Київський національний університет
імені Т.Шевченка

Звіт

до лабораторної роботи №2
на тему:
«Рейтинг футбольних клубів України»

Студента 4-го курсу
Бурлаки Владислава
Варіант №9

Київ-2023

Зміст

Постановка задачі

Метою роботи є формування базових знань та умінь розробляти для мобільного пристрою програми, яка використовує збереження даних у БД, використовує адресну книгу, обробляє і візуалізує геопросторові дані з використанням відповідних API. Варіант №9: id, назва клубу, місто, рік заснування, рейтинг UEFA. Показати назви і рейтинги всіх українських клубів. Величина для обчислення: Сумарний рейтинг клубів з Києва. Показати вибірку контактів, у яких електронна пошта закінчується на "@knu.ua". Побудувати маршрут до обраного міста одного з футбольних клубів.

Короткий виклад теорії

- Використання API в Реакт Нейтів: Fetch та Axios: Для взаємодії з API в React Native використовуються асинхронні запити, зазвичай за допомогою функцій fetch або бібліотеки Axios. Обробка JSON: Зазвичай API повертає дані у форматі JSON, які потім можна обробляти та використовувати в додатку.
- Адресна Книга Контактів Телефону: Використання Бібліотек: Для отримання доступу до адресної книги телефону в React Native використовують бібліотеки, такі як react-native-contacts. Доступ до Контактів: Зазвичай дозвіл на доступ до контактів здійснюється через спеціальний дозвіл в мобільній операційній системі.
- Робота з Google API: Картографія та Місця: Для інтеграції з Google Maps API використовують бібліотеки, наприклад react-native-maps для відображення карт та react-native-google-places-autocomplete для пошуку місць. Отримання Ключа API: Для використання Google API потрібен ключ API, який можна отримати в консолі розробника Google. Бекенд на JavaScript:
- Вибір Фреймворку: Для розробки бекенду на JavaScript можна використовувати фреймворки, такі як Express.js, NestJS, або інші, залежно від ваших потреб та вмінь. Express.js: Це легкий та популярний фреймворк для створення серверів та API на JavaScript. NestJS: Це фреймворк, який використовує TypeScript та пропонує об'єктно-орієнтовану архітектуру.
- Робота з Базою Даних: Вибір Бази Даних: В залежності від вимог вашого додатку, виберіть підходящу базу даних, таку як MongoDB, MySQL, PostgreSQL тощо. ORM або ODM: Використання ORM (Object-Relational Mapping) або ODM (Object-Document Mapping) для спрощення взаємодії з базою даних.
- Робота з Запитами та Відповідями: Маршрутизація: Визначення маршрутів для обробки запитів від клієнта. Взаємодія з Клієнтом: Отримання та обробка запитів від клієнта, надсилання відповідей, робота з HTTP-протоколом.

Опис алгоритмів, які застосовуються для обчислень

- Отримання ключа API:

Спочатку потрібно отримати ключ API від Google Cloud Console для використання сервісів Google Maps API, таких як отримання карт, маршрутизація тощо.

- Знаходження місць:

Для знаходження місць, ви можете використовувати Google Places API. Запити можна виконувати, наприклад, через fetch або використовуючи бібліотеку, таку як axios.

- Робота з контактами телефону:

1. Отримання дозволів

Забезпечте відповідні дозволи в маніфесті вашого додатка для доступу до контактів.

2. Взаємодія з контактами:

Використовуйте react-native-contacts або інші бібліотеки для роботи з контактами в React Native. Зазвичай вони надають методи для читання, запису та видалення контактів.

- Робота з взаємодією бекенда та MongoDB:

1. Створення API на бекенді:

Створіть API на своєму бекенді для обробки запитів від фронтенда. Використовуйте Express, Django, Flask або інший фреймворк залежно від ваших потреб.

2. Взаємодія фронтенду та бекенду:

Використовуйте бібліотеки, такі як axios, для здійснення HTTP-запитів з фронтенду до вашого бекенду.

3. Робота з MongoDB:

Використовуйте офіційний драйвер MongoDB або ORM (наприклад, Mongoose для Node.js) для зручної взаємодії з базою даних MongoDB.

Опис та демонстрація роботи програми

Ця програма призначена для перегляду та модифікації записів нереляційної бази даних, перегляд контактів телефону та вибірки контактів, відповідно до варіанту та функціонал пошуку шляху по гугл картах.

