App.js

```
import React, { useState } from 'react';
import {
  View,
  Text,
  TextInput,
  TouchableOpacity,
  StyleSheet,
  FlatList,
  Alert,
  SafeAreaView,
  ScrollView,
  Dimensions
} from 'react-native';

const { width, height } = Dimensions.get('window');

// Simple icon replacement using text/emojis
const Icon = ({ name, size = 24, color = 'black' }) => {
  const iconMap = {
    'checkmark-circle': '✓',
    'location': '📍',
    'trash': '🗑️',
    'close-circle': '✗',
    'search': '🔍',
    'person-circle-outline': '👤',
    'eye': '👁️'
  };

  return (
    <Text style={{ fontSize: size, color }}>
      {iconMap[name] || 'o'}
    </Text>
  );
};

export default function App() {
```

```
const [currentScreen, setCurrentScreen] = useState('DeveloperMenu');

// DeveloperMenu Component
const DeveloperMenu = () => {
  const handleLogout = () => {
    Alert.alert(
      'Logout',
      'Are you sure you want to logout?',
      [
        { text: 'Cancel', style: 'cancel' },
        {
          text: 'Logout',
          style: 'destructive',
          onPress: () => {
            Alert.alert('Success', 'Logged out successfully');
          }
        }
      ]
    );
  };

  return (
    <SafeAreaView style={styles.container}>
      <View style={styles.lowerHeader}>
        <Text style={styles.title}>DEVELOPER MENU</Text>
        <View style={styles.line} />
      </View>

      <View style={styles.centerContent}>
        <TouchableOpacity
          style={styles.button}
          onPress={() => setCurrentScreen('RegistrationRequests')}
        >
          <Text style={styles.buttonText}>Registration Requests</Text>
        </TouchableOpacity>

        <TouchableOpacity
          style={styles.button}
          onPress={() => setCurrentScreen('DeleteEvacuationSites')}
```

```
          >
            <Text style={styles.buttonText}>Delete Evacuation
Sites</Text>
          </TouchableOpacity>

          <TouchableOpacity
            style={styles.button}
            onPress={() => setCurrentScreen('AddEvacuationSite')}
          >
            <Text style={styles.buttonText}>Add Evacuation Sites</Text>
          </TouchableOpacity>

          <TouchableOpacity
            style={styles.button}
            onPress={() => setCurrentScreen('SiteManagerDeletion')}
          >
            <Text style={styles.buttonText}>Site Manager Deletion</Text>
          </TouchableOpacity>

          <TouchableOpacity style={styles.logoutButton}
onPress={handleLogout}>
            <Text style={styles.logoutText}>Log out</Text>
          </TouchableOpacity>
        </View>
      </SafeAreaView>
    );
  };

  // AddEvacuationSite Component
  const AddEvacuationSite = () => {
    const [site, setSite] = useState('');
    const [added, setAdded] = useState(false);

    const handleAddSite = () => {
      if (!site.trim()) {
        Alert.alert('Error', 'Please enter a location for the evacuation
site');
        return;
      }
```

```
      setAdded(true);
    };

    const handleReset = () => {
      setSite('');
      setAdded(false);
    };

    return (
      <View style={styles.container}>
        <View style={styles.lowerHeader}>
          <Text style={styles.title}>ADD EVACUATION SITE</Text>
          <View style={styles.line} />
        </View>

        <View style={styles.centerContent}>
          {!added ? (
            <>
              <TextInput
                placeholder="Input the location of the evacuation site"
                style={styles.input}
                value={site}
                onChangeText={setSite}
              />
              <TouchableOpacity
                style={[styles.addButton, !site.trim() &&
styles.disabledButton]}
                onPress={handleAddSite}
                disabled={!site.trim()}
              >
                <Icon name="checkmark-circle" size={24} color="#fff" />
                <Text style={styles.buttonText}>Add Site</Text>
              </TouchableOpacity>
            </>
          ) : (
