

UNIwersytet Rzeszowski
Wydział Nauk Ścisłych i Technicznych
Instytut Informatyki



Lukasz Paż
134956
Informatyka

Dokumentacja Aplikacji Steam

Rzeszów 2025r

Spis treści

1	Project assumptions	1
2	Opis założeń projektu	1
2.1	Wymagania funkcjonalne	2
2.2	Wymagania niefunkcjonalne	2
3	Opis struktury projektu	3
3.1	Zarządzanie danymi i baza danych	3
3.1.1	klienci	4
3.1.2	grywbibliotece	4
3.1.3	grywsklepie	4
3.2	Najważniejsze metody	4
4	Harmonogram realizacji projektu	5
5	Prezentacja warstwy użytkowej projektu	6
6	Podsumowanie	11

1 Project assumptions

The goal of the project is to create a desktop application inspired by the Steam platform, allowing users to buy and sell digital games and manage their funds. The project involves the development of a transaction system in which users can freely trade games with one another. The application is programmed in Java using the Swing library for building the graphical user interface and JDBC for database communication. The system requires a constant connection to the database and does not function offline.

The main problem the project aims to address is the lack of flexibility and freedom in trading digital games on popular distribution platforms. On most commercial systems, users are not allowed to resell games or have full control over transactions. The application presented in this project fills that gap by offering a simple yet fully functional game exchange system, where users can create their own sales offers by manually entering information about the game being sold — regardless of their library content.

The purchase process works as follows: a user selects a game listed by another player, makes the payment, and the full amount is transferred to the seller. The system automatically assigns the purchased game to the buyer's library and updates account balances and the list of available offers. A user cannot buy a game they have listed themselves. The project implementation included creating the database, developing the application logic, designing the interface, and handling transaction operations.

2 Opis założeń projektu

Celem projektu jest stworzenie desktopowej aplikacji wzorowanej na platformie Steam, umożliwiającej użytkownikom dokonywanie zakupu i sprzedaży gier cyfrowych, zarządzanie środkami pieniężnymi. Projekt zakłada stworzenie systemu transakcyjnego, w którym

użytkownicy mogą swobodnie handlować grami między sobą. Aplikacja została zaprogramowana w języku Java z wykorzystaniem biblioteki Swing do budowy graficznego interfejsu użytkownika oraz JDBC do obsługi komunikacji z bazą danych. System wymaga stałego połączenia z bazą danych i nie funkcjonuje w trybie offline.

Głównym problemem, który projekt ma rozwiązać, jest brak elastyczności i swobody w handlu grami cyfrowymi na popularnych platformach dystrybucji. Na większości komercyjnych systemów użytkownicy nie mają możliwości odsprzedaży gier ani pełnej kontroli nad transakcjami. Aplikacja przedstawiana w projekcie wypełnia te luki, oferując prosty, ale w pełni funkcjonalny system wymiany gier, w którym użytkownicy mogą wystawiać własne oferty sprzedaży, podając ręcznie dane sprzedawanej gry – niezależnie od zawartości swojej biblioteki. Proces zakupu przebiega w ten sposób, że użytkownik wybiera grę wystawioną przez innego gracza, dokonuje płatności, a pełna kwota zostaje przekazana sprzedawcy. System automatycznie przypisuje zakupioną grę do biblioteki kupującego i aktualizuje stany kont oraz listę dostępnych ofert. Użytkownik nie może kupić gry, którą sam wystawił. Realizacja projektu obejmowała stworzenie bazy danych, implementację logiki aplikacji, zaprojektowanie interfejsu oraz obsługę operacji transakcyjnych.

