Politechnika Białostocka

Dokumentacja Projektu

Temat: Aplikacja mobilna “Travel Helper”

Systemy Mobilne

Wykonujący projekt:
Gabriel Czajkowski

Studia dzienne
Kierunek: Informatyka
Semestr: V
Grupa zajęciowa: PS1

Prowadzący:
dr inż. Adam Borowicz

1. OPIS APLIKACJI

Głównym założeniem oraz przeznaczeniem aplikacji “Travel Helper” jest pomoc w podróżowaniu. Aplikacja pozwala między innymi na publikowanie ogłoszeń z zapotrzebowaniem na transport. Użytkownik uzupełnia informację na temat lokalizacji początkowej, docelowej oraz czasu w którym chciałby rozpocząć podróż. Ponadto osoby korzystające z aplikacji mają możliwość sprawdzenia swojego aktualnego położenia. Oprócz samego miasta podane są też jego podstawowe informacje takie jak populacja, województwo i kraj w którym się ono znajduje. Użytkownik ma również możliwość szybkiego przejścia do aplikacji YouTube w celu urozmaicenia ciągnącej się podróży. Aby skorzystać z “Travel Helper” należy być zarejestrowanym.

2. OPIS PODSTAWOWYCH FUNKCJONALNOŚCI

- Rejestracja,
- Logowanie,
- W przypadku zapomnienia hasła, możliwość jego zmiany poprzez link wysłany na podany podczas rejestracji adres e-mail,
- W widoku profilu możliwość zmiany:
    - Imienia i nazwiska,
    - Miejsca zamieszkania (miasta),
    - Numeru telefonu.
- Możliwość ustawienia lub zmiany zdjęcia profilowego. Może być ono wybrane z galerii lub innych źródeł bądź też wykonane aparatem fotograficznym oraz odpowiednio przycięte.
- Wylogowanie się z konta (domyślanie aplikacja zapamiętuje zalogowanego użytkownika i nie jest konieczne logowanie się przy każdym włączeniu aplikacji),
- Usunięcie konta (wszystkie dane powiązane z kontem zostaną usunięte z bazy danych),
- Możliwość sprawdzenia swojej aktualnej lokalizacji zawierającej takie informacje jak:
    - Miasto w którym się znajdujemy,
    - Populację miasta,
    - Województwo,
    - Kraj.
- Dane o lokalizacji pobierane są z odpowiedniego web serwisu (API): https://rapidapi.com/Spott/api/spott?endpoint=apiendpoint_e9adda0f-11fc-4aa4-bc16-a0007d938ab1,
- Możliwość szybkiego przejścia do aplikacji YouTube,
- Wyświetlenie wszystkich ogłoszeń o chęci podróży na przewijanej liście RecyclerView,
- Każde ogłoszenie zawiera informacje o zamieszczającym takie jak:
    - Zdjęcie profilowe,
    - Imię i nazwisko,
    - Numer telefonu,
    - Lokalizacja początkowa (miasto oraz ulica),
    - Lokalizacja docelowa (miasto oraz ulica),
    - Data oraz godzina chęci rozpoczęcia podróży.
- Możliwość dodania własnego ogłoszenia,
- Podczas zamieszczania swojego ogłoszenia konieczność podania takich informacji jak:
    - Lokalizacja początkowa (miasto oraz ulica),
    - Lokalizacja docelowa (miasto oraz ulica),
    - Data (dzień, miesiąc, rok),
    - Godzina (godzina, minuta).

- W każdym polu do wprowadzenia danych zastosowano walidację. Przykładowe z nich:
    - Pola nie mogą być puste,
    - Hasło musi składać się z co najmniej 4 znaków (istnieje możliwość jego odsłonięcia oraz ponownego zasłonienia),
    - E-mail musi mieć odpowiednią składnię,
    - Rok chęci rozpoczęcia podróży nie może być mniejszy od aktualnego,
    - Numer telefonu musi składać się dokładnie z 9 cyfr.
- Aplikacja może być wyświetlana w jednym z trzech języków:
    - Polski,
    - Angielski,
    - Rosyjski.
- Wszystkie informacje na temat użytkownika i ogłoszeń zapisywane są w bazie danych Firebase firmy Google,
- Do poruszania się po aplikacji (przełączania między fragmentami) służy pasek nawigacji znajdujący się na dole ekranu.

