Cross Platform App Development Lab Experiment No. 8

Aim: Implementation of asynchronous functions and outside API calls in JavaScript, React Native, and .NET MAUI.

Objectives:
1. Implement asynchronous functions in JavaScript.
2. Execute API calls in a React Native application asynchronously.
3. Apply asynchronous functions in a .NET MAUI project.
4. Perform outside API calls in .NET MAUI asynchronously.

Theory:
- **Asynchronous Functions:**
  - Functions that operate independently of the main program flow.
  - Allow concurrent execution of multiple tasks.
  - Improve performance and responsiveness by preventing blocking.
- **API Calls in React Native:**
  - Utilize libraries like Axios or fetch for HTTP requests.
  - Implement asynchronous functions to handle API responses.
- **Asynchronous Programming in .NET MAUI:**
  - Utilize asynchronous programming with `async` and `await` keywords.
  - Use the HttpClient class for making asynchronous API calls.

Requirements:
- A basic understanding of JavaScript for React Native.
- Familiarity with C# for .NET MAUI.

Tools:
- For JavaScript and React Native:
  - Any text editor (e.g., Visual Studio Code).
  - React Native development environment.
- For .NET MAUI:
  - Visual Studio or Visual Studio Code.
  - .NET MAUI development environment.

Implementation/Code:-

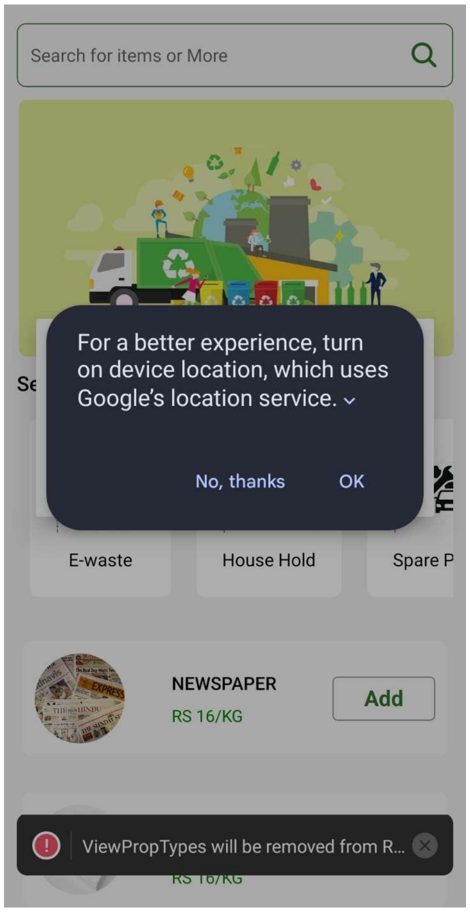
- Creating an API call for location service for our application

```
const [displayCurrentAddress, setdisplayCurrentAddress] = useState(
    "VJTI College matunga 400019"
  );
  const [locationServicesEnabled, setLocationServicesEnabled] = useState(false);
  useEffect(() => {
    checkIfLocationEnabled();
    getCurrentLocation();
  }, []);

  const checkIfLocationEnabled = async () => {
    let enabled = await Location.hasServicesEnabledAsync();
    if (!enabled) {
      Alert.alert(
        "Location services not enabled",
        "Please enable the location services",
        [
          {
            text: "Cancel",
            onPress: () => console.log("Cancel Pressed"),
            style: "cancel",
          },
          { text: "OK", onPress: () => console.log("OK Pressed") },
        ],
        { cancelable: false }
      );
    } else {
      setlocationServicesEnabled(enabled);
    }
  };
```

