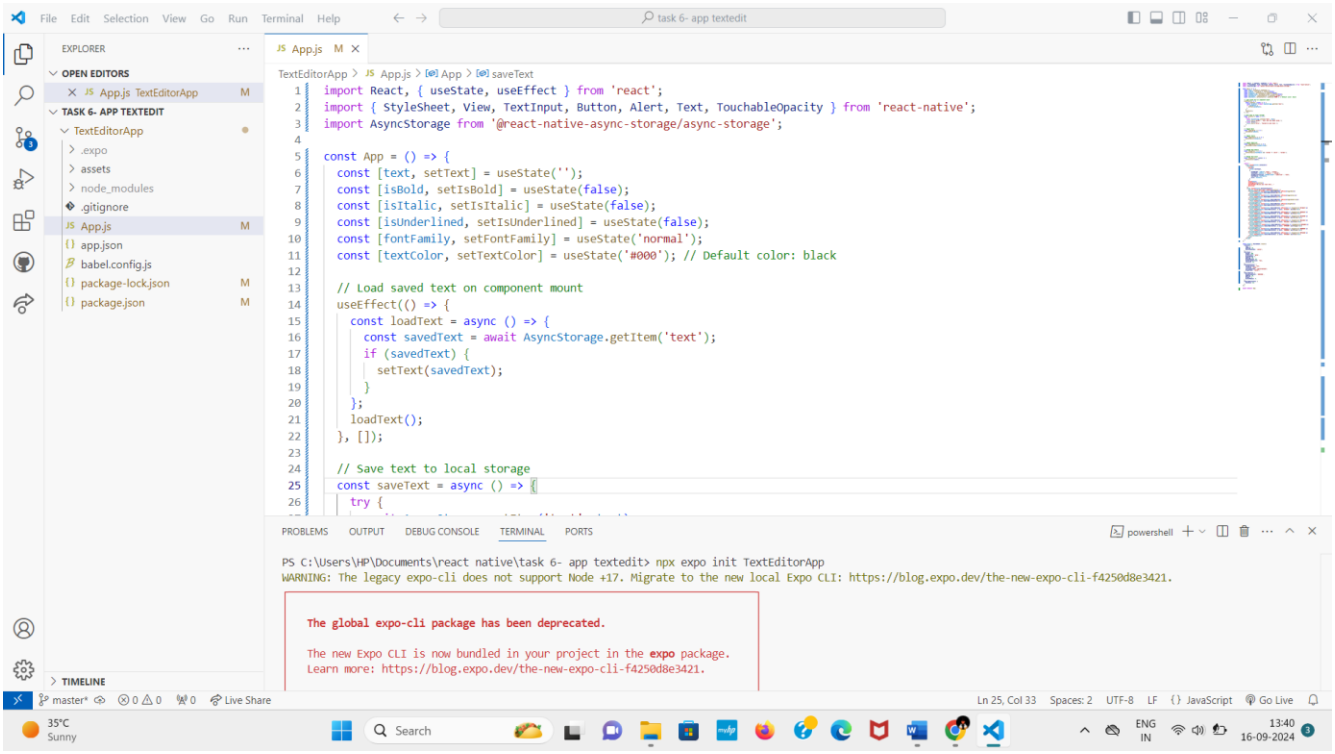
```
    Alert.alert('Error', 'Failed to save text.');
  }
};

// Toggle Bold
const toggleBold = () => {
  setIsBold(!isBold);
};

// Toggle Italic
const toggleItalic = () => {
  setIsItalic(!isItalic);
};

// Toggle Underline
const toggleUnderline = () => {
  setIsUnderlined(!isUnderlined);
};

// Change Font Family
const changeFont = () => {
  setFontFamily(fontFamily === 'normal' ? 'serif' : 'normal');
};

// Change Text Color
const changeColor = (color) => {
  setTextColor(color);
};

return (
  <View style={styles.container}>
    <TextInput
```