            <View style={styles.successBox}>
              <Text style={styles.successText}>Evacuation site located
@{site}</Text>
              <Text style={styles.successText}>Has been successfully
```

```
added</Text>
              <TouchableOpacity style={styles.resetButton}
onPress={handleReset}>
                <Text style={styles.resetText}>Add Another Site</Text>
              </TouchableOpacity>
            </View>
          )}

          <TouchableOpacity style={styles.backButton} onPress={() =>
setCurrentScreen('DeveloperMenu')}>
            <Text style={styles.backText}>Return to menu</Text>
          </TouchableOpacity>
        </View>
      </View>
    );
  };

  // DeleteEvacuationSites Component
  const DeleteEvacuationSites = () => {
    const initialSites = [
      'Site 1 - Main Evacuation Center',
      'Site 2 - Community Hall',
      'Site 3 - School Gym',
      'Site 4 - Sports Complex',
      'Site 5 - Public Library',
      'Site 6 - Park Pavilion',
      'Site 7 - Emergency Shelter',
      'Site 8 - Community Center',
      'Site 9 - High School Auditorium',
      'Site 10 - Municipal Building'
    ];
    const [sites, setSites] = useState(initialSites);

    const handleDeleteSite = (siteToDelete) => {
      Alert.alert(
        'Confirm Deletion',
        `Are you sure you want to delete "${siteToDelete}"?`,
        [
          { text: 'Cancel', style: 'cancel' },
```

```
          {
            text: 'Delete',
            style: 'destructive',
            onPress: () => {
              setSites(sites.filter(site => site !== siteToDelete));
              Alert.alert('Success', 'Evacuation site deleted
successfully');
            }
          }
        ]
      );
    };

    const handleViewOnMap = (site) => {
      Alert.alert('View on Map', `Showing location of: ${site}`);
    };

    return (
      <View style={styles.container}>
        <View style={styles.lowerHeader}>
          <Text style={styles.title}>EVAC SITE DELETION</Text>
          <View style={styles.line} />
        </View>

        <FlatList
          data={sites}
          keyExtractor={(item) => item}
          renderItem={({ item }) => (
            <View style={styles.listItem}>
              <Text style={styles.listText}>{item}</Text>
              <View style={styles.iconContainer}>
                <TouchableOpacity onPress={() => handleViewOnMap(item)}>
                  <Icon name="location" size={22} color="red" />
                </TouchableOpacity>
                <TouchableOpacity onPress={() =>
handleDeleteSite(item)}>
                  <Icon name="trash" size={22} color="red" />
                </TouchableOpacity>
              </View>
```

```
            </View>
          )}
          style={styles.list}
          contentContainerStyle={styles.listContent}
          showsVerticalScrollIndicator={false}
        />

        <TouchableOpacity style={styles.backButton} onPress={() =>
setCurrentScreen('DeveloperMenu')}>
          <Text style={styles.backText}>Return to menu</Text>
        </TouchableOpacity>
      </View>
    );
  };

  // RegistrationRequests Component
  const RegistrationRequests = () => {
    const initialUsers = [
      'John Doe - 09956982021',
      'Jane Smith - 09556323456',
      'Mike Johnson - 09451235752',
      'Sarah Wilson - 09564123789',
      'Tom Brown - 091542032',
      'Emily Davis - 09452163652',
      'Robert Wilson - 09746985210',
      'Lisa Anderson - 09432612056'
    ];
    const [users, setUsers] = useState(initialUsers);

    const handleApprove = (user) => {
      Alert.alert(
        'Approve Registration',
        `Approve registration for ${user}?`,
        [
          { text: 'Cancel', style: 'cancel' },
          {
            text: 'Approve',
            onPress: () => {
              setUsers(users.filter(u => u !== user));
```

```
              Alert.alert('Success', 'Registration approved
successfully');
            }
          }
        ]
      );
    };

    const handleReject = (user) => {