Функціонал застосунку розподілений по 5 сторінкам. Одна з них призначена для ознайомлення з автором роботи, перегляду його фото та імені (рис. 1).

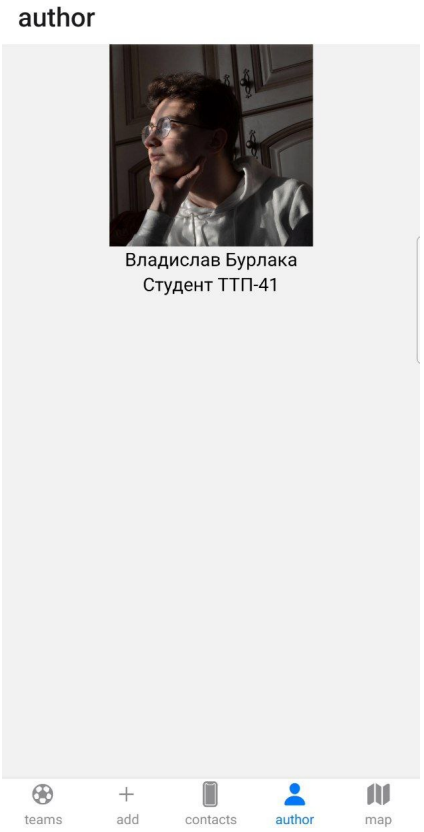

Рисунок 1 - сторінка з автором

Список контактів користувача та їх вибірка за заданим фільтром розміщені на окремій сторінці (рис. 2).

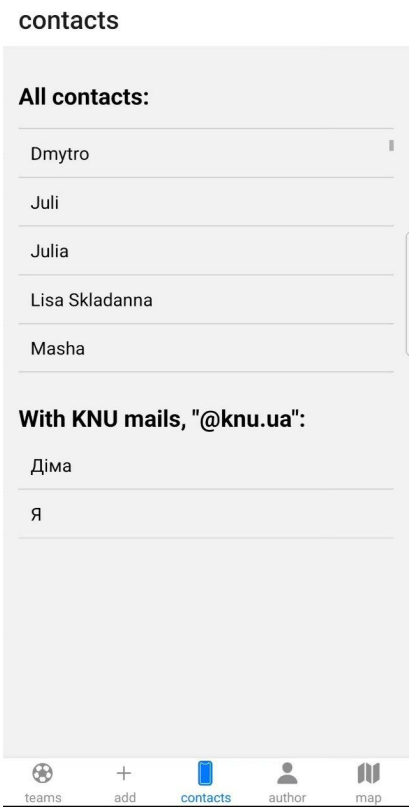

Рисунок 2 - сторінка з контактами

Перегляд записів бази даних та підрахунок сумарного рейтингу команд згідно варіанту доступні на сторінці клубів (рис. 3).

Рисунок 3 - сторінка футбольних клубів

Функція додавання нових записів в базу даних розміщена на окремій сторінці (рис. 4). Тут користувач має заповнити усі необхідні поля, що стосуються футбольного клубу.

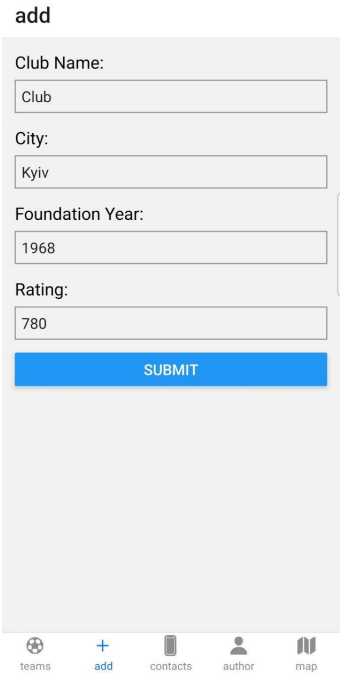

Рисунок 4 - сторінка додавання нових записів

Інтерактивна карта з можливістю побудови маршруту - остання з доступних користувачу сторінок (рис. 5). Для її використання необхідний доступ до геоданих користувача. Обираючи клуб додаток будує маршрут від вашого поточного місця перебування до міста, в якому він заснований