2.1 Wymagania funkcjonalne

- Użytkownik może zarejestrować nowe konto oraz zalogować się do aplikacji.
- Użytkownik może doładować swoje konto środkami pieniężnymi.
- Użytkownik może ręcznie wystawić nową grę na sprzedaż, podając jej nazwę, gatunek i cenę.
- Użytkownik może przeglądać listę gier dostępnych w sklepie.
- Użytkownik może kupić grę wystawioną przez innego użytkownika (z wyjątkiem własnych ofert).
- Po dokonaniu zakupu gra jest przypisywana do biblioteki kupującego.
- Cała zapłacona kwota trafia bezpośrednio na konto sprzedawcy.
- Użytkownik może przeglądać swoją bibliotekę gier.
- Użytkownik może przeglądać historie doładowań i transakcji.
- Aplikacja obsługuje walidację danych logowania oraz informuje o błędnych próbach.

2.2 Wymagania нефunkcjonalne

- Aplikacja wymaga aktywnego połączenia z bazą danych w celu działania.
- Aplikacja powinna umożliwiać poprawne wykonanie transakcji pomiędzy użytkownikami bez utraty danych.
- Aplikacja powinna uruchamiać się w czasie nie dłuższym niż 3 sekundy na standardowym komputerze użytkownika.
- Interfejs graficzny powinien być przejrzysty, czytelny i responsywny.

- Aplikacja powinna być odporna na próby zakupu własnych ofert sprzedaży.
- Aplikacja powinna umożliwiać jednoczesne działanie wielu użytkowników bez kolizji danych.
- System powinien zapewniać integralność danych, w szczególności dotyczących środków na kontach i przypisań gier.
- Aplikacja powinna być łatwa w utrzymaniu i rozwijaniu, z czytelną strukturą kodu.
- System powinien być skalowalny, z możliwością przyszłego rozszerzenia o dodatkowe funkcje.

3 Opis struktury projektu

Struktura projektu została podzielona na moduły zgodne z zasadą rozdzielania warstw logiki aplikacji, interfejsu użytkownika oraz obsługi danych. Każde okno graficzne aplikacji ma swoją osobną klasę GUI, która dziedziczy po klasie **Frame**. Klasa **Frame** odpowiada za ustawienie wspólnych elementów wyglądu i zachowania wszystkich okien aplikacji (np. rozmiar, kolory, czcionki) poprzez odpowiednia konfigurację w konstruktorze. Przykładowo, okno główne reprezentowane przez klasę **AplikacjaGUI** dziedziczy po **Frame** i ma przypisaną klasę kontrolera **AplikacjaController**.

Analogiczny podział zastosowano dla pozostałych modułów:

- **LoginGUI** ↔ **LoginController**
- **RejestracjaGUI** ↔ **RejestracjaController**
- **SklepGUI** ↔ **SklepController**
- **SprzedajGreGUI** ↔ **SprzedajGreController**
- **DoladujGUI** ↔ **DoladujController**

Komunikaty wyświetlane użytkownikowi są obsługiwane przez klasę **Komunikat**, a ogólne dane dotyczące użytkownika i gier są reprezentowane przez klasy **Uzytkownik** oraz **Gra**.

3.1 Zarządzanie danymi i baza danych

Aplikacja wykorzystuje technologie JDBC do komunikacji z relacyjną bazą danych, zdefiniowaną w klasie **BazaDanych**. Klasa ta odpowiada za nawiązywanie połączenia z bazą oraz wykonywanie zapytań SQL, takich jak logowanie, rejestracja użytkownika, zakup gry, wystawienie gry na sprzedaż czy doładowanie konta. Wszystkie transakcje są przetwarzane bezpośrednio na poziomie bazy danych w czasie rzeczywistym – aplikacja nie działa offline. Tabele bazy danych "AplikacjaSteam2":

3.1.1 klienci

- id (int) (primary key)
- login (text)
- haslo (text)
- imie (double)

3.1.2 grywbibliotece

- id (int) (primary key)
- wlasciciel_id (int)
- nazwa (text)
- gatunek (text)

3.1.3 grywsklepie

- id (int) (primary key)
- sprzedawca_id (int)
- nazwa (text)
- gatunek (text)
- cena (double)