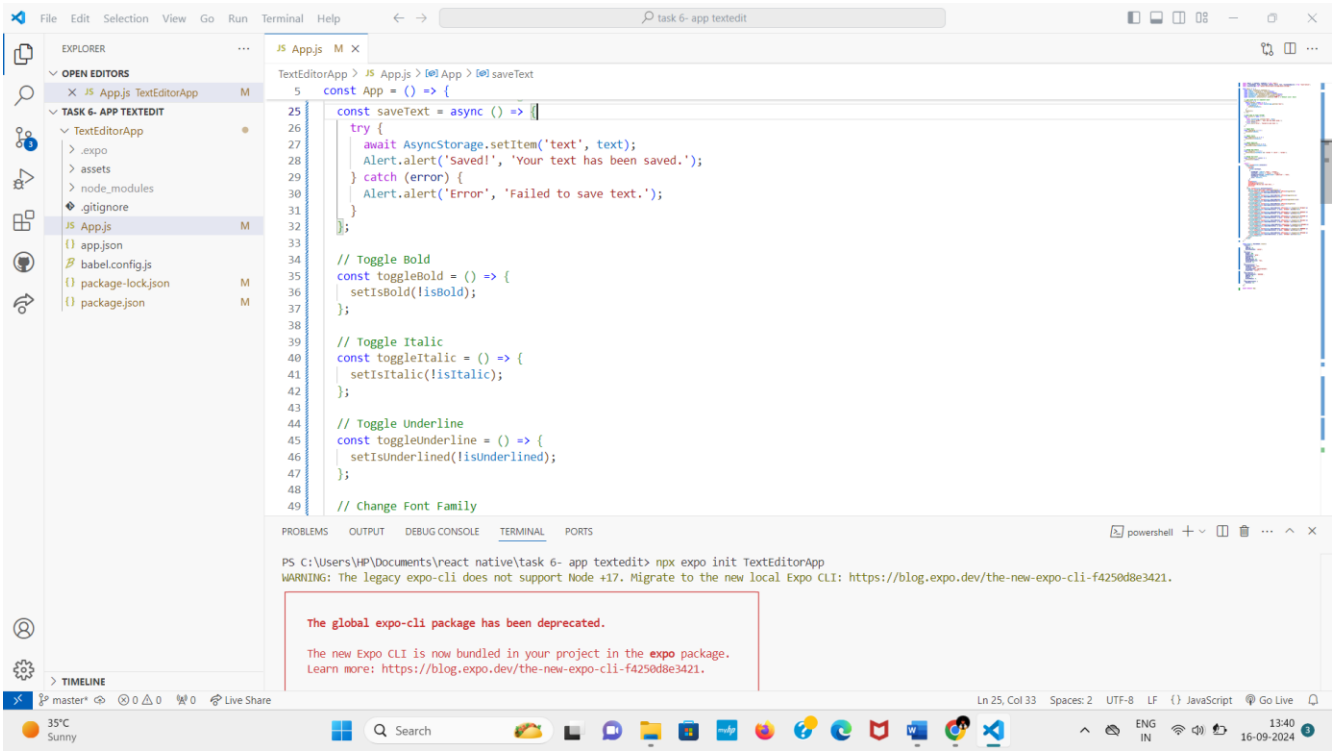
```
  style={[
    styles.textInput,
    {
      fontWeight: isBold ? 'bold' : 'normal',
      fontStyle: isItalic ? 'italic' : 'normal',
      textDecorationLine: isUnderlined ? 'underline' : 'none',
      fontFamily: fontFamily,
      color: textColor,
    }
  ]}
  value={text}
  onChangeText={setText}
  placeholder="Write your text here..."
  multiline
/>
<View style={styles.buttonContainer}>
  <Button title="Save Text" onPress={saveText} />
  <TouchableOpacity style={styles.featureButton} onPress={toggleBold}>
    <Text style={styles.featureButtonText}>B</Text>
  </TouchableOpacity>
  <TouchableOpacity style={styles.featureButton} onPress={toggleItalic}>
    <Text style={styles.featureButtonText}>I</Text>
  </TouchableOpacity>
  <TouchableOpacity style={styles.featureButton} onPress={toggleUnderline}>
    <Text style={styles.featureButtonText}>U</Text>
  </TouchableOpacity>
  <TouchableOpacity style={styles.featureButton} onPress={changeFont}>
    <Text style={styles.featureButtonText}>Font</Text>
  </TouchableOpacity>
  <TouchableOpacity style={styles.featureButton} onPress={() => changeColor('#ff6347')}>
    <Text style={[styles.featureButtonText, { color: '#ff6347' }]}>Red</Text>
  </TouchableOpacity>
```

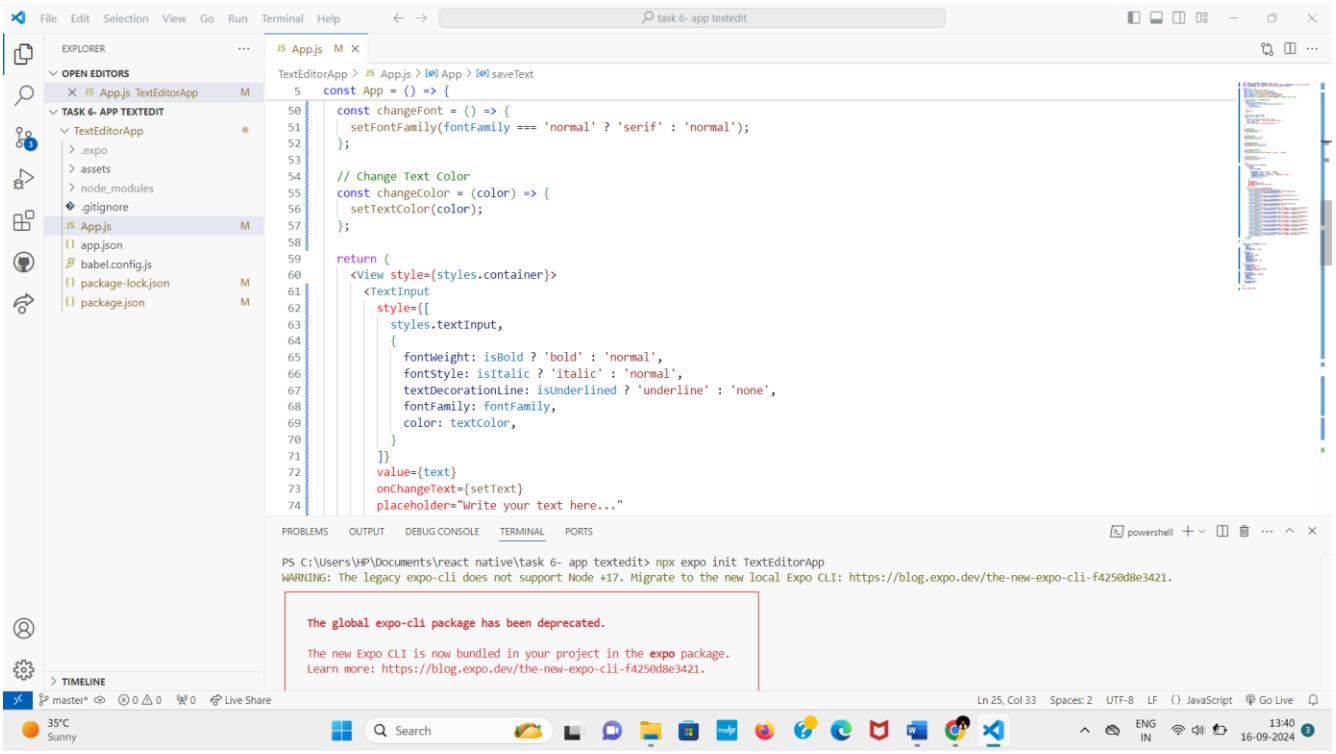
```
      <TouchableOpacity style={styles.featureButton} onPress={() => changeColor('#4682b4')}>
        <Text style={[styles.featureButtonText, { color: '#4682b4' }]}>Blue</Text>
      </TouchableOpacity>
      <TouchableOpacity style={styles.featureButton} onPress={() => changeColor('#ffd700')}>
        <Text style={[styles.featureButtonText, { color: '#ffd700' }]}>Yellow</Text>
      </TouchableOpacity>
      <TouchableOpacity style={styles.featureButton} onPress={() => changeColor('#32cd32')}>
        <Text style={[styles.featureButtonText, { color: '#32cd32' }]}>Green</Text>
      </TouchableOpacity>
       <TouchableOpacity style={styles.featureButton} onPress={() => changeColor('#FFA500')}>
        <Text style={[styles.featureButtonText, { color: '#FFA500' }]}>Orange</Text>
      </TouchableOpacity>
      <TouchableOpacity style={styles.featureButton} onPress={() => changeColor('#800080')}>
        <Text style={[styles.featureButtonText, { color: '#800080' }]}>Purple</Text>
      </TouchableOpacity>
      <TouchableOpacity style={styles.featureButton} onPress={() => changeColor('#FFC0CB')}>
        <Text style={[styles.featureButtonText, { color: '#FFC0CB' }]}>Pink</Text>
      </TouchableOpacity>
    </View>
  </View>
 );
};

const styles = StyleSheet.create({
 container: {
  flex: 1,
  padding: 16,
  justifyContent: 'center',
 },
 textInput: {
  height: 200,
  borderColor: 'gray',
```