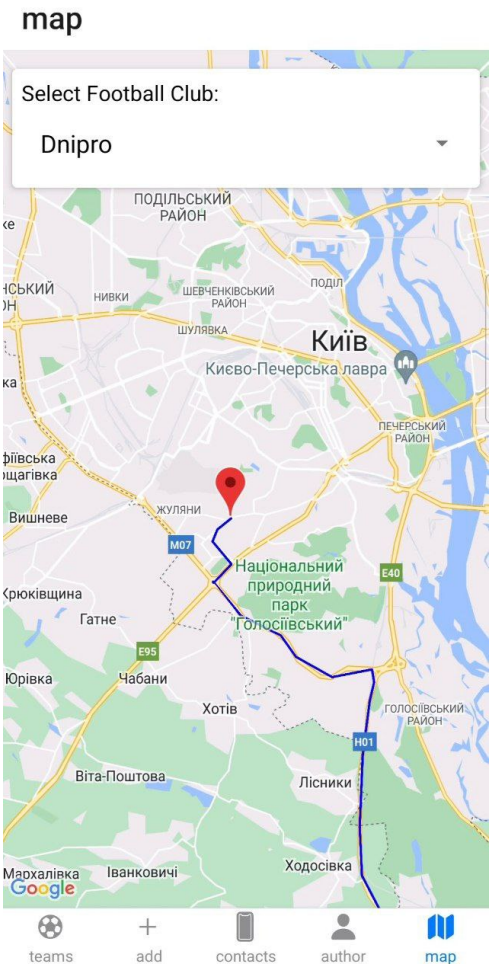

Рисунок 5 - сторінка з інтерактивною картою

Список використаних джерел

1. React Native Documentation URL:https://reactnative.dev/docs/getting-started
2. React Native Contacts - Tutorial URL: https://heartbeat.fritz.ai/accessing-contacts-in-react-native-expo-using-expo-contacts-and-react-native-contacts-57e4bd9d7011
3. MongoDB University - Free MongoDB Courses URL:https://university.mongodb.com/
4. Django - Getting Started URL:https://docs.djangoproject.com/en/3.2/intro/install/
5. Mongoose (Node.js ODM for MongoDB) Documentation URL: https://mongoosejs.com/docs/
6. Google Maps JavaScript API URL: https://developers.google.com/maps/documentation/javascript
7. Google Places API URL: https://developers.google.com/maps/documentation/places/web-service/overview
8. Google Directions API URL: https://developers.google.com/maps/documentation/directions/overview
9. Google Maps Platform - Get Started URL: https://developers.google.com/maps/gmp-get-started
10. Express.js - Getting Started URL: https://expressjs.com/en/starter/installing.html
11. Axios GitHub Repository URL: https://github.com/axios/axios
12. Axios - Making Requests URL: https://axios-http.com/docs/making-requests