3.2 Najważniejsze metody

- `Zaloguj()` – metoda odpowiedzialna za logowanie użytkownika. Pobiera dane logowania z interfejsu i weryfikuje je w bazie danych. W przypadku poprawnych danych uruchamiane jest główne okno aplikacji, a poprzednie zostaje zamknięte.
- `Zarejestruj()` – metoda służąca do tworzenia nowego konta użytkownika. Sprawdza poprawność danych wejściowych i unikalność loginu, a następnie wstawia nowego użytkownika do bazy danych.
- `CzyIstniejeTakiUzytkownik(String login)` – metoda pomocnicza, która sprawdza, czy dany login istnieje już w bazie danych.
- `Doladuj()` – metoda pozwalająca na doładowanie konta użytkownika o określona kwotę. Zmiana salda zapisywana jest w bazie danych oraz aktualizowana lokalnie.
- `OdswiezListeGier()` – metoda, która pobiera z bazy danych gry należące do zalogowanego użytkownika i aktualizuje model listy wyświetlanej w interfejsie.
- `KupGre()` – metoda umożliwiająca zakup gry ze sklepu. Sprawdza, czy użytkownik ma wystarczającą ilość środków oraz czy nie próbuje kupić własnej gry. Następnie przenosi gre do biblioteki użytkownika, usuwa ją ze sklepu i aktualizuje stan konta kupującego i sprzedającego.

- `SprzedajGre()` – metoda umożliwiająca wystawienie nowej gry na sprzedaż. Użytkownik podaje nazwę, gatunek oraz cenę, które są następnie zapisywane w bazie danych jako nowy wpis w sklepie.

Wykorzystane technologie

- Java 24
- Swing (GUI)
- JDBC (połączenie z bazą danych)
- MySQL 10.4.32-MariaDB

Minimalne wymagania sprzętowe

- System operacyjny: Windows/Linux/macOS
- Procesor: dwurdzeniowy 1.8 GHz lub wyższy
- Pamięć RAM: 4 GB
- Połączenie z bazą danych (lokalne lub zdalne)

Dodatkowe narzędzia

- IntelliJ IDEA (do budowy i edycji projektu)
- phpMyAdmin (do zarządzania bazą danych)
- Git (kontrola wersji, opcjonalnie)

4 Harmonogram realizacji projektu

Poniżej przedstawiono harmonogram realizacji projektu w postaci diagramu Gantta. Diagram ten obrazuje podział prac nad projektem w czasie, z uwzględnieniem głównych etapów takich jak projektowanie, implementacja, testowanie oraz dokumentacja.

Funkcjonalność	Tydzień 1	Tydzień 2	Tydzień 3	Tydzień 4	Tydzień 5	Tydzień 6
1. Logowanie i rejestracja	3 dni					
2. Ekran główny po zalogowaniu (lista gier, stan konta, nick)		4 dni				
3. Funkcja Doładowania konta (okno i logika)			3 dni			
4. Sklep - wyświetlanie listy gier do kupienia i wystawionych na sprzedaż				4 dni		
5. Funkcjonalność kupna i sprzedaży gier (logika + UI)					5 dni	
6. Przycisk "Wróć" i nawigacja między ekranami						3 dni
7. Testy i poprawki końcowe						2 dni

Rysunek 1: Diagram Gantta przedstawiający harmonogram realizacji projektu

Projekt rozpoczęto od zaprojektowania bazy danych oraz struktury klas. Kolejnym krokiem była implementacja poszczególnych modułów aplikacji, takich jak logowanie, rejestracja, interfejs główny, obsługa sklepu, doładowanie konta i zakup gier. W dalszej kolejności przeprowadzono testowanie funkcjonalności, poprawki błędów oraz przygotowanie dokumentacji.