3. UŻYTE BIBLIOTEKI

- Baza danych Firebase:
    - 'com.google.firebase:firebase-auth:20.0.1',
    - 'com.google.firebase:firebase-firestore:22.0.1',
    - 'com.google.firebase:firebase-storage:19.2.1',
    - 'com.google.firebase:firebase-database:19.6.0'.
- API:
    - 'com.squareup.retrofit2:converter-gson:2.6.2',
    - 'com.squareup.okhttp3:logging-interceptor:4.9.0'.
- Lokalizacja:
    - 'com.google.android.gms:play-services-location:17.0.0'
- Przycinanie i ustawianie zdjęcia profilowego:
    - 'com.theartofdev.edmodo:android-image-cropper:2.8.0',
    - 'com.github.bumptech.glide:glide:4.11.0',
    - 'com.github.bumptech.glide:compiler:4.11.0'.
- Wyświetlanie okrągłego zdjęcia:
    - 'de.hdodenhof:circleimageview:3.1.0'

4. NAJWAŻNIEJSZE FRAGMENTY APLIKACJI

Pierwszym ważnym, o ile nie najważniejszym elementem (bez niego nie możliwe byłoby korzystanie z aplikacji) jest rejestracja. Poniższa metoda zapisuje wpisane przez użytkownika dane rejestracyjne oraz aktualną datę do bazy danych.

```
private void SaveUserData(String name, String email, String phoneNumber, String city) {
    @SuppressLint("SimpleDateFormat") SimpleDateFormat simpleDateFormat = new SimpleDateFormat( pattern: "dd-MM-yyyy");
    String currentTime = simpleDateFormat.format(new Date());

    //https://firebase.google.com/docs/firestore/manage-data/add-data
    String userId = firebaseAuth.getCurrentUser().getUid();
    documentReference = firebaseFirestore.collection( collectionPath: "Users").document(userId);

    Map<String, Object> userMap = new HashMap<>();
    userMap.put("Username", name);
    userMap.put("E-mail", email);
    userMap.put("City", city);
    userMap.put("Phone number", phoneNumber);
    userMap.put("Date of account creation", currentTime);

    documentReference.set(userMap).addOnSuccessListener(aVoid -> {
        loadingDialog.DismissDialog();
        Toast.makeText( context: NewUserInformationActivity.this, "User successfully created.", Toast.LENGTH_SHORT).show();
        startActivity(new Intent(getApplicationContext(), MainActivity.class));
        finish();
    }).addOnFailureListener(e -> Toast.makeText(getApplicationContext(), "The user data could not be saved.", Toast.LENGTH_LONG).show());
}
```

Obraz 1. Zapis danych użytkownika do bazy danych.

Logowanie do aplikacji - polega ono na pobraniu wpisanych przed użytkownika danych, ich walidacji oraz sprawdzenia, czy podane dane zgadzają się z tymi w bazie danych.

```
signInButton.setOnClickListener(v -> {
    String email = mEmail.getEditText().getText().toString().trim();
    String password = mPassword.getEditText().getText().toString().trim();

    if (!ValidateEmail() | !ValidatePassword()) {
        if (!ValidateEmail()) mEmail.requestFocus();
        else mPassword.requestFocus();
        return;
    }

    loadingDialog = new LoadingDialog( myActivity: LoginActivity.this,  cancelable: false);
    loadingDialog.StartLoadingDialog();

    firebaseAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            loadingDialog.DismissDialog();
            Toast.makeText( context: LoginActivity.this, "Successfully logged in", Toast.LENGTH_SHORT).show();
            startActivity(new Intent(getApplicationContext(), MainActivity.class));
            finish();
        } else {
            loadingDialog.DismissDialog();
            Toast.makeText( context: LoginActivity.this, "Login error", Toast.LENGTH_LONG).show();
        }
    }).addOnFailureListener(e -> {
        signInError.setVisibility(View.VISIBLE);
        loadingDialog.DismissDialog();
    });
});
```

Obraz 2. Logowanie do aplikacji.

Resetowanie hasła – sprawdzane jest, czy istnieje w bazie danych użytkownik o podanym adresie e-mail. Jeżeli tak, wysyłany jest link do zmiany hasła na wpisany w polu tekstowym adres e-mail. W przeciwnym wypadku zostanie zwrócona informacja, że w bazie nie ma zarejestrowanego użytkownika o wprowadzonym adresie e-mail.

```
resetButton.setOnClickListener(v -> {
    String email = mEmail.getEditText().getText().toString().trim();

    if (!ValidateEmail()) return;

    //Checking whether the e-mail address provided exists
    //https://firebase.google.com/docs/auth/web/manage-users
    firebaseAuth.fetchSignInMethodsForEmail(email).addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            List<String> methods = Objects.requireNonNull(task.getResult()).getSignInMethods();
            if (methods.isEmpty()) {
                mEmail.setError("There is no user with a given email address.");
                mEmail.requestFocus();
            } else {
                alertDialog.dismiss();
                firebaseAuth.sendPasswordResetEmail(email).addOnSuccessListener(aVoid -> {
                    Toast.makeText(activity.getApplicationContext(), "A link to change your password has been sent to your e-...", Toast.LENGTH_LONG).show();
                }).addOnFailureListener(e -> {
                    Toast.makeText(activity.getApplicationContext(), "Something has gone wrong. Try again later.", Toast.LENGTH_LONG).show();
                });
            }
        }
    });
});
```

Obraz 3. Resetowanie hasła.

