BAZKOR

Ez az applikáció egy orvosi időpontfoglalási rendszer, amely lehetővé teszi a felhasználók számára, hogy könnyedén foglaljanak időpontokat a Borsod-Abaúj-Zemplén Vármegyei Központi Kórház és Egyetemi Oktatókórházba. Az alkalmazás főbb funkciói közé tartozik a regisztráció és bejelentkezés, amelyet a Firebase hitelesítési szolgáltatása kezel. A felhasználók az alkalmazásban megadhatják a foglalás dátumát, az időpontot, valamint a probléma rövid leírását, majd sikeres foglalás után értesítést kapnak (visszajelzést a foglalásról).

Az alkalmazás egy modern, könnyen használható felhasználói felülettel rendelkezik, amely világos és sötét témát is támogat. Az értesítések fülön a felhasználók nyomon követhetik a korábbi foglalásaikat, és a beállítások menüben lehetőségük van a fiókjuk törlésére vagy a témaváltásra. Az egyszerű navigációs rendszer lehetővé teszi a gyors és zökkenőmentes használatot, biztosítva, hogy az időpontfoglalás egyszerű és hatékony legyen.

Használt technológiák

I. React Native

- **Leírás**: Egy népszerű JavaScript-keretrendszer, amely lehetővé teszi a natív mobilalkalmazások fejlesztését Android és iOS platformokra egyetlen kódbázissal.
- **Funkciók**: UI-komponensek létrehozása, állapotkezelés, platformfüggetlen fejlesztés.

II. Firebase

- **Firebase Authentication**: Felhasználói hitelesítés regisztrációhoz, bejelentkezéshez és fióktörléshez. Gondoskodik az adatok biztonságos kezeléséről.
- **Firebase Initialization**: Az alkalmazás konfigurálása és inicializálása a Firebase projekt adataival.

III. React Navigation

- **Leírás**: Egy erőteljes navigációs könyvtár, amely támogatja a különböző navigációs mintákat, például az alsó tab navigációt.
- **Funkciók**: Könnyű és intuitív navigáció a képernyők között.

IV. JavaScript (ES6)

- **Leírás**: Az alkalmazás fő fejlesztési nyelve, amely biztosítja az alkalmazás logikáját, interakcióit és állapotkezelését.
- **Funkciók**: Állapotkezelés, eseménykezelés, és a React komponensek kezelése.

V. React Hooks

- **useState**: Állapotok kezelésére a különböző komponensekben.
- **useEffect**: (Nincs használatban a bemutatott kódban, de gyakran használt React Hook a mellékhatások kezelésére).

VI. CSS-in-JS (StyleSheet API)

- **Leírás**: A React Native StyleSheet API-ját használjuk a stílusok definiálására, amely optimalizált natív teljesítményt nyújt.
- **Funkciók**: Egységes és platformfüggetlen stílusok alkalmazása az UI-komponenseken.

1. **Firebase Konfiguráció és Inicializálás**

```
const firebaseConfig = {
  apiKey: "AIzaSyBRDW6IfIKNEcKrI8PQZ6GyuMZ6ULtn5Qo",
  authDomain: "beadando1-10f1d.firebaseapp.com",
  projectId: "beadando1-10f1d",
  storageBucket: "beadando1-10f1d.firebasestorage.app",
  messagingSenderId: "1002869907390",
  appId: "1:1002869907390:web:eefc7d64a29b9c72e663ef",
  measurementId: "G-P4WTQRF3K6"
};

if (!getApps().length) {
  initializeApp(firebaseConfig);
}
const auth = getAuth();
```

- **firebaseConfig**: Tartalmazza a Firebase projekt konfigurációs adatait, amelyek szükségesek a Firebase szolgáltatások használatához.
- **Firebase inicializálás**: Ellenőrzi, hogy a Firebase már inicializálva van-e. Ha nem, inicializálja a Firebase alkalmazást.
- **auth**: A Firebase Authentication objektum, amely kezeli a felhasználói hitelesítést.