Do zarządzania kodem źródłowym wykorzystano system kontroli wersji **Git**, a repozytorium projektu zostało umieszczone na platformie **GitHub** pod adresem:

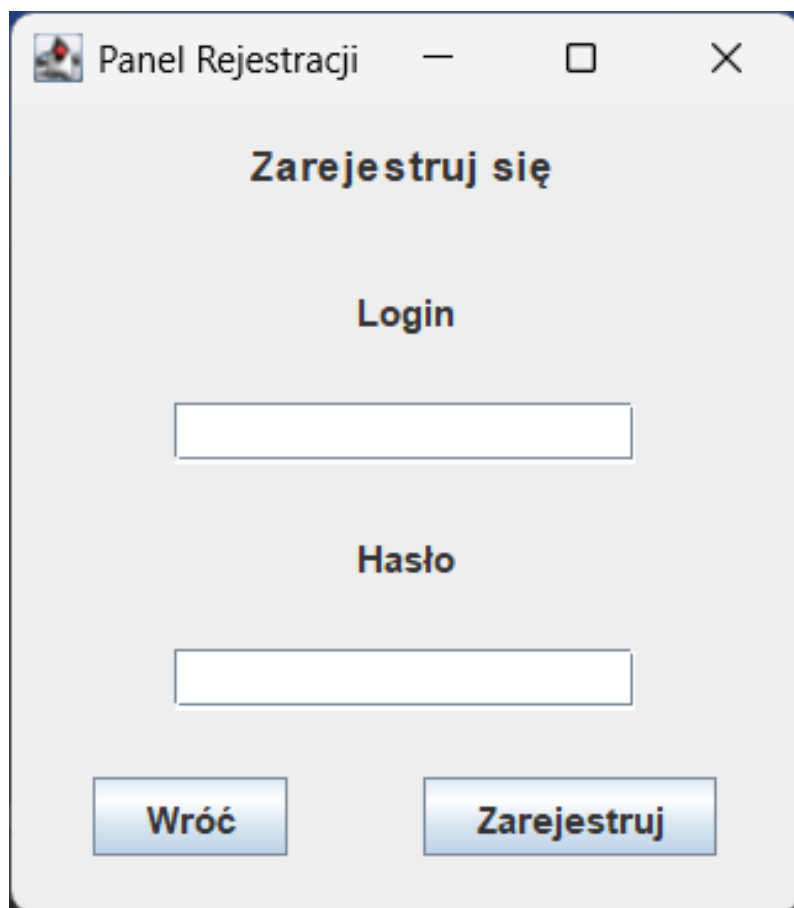
<https://github.com/Lukasz432/P0-LUKASZ-PAZ>

5 Prezentacja warstwy użytkowej projektu

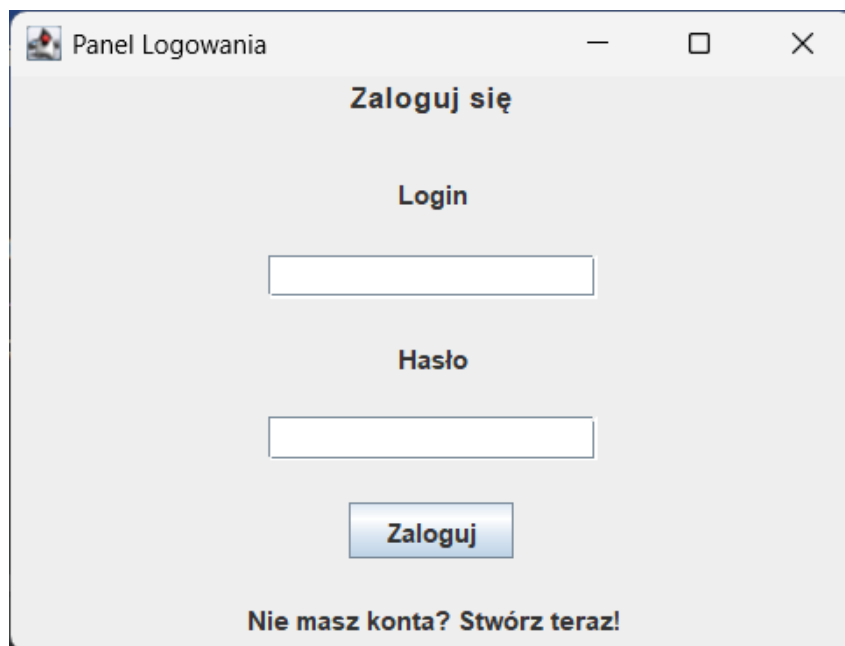
Aplikacja desktopowa Steam została napisana w Javie z użyciem biblioteki Swing do tworzenia interfejsu użytkownika oraz JDBC do komunikacji z bazą danych. Pozwala użytkownikowi na logowanie i rejestrację, przeglądanie i zarządzanie swoimi grami oraz stanem konta, a także dokonywanie zakupów i sprzedaży gier w sklepie.