Додаток (код основної частини програми)

```
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import { Ionicons } from '@expo/vector-icons';
import AuthorScreen from './screens/Author';
import ClubMap from './screens/Map';
import AddFootballClubScreen from './screens/AddTeam.jsx';
import FootballClubListScreen from './screens/FootballClubList.jsx';
import ContactsScreen from './screens/Contacts';

const Stack = createStackNavigator();
const Tab = createBottomTabNavigator();



const FootballClubsInfo = () => (
 <Stack.Navigator screenOptions={{ headerShown: false }}>
   <Stack.Screen name="ClubList" component={FootballClubListScreen} />
 </Stack.Navigator>
);
const MapRoute = () => (
 <Stack.Navigator screenOptions={{ headerShown: false }}>
   <Stack.Screen name="MapRoute" component={ClubMap} />
 </Stack.Navigator>
);
const FootballClubAdd = () => (
 <Stack.Navigator screenOptions={{ headerShown: false }}>
   <Stack.Screen name="AddClub" component={AddFootballClubScreen} />
 </Stack.Navigator>
);
const Author = () => (
 <Stack.Navigator screenOptions={{ headerShown: false }}>
   <Stack.Screen name="Author" component={AuthorScreen} />
 </Stack.Navigator>
);
const Contacts = () => (
 <Stack.Navigator screenOptions={{ headerShown: false }}>
   <Stack.Screen name="Contacts" component={ContactsScreen} />
 </Stack.Navigator>
);
```

```
const App = () => {
 return (
  <NavigationContainer>
    <Tab.Navigator
      screenOptions={({ route }) => ({
        tabBarIcon: ({ color, size }) => {
          let iconName;
          if (route.name === 'teams') {
            iconName = 'ios-football';
          } else if (route.name === 'add') {
            iconName = 'ios-add';
          } else if (route.name === 'author') {
            iconName = 'ios-person';
          } else if (route.name === 'contacts') {
            iconName = 'ios-phone-portrait';
          } else if (route.name === 'map') {
            iconName = 'ios-map';
          }
          return <Ionicons name={iconName} size={size} color={color} />;
        },
      })}
    >
      <Tab.Screen name="teams" component={FootballClubsInfo} />
      <Tab.Screen name="add" component={FootballClubAdd} />
      <Tab.Screen name="contacts" component={Contacts} />
      <Tab.Screen name="author" component={Author} />
      <Tab.Screen name="map" component={MapRoute} />
    </Tab.Navigator>
  </NavigationContainer>
 );
};

export default App;
```

```
import React, { useState } from 'react';
import { View, Text, TextInput, Button, StyleSheet, Alert } from 'react-native';

const AddFootballClubScreen = () => {
 const [name, setName] = useState('');
 const [city, setCity] = useState('');
```

```
const [foundationYear, setFoundationYear] = useState('');
const [rating, setRating] = useState('');

const handleSubmit = async () => {

 if (/^\d+$/.test(city) ) {
  Alert.alert('Validation Error', 'Please fill real city');
  return;
 }

 if (!name || !city || !foundationYear || !rating) {
  Alert.alert('Validation Error', 'Please fill in all fields');
  return;
 }
 if (foundationYear > 2023 || foundationYear < 1920) {
  Alert.alert('Validation Error', 'Please fill real foundation date');
  return;
 }


 try {
  const response = await fetch('http://192.168.31.171:3000/addClub', {
   method: 'POST',
   headers: {
    'Content-Type': 'application/json',
   },
   body: JSON.stringify({
    name,
    city,
    foundationYear: parseInt(foundationYear),
    rating: parseInt(rating),
   }),
  });
  setName('');
  setCity('');
  setFoundationYear('');
  setRating('');
  if (response.ok) {
   console.log('Football club added successfully');
  } else {
   console.error('Error adding football club');
  }
```

```
    } catch (error) {
      console.error('Network error:', error);
    }
  };

  return (
    <View style={styles.container}>
      <Text style={styles.label}>Club Name:</Text>
      <TextInput
        style={styles.input}
        value={name}
        onChangeText={(text) => setName(text)}
      />

      <Text style={styles.label}>City:</Text>
      <TextInput
        style={styles.input}
        value={city}
        onChangeText={(text) => setCity(text)}
      />

      <Text style={styles.label}>Foundation Year:</Text>
      <TextInput
        style={styles.input}
        value={foundationYear}
        onChangeText={(text) => setFoundationYear(text)}
        keyboardType="numeric"
      />

      <Text style={styles.label}>Rating:</Text>
      <TextInput
        style={styles.input}
        value={rating}
        onChangeText={(text) => setRating(text)}
        keyboardType="numeric"
      />

      <Button title="Submit" onPress={handleSubmit} />
    </View>
  );
};
```

```
const styles = StyleSheet.create({
 container: {
  flex: 1,
  padding: 16,
 },
 label: {
  fontSize: 16,
  marginBottom: 8,
 },
 input: {
  height: 40,