Zmiana danych takich jak imię i nazwisko, miejsce zamieszkania (miasto) i numer telefonu. Polega ona na wywołaniu odpowiednich metod odpowiadających za walidację i aktualizację danych.

```
saveButton.setOnClickListener(v -> {

    if (!ValidateCity() | !ValidatePhoneNumber()) {
        if (!ValidateCity()) city.requestFocus();
        else mPhoneNumber.requestFocus();
        return;
    }

    user.setUserName(userName.getText().toString());
    user.setCity(city.getEditText().getText().toString());
    user.setPhoneNumber(mPhoneNumber.getEditText().getText().toString());
    UpdateUserDataFirebase();
});
```

Obraz 4. Zmiana danych.

```
public void UpdateUserDataFirebase() {

    DocumentReference documentReference = FirebaseFirestore.getInstance().collection( collectionPath: "Users").document(user.getId());

    Map<String, Object> userMap = new HashMap<>();

    userMap.put("Username", user.getUserName());
    userMap.put("City", user.getCity());
    userMap.put("Phone number", user.getPhoneNumber());

    documentReference.update(userMap).addOnCompleteListener(task -> {
        Toast.makeText(getContext(), "Data has been updated", Toast.LENGTH_SHORT).show();
    }).addOnFailureListener(e -> Toast.makeText(getContext(), "Error during data saving", Toast.LENGTH_LONG).show());
}
```

Obraz 5. Zapis zmienionych danych do bazy danych.

Ustawienie lub zmiana zdjęcia profilowego.

```
profileImage.setOnClickListener(view1 -> CropImage.activity()
        .setGuidelines(CropImageView.Guidelines.ON).setCropShape(CropImageView.CropShape.OVAL)
        .start(getActivity()));
```

Obraz 6. Ustawienie lub zmiana zdjęcia profilowego.

```
@Override
public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == CropImage.CROP_IMAGE_ACTIVITY_REQUEST_CODE) {
        CropImage.ActivityResult result = CropImage.getActivityResult(data);
        if (resultCode == RESULT_OK) {
            resultUri = result.getUri();
            dataLoadedFlag = true;
            SetProfileImage(resultUri);
            user.setProfileImage(resultUri);
            LoadUserProfileImageToFirebase();
        } else if (resultCode == CropImage.CROP_IMAGE_ACTIVITY_RESULT_ERROR_CODE) {
            Toast.makeText(getContext(), "Error", Toast.LENGTH_LONG).show();
        }
    }
}

private void SetProfileImage(Uri imageUri) {
    if (getActivity() == null) return;
    Glide.with(getActivity()).load(imageUri).placeholder(R.drawable.ic_account).error(R.drawable.ic_account).into(profileImage);
}

private void LoadUserProfileImageToFirebase() {

    fileRef = firebaseStorage.getReference().child("Users/" + firebaseAuth.getUid() + "/User photo");
    fileRef.putFile(user.getProfileImage()).addOnSuccessListener(taskSnapshot -> {
        Toast.makeText(myContext, "The profile photo has been changed", Toast.LENGTH_SHORT).show();
    }).addOnFailureListener(e -> {
        Toast.makeText(myContext, "Error during image setting", Toast.LENGTH_LONG).show();
    });
}
```

Obraz 7. Zmiana oraz zapis zdjęcia profilowego do bazy danych.

Kolejnym ciekawym i wymagającym elementem było usuwanie konta, ponieważ trzeba było usunąć również wszystkie powiązane z danym kontem dane znajdujące się w bazie danych. Realizowane jest to poprzez wywołanie odpowiedniego dialogu z zapytaniem o pewności chęci usunięcia konta. W momencie potwierdzenia, usuwane są wszystkie powiązane z danym kontem dane.

```
} else if (id == R.id.user_menu_thrash_can) {
    deleteAccountDialog = new DeleteAccountDialog(myActivity);
    deleteAccountDialog.StartDeleteAccountDialog();
    return true;
}
```

Obraz 8. Wywołanie dialogu usuwania konta.

```
FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
String userId = user.getUid();
DocumentReference usersReference = FirebaseFirestore.getInstance().collection( collectionPath: "Users").document(userId);
CollectionReference needRideReference = FirebaseFirestore.getInstance().collection( collectionPath: "Need ride");
StorageReference profilePhoto = FirebaseStorage.getInstance().getReference().child("Users/" + userId + "/User photo");

yesButton.setOnClickListener(v -> {
    usersReference.delete().addOnCompleteListener(aVoid -> Toast.makeText(myContext,  text: "User deleted", Toast.LENGTH_LONG).show());
    needRideReference.get().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            for (QueryDocumentSnapshot documentSnapshot : task.getResult()) {
                String documentId = documentSnapshot.getId();
                if (documentSnapshot.getString( field: "User ID").equals(userId)) {
                    DocumentReference documentReference2 = FirebaseFirestore.getInstance().collection( collectionPath: "Need ride").document(documentId);
                    documentReference2.delete();
                }
            }
        }
    });
    profilePhoto.delete();
    user.delete();
    FirebaseFirestore.getInstance().clearPersistence();

    alertDialog.dismiss();
    myActivity.startActivity(new Intent(myContext, LoginActivity.class));
    myActivity.finish();
});

noButton.setOnClickListener(v -> {
    alertDialog.dismiss();
});
```

Obraz 9. Usunięcie konta z bazy danych.