---

2. **Navigációs Sáv Létrehozása**

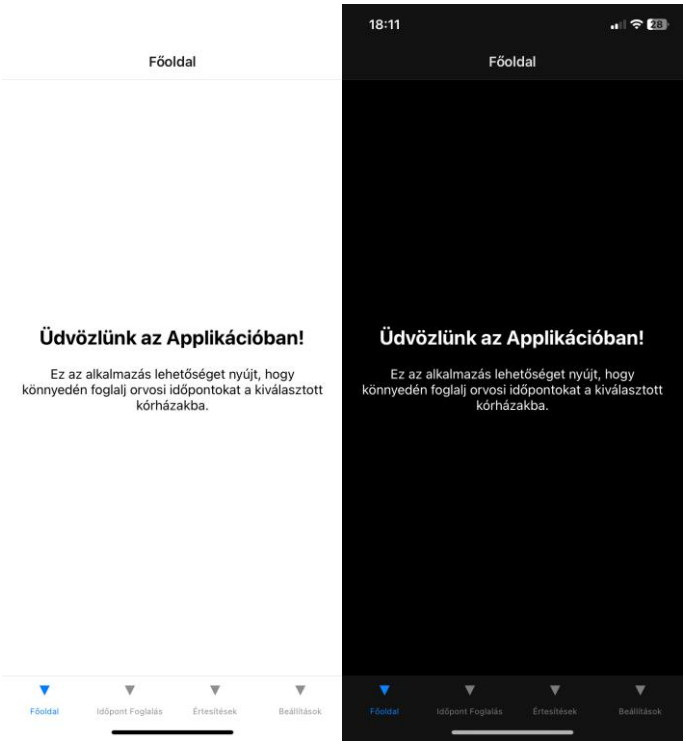
```
const Tab = createBottomTabNavigator();
```

- **Tab Navigator**: Létrehozza az alsó navigációs sávot, amely lehetővé teszi a különböző képernyők közötti navigációt.

3. **HomeScreen Komponens**

```
function HomeScreen({ theme }) {
 return (
  <View style={[styles.container, { backgroundColor: theme === 'dark' ? '#000' : '#fff' }]}>
   <Text style={[styles.title, { color: theme === 'dark' ? '#fff' : '#000' }]}>
    Üdvözlünk az Applikációban!
   </Text>
   <Text style={[styles.text, { color: theme === 'dark' ? '#fff' : '#000' }]}>
    Ez az alkalmazás lehetőséget nyújt, hogy könnyedén foglalj orvosi időpontokat a kiválasztott kórházakba.
   </Text>
  </View>
 );
}
```

- **Üdvözlő képernyő**: Egy egyszerű képernyő, amely üdvözli a felhasználót, és rövid információt ad az alkalmazásról.
- **Téma kezelés**: A szöveg és a háttér színe a kiválasztott témától (világos vagy sötét) függően változik.



4. **BookingScreen Komponens**

```
function BookingScreen({ theme, onBookingConfirmed }) {
 const hospital = 'Borsod-Abaúj-Zemplén Vármegyei Központi Kórház és Egyetemi Oktatókórház';
 const [date, setDate] = useState('');
 const [time, setTime] = useState('');
 const [description, setDescription] = useState('');

 const handleBooking = () => {
  if (date.trim() === '' || time.trim() === '' || description.trim() === '') {
   Alert.alert('Hiba', 'Minden mezőt ki kell tölteni!');
   return;
  }

  const timeRegex = /^([0-1][0-9]|2[0-3]):([0-5][0-9])$/;
  if (!timeRegex.test(time)) {
   Alert.alert('Hiba', 'Az időpont formátuma HH:MM (pl. 06:00 vagy 16:00) legyen.');
   return;
  }

  const [hours] = time.split(':').map(Number);
  if (hours < 6 || hours > 16) {
   Alert.alert('Hiba', 'Az időpontnak 06:00 és 16:00 között kell lennie.');
   return;
```

```
  }

  const bookingDetails = `Sikeres időpont foglalás: ${date} ${time}`;
  onBookingConfirmed(bookingDetails); // Értesítés hozzáadása
  Alert.alert('Foglalás sikeres!', bookingDetails);
 };

 return (
  <View style={[styles.container, { backgroundColor: theme === 'dark' ? '#000' : '#fff' }]}>
   <Text style={[styles.title, { color: theme === 'dark' ? '#fff' : '#000' }]}>Időpont
Foglalás</Text>
   <Text style={{ color: theme === 'dark' ? '#fff' : '#000' }}>Kórház: {hospital}</Text>

   <TextInput
    style={[styles.input, { color: theme === 'dark' ? '#fff' : '#000' }]}
    placeholder="Dátum (pl. 2024-11-05)"
    placeholderTextColor={theme === 'dark' ? '#ccc' : '#666'}
    value={date}
    onChangeText={setDate}
   />

   <TextInput
    style={[styles.input, { color: theme === 'dark' ? '#fff' : '#000' }]}
    placeholder="Idő (pl. 06:00)"
    placeholderTextColor={theme === 'dark' ? '#ccc' : '#666'}
    value={time}
    onChangeText={setTime}
   />

   <TextInput
    style={[styles.input, { color: theme === 'dark' ? '#fff' : '#000' }]}
    placeholder="Írja le panaszát"
    placeholderTextColor={theme === 'dark' ? '#ccc' : '#666'}
    value={description}
    onChangeText={setDescription}
   />

   <TouchableOpacity style={styles.button} onPress={handleBooking}>
    <Text style={styles.buttonText}>Időpont Foglalása</Text>
   </TouchableOpacity>
```

```
  </View>
 );
}
```