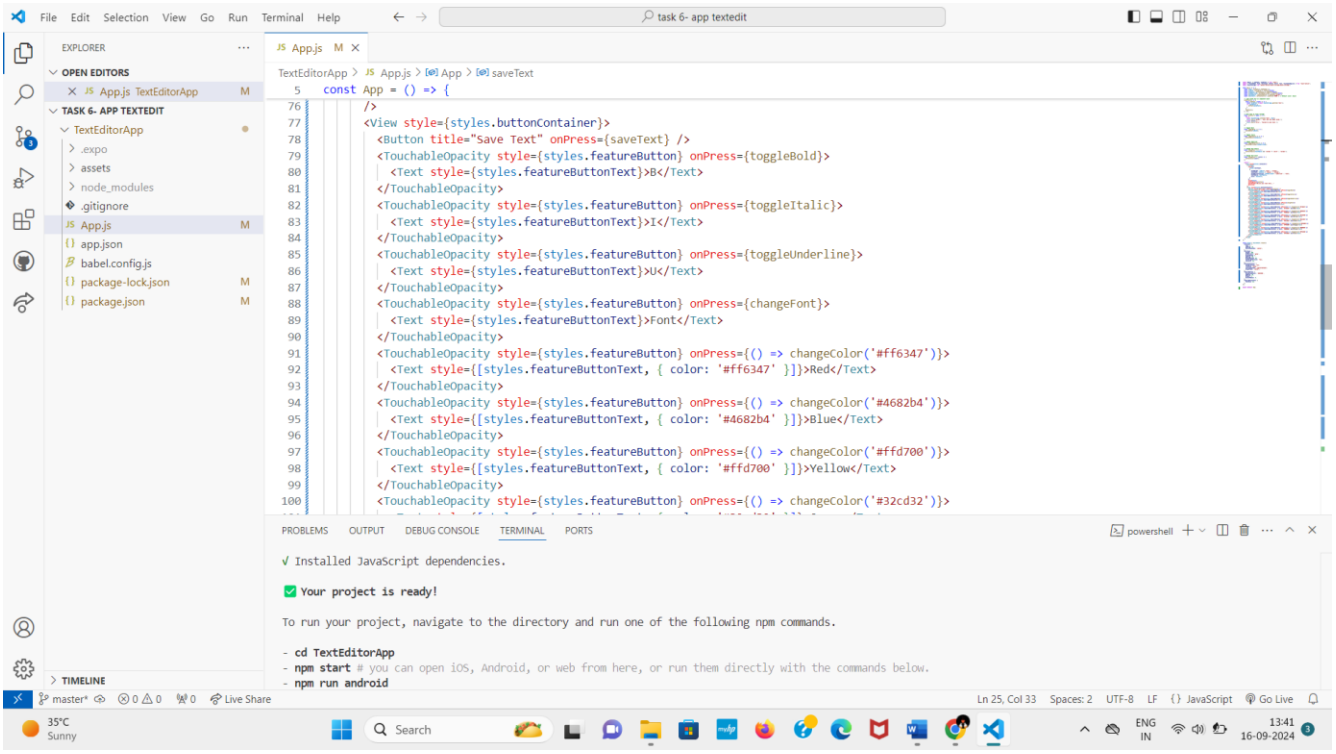
```
    borderWidth: 1,
    padding: 10,
    marginBottom: 20,
    textAlignVertical: 'top',
    fontSize: 18,
  },
  buttonContainer: {
    flexDirection: 'row',
    flexWrap: 'wrap',
    justifyContent: 'space-between',
    alignItems: 'center',
  },
  featureButton: {
    backgroundColor: '#f0f0f0',
    padding: 10,
    margin: 5,
    borderRadius: 5,
  },
  featureButtonText: {
    fontSize: 16,
  },
});

export default App;
```