Widok ładowania strony – zastosowano go wszędzie tam, gdzie trzeba było chociaż przez chwilę poczekać na załadowanie się danych. Dzięki temu użytkownik wie, że aplikacja nie zawiesiła się.

```
public class LoadingDialog {

    //region VARIABLES
    //OTHERS
    private final Activity activity;
    private AlertDialog alertDialog;
    private Boolean cancelable;
    //endregion

    public LoadingDialog(Activity myActivity, Boolean cancelable) {
        activity = myActivity;
        this.cancelable = cancelable;
    }

    public void StartLoadingDialog() {
        AlertDialog.Builder builder = new AlertDialog.Builder(activity);
        LayoutInflater inflater = activity.getLayoutInflater();

        builder.setView(inflater.inflate(R.layout.dialog_loading,  root: null));
        builder.setCancelable(cancelable);

        alertDialog = builder.create();
        alertDialog.getWindow().setBackgroundDrawableResource(android.R.color.transparent);
        alertDialog.show();
    }

    public void DismissDialog() { alertDialog.dismiss(); }
}
```

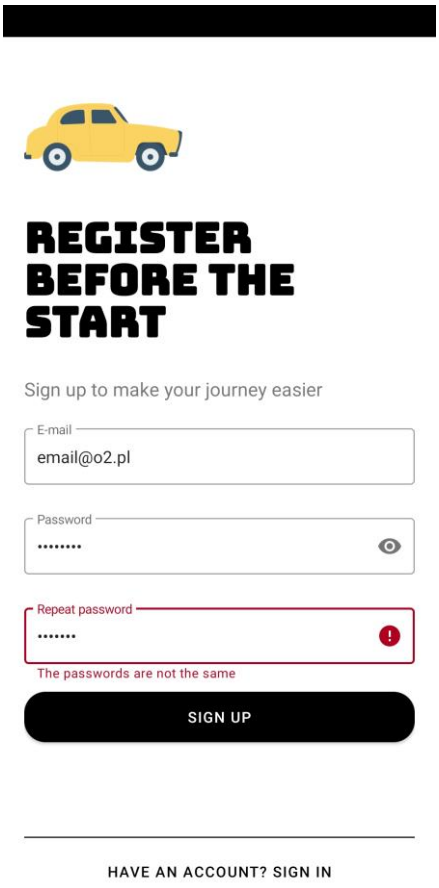
Obraz 10. Loading Dialog.

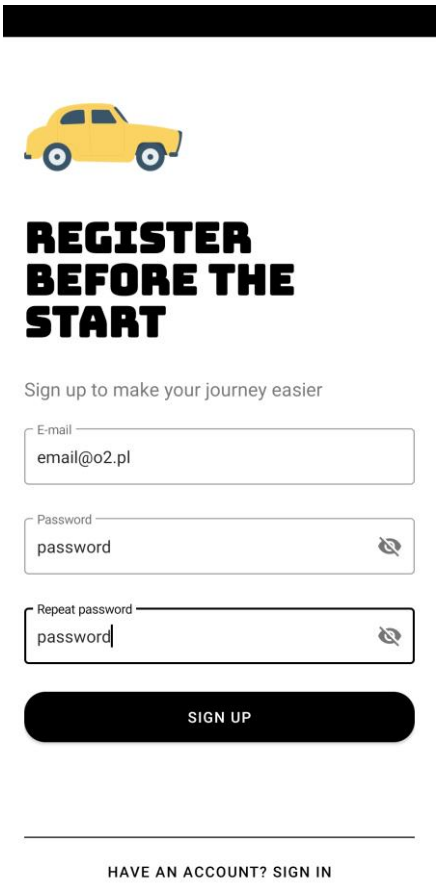
Otwieranie aplikacji YouTube – po naciśnięciu ikony zostanie uruchomiona aplikacja YouTube.

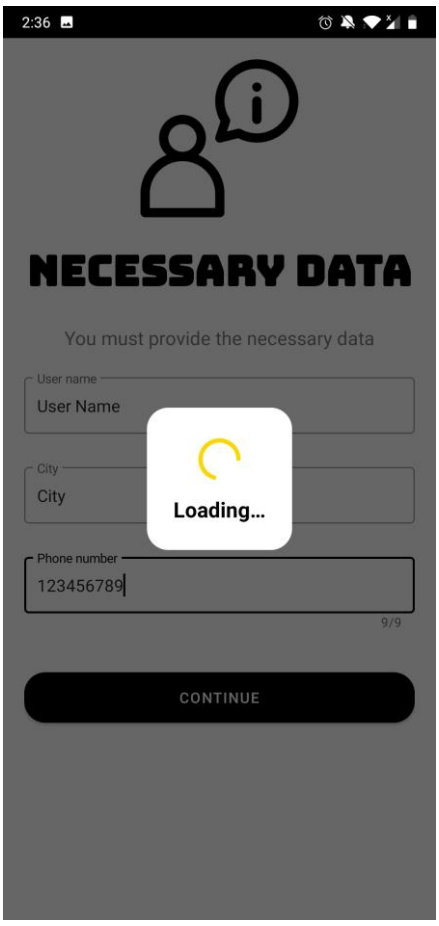
```
YT.setOnClickListener(v -> {
    Intent launchIntent = myContext.getPackageManager().getLaunchIntentForPackage( packageName: "com.google.android.youtube");
    if (launchIntent != null) startActivity(launchIntent);
    else
        Toast.makeText(myActivity, "Error", Toast.LENGTH_LONG).show();
});
```