Interfejs użytkownika składa się z kilku paneli GUI, które odpowiadają poszczególnym funkcjonalnościom:

- **RejestracjaGUI** – panel odpowiedzialny za rejestrację nowych użytkowników. Pozwala na wprowadzenie danych takich jak login i hasło.

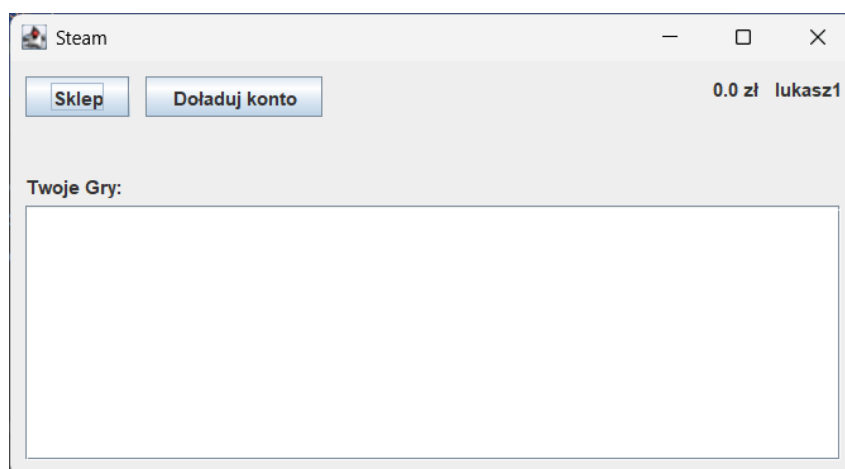


LogowanieGUI – panel do logowania istniejących użytkowników. Po poprawnym zalogowaniu użytkownik zostaje przekierowany do panelu głównego.



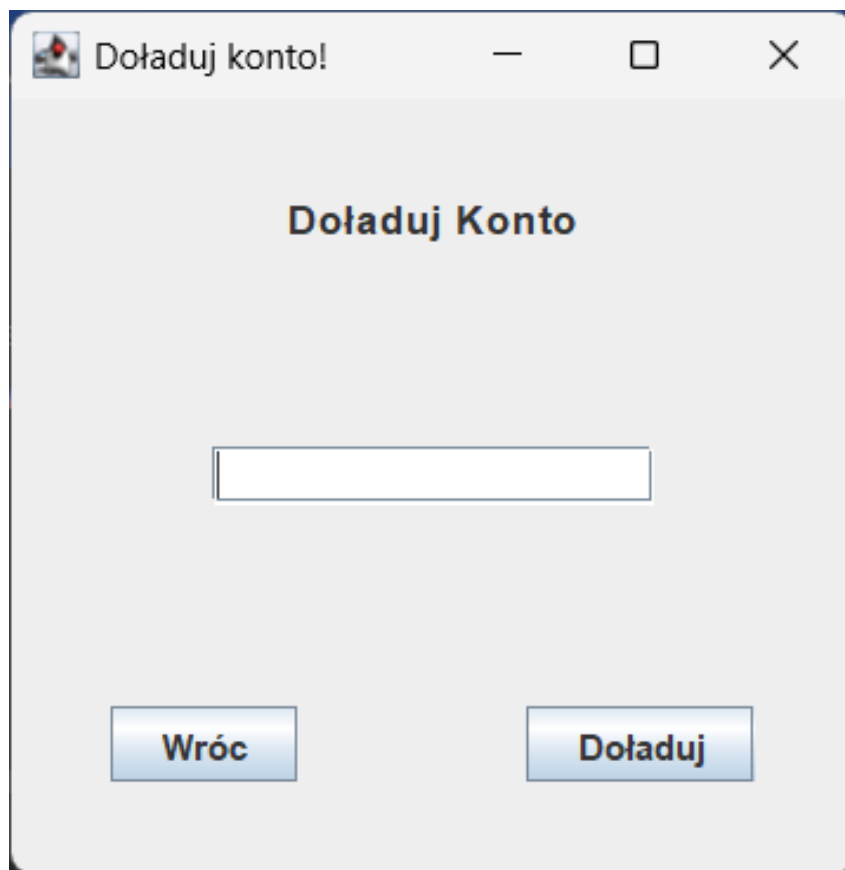
Rysunek 2: Panel logowania użytkownika (LogowanieGUI).