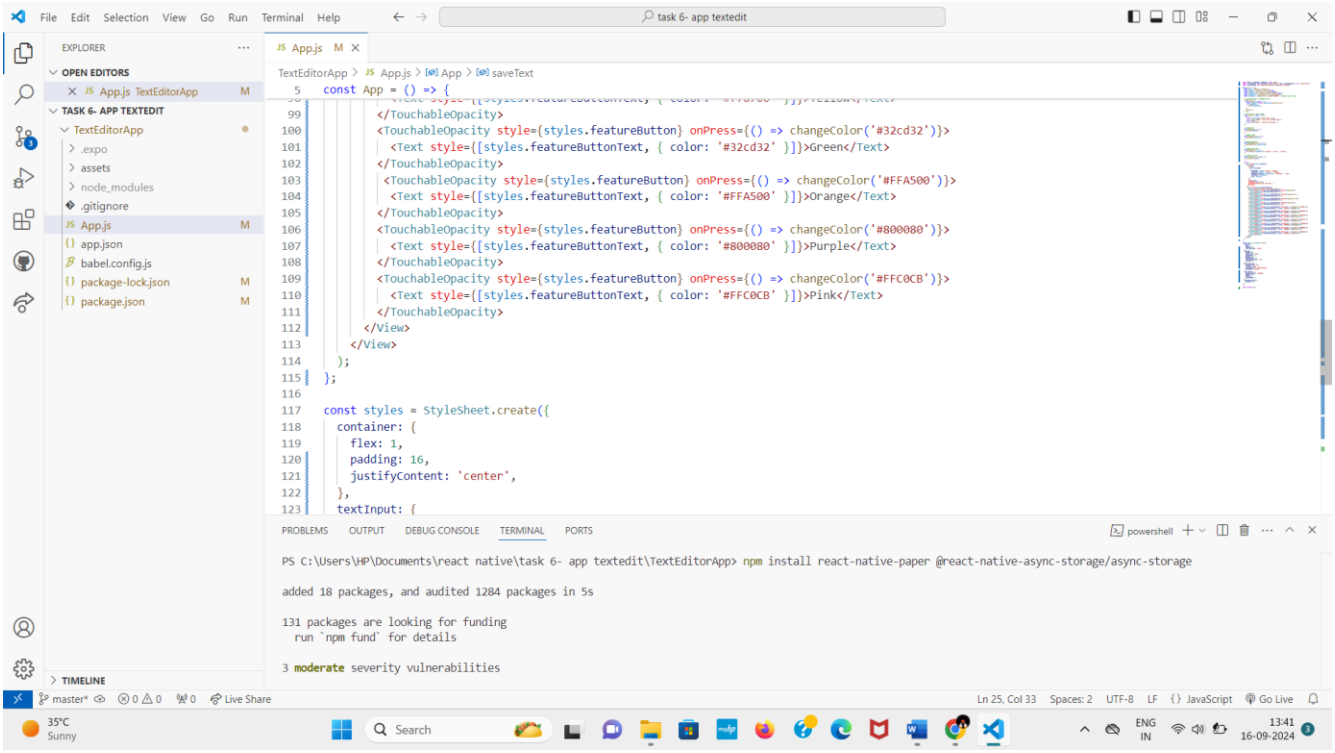
SCREENSHOT OF CODE IN VS CODE

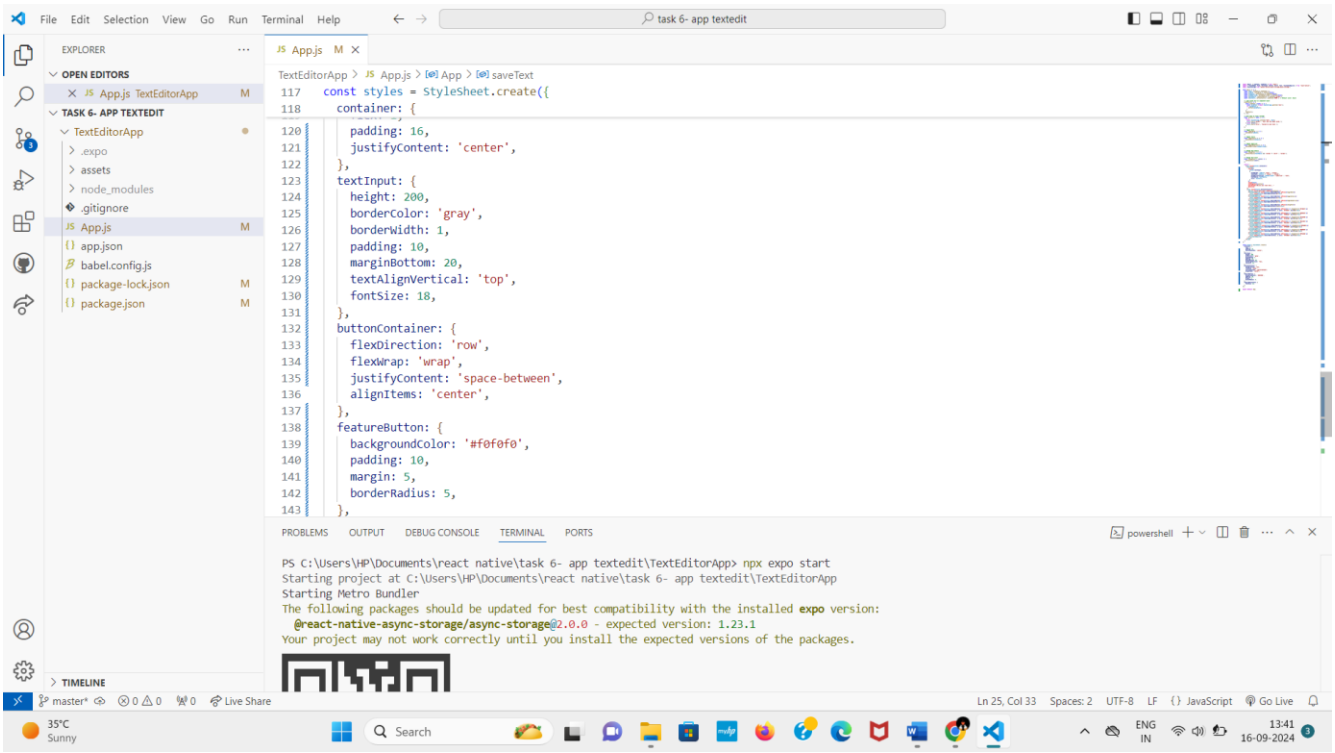


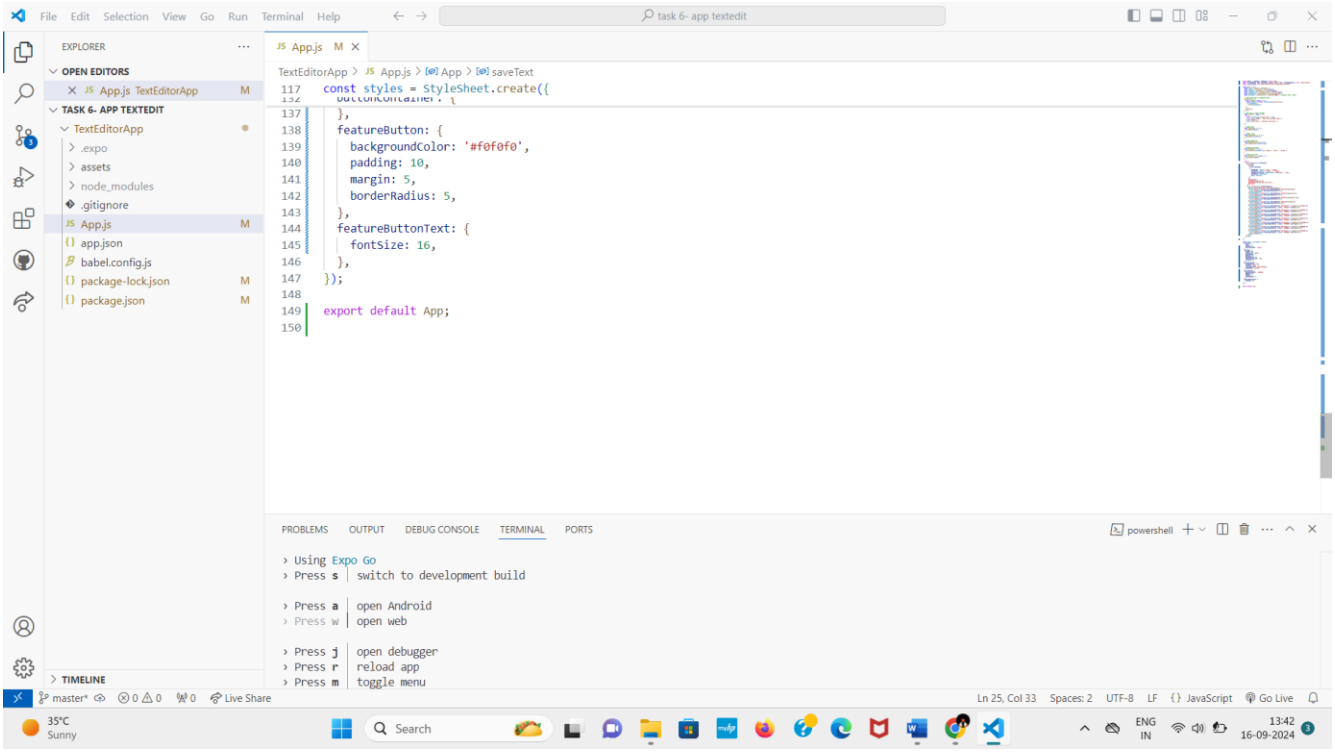



File Edit Selection View Go Run Terminal Help
task 6- app textedit
EXPLORER
OPEN EDITORS
App.js TextEditorApp M
TASK 6- APP TEXTEDIT
TextEditorApp
.expo
assets
node_modules
.gitignore
App.js M
app.json
babel.config.js
package-lock.json M
package.json M
TIMELINE
App.js M
TextEditorApp > App.js > App > saveText
5 const App = () => {
50 const changeFont = () => {
51 setFontFamily(fontFamily === 'normal' ? 'serif' : 'normal');
52 };
53
54 // Change Text Color
55 const changeColor = (color) => {
56 setTextColor(color);
57 };
58
59 return (
60 <View style={styles.container}>
61 <TextInput
62 style={[
63 styles.textInput,
64 {
65 fontWeight: isBold ? 'bold' : 'normal',
66 fontStyle: isItalic ? 'italic' : 'normal',
67 textDecorationLine: isUnderlined ? 'underline' : 'none',
68 fontFamily: fontFamily,
69 color: textColor,
70 }
71]}
72 value={text}
73 onChangeText={setText}
74 placeholder="Write your text here..."
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
PS C:\Users\HP\Documents\react native\task 6- app textedit> npx expo init TextEditorApp
WARNING: The legacy expo-cli does not support Node +17. Migrate to the new local Expo CLI: https://blog.expo.dev/the-new-expo-cli-f4250d8e3421.
The global expo-cli package has been deprecated.
The new Expo CLI is now bundled in your project in the expo package.
Learn more: https://blog.expo.dev/the-new-expo-cli-f4250d8e3421.
master* Live Share
Ln 25, Col 33 Spaces: 2 UTF-8 LF {} JavaScript Go Live
35°C Sunny
Search
ENG IN
13:40 16-09-2024