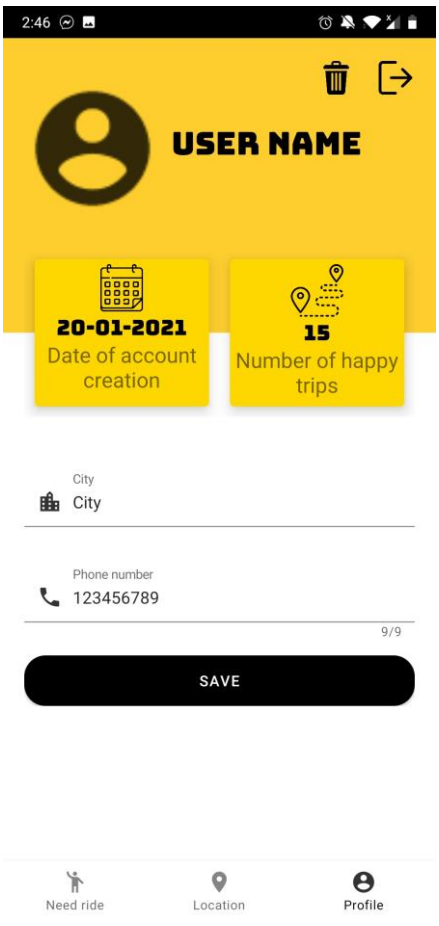
Obraz 11. Włączanie aplikacji YouTube.