- **Időpont foglalás**: Egy képernyő, ahol a felhasználó megadhatja az időpont foglalásának adatait (dátum, időpont, probléma leírása).
- **Érvényesítés**: Ellenőrzi, hogy minden mező kitöltött-e, és hogy az időpont megfelelő-e (06:00 és 16:00 között).
- **Foglalás kezelése**: Sikeres foglalás után értesítést küld a felhasználónak.

5. **NotificationsScreen Komponens**

```
function NotificationsScreen({ theme, notifications }) {
 return (
  <View style={[styles.container, { backgroundColor: theme === 'dark' ? '#000' : '#fff' }]}>
   <Text style={[styles.title, { color: theme === 'dark' ? '#fff' : '#000' }]}>Értesítések</Text>
   {notifications.length === 0 ? (
    <Text style={{ color: theme === 'dark' ? '#fff' : '#000' }}>Nincsenek értesítések.</Text>
   ) : (
    notifications.map((notification, index) => (
```

```
      <View key={index} style={styles.notificationContainer}>
       <Text style={{ color: theme === 'dark' ? '#fff' : '#000' }}>{notification}</Text>
       {index < notifications.length - 1 && <View style={styles.separator} />}
      </View>
     ))
    )}
  </View>
 );
}
```

- **Értesítések**: Megjeleníti a felhasználó által lefoglalt időpontokat. Ha nincs értesítés, egy üzenetet jelenít meg.
- **Téma kezelés**: A szöveg és a háttér színe a kiválasztott témától függően változik.