File Edit Selection View Go Run Terminal Help
task 6- app textedit
EXPLORER
OPEN EDITORS
App.js TextEditorApp M
TASK 6- APP TEXTEDIT
TextEditorApp
.expo
assets
node_modules
.gitignore
App.js M
app.json
babel.config.js
package-lock.json M
package.json M
TIMELINE
App.js M
TextEditorApp > App.js > App > saveText
5 const App = () => {
76 />
77 <View style={styles.buttonContainer}>
78 <Button title="Save Text" onPress={saveText} />
79 <TouchableOpacity style={styles.featureButton} onPress={toggleBold}>
80 <Text style={styles.featureButtonText}>B</Text>
81 </TouchableOpacity>
82 <TouchableOpacity style={styles.featureButton} onPress={toggleItalic}>
83 <Text style={styles.featureButtonText}>I</Text>
84 </TouchableOpacity>
85 <TouchableOpacity style={styles.featureButton} onPress={toggleUnderline}>
86 <Text style={styles.featureButtonText}>U</Text>
87 </TouchableOpacity>
88 <TouchableOpacity style={styles.featureButton} onPress={changeFont}>
89 <Text style={styles.featureButtonText}>Font</Text>
90 </TouchableOpacity>
91 <TouchableOpacity style={styles.featureButton} onPress={() => changeColor('#ff6347')}>
92 <Text style={[styles.featureButtonText, { color: '#ff6347' }]}>Red</Text>
93 </TouchableOpacity>
94 <TouchableOpacity style={styles.featureButton} onPress={() => changeColor('#4682b4')}>
95 <Text style={[styles.featureButtonText, { color: '#4682b4' }]}>Blue</Text>
96 </TouchableOpacity>
97 <TouchableOpacity style={styles.featureButton} onPress={() => changeColor('#ffd700')}>
98 <Text style={[styles.featureButtonText, { color: '#ffd700' }]}>Yellow</Text>
99 </TouchableOpacity>
100 <TouchableOpacity style={styles.featureButton} onPress={() => changeColor('#32cd32')}>
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
√ Installed JavaScript dependencies.
Your project is ready!
To run your project, navigate to the directory and run one of the following npm commands.
- cd TextEditorApp
- npm start # you can open iOS, Android, or web from here, or run them directly with the commands below.
- npm run android
master* Live Share
Ln 25, Col 33 Spaces: 2 UTF-8 LF {} JavaScript Go Live
35°C Sunny
Search
ENG IN
13:41 16-09-2024


File Edit Selection View Go Run Terminal Help
task 6- app textedit
EXPLORER
OPEN EDITORS
App.js TextEditorApp M
TASK 6- APP TEXTEDIT
TextEditorApp
.expo
assets
node_modules
.gitignore
App.js M
app.json
babel.config.js
package-lock.json M
package.json M
TIMELINE
App.js M
TextEditorApp > App.js > App > saveText
99 </TouchableOpacity>
100 <TouchableOpacity style={styles.featureButton} onPress={() => changeColor('#32cd32')}>
101 <Text style={[styles.featureButtonText, { color: '#32cd32' }]}>Green</Text>
102 </TouchableOpacity>
103 <TouchableOpacity style={styles.featureButton} onPress={() => changeColor('#FFA500')}>
104 <Text style={[styles.featureButtonText, { color: '#FFA500' }]}>Orange</Text>
105 </TouchableOpacity>
106 <TouchableOpacity style={styles.featureButton} onPress={() => changeColor('#800080')}>
107 <Text style={[styles.featureButtonText, { color: '#800080' }]}>Purple</Text>
108 </TouchableOpacity>
109 <TouchableOpacity style={styles.featureButton} onPress={() => changeColor('#FFC0CB')}>
110 <Text style={[styles.featureButtonText, { color: '#FFC0CB' }]}>Pink</Text>
111 </TouchableOpacity>
112 </View>
113 </View>
114);
115 };
116
117 const styles = StyleSheet.create({
118 container: {
119 flex: 1,
120 padding: 16,
121 justifyContent: 'center',
122 },
123 textInput: {
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
PS C:\Users\HP\Documents\react native\task 6- app textedit\TextEditorApp> npm install react-native-paper @react-native-async-storage/async-storage
added 18 packages, and audited 1284 packages in 5s
131 packages are looking for funding
run `npm fund` for details
3 moderate severity vulnerabilities
master* 0 0 0 Live Share
Ln 25, Col 33 Spaces: 2 UTF-8 LF JavaScript Go Live
35°C Sunny
Search
ENG IN
13:41 16-09-2024


