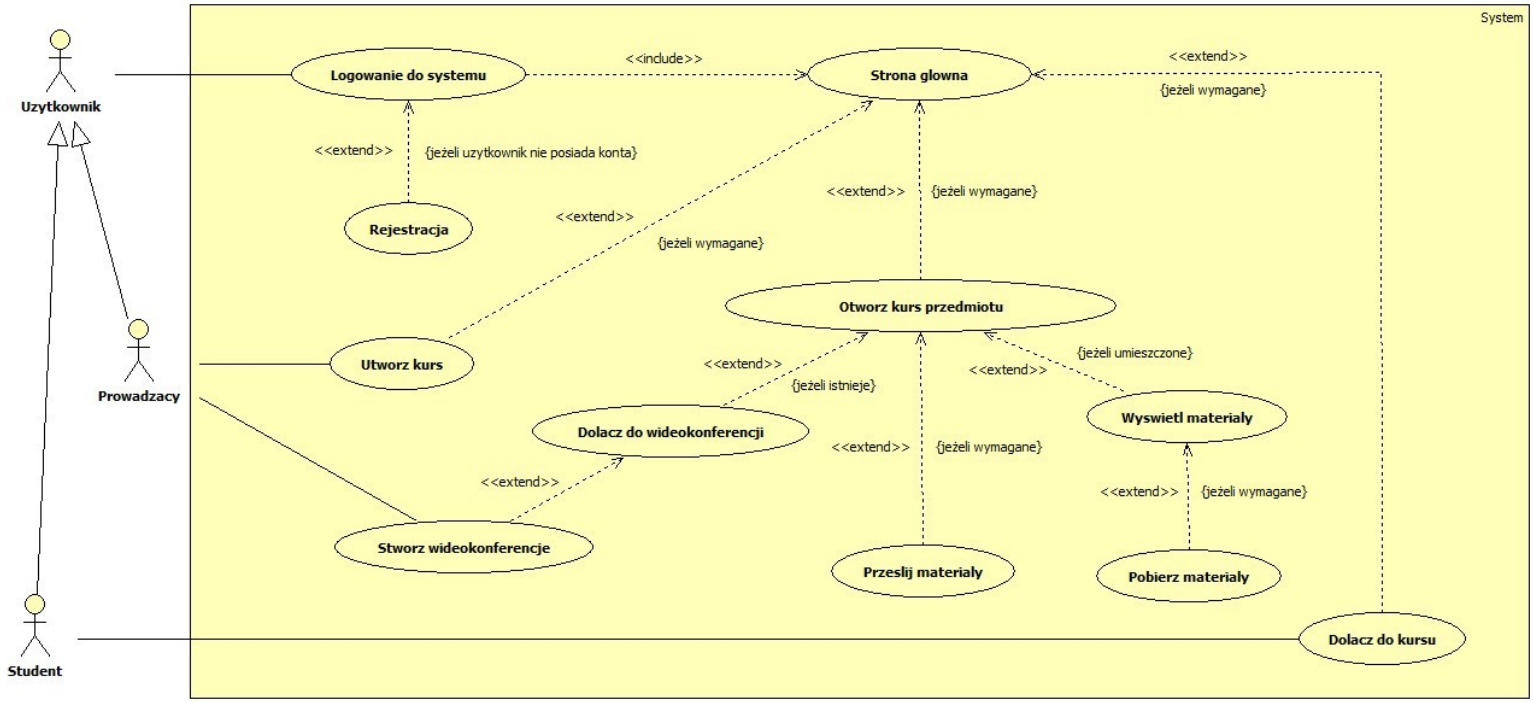