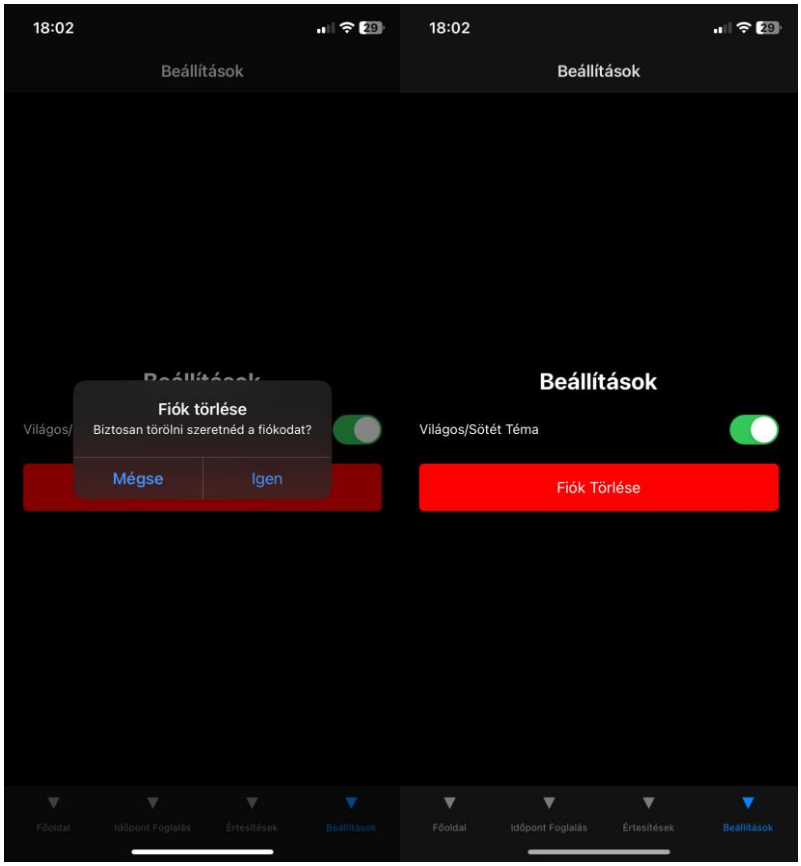
6. SettingsScreen Komponens

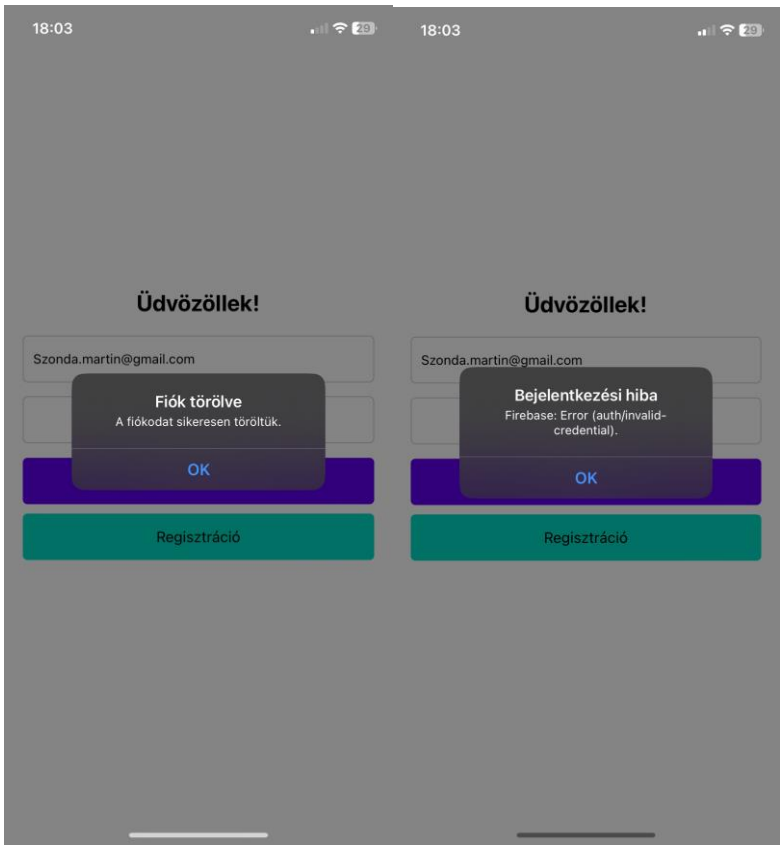
```
function SettingsScreen({ theme, toggleTheme, onAccountDeleted }) {
 const handleDeleteAccount = () => {
  Alert.alert('Fiók törlése', 'Biztosan törölni szeretnéd a fiókodat?', [
   { text: 'Mégse', style: 'cancel' },
   {
    text: 'Igen',
    onPress: () => {
     const user = auth.currentUser;
     if (user) {
      deleteUser(user)
       .then(() => {
        Alert.alert('Fiók törölve', 'A fiókodat sikeresen töröltük.');
```

```
            onAccountDeleted();
          })
          .catch((error) => {
            Alert.alert('Hiba', error.message);
          });
        } else {
          Alert.alert('Hiba', 'Nincs bejelentkezett felhasználó.');
        }
      },
    },
  ]);
};

return (
  <View style={[styles.container, { backgroundColor: theme === 'dark' ? '#000' : '#fff' }]}>
    <Text style={[styles.title, { color: theme === 'dark' ? '#fff' : '#000' }]}>Beállítások</Text>
    <View style={styles.switchContainer}>
      <Text style={{ color: theme === 'dark' ? '#fff' : '#000' }}>Világos/Sötét Téma</Text>
      <Switch value={theme === 'dark'} onValueChange={toggleTheme} />
    </View>
    <TouchableOpacity style={styles.deleteButton} onPress={handleDeleteAccount}>
      <Text style={styles.deleteButtonText}>Fiók Törlése</Text>
    </TouchableOpacity>
  </View>
);
}
```

- **Téma váltás**: A felhasználó válthat a világos és sötét téma között.
- **Fiók törlése**: Egy gomb, amely törli a felhasználói fiókot, miután megerősítést kér.





