


Politechnika Bydgoska im. Jana i Jędrzeja Śniadeckich Wydział Telekomunikacji, Informatyki i Elektrotechniki al. prof. S. Kaliskiego 7, 85-796 Bydgoszcz				 <b>POLITECHNIKA BYDGOSKA</b> Wydział Telekomunikacji, Informatyki i Elektrotechniki	
Przedmiot	Programowanie urządzeń mobilnych			Kierunek/Tryb	IS/ST
Nr laboratorium	3	Data wykonania	24.03.2025	Grupa	1
Ocena		Data oddania	30.03.2025	Imię Nazwisko	Cezary Tytko
Nazwa ćwiczenia	Interfejs użytkownika i obsługa zdarzeń				

## 2. Cel ćwiczenia

Celem ćwiczenia jest zapoznanie studentów z tworzeniem prostej gry mobilnej na platformie Android przy użyciu interfejsu użytkownika EditText, Button, TextView, EditText. Studenci mają nauczyć się:

1. Projektować interfejs użytkownika, który obejmuje wprowadzanie danych, przycisk do sprawdzania odpowiedzi oraz wyświetlanie wyników i komunikatów.
2. Programować obsługę zdarzeń związaną z rozgrywką.
3. Wykonywać proste operacje na danych wejściowych.
4. Wyświetlać wynik i komunikaty na ekranie w czytelny sposób.
5. Obsługiwać potencjalne błędy podczas wprowadzania danych.

## 3. Opis projektu

Gra "Wisielec" to aplikacja mobilna na system Android, w której użytkownik ma za zadanie odgadnąć ukryte słowo, zgadując pojedyncze litery. Aplikacja zapewnia interfejs użytkownika umożliwiający wprowadzanie liter, wyświetlanie aktualnego stanu słowa oraz zarządzanie liczbą pozostałych prób. Gra kończy się wygraną, jeśli użytkownik odgadnie wszystkie litery, lub przegraną, jeśli wyczerpie dostępne próby.

Interfejs użytkownika zawiera następujące elementy:

- **TextView:** do wyświetlania aktualnego stanu zgadywanego słowa,
- **TextView:** do wyświetlania liczby pozostałych prób,
- **TextView:** do wyświetlania komunikatów o stanie gry,
- **EditText:** pole do wpisywania liter przez użytkownika,
- **Button:** przycisk do sprawdzania wprowadzonej litery,
- **Button:** przycisk do resetowania gry.

## 4. Implementacja

Kod został napisany w języku Kotlin w środowisku Android Studio.

MainActivity.kt:

```

1. package com.example.pumlab3
2.
3. import android.os.Bundle
4. import android.util.Log
5. import android.view.View
6. import android.widget.*
7. import androidx.appcompat.app.AppCompatActivity
8. import kotlin.random.Random