5. ZRZUTY EKRANU



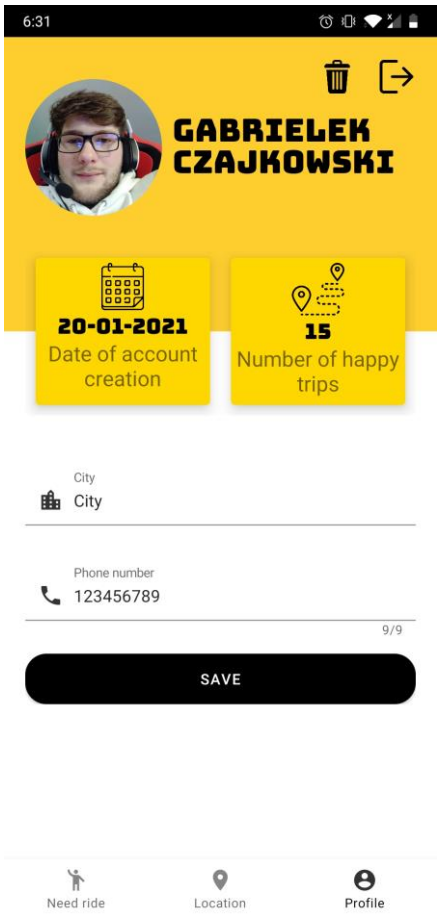


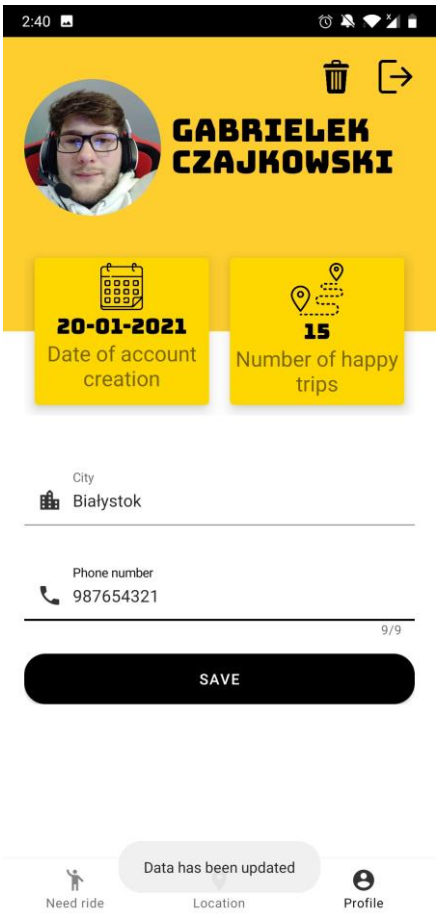


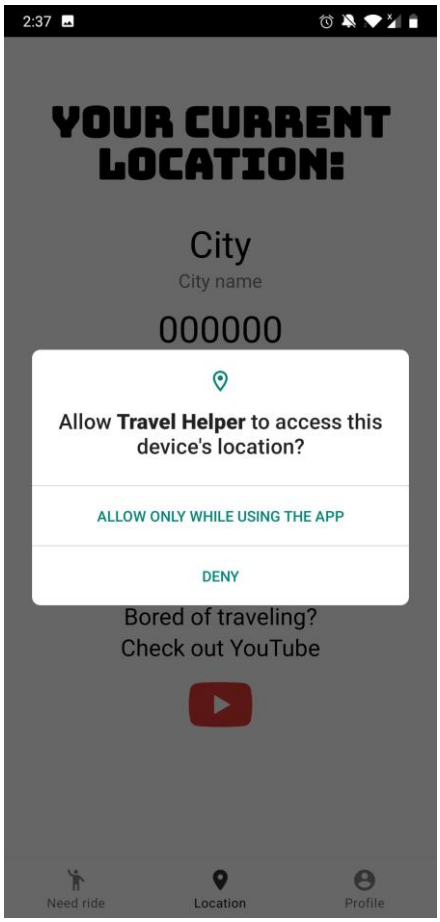


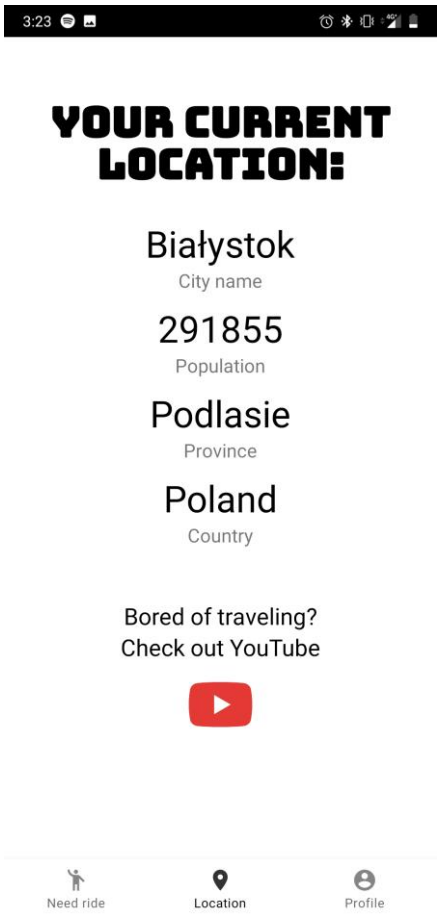

2:43
CROP
SELECT SOURCE
Camera
Files
Bitmoji

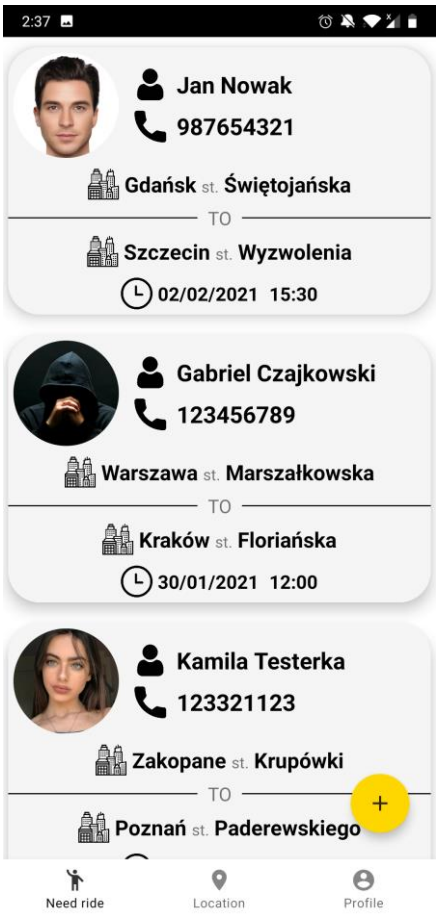

2:38
CROP

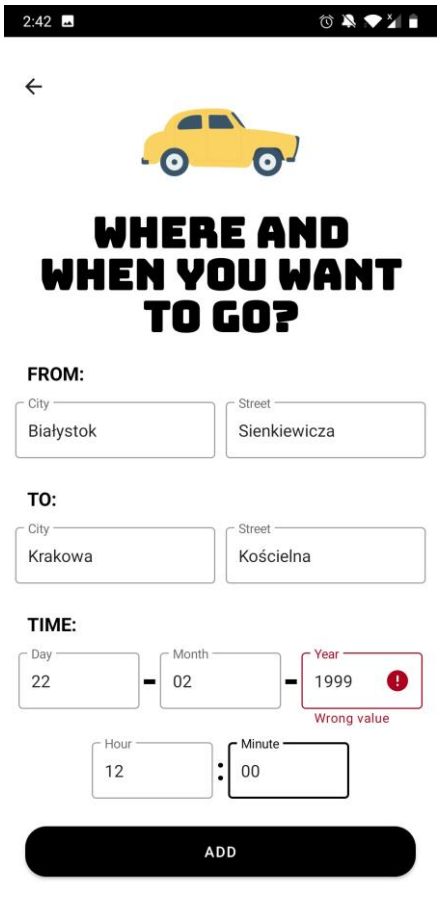

6:31
GABRIELEK CZAJKOWSKI
20-01-2021
Date of account creation
15
Number of happy trips
City
City
Phone number
123456789
9/9
SAVE
Need ride
Location
Profile