7. App Főkomponens

```
export default function App() {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [isAuthenticated, setIsAuthenticated] = useState(false);
  const [theme, setTheme] = useState('light');
  const [notifications, setNotifications] = useState([]);

  const toggleTheme = () => {
    setTheme(theme === 'light' ? 'dark' : 'light');
  };

  const handleSignUp = async () => {
    try {
      await createUserWithEmailAndPassword(auth, email, password);
      Alert.alert('Sikeres regisztráció', 'Fiók létrehozva!');
      setIsAuthenticated(true);
    } catch (error) {
      Alert.alert('Regisztrációs hiba', error.message);
    }
  };

  const handleLogin = async () => {
    try {
      await signInWithEmailAndPassword(auth, email, password);
      Alert.alert('Sikeres bejelentkezés', 'Bejelentkeztél!');
      setIsAuthenticated(true);
    } catch (error) {
      Alert.alert('Bejelentkezési hiba', error.message);
    }
  };

  const handleAccountDeleted = () => {
    setIsAuthenticated(false);
  };

  const handleBookingConfirmed = (bookingDetails) => {
    setNotifications([...notifications, bookingDetails]);
  };

  if (!isAuthenticated) {
```

```
  return (
    <View style={styles.container}>
      <Text style={styles.title}>Üdvözöllek!</Text>
      <TextInput
        style={styles.input}
        placeholder="Email"
        placeholderTextColor="#ccc"
        value={email}
        onChangeText={(text) => setEmail(text)}
        keyboardType="email-address"
      />
      <TextInput
        style={styles.input}
        placeholder="Jelszó"
        placeholderTextColor="#ccc"
        value={password}
        onChangeText={(text) => setPassword(text)}
        secureTextEntry
      />
      <TouchableOpacity style={styles.button} onPress={handleLogin}>
        <Text style={styles.buttonText}>Bejelentkezés</Text>
      </TouchableOpacity>
      <TouchableOpacity style={styles.buttonSecondary} onPress={handleSignUp}>
        <Text style={styles.buttonSecondaryText}>Regisztráció</Text>
      </TouchableOpacity>
    </View>
  );
}

 return (
  <NavigationContainer theme={theme === 'dark' ? DarkTheme : DefaultTheme}>
    <Tab.Navigator>
      <Tab.Screen name="Főoldal">
        {() => <HomeScreen theme={theme} />}
      </Tab.Screen>
      <Tab.Screen name="Időpont Foglalás">
        {() => <BookingScreen theme={theme}
onBookingConfirmed={handleBookingConfirmed} />}
      </Tab.Screen>
      <Tab.Screen name="Értesítések">
```

```
      {() => <NotificationsScreen theme={theme} notifications={notifications} />}
    </Tab.Screen>
    <Tab.Screen name="Beállítások">
      {() => <SettingsScreen theme={theme} toggleTheme={toggleTheme}
onAccountDeleted={handleAccountDeleted} />}
    </Tab.Screen>
   </Tab.Navigator>
  </NavigationContainer>
 );
}
```

- **Állapotkezelés**: Tárolja a felhasználó bejelentkezési adatait, a hitelesítés állapotát, a témát és az értesítéseket.
- **Bejelentkezés és regisztráció**: Kezeli a felhasználói hitelesítést a Firebase használatával.
- **Téma váltás**: Kezeli a világos és sötét téma közötti váltást.
- **Navigációs struktúra**: Az alkalmazás fő navigációs felépítése, amely tartalmazza az összes képernyőt.

8. Stílusok (StyleSheet)

```
const styles = StyleSheet.create({...});
```

- **Stílusok definiálása**: A React Native StyleSheet API-jával hozza létre a különböző UI elemek stílusát, például gombokat, szövegmezőket, és a konténerek elrendezését.