  borderColor: 'gray',
  borderWidth: 1,
  marginBottom: 16,
  paddingHorizontal: 8,
 },
});

export default AddFootballClubScreen;
```

```
import React from 'react';
import { Text } from 'react-native';
import {View, Image } from 'react-native';

const AuthorScreen = () => {
 return (
  <View style={{display: 'flex', justifyContent: "center", alignItems:'center'}}>
     <Image source={require('../assets/myava.jpg')}
     style={{width: 200, height: 200}} />
     <Text>Владислав Бурлака</Text>
     <Text>Студент ТТП-41</Text>
  </View>
 );
};

export default AuthorScreen;
```

```
import React, { useEffect, useState } from 'react';
import { StyleSheet, View, Text, FlatList } from 'react-native';
import * as Contacts from 'expo-contacts';
```

```
export default function ContactsScreen() {
 const [contacts, setContacts] = useState([]);
 const [knuContacts, setKnuContacts] = useState([]);

 useEffect(() => {
  const loadContacts = async () => {
   const { status } = await Contacts.requestPermissionsAsync();
   if (status === 'granted') {
    const { data } = await Contacts.getContactsAsync({
     fields: [Contacts.Fields.Emails],
    });

    if (data.length > 0) {
     setContacts(data);

const knuFilteredContacts = data.filter(contact =>
  contact.emails?.some(emailObj => emailObj.email.endsWith('@knu.ua'))
);
setKnuContacts(knuFilteredContacts);
    }
   }
  };

  loadContacts();
 }, []);

 return (
  <View style={styles.container}>
  <View style={styles.section}>
   <Text style={styles.sectionTitle}>All contacts:</Text>
   <FlatList
    style={styles.list}
    data={contacts}
    keyExtractor={item => item.id.toString()}
    renderItem={({ item }) => (
     <View style={styles.listItem}>
      <Text>{item.firstName}</Text>
     </View>
    )}
   />
  </View>
```

```
    <View style={styles.section}>
     <Text style={styles.sectionTitle}>With KNU mails, "@knu.ua":</Text>
     <FlatList
      style={styles.list}
      data={knuContacts}
      keyExtractor={item => item.id.toString()}
      renderItem={({ item }) => (
       <View style={styles.listItem}>
        <Text>{item.firstName}</Text>
       </View>
      )}
     />
    </View>
   </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    padding: 16,
  },
  section: {
    height: "45%",
    width: '100%',
    marginBottom: 16,
  },
  sectionTitle: {
    fontSize: 18,
    fontWeight: 'bold',
    marginBottom: 8,
  },
  list: {
    width: '100%',
  },
  listItem: {
    padding: 12,
    borderBottomWidth: 1,
    borderBottomColor: '#ccc',
  },
```

```
});
```

```
import React, { useEffect, useState, useCallback } from 'react';
import { View, Text, FlatList, Button, StyleSheet, Alert } from 'react-native';
import { useFocusEffect } from '@react-navigation/native';

const FootballClubListScreen = () => {
 const [footballClubData, setFootballClubData] = useState([]);
 const [totalRatingKyiv, setTotalRatingKyiv] = useState(0);

 const fetchFootballClubData = useCallback(() => {
  fetch('http://192.168.31.171:3000/getAllClubs')
   .then((response) => response.json())
   .then((data) => setFootballClubData(data))
   .catch((error) => console.error('Error fetching football club data:', error));
 }, []);

 const handleDeleteFootballClub = async (id) => {
  try {
   const response = await fetch(`http://192.168.31.171:3000/deleteClub/${id}`, {
    method: 'DELETE',
   });

   if (response.ok) {
    console.log('Football club deleted successfully');
    fetchFootballClubData();
   } else {
    console.error('Error deleting football club');
   }
  } catch (error) {
   console.error('Network error:', error);
  }
 };

 useEffect(() => {
  const kyivClubs = footballClubData.filter((club) => club.city.toLowerCase() === 'kyiv' ||
club.city.toLowerCase() === 'київ');
  console.log(kyivClubs)
  const sumRating = kyivClubs.reduce((sum, club) => sum + club.rating, 0);
  setTotalRatingKyiv(sumRating);
 }, [footballClubData]);
```

```
useFocusEffect(
  React.useCallback(() => {
    fetchFootballClubData();
  }, [fetchFootballClubData])
);

const confirmDelete = (id, name) => {
  Alert.alert(
    'Confirm Deletion',
    `Are you sure you want to delete the club ${name}?`,
    [
      { text: 'Cancel', style: 'cancel' },
      { text: 'Delete', onPress: () => handleDeleteFootballClub(id) },
    ],
    { cancelable: false }
  );
};

return (
  <View style={styles.container}>
    <Text style={styles.header}>Football Clubs</Text>
    <FlatList
      data={footballClubData}
      keyExtractor={(item) => item._id}
      renderItem={({ item }) => (
        <View style={styles.item}>
          <Text>Name: {item.name}</Text>
          <Text>Rating: {item.rating}</Text>
          <Button
            title="Delete"
            onPress={() => confirmDelete(item._id, item.name)}
            color="red"
          />
        </View>