```

```

9.
10. class MainActivity : AppCompatActivity() {
11.     private lateinit var wordTextView: TextView
12.     private lateinit var triesTextView: TextView
13.     private lateinit var statusTextView: TextView
14.     private lateinit var letterInput: EditText
15.     private lateinit var checkButton: Button
16.     private lateinit var resetButton: Button
17.
18.     private val TAG = "MainActivity"
19.     private var wordToGuess = ""
20.     private var guessedWord = charArrayOf()
21.     private var triesLeft = 6
22.     private val guessedLetters = mutableSetOf<Char>()
23.
24.     override fun onCreate(savedInstanceState: Bundle?) {
25.         super.onCreate(savedInstanceState)
26.         setContentView(R.layout.activity_main)
27.
28.         wordTextView = findViewById(R.id.wordTextView)
29.         triesTextView = findViewById(R.id.triesTextView)
30.         statusTextView = findViewById(R.id.statusTextView)
31.         letterInput = findViewById(R.id.letterInput)
32.         checkButton = findViewById(R.id.checkButton)
33.         resetButton = findViewById(R.id.resetButton)
34.
35.         resetGame()
36.
37.         checkButton.setOnClickListener { checkLetter() }
38.         resetButton.setOnClickListener { resetGame() }
39.     }
40.
41.     private fun resetGame() {
42.         wordToGuess = WORDS[Random.nextInt(WORDS.size)]
43.         guessedWord = CharArray(wordToGuess.length) { '_' }
44.         guessedLetters.clear()
45.         updateGuessedWord(' ')
46.         triesLeft = 6
47.         checkButton.isEnabled = true
48.         statusTextView.text = ""
49.         Log.d(TAG, wordToGuess)
50.         updateWordDisplay()
51.     }
52.
53.     private fun updateWordDisplay() {
54.         val displayedWord = wordToGuess.mapIndexed { index, char ->
55.             if (char.lowercaseChar() in guessedLetters) char else '_'
56.         }.joinToString(" ")
57.
58.         wordTextView.text = displayedWord
59.         triesTextView.text = "Pozostałe próby: $triesLeft"
60.     }
61.
62.     private fun updateGuessedWord(letter: Char) {
63.         if (!guessedLetters.contains(letter)) {
64.             guessedLetters.add(letter)
65.         }
66.         if (wordToGuess.lowercase().contains(letter)) {
67.             for (i in wordToGuess.indices) {
68.                 if (wordToGuess[i] == letter) {
69.                     guessedWord[i] = letter
70.                 }
71.             }
72.         } else {
73.             triesLeft--
74.         }
75.     }
76.
77.     private fun checkLetter() {
78.         val input = letterInput.text.toString()
79.         if (input.isEmpty() || input.length > 1) {
80.             Toast.makeText(this, "Podaj jedną literę!", Toast.LENGTH_SHORT).show()
81.             return
82.         }
83.
84.         val letter = input[0].lowercaseChar()
85.         letterInput.text.clear()
86.

```

```

87.         if (letter in guessedLetters) {
88.             Toast.makeText(this, "Ta litera była już zgadywana!", Toast.LENGTH_SHORT).show()
89.             return
90.         }
91.
92.         updateGuessedWord(letter)
93.         updateWordDisplay()
94.         checkGameStatus()
95.     }
96.
97.     private fun checkGameStatus() {
98.         if (String(guessedWord) == wordToGuess.lowercase()) {
99.             statusTextView.text = "Gratulacje! Odgadłeś słowo!"
100.            checkButton.isEnabled = false
101.        } else if (triesLeft == 0) {
102.            statusTextView.text = "Przegrałeś! Słowo to: $wordToGuess"
103.            checkButton.isEnabled = false
104.        }
105.    }
106.
107.    private val WORDS = listOf(
108.        "komputer", "programowanie", "system", "algorytm", "aplikacja", "internet", "dane", "kod", "sieć",
109.        "web",
110.        "serwer", "framework", "grafika", "programista", "strona", "dysk", "RAM", "CPU", "GPU", "baza
111.        danych",
112.        "linux", "Windows", "macOS", "sztuczna inteligencja", "machine learning", "cyberbezpieczeństwo",
113.        "komponent",
114.        "API", "git", "repository", "debugowanie", "skrypt", "hasło", "router", "firewall", "monitor",
115.        "cyfrowy",
116.        "cryptocurrency", "kodowanie", "architektura", "wersjonowanie", "debugowanie", "interfejs",
117.        "responsywność",
118.        "frontend", "backend", "hosting", "cyberatak", "synchronizacja", "dokumentacja", "proxy", "wirus",
119.        "malware",
120.        "VPN", "sterownik", "intranet", "sieć neuronowa", "framework", "kryptografia", "algorithm",
121.        "program",
122.        "framework", "obiekt", "serwer", "komponent", "aplikacja mobilna", "rozwiązanie", "komunikacja",
123.        "testowanie",
124.        "konfiguracja", "edycja", "plugin", "archiwum", "algorytm", "adres IP", "usługa", "web design",
125.        "klient",
126.        "terminal", "zaszyfrowany", "deklaracja", "monitoring", "renderowanie", "API key", "kompilator",
127.        "hardware",
128.        "serwer plików", "debugowanie", "rozpoznawanie", "zarządzanie danymi", "zintegrowany", "data
129.        mining",
130.        "sieć społecznościowa", "ciasteczko", "wywołanie", "zabezpieczenie", "operating system", "usługa
131.        chmurowa",
132.        "framework", "transmisja", "responsive design", "open-source", "hosting", "platforma", "komunikat",
133.        "serializacja",
134.        "aplikacja webowa", "cache", "testing", "storage", "asynchroniczny", "rozwiązanie", "algorytmiczny",
135.        "rozpoznawanie",
136.        "klucz API", "synchronizacja", "platforma cyfrowa", "zaszyfrowany", "synchronizacja", "open-source",
137.        "profil",
138.        "cache", "komunikacja", "algorithm", "kompilator", "zabezpieczenie"
139.    )
140. }
141.