AplikacjaGUI – główny panel użytkownika, gdzie wyświetlana jest lista jego gier, aktualny stan konta oraz nick (login). Z tego poziomu można przejść do innych funkcji aplikacji.



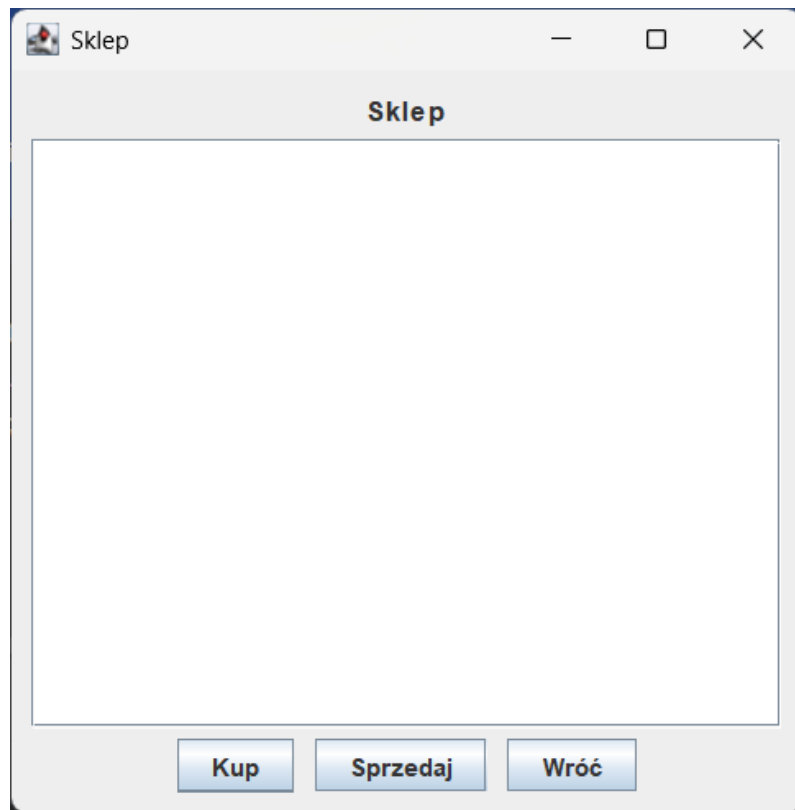
Rysunek 3: Główny panel użytkownika po zalogowaniu (AplikacjaGUI).

DoładujGUI – panel służący do doładowania konta użytkownika. Pozwala na wpisanie kwoty do doładowania i zatwierdzenie operacji.