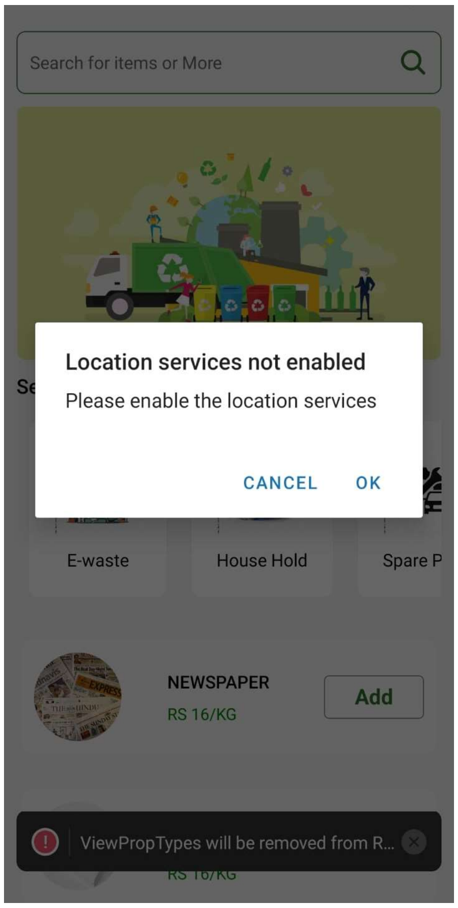
- Code for the part fetching location

```
 const getCurrentLocation = async () => {
    let { status } = await Location.requestForegroundPermissionsAsync();

    if (status !== "granted") {
      Alert.alert(
        "Permission denied",
        "allow the app to use the location services",
        [
          {
            text: "Cancel",
            onPress: () => console.log("Cancel Pressed"),
            style: "cancel",
          },
          { text: "OK", onPress: () => console.log("OK Pressed") },
        ],
        { cancelable: false }
      );
    }

    const { coords } = await Location.getCurrentPositionAsync();
    // console.log(coords)
    if (coords) {
      const { latitude, longitude } = coords;

      let response = await Location.reverseGeocodeAsync({
        latitude,
        longitude,
      });
```

- So we can press OK and give permission to access location of our device.
- For the same we can cancel the pop-up we can see in below screenshots


For a better experience, turn on device location, which uses Google's location service.

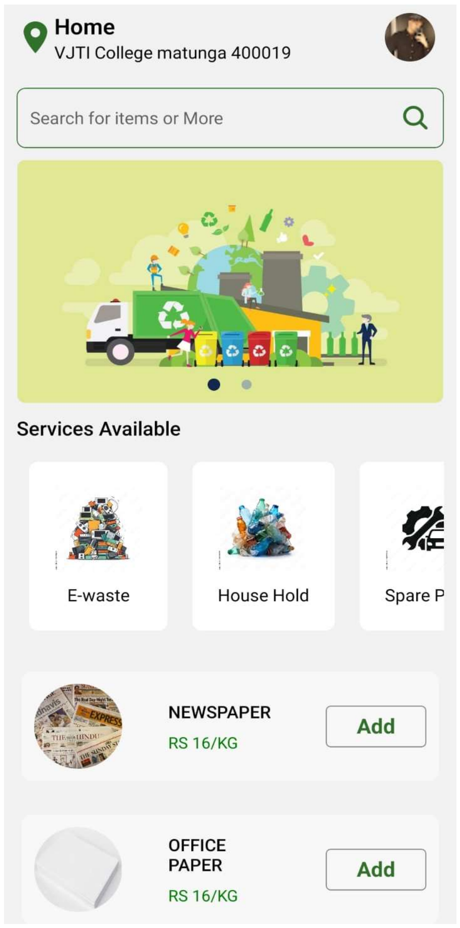
ViewPropTypes will be removed from R...


Location services not enabled

Please enable the location services

ViewPropTypes will be removed from R...

- Now, we can see location on the top of application.



Conclusion:

We learnt to implement of asynchronous functions and outside API calls in React Native and handling outside API calls in JavaScript, React Native, we make sure efficient execution of tasks, improving the overall responsiveness of our applications.

References:

1. MDN Web Docs - Asynchronous programming in JavaScript: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous
2. React Native Networking: https://reactnative.dev/docs/network