      )}
    />
    <Text style={styles.header}>Kyiv Clubs</Text>
    <Text style={styles.totalRatingText}>
      Total Rating of Kyiv Clubs: {totalRatingKyiv ? totalRatingKyiv.toFixed(2) : "0"}
    </Text>
    <FlatList
```

```
        data={footballClubData.filter((club) => club.city === 'Kyiv')}
        keyExtractor={(item) => item._id}
        renderItem={({ item }) => (
          <View style={styles.item}>
            <Text>Name: {item.name}</Text>
            <Text>Rating: {item.rating}</Text>
            <Button
              title="Delete"
              onPress={() => confirmDelete(item._id, item.name)}
              color="red"
            />
          </View>
        )}
      />
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 16,
  },
  header: {
    marginLeft:"35%",
    fontSize: 24,
    fontWeight: 'bold',
    marginBottom: 16,
  },
  item: {
    marginBottom: 8,
    borderWidth: 1,
    borderColor: 'gray',
    padding: 5,
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
  },
  totalRatingText: {
    fontSize: 18,
    fontWeight: 'bold',
    marginBottom: 8,
```

```
  },
});

export default FootballClubListScreen;
```

```
import React, { useEffect, useState } from "react";
import { Dimensions, StyleSheet, Text, View } from "react-native";
import { Picker } from "@react-native-picker/picker";
import MapView, { Marker } from "react-native-maps";
import MapViewDirections from 'react-native-maps-directions';
import * as Location from 'expo-location';
import axios from 'axios';


const FootballClubRoute = () => {
 const [location, setLocation] = useState(null);
 const [errorMsg, setErrorMsg] = useState(null);
 const [selectedClub, setSelectedClub] = useState(null);
 const [clubs, setClubs] = useState([]);
 const [destination, setDestination] = useState(null);

 useEffect(() => {
  (async () => {
    let { status } = await Location.requestForegroundPermissionsAsync();
    if (status !== 'granted') {
      setErrorMsg('Permission to access location was denied');
      return;
    }

    let location = await Location.getCurrentPositionAsync({});
    setLocation(location.coords);
  })();

 axios.get("http://192.168.31.171:3000/getAllClubs")
      .then(response => {
        setClubs(response.data);
      })
      .catch(error => {
        console.error("Error fetching clubs:", error);
      });
}, []);
```

```
  const handleClubChange = (club) => {
    setSelectedClub(club);
    getClubCoordinates(club.city);
  console.log(club.city)
  };

  const getClubCoordinates = (city) => {
    const apiKey = '';
    const geocodeUrl =
`https://maps.googleapis.com/maps/api/geocode/json?address=${encodeURIComponent(city)}&key=${api
Key}`;

    axios.get(geocodeUrl)
      .then(response => {
        const { results } = response.data;
        if (results.length > 0) {
          const { geometry } = results[0];
          const { location } = geometry;
          setDestination(location);
        } else {
          console.error('No results found for the city:', city);
        }
      })
      .catch(error => {
        console.error('Error fetching coordinates:', error);
      });
  };

  return (
    <View style={styles.container}>
      {location && (
        <MapView
          style={styles.map}
          initialRegion={{
            latitude: location.latitude,
            longitude: location.longitude,
            latitudeDelta: 0.0922,
            longitudeDelta: 0.0421,
          }}
        >
          <Marker
```

```
          coordinate={{ latitude: location.latitude, longitude: location.longitude }}
        />
        {destination && (
          <>
            <Marker
              coordinate={{ latitude: destination.lat, longitude: destination.lng }}
            />
      <MapViewDirections
        origin={{ latitude: location.latitude, longitude: location.longitude }}
     destination={{ latitude: destination.lat, longitude: destination.lng }}
          apikey=""
         strokeWidth={2}
        strokeColor="blue"
        />
         </>
        )}
      </MapView>
    )}

    <View style={styles.pickerContainer}>
      <Text>Select Football Club: </Text>
      <Picker
        selectedValue={selectedClub}
        onValueChange={(itemValue) => handleClubChange(itemValue)}
      >
        {clubs.map((club, index) => (
          <Picker.Item  label={club.name} value={club} />
        ))}
      </Picker>
    </View>
  </View>
 );
};

const styles = StyleSheet.create({
 container: {
  ...StyleSheet.absoluteFillObject,
  justifyContent: 'flex-end',
  alignItems: 'center',
 },
 map: {
  ...StyleSheet.absoluteFillObject,
```

```
  },
  pickerContainer: {
    position: 'absolute',
    top: 16,
    left: 8,
    right: 8,
    backgroundColor: 'white',
    borderRadius: 5,
    padding: 8,
    elevation: 4,
  },
});

export default FootballClubRoute;
```