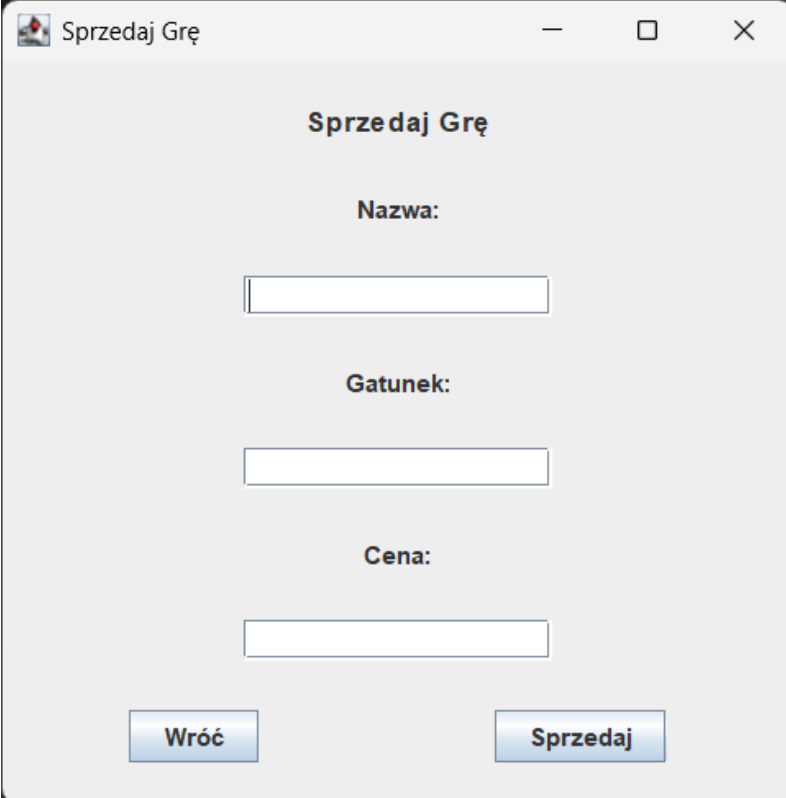
Rysunek 4: Panel doładowania konta (DoladujGUI).

- **SklepGUI** – panel sklepu, w którym użytkownik widzi listę gier dostępnych do zakupu oraz gry, które sam wystawił na sprzedaż.



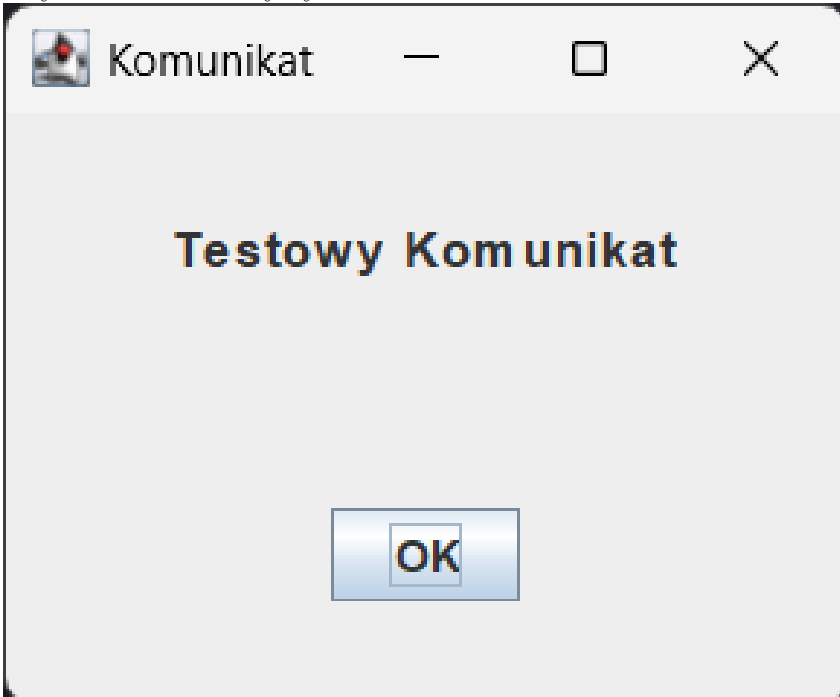
Rysunek 5: Panel sklepu z listą gier (SklepGUI).

- **SprzedajGreGUI** – panel dedykowany wystawianiu gier na sprzedaż. Umożliwia wpisanie potrzebnych danych gry oraz cene.



Rysunek 6: Panel sprzedaży gier (SprzedajGreGUI).