```

### activity\_main.xml:

```

1.  <?xml version="1.0" encoding="utf-8"?>
2.  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.      android:layout_width="match_parent"
4.      android:layout_height="match_parent"
5.      android:orientation="vertical"
6.      android:padding="16dp"
7.      android:gravity="center">
8.
9.      <TextView
10.         android:id="@+id/wordTextView"
11.         android:layout_width="wrap_content"
12.         android:layout_height="wrap_content"
13.         android:textSize="24sp"
14.         android:textStyle="bold"
15.         android:text=" _ _ _ _ "
16.         android:padding="16dp"/>
17.
18.      <TextView
19.         android:id="@+id/triesTextView"

```

```

20.         android:layout_width="wrap_content"
21.         android:layout_height="wrap_content"
22.         android:text="Pozostałe próby: 6"
23.         android:textSize="18sp"
24.         android:padding="8dp"/>
25.
26.     <EditText
27.         android:id="@+id/letterInput"
28.         android:layout_width="wrap_content"
29.         android:layout_height="wrap_content"
30.         android:hint="Podaj literę"
31.         android:maxLength="1"
32.         android:inputType="text"
33.         android:textSize="18sp"
34.         android:gravity="center"/>
35.
36.     <Button
37.         android:id="@+id/checkButton"
38.         android:layout_width="wrap_content"
39.         android:layout_height="wrap_content"
40.         android:text="Sprawdź literę"
41.         android:padding="8dp"
42.         android:layout_marginTop="10dp"/>
43.
44.         <Button
45.             android:id="@+id/resetButton"
46.             android:layout_width="wrap_content"
47.             android:layout_height="wrap_content"
48.             android:text="Nowa gra"
49.             android:padding="8dp"
50.             android:layout_marginTop="10dp"/>
51.
52.     <TextView
53.         android:id="@+id/statusTextView"
54.         android:layout_width="wrap_content"
55.         android:layout_height="wrap_content"
56.         android:text=""
57.         android:textSize="20sp"
58.         android:textStyle="bold"
59.         android:padding="10dp"/>
60. </LinearLayout>
61.

```

## 5. Funkcje kluczowe

- **Obsługa zgadywania liter** – użytkownik może wpisywać pojedyncze litery i sprawdzać, czy występują w ukrytym słowie.
- **Zarządzanie stanem gry** – aplikacja aktualizuje wyświetlane słowo oraz liczbę pozostałych prób.
- **Wyświetlanie komunikatów** – informowanie użytkownika o poprawnym lub błędnym odgadnięciu litery, wygranej lub przegranej.
- **Resetowanie gry** – użytkownik może rozpocząć nową rozgrywkę.