      Alert.alert(
        'Reject Registration',
        `Reject registration for ${user}?`,
        [
          { text: 'Cancel', style: 'cancel' },
          {
            text: 'Reject',
            style: 'destructive',
            onPress: () => {
              setUsers(users.filter(u => u !== user));
              Alert.alert('Success', 'Registration rejected
successfully');
            }
          }
        ]
      );
    };

    return (
      <View style={styles.container}>
        <View style={styles.lowerHeader}>
          <Text style={styles.title}>REGISTRATION REQUESTS</Text>
          <View style={styles.line} />
        </View>

        <FlatList
          data={users}
          keyExtractor={(item) => item}
          renderItem={({ item }) => (
            <View style={styles.listItem}>
```

```
              <Text style={styles.listText}>{item}</Text>
              <View style={styles.iconContainer}>
                <TouchableOpacity onPress={() => handleApprove(item)}>
                  <Icon name="checkmark-circle" size={24} color="green"
/>
                </TouchableOpacity>
                <TouchableOpacity onPress={() => handleReject(item)}>
                  <Icon name="close-circle" size={24} color="red" />
                </TouchableOpacity>
              </View>
            </View>
          )}
          style={styles.list}
          contentContainerStyle={styles.listContent}
          showsVerticalScrollIndicator={false}
        />

        <TouchableOpacity style={styles.backButton} onPress={() =>
setCurrentScreen('DeveloperMenu')}>
          <Text style={styles.backText}>Return to menu</Text>
        </TouchableOpacity>
      </View>
    );
  };

  // SiteManagerDeletion Component
  const SiteManagerDeletion = () => {
    const [users, setUsers] = useState([
      { id: "1", name: "John Doe - Main Center" },
      { id: "2", name: "Jane Smith - Community Hall" },
      { id: "3", name: "Mike Johnson - School Gym" },
      { id: "4", name: "Sarah Wilson - Sports Complex" },
      { id: "5", name: "Tom Brown - Public Library" },
      { id: "6", name: "Emily Davis - Park Pavilion" },
      { id: "7", name: "Robert Lee - Emergency Shelter" },
      { id: "8", name: "Maria Garcia - Community Center" },
    ]);

    const handleViewDetails = (user) => {
```

```
      Alert.alert('Manager Info', `Viewing details for: ${user.name}`);
    };

    const handleDelete = (user) => {
      Alert.alert(
        "Delete Site Manager",
        `Are you sure you want to delete "${user.name}"?`,
        [
          { text: "Cancel", style: "cancel" },
          {
            text: "Delete",
            style: "destructive",
            onPress: () => {
              setUsers(users.filter(u => u.id !== user.id));
              Alert.alert("Success", "Site manager deleted
successfully");
            }
          }
        ]
      );
    };

    return (
      <View style={styles.container}>
        <View style={styles.lowerHeader}>
          <Text style={styles.title}>SITE MANAGER DELETION</Text>
          <View style={styles.line} />
        </View>

        <FlatList
          data={users}
          keyExtractor={(item) => item.id}
          renderItem={({ item }) => (
            <View style={styles.listItem}>
              <Icon name="person-circle-outline" size={24} color="black"
/>
              <Text style={styles.itemText}>{item.name}</Text>
              <View style={styles.iconContainer}>
                <TouchableOpacity onPress={() =>
```

```
handleViewDetails(item)}>
                  <Icon name="eye" size={24} color="blue" />
                </TouchableOpacity>
                <TouchableOpacity onPress={() => handleDelete(item)}>
                  <Icon name="trash" size={24} color="red" />
                </TouchableOpacity>
              </View>
            </View>
          )}
          style={styles.list}
          contentContainerStyle={styles.listContent}
          showsVerticalScrollIndicator={false}
        />