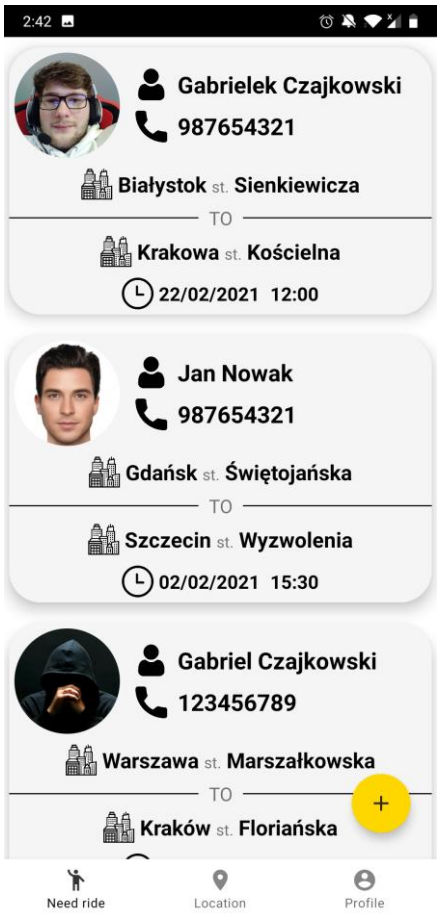

2:40
GABRIELEK CZAJKOWSKI
20-01-2021
Date of account creation
15
Number of happy trips
City
Białystok
Phone number
987654321
9/9
SAVE
Data has been updated
Need ride
Location
Profile

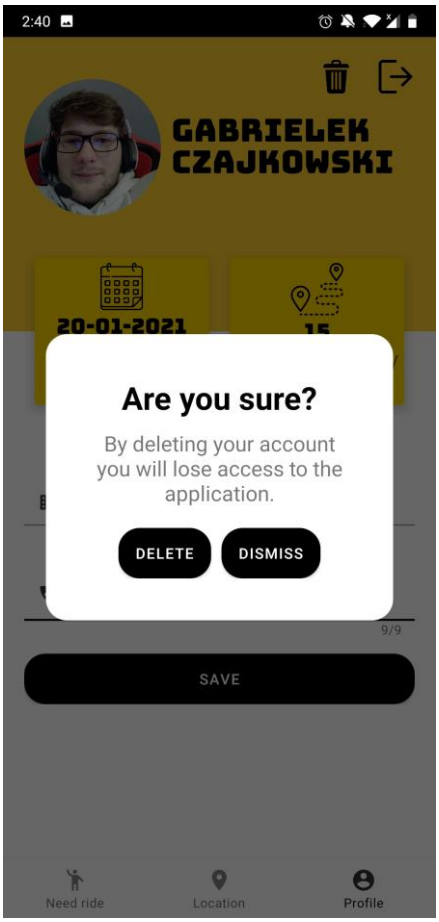

2:37
YOUR CURRENT LOCATION:
City
City name
000000
Allow Travel Helper to access this device's location?
ALLOW ONLY WHILE USING THE APP
DENY
Bored of traveling?
Check out YouTube
Need ride
Location
Profile

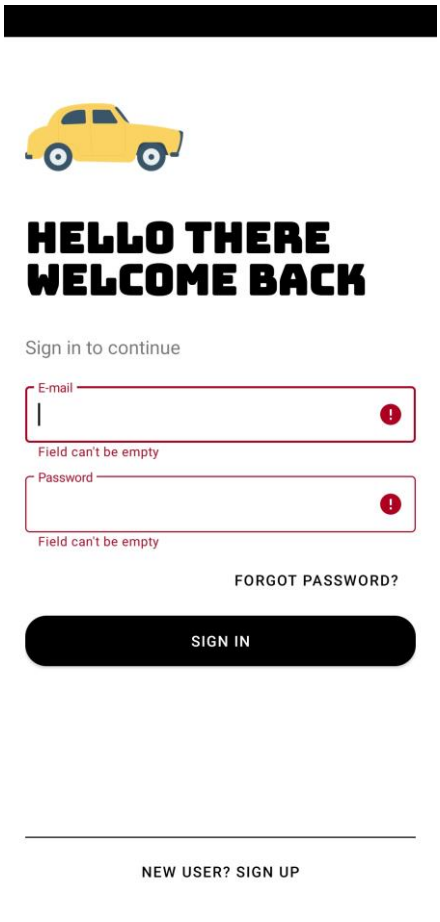

3:23
YOUR CURRENT LOCATION:
Białystok
City name
291855
Population
Podlasie
Province
Poland
Country
Bored of traveling?
Check out YouTube
Need ride
Location
Profile