File Edit Selection View Go Run Terminal Help
task 6- app textedit
EXPLORER
OPEN EDITORS
App.js TextEditorApp M
TASK 6- APP TEXTEDIT
TextEditorApp
.expo
assets
node_modules
.gitignore
App.js M
app.json
babel.config.js
package-lock.json M
package.json M
TIMELINE
App.js M
TextEditorApp > App.js > App > saveText
117 const styles = StyleSheet.create({
118 container: {
120 padding: 16,
121 justifyContent: 'center',
122 },
123 textInput: {
124 height: 200,
125 borderColor: 'gray',
126 borderWidth: 1,
127 padding: 10,
128 marginBottom: 20,
129 textAlignVertical: 'top',
130 fontSize: 18,
131 },
132 buttonContainer: {
133 flexDirection: 'row',
134 flexWrap: 'wrap',
135 justifyContent: 'space-between',
136 alignItems: 'center',
137 },
138 featureButton: {
139 backgroundColor: '#f0f0f0',
140 padding: 10,
141 margin: 5,
142 borderRadius: 5,
143 },
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
PS C:\Users\HP\Documents\react native\task 6- app textedit\TextEditorApp> npx expo start
Starting project at C:\Users\HP\Documents\react native\task 6- app textedit\TextEditorApp
Starting Metro Bundler
The following packages should be updated for best compatibility with the installed expo version:
@react-native-async-storage/async-storage@2.0.0 - expected version: 1.23.1
Your project may not work correctly until you install the expected versions of the packages.
master* 0 0 0 Live Share
Ln 25, Col 33 Spaces: 2 UTF-8 LF JavaScript Go Live
35°C Sunny
Search
ENG IN
13:41 16-09-2024


File Edit Selection View Go Run Terminal Help
task 6- app textedit
EXPLORER
OPEN EDITORS
App.js TextEditorApp M
TASK 6- APP TEXTEDIT
TextEditorApp
.expo
assets
node_modules
.gitignore
App.js M
app.json
babel.config.js
package-lock.json M
package.json M
TIMELINE
App.js M
TextEditorApp > App.js > App > saveText
117 const styles = StyleSheet.create({
137 },
138 featureButton: {
139 backgroundColor: '#f0f0f0',
140 padding: 10,
141 margin: 5,
142 borderRadius: 5,
143 },
144 featureButtonText: {
145 fontSize: 16,
146 },
147 });
148
149 export default App;
150
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
› Using Expo Go
› Press s | switch to development build
› Press a | open Android
› Press w | open web
› Press j | open debugger
› Press r | reload app
› Press m | toggle menu
master* 0 0 0 Live Share
Ln 25, Col 33 Spaces: 2 UTF-8 LF JavaScript Go Live
35°C Sunny
Search
ENG IN
13:42 16-09-2024

METHODOLOGY USED FOR TEXT EDITOR APP

The methodology for developing this text editor app involves a step-by-step approach to implement its various features such as text input, styling (bold, italic, underline), font family selection, text color changes, and saving text data using **AsyncStorage**.

1. Initialize State Variables

- The first step is to initialize state variables using the useState hook to handle:
    - **Text content** (text)
    - **Bold styling** (isBold)
    - **Italic styling** (isItalic)
    - **Underline styling** (isUnderlined)
    - **Font family** (fontFamily)
    - **Text color** (textColor)

Each of these state variables controls different aspects of the text display and interaction

```
const [text, setText] = useState('');
const [isBold, setIsBold] = useState(false);
const [isItalic, setIsItalic] = useState(false);
const [isUnderlined, setIsUnderlined] = useState(false);
const [fontFamily, setFontFamily] = useState('normal');
const [textColor, setTextColor] = useState('#000'); // Default color: black
```

2. **Loading Data from AsyncStorage**

- Upon loading the app, the useEffect hook is triggered to retrieve any previously saved text from **AsyncStorage**.
- If data exists, it is assigned to the text state so users can continue editing from where they left off.

```
useEffect(() => {
        const loadText = async () => {
        const savedText = await AsyncStorage.getItem('text');
        if (savedText) {
        setText(savedText);
        }
        };
        loadText();
        }, []);
```

3. TextInput for User Input

- The TextInput component is used to capture user text input. The style of this component is dynamically set using conditional checks on the state variables (bold, italic, underline, font family, and color).
- The multiline prop allows for multiple lines of text, making it more suitable for a note-taking app or text editor.

```
<TextInput
 style={[
  styles.textInput,
  {
   fontWeight: isBold ? 'bold' : 'normal',
   fontStyle: isItalic ? 'italic' : 'normal',
   textDecorationLine: isUnderlined ? 'underline' : 'none',
   fontFamily: fontFamily,
   color: textColor,
  }
 ]}
 value={text}
 onChangeText={setText}
```

```
  placeholder="Write your text here..."
  multiline
/>
```

4. Handling Button Actions for Text Formatting

- Several buttons are provided to change text styling. Each button toggles a specific style state:
    - **Bold**: Controlled by the setIsBold function.
    - **Italic**: Controlled by the setIsItalic function.
    - **Underline**: Controlled by the setIsUnderlined function.
    - **Font Family**: Switches between normal and serif.
    - **Text Color**: Changes to various preset colors like red, blue, yellow, and green.

```
 <TouchableOpacity style={styles.featureButton} onPress={toggleBold}>
  <Text style={styles.featureButtonText}>Bold</Text>
</TouchableOpacity>
<TouchableOpacity style={styles.featureButton} onPress={() => changeColor('#4682b4')}>
  <Text style={[styles.featureButtonText, { color: '#4682b4' }]}>Blue</Text>
</TouchableOpacity>
```

5. Saving Data to AsyncStorage

- The saveText function is triggered when the **Save Text** button is pressed. This function saves the current text state to **AsyncStorage**, ensuring that the user's input persists even after the app is closed.

```
const saveText = async () => {
  try {
    await AsyncStorage.setItem('text', text);
    Alert.alert('Saved!', 'Your text has been saved.');
  } catch (error) {
```

```
    Alert.alert('Error', 'Failed to save text.');
  }
};
```

6. UI Layout and Styling

- The layout is designed using a View container, and the buttons are styled with TouchableOpacity for a better user experience. Buttons are wrapped inside a View with flexDirection: 'row' and flexWrap: 'wrap' to arrange them in a responsive grid-like pattern.