Komunikat – panel wyświetlający informacje i powiadomienia dla użytkownika, np. potwierdzenia, błędy lub komunikaty systemowe.



Rysunek 7: Panel komunikatów systemowych (Komunikat).

6 Podsumowanie

W ramach realizacji projektu aplikacji desktopowej Steam zostały zaprojektowane i zaimplementowane wszystkie kluczowe funkcjonalności, takie jak logowanie i rejestracja użytkowników, zarządzanie biblioteką gier, obsługa stanu konta oraz mechanizmy kupna i sprzedaży gier. Interfejs użytkownika został zrealizowany przy użyciu biblioteki Swing, co pozwoliło na stworzenie intuicyjnego i responsywnego środowiska pracy.

Projekt obejmował również integrację z bazą danych za pomocą JDBC, co umożliwiło trwałe przechowywanie informacji o użytkownikach, grach oraz transakcjach. Przeprowadzone testy potwierdziły poprawność działania aplikacji oraz spełnienie założonych wymagań funkcjonalnych.

Możliwe kierunki rozwoju projektu obejmują:

- Rozszerzenie aplikacji o mechanizmy rekomendacji gier na podstawie historii zakupów i ocen użytkownika.
- Implementację systemu ocen i recenzji gier, co zwiększyłoby interakcje między użytkownikami.
- Dodanie obsługi powiadomień w czasie rzeczywistym, np. o nowych ofertach w sklepie czy promocjach.
- Ulepszenie interfejsu użytkownika poprzez zastosowanie bardziej nowoczesnych frameworków UI lub integrację z technologiami webowymi.
- Zabezpieczenie aplikacji na poziomie transmisji danych, np. poprzez szyfrowanie połączeń i uwierzytelnianie dwuskładnikowe.

Realizacja powyższych rozszerzeń pozwoliłaby na podniesienie funkcjonalności i komfortu użytkownika aplikacji, zwiększając jej atrakcyjność na rynku.

Spis rysunków

1	Diagram Gantta przedstawiający harmonogram realizacji projektu	5
2	Panel logowania użytkownika (LogowanieGUI).	7
3	Główny panel użytkownika po zalogowaniu (AplikacjaGUI).	7
4	Panel doładowania konta (DoładujGUI).	8
5	Panel sklepu z listą gier (SklepGUI).	9
6	Panel sprzedaży gier (SprzedajGreGUI).	10
7	Panel komunikatów systemowych (Komunikat).	10

Załącznik nr 2 do Zarządzenia nr 228/2021 Rektora Uniwersytetu Rzeszowskiego z dnia 1 grudnia 2021 roku w sprawie ustalenia procedury antyplagiatowej w Uniwersytecie Rzeszowskim

OŚWIADCZENIE STUDENTA O SAMODZIELNOŚCI PRACY

Bukacz P.

Imię (imiona) i nazwisko studenta

Wydział Nauk Ścisłych i Technicznych

Informatyka

Nazwa kierunku

134956

Numer albumu

1. Oświadczam, że moja praca projektowa pt.: Przygotowanie dokumentacji do projektu w systemie L^AT_EX^{*}
 - 1) została przygotowana przeze mnie samodzielnie*,
 - 2) nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2021 r., poz. 1062) oraz dóbr osobistych chronionych prawem cywilnym,
 - 3) nie zawiera danych i informacji, które uzyskałem/am w sposób niedozwolony,
 - 4) nie była podstawą otrzymania oceny z innego przedmiotu na uczelni wyższej ani mnie, ani innej osobie.
2. Jednocześnie wyrażam zgodę/~~nie wyrażam zgody~~** na udostępnienie mojej pracy projektowej do celów naukowo-badawczych z poszanowaniem przepisów ustawy o prawie autorskim i prawach pokrewnych.

Rzeszów 11.06.2025

(miejscowość, data)

Bukacz P.

(czytelny podpis studenta)

* Uwzględniając merytoryczny wkład prowadzącego przedmiot

** – niepotrzebne skreślić