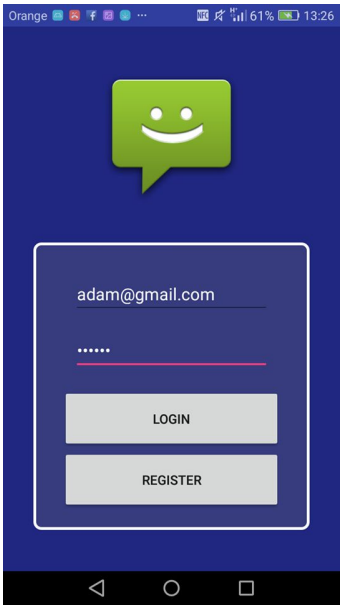
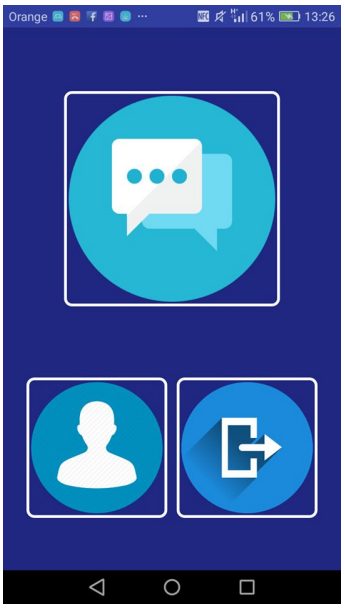
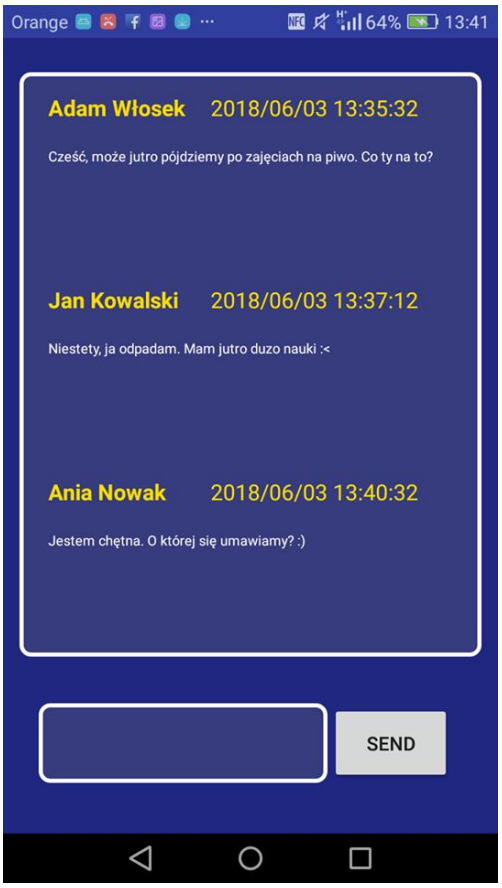
Chat App

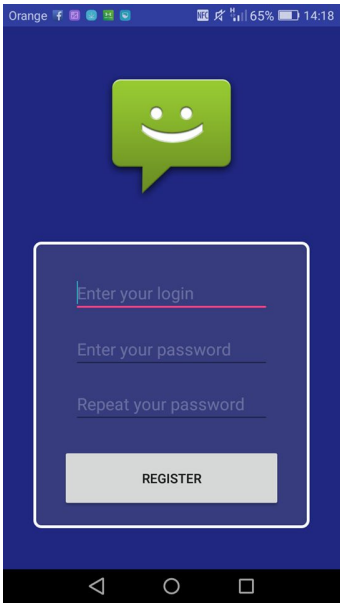
Adam Włosek

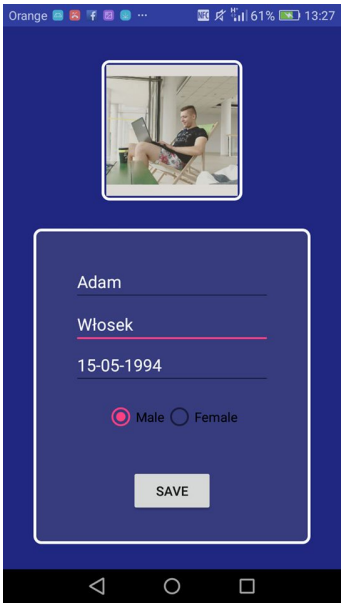
1. Prezentacja aplikacji











2. Wstęp Teoretyczny

Chat App jest aplikacją stworzoną do komunikacji pomiędzy zarejestrowanymi w systemie użytkownikami. Każdy użytkownik posiada osobiste konto zawierające podstawowe informacje o sobie (Imię, Nazwisko,Data urodzenia, płeć). Po uzupełnieniu profilu, można dołączyć do globalnego czatu i rozpocząć rozmowę.

3. Dokumentacja projektu

Projekt opiera się w dużej mierze o narzędzie Firebase, udostępniające bazę danych w formacie JSON, oraz prosty storage zdjęć.