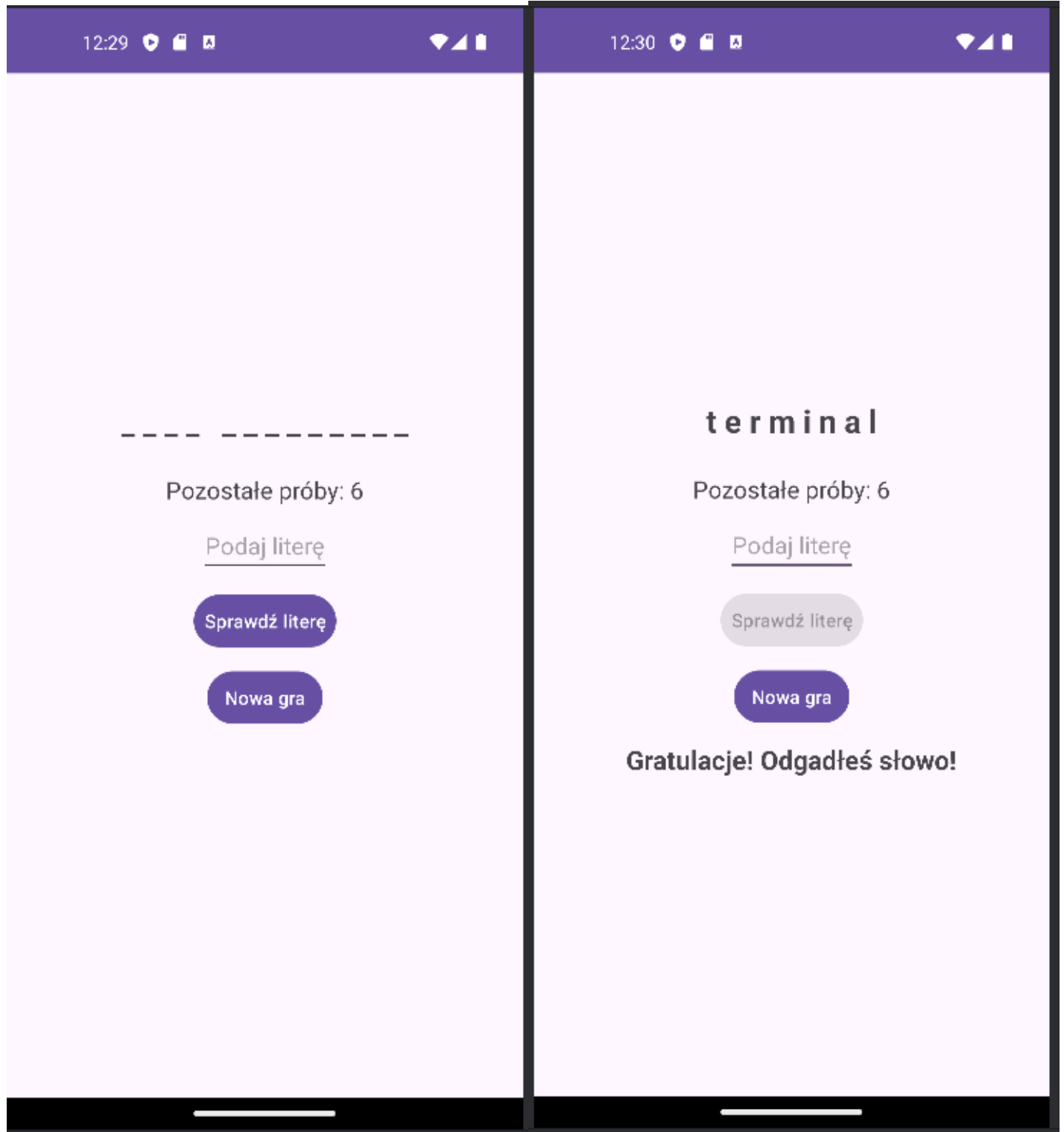
## 6. Testowanie

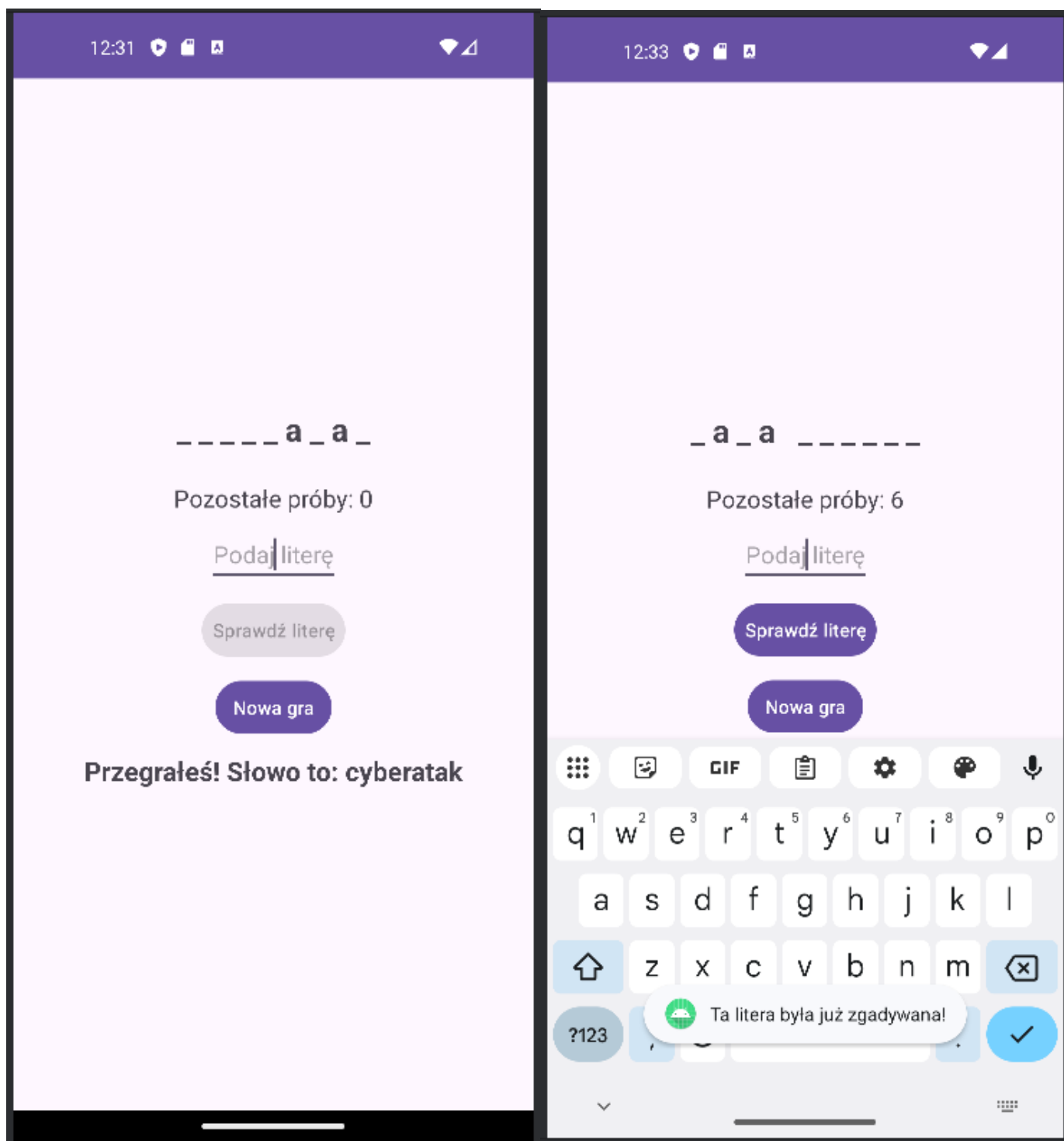
Testowanie gry obejmowało:

- Sprawdzenie poprawności wyświetlania ukrytego słowa i zgadywanych liter,
- Obsługę błędnych wejść (np. wpisanie więcej niż jednej litery),
- Weryfikację poprawnego zakończenia gry po odgadnięciu słowa lub przekroczeniu limitu prób.

Test	Wejście	Oczekiwany wynik
1	Wpisanie poprawnej litery	Odśloniecie litery w słowie
2	Wpisanie błędnej litery	Zmniejszenie liczby prób
3	Wpisanie tej samej litery ponownie	Komunikat o powtórzeniu litery
4	Odgadnięcie całego słowa	Komunikat o wygranej
5	Wyczerpanie prób	Komunikat o przegranej i ujawnienie słowa

## 7. Wyniki





## 8. Podsumowanie

Projekt realizuje klasyczną grę "Wisielec" w aplikacji mobilnej. Implementacja pozwoliła na zdobycie doświadczenia w tworzeniu aplikacji na Androida, obsłudze interakcji użytkownika oraz zarządzaniu stanem gry. Ćwiczenie umożliwiło lepsze zrozumienie obsługi zdarzeń i logiki gry.

## 9. Trudności i błędy

- Nie wystąpiły żadne trudności ani błędy.

## 10. Źródła i odniesienia

- Nie korzystano ze źródeł i odniesień innych niż ta instrukcja.

## 11. Dodatkowe materiały

- Nie korzystano z dodatkowych materiałów.