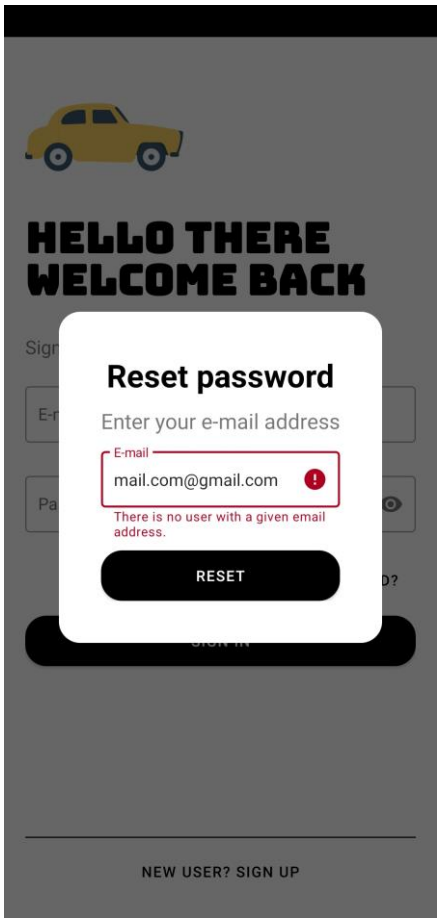

2:37
Jan Nowak
987654321
Gdańsk st. Świętojańska
TO
Szczecin st. Wyzwolenia
02/02/2021 15:30
Gabriel Czajkowski
123456789
Warszawa st. Marszałkowska
TO
Kraków st. Floriańska
30/01/2021 12:00
Kamila Testerka
123321123
Zakopane st. Krupówki
TO
Poznań st. Paderewskiego
Need ride
Location
Profile


2:42
WHERE AND WHEN YOU WANT TO GO?
FROM:
City
Białystok
Street
Sienkiewicza
TO:
City
Krakowa
Street
Kościelna
TIME:
Day
22
Month
02
Year
1999
Wrong value
Hour
12
Minute
00
ADD


2:42
Gabrielek Czajkowski
987654321
Białystok st. Sienkiewicza
TO
Krakowa st. Kościelna
22/02/2021 12:00
Jan Nowak
987654321
Gdańsk st. Świętojańska
TO
Szczecin st. Wyzwolenia
02/02/2021 15:30
Gabriel Czajkowski
123456789
Warszawa st. Marszałkowska
TO
Kraków st. Floriańska
Need ride
Location
Profile


2:40
GABRIELEK CZAJKOWSKI
20-01-2021
Are you sure?
By deleting your account you will lose access to the application.
DELETE
DISMISS
SAVE
Need ride
Location
Profile


HELLO THERE WELCOME BACK
Sign in to continue
E-mail
Field can't be empty
Password
Field can't be empty
FORGOT PASSWORD?
SIGN IN
NEW USER? SIGN UP


HELLO THERE WELCOME BACK
Reset password
Enter your e-mail address
E-mail
mail.com@gmail.com
There is no user with a given email address.
RESET
NEW USER? SIGN UP

6. WNIOSKI

Aplikacja “Travel Helper” spełnia wszystkie wymagania i założenia postawione przed jej stworzeniem. Jej użytkowanie jest atrakcyjne i intuicyjne. Posiada wiele widoków oraz wykorzystuje zaawansowane elementy układu interfejsu użytkownika takie jak: fragmenty, CustomView, listę typu RecyclerView z własnym adapterem oraz holderem, CardView oraz inne. Zaimplementowano w niej umiędzynarodowienie (może być wyświetlana w trzech językach: Polskim, Angielskim i Rosyjskim), zapewniono również działanie na urządzeniach o różnych rozdzielczościach oraz wydzielono stałe do odpowiednich plików z zasobami. Podczas pisania projektu zastosowano dobre praktyki programistyczne opisane w Material Design. Ponadto “Travel Helper” korzysta z zasobów sprzętowych takich jak lokalizacja. Jest on również w stanie uruchomić inne aplikacje takie jak aparat fotograficzny (lub inne źródła obrazów takie jak galeria, dysk itp.) lub YouTube. W celu wyświetlenia informacji o lokalizacji skorzystano z web serwisu. Do zapisywania niezbędnych danych wykorzystano bazę danych Firebase firmy Google. Podczas pisania aplikacji nie napotkano znaczących problemów, których nie udało się rozwiązać. Projekt uświadamia, jak ważne w dzisiejszych czasach są aplikacje mobilne, w których prawie każdy posiada smartfon lub inne urządzenie mobilne. Program jest w pełni sprawny i nie posiada błędów. Wszystko działa tak jak powinno dzięki poświęceniu dużej ilości czasu i zaangażowania. Jestem zadowolony z efektu końcowego mojej pracy.