        <TouchableOpacity
          style={styles.backButton}
          onPress={() => setCurrentScreen('DeveloperMenu')}
        >
          <Text style={styles.backText}>Return to Menu</Text>
        </TouchableOpacity>
      </View>
    );
  };

  // Render current screen
  const renderScreen = () => {
    switch (currentScreen) {
      case 'RegistrationRequests':
        return <RegistrationRequests />;
      case 'DeleteEvacuationSites':
        return <DeleteEvacuationSites />;
      case 'AddEvacuationSite':
        return <AddEvacuationSite />;
      case 'SiteManagerDeletion':
        return <SiteManagerDeletion />;
      default:
        return <DeveloperMenu />;
    }
  };
```

```
  return renderScreen();
}

// Styles
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#f2f2f2',
  },
  lowerHeader: {
    alignItems: 'center',
    paddingTop: height * 0.05,
    paddingBottom: 10,
    backgroundColor: '#f2f2f2',
  },
  centerContent: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
    paddingHorizontal: 20,
  },
  scrollContent: {
    flex: 1,
    width: '100%',
  },
  scrollContainer: {
    alignItems: 'center',
    padding: 20,
    paddingTop: 10,
  },
  list: {
    flex: 1,
    width: '100%',
  },
  listContent: {
    padding: 20,
    paddingBottom: 80,
  },
```

```
  title: {
    fontSize: width * 0.05,
    fontWeight: 'bold',
    textAlign: 'center',
    color: '#333',
  },
  line: {
    width: '60%',
    height: 2,
    backgroundColor: '#333',
    marginVertical: 10
  },
  input: {
    width: '90%',
    backgroundColor: '#fff',
    borderRadius: 10,
    padding: 15,
    marginBottom: 20,
    fontSize: 16,
    borderWidth: 1,
    borderColor: '#ddd',
    minHeight: 50,
  },
  addButton: {
    backgroundColor: '#007BFF',
    padding: 15,
    borderRadius: 25,
    flexDirection: 'row',
    alignItems: 'center',
    gap: 10,
    minWidth: 150,
    justifyContent: 'center',
    minHeight: 50,
  },
  disabledButton: {
    backgroundColor: '#ccc',
  },
  button: {
    backgroundColor: '#FFA500',
```

```
    borderRadius: 25,
    paddingVertical: 15,
    paddingHorizontal: 30,
    marginVertical: 10,
    width: width * 0.8,
    alignItems: 'center',
    minHeight: 50,
    justifyContent: 'center',
  },
  buttonText: {
    color: '#fff',
    fontWeight: 'bold',
    fontSize: 16
  },
  successBox: {
    backgroundColor: '#fff',
    padding: 25,
    borderRadius: 15,
    alignItems: 'center',
    width: '90%',
    borderWidth: 1,
    borderColor: '#ddd',
    marginBottom: 20,
  },
  successText: {
    fontSize: 16,
    marginBottom: 5,
    textAlign: 'center'
  },
  resetButton: {
    backgroundColor: '#28a745',
    paddingVertical: 12,
    paddingHorizontal: 25,
    borderRadius: 20,
    marginTop: 15,
    minHeight: 44,
    justifyContent: 'center',
  },
  resetText: {
```

```
    color: '#fff',
    fontWeight: 'bold',
    fontSize: 14,
  },
  backButton: {
    backgroundColor: '#007BFF',
    paddingVertical: 12,
    paddingHorizontal: 30,
    borderRadius: 20,
    marginTop: 20,
    marginBottom: 20,
    minHeight: 44,
    justifyContent: 'center',
    alignSelf: 'center',
  },
  backText: {
    color: '#fff',
    fontWeight: 'bold',
    fontSize: 16
  },
  logoutButton: {
    backgroundColor: '#007BFF',
    borderRadius: 20,
    paddingVertical: 12,
    paddingHorizontal: 30,
    marginTop: 20,
    minHeight: 44,
    justifyContent: 'center',
  },
  logoutText: {
    color: '#fff',
    fontWeight: 'bold',
    fontSize: 16
  },
  listItem: {
    width: '100%',
    backgroundColor: '#fff',
    borderRadius: 15,
    padding: 15,
```

```
    marginBottom: 10,
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
    borderWidth: 1,
    borderColor: '#ddd',
    minHeight: 60,
  },
  listText: {
    fontSize: 16,
    flex: 1,
    marginRight: 10
  },
  itemText: {
    flex: 1,
    marginLeft: 10,
    fontSize: 16,
  },
  iconContainer: {
    flexDirection: 'row',
    gap: 15
  },
});
```

package.json

```
{
  "dependencies": {}
}
```