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 16,
    justifyContent: 'center',
  },
  buttonContainer: {
    flexDirection: 'row',
    flexWrap: 'wrap',
    justifyContent: 'space-between',
    alignItems: 'center',
  },
});
```

CODE EXPLANATION

1. **Project Setup:**
   - To set up the project, follow these steps
   - Install the necessary dependencies for React Native:

```
npx react-native init TextEditorApp
cd TextEditorApp
npm install @react-native-async-storage/async-storage
```

2. **App State Management:**
   - The app uses React hooks (useState, useEffect) to manage the text content, styling, and interactions.
   - States such as text, isBold, isItalic, and isUnderlined are used to keep track of the text formatting options selected by the user.
   - Font Family and Text Color are also stored as states to dynamically change the appearance of the text.

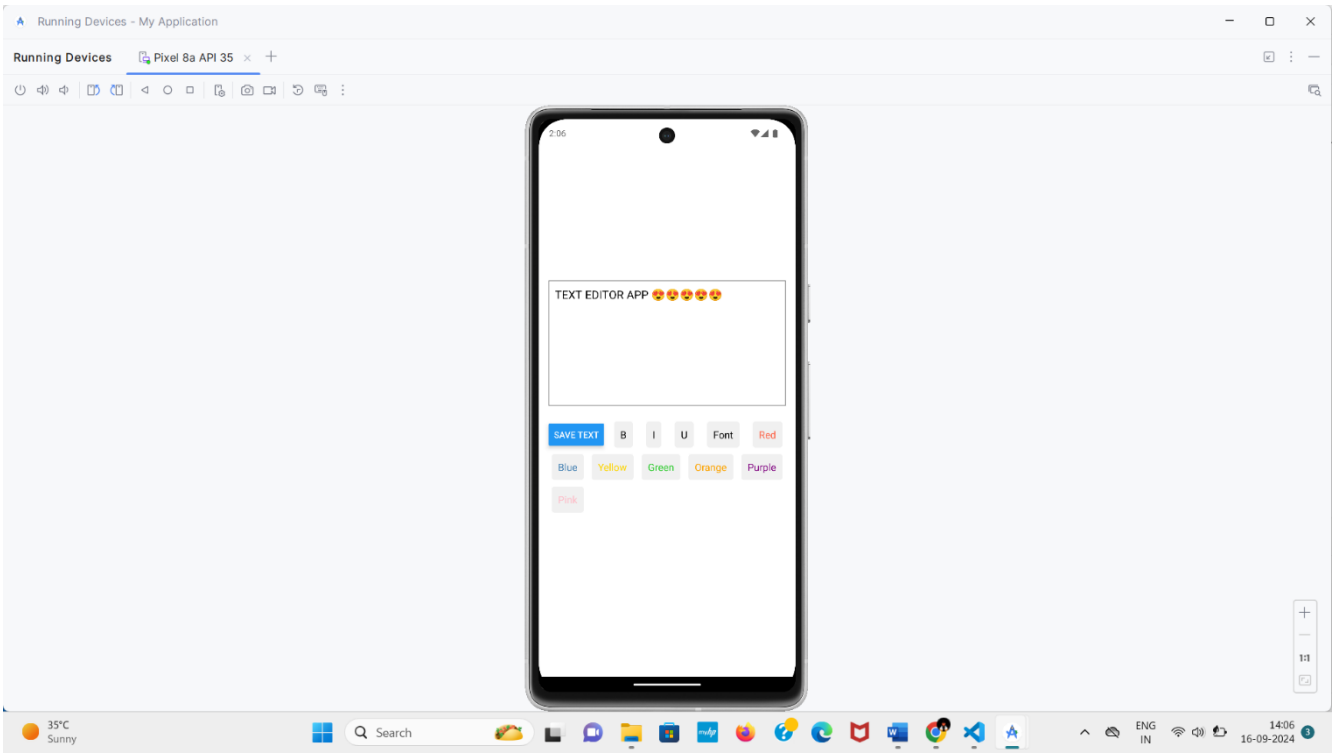
3. **Saving and Loading Text:**
   - The app uses AsyncStorage to save the user's input locally, ensuring persistence even when the app is closed.
   - On app startup (useEffect), the previously saved text is loaded from storage if available.

4. **Text Formatting Options:**
   - Bold, Italic, and Underline are toggled via buttons. These modify the corresponding state, which updates the TextInput style dynamically.
   - The font is switched between two styles (normal and serif) with a button.

- Text color is changed using color buttons. Colors like red, blue, yellow, and more are available, each associated with its own button.

5. Text Editor UI:
    - The TextInput is styled to reflect the selected formatting options, including bold, italic, underlined, color, and font family.
    - Buttons for text features are created using TouchableOpacity, and custom styles are applied to make them look like a toolbar.

OUTPUT


Running Devices - My Application
Running Devices
Pixel 8a API 35
2:06
TEXT EDITOR APP 😍😍😍😍😍
SAVE TEXT
B
I
U
Font
Red
Blue
Yellow
Green
Orange
Purple
Pink
35°C
Sunny
Search
ENG
IN
14:06
16-09-2024

CONCLUSION

The Text Editor App is a robust and user-friendly application built using React Native that allows users to perform basic text formatting operations such as bold, italic, underline, and color changes. The inclusion of local storage through AsyncStorage ensures that user data is saved and retrieved seamlessly, enhancing the overall user experience. By leveraging tools like VS Code for development and Android Studio for testing, this project showcases how efficient and flexible app development can be achieved using modern technologies. The app's core functionality, combined with its simplicity, makes it an excellent starting point for more complex text editing applications in the future. Through this project, we demonstrate the versatility of React Native and its potential for creating cross-platform apps with minimal effort.