Opis Klas

Lp.	Nazwa	Typ	Pakiet	Opis
1	ChatPanel	Klasa	chat	Odpowiedzialna za wygląd i funkcjonalność czatu
2	Message	Klasa	chat	Reprezentuje obiekt pojedynczej wiadomości
3	MessageAdapter	Klasa	chat	Adapter RecyclerView wiadomości
4	LoginPanel	Klasa	init	Odpowiedzialna za wygląd i funkcjonalność panelu logowania
5	RegisterPanel	Klasa	init	Odpowiedzialna za wygląd i funkcjonalność panelu rejestracji
6	User	Klasa	user	Reprezentuje obiekt pojedynczego użytkownika
7	UserInformation Panel	Klasa	user	Odpowiedzialna za wygląd i funkcjonalność panelu informacyjnego użytkownika
8	MainPanel	Klasa	user	Odpowiedzialna za wygląd i funkcjonalność panelu głównego po zalogowaniu

4. Wybrane fragmenty implementacji

4.1.Autoryzacja

```
private FirebaseAuth mAuth;
private FirebaseAuth.AuthStateListener mAuthListener;
```

Tworzone są dwa obiekty: mAuth i mAuthListener, odpowiedzialne za autoryzację użytkownika próbującego zalogować się do systemu.

```
@Override
protected void onStart() {
    super.onStart();
    mAuth.addAuthStateListener(mAuthListener);
}
```

W metodzie onStart() wywołujemy metode addAuthStateListener() nasłuchując czy login i hasło są poprawne.

```
public void login(View view) {
    String email = this.etEmail.getText().toString().trim();
    String password = this.etPassword.getText().toString().trim();

    if(email.isEmpty()){
        this.etEmail.setError("E-mail is required");
        this.etEmail.requestFocus();
    }
    if(password.isEmpty()){
        this.etPassword.setError("Password is required");
        this.etPassword.requestFocus();
    }

    mAuth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener( activity: this, (task) -> {
                    if (!task.isSuccessful()) {
                        FirebaseUser user = mAuth.getCurrentUser();
                        Toast.makeText( context: LoginPanel.this, text: "Authentication failed.",
                                Toast.LENGTH_SHORT).show();
                    }
                    else {
                        Toast.makeText( context: LoginPanel.this, text: "Authentication succesed.",
                                Toast.LENGTH_SHORT).show();
                        finish();
                        startActivity(new Intent(getApplicationContext(), MainPanel.class));
                    }
            });
}
```

W momencie kliknięcia w panelu logowania przycisku Login

wywołana zostaje metoda login(), następnie metoda weryfikacji logowania (signInWithEmailAndPassword(email,password)). Jeśli task zakończy się sukcesem, użytkownik zostaje przekierowany do panelu klasy MainPanel.class. Jeśli nie, wyświetlany jest Toast z komunikatem o błędzie logowania.

4.2 Dodawanie avatara

```
ImageButton avatarButton;
avatarButton = (ImageButton) findViewById(R.id.user_info);
```

Tworzony jest obiekt klasy ImageButton reprezentujący avatar użytkownika, a następnie przypisywany do określonego identyfikatora.

```
avatarButton.setOnClickListener((v) -> {
        Intent intent = new Intent(Intent.ACTION_PICK);
        intent.setType("image/*");
        startActivityForResult(intent,GALLERY_INTENT);
});
```

W metodzie OnCreate() tworzony jest listener po którym kliknięciu w przycisk avatarButton, tworzone jest nowe zdarzenie wybierania zdjęcia z galerii ,a następnie uruchamiana jest metoda startActivityForResult().

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

    if(requestCode == GALLERY_INTENT && resultCode == RESULT_OK){
        Uri uri = data.getData();

        StorageReference filepath = mStorage.child("ProfilePhoto").child(FirebaseAuth.getInstance().getCurrentUser().getUid());

        filepath.putFile(uri).addOnSuccessListener((OnSuccessListener) (taskSnapshot) -> {
                Toast.makeText(getApplicationContext(), text: "Upload Done",Toast.LENGTH_SHORT).show();

                Uri downloadUri = taskSnapshot.getDownloadUrl();
                Log.i( tag: "downloadURI", msg: ""+ downloadUri);

                Picasso.get().load(downloadUri).fit().centerCrop().into(avatarButton);


        });

    }
}
```

Do obiektu uri przypisywana jest referencja zdjęcia w pamięci. Następnie tworzona jest referencja StorageReference filepath, przechowująca dokładną ścieżkę miejsca przechowywania zdjęcia w chmurze. Wywołując metodęfilepath.putFile(uri). addOnSuccesListener()
następuje próba dodania zdjęcia. W chwili powodzenia wyświetlany jest Toast o poprawnym dodaniu pliku. Ostatnia linijka kodu to obiekt klasy Picasso - narzędzia wykorzystanego do załadowania z URL zdjęcia avatara do obiektu avatarButton.

5. **Podsumowanie**
   Motywacją do stworzenia aplikacji Chat App jest chęć poznania działania mechanizmów komunikacji pomiędzy użytkownikami w trybie online, które są podstawą aplikacji społecznościowych.

   Podczas pisania aplikacji, wystąpiło wiele problemów wynikających z nieznajomości narzędzia Firebase. Po zapoznaniu się z dokumentacją, pobieranie,wysyłanie oraz zapisywanie danych i zdjęć do bazy stało się proste.

   Ponadto nauczyłem się tworzyć bazę kont użytkowników przechowywanych przez Firebase, tak by każdy użytkownik miał dostęp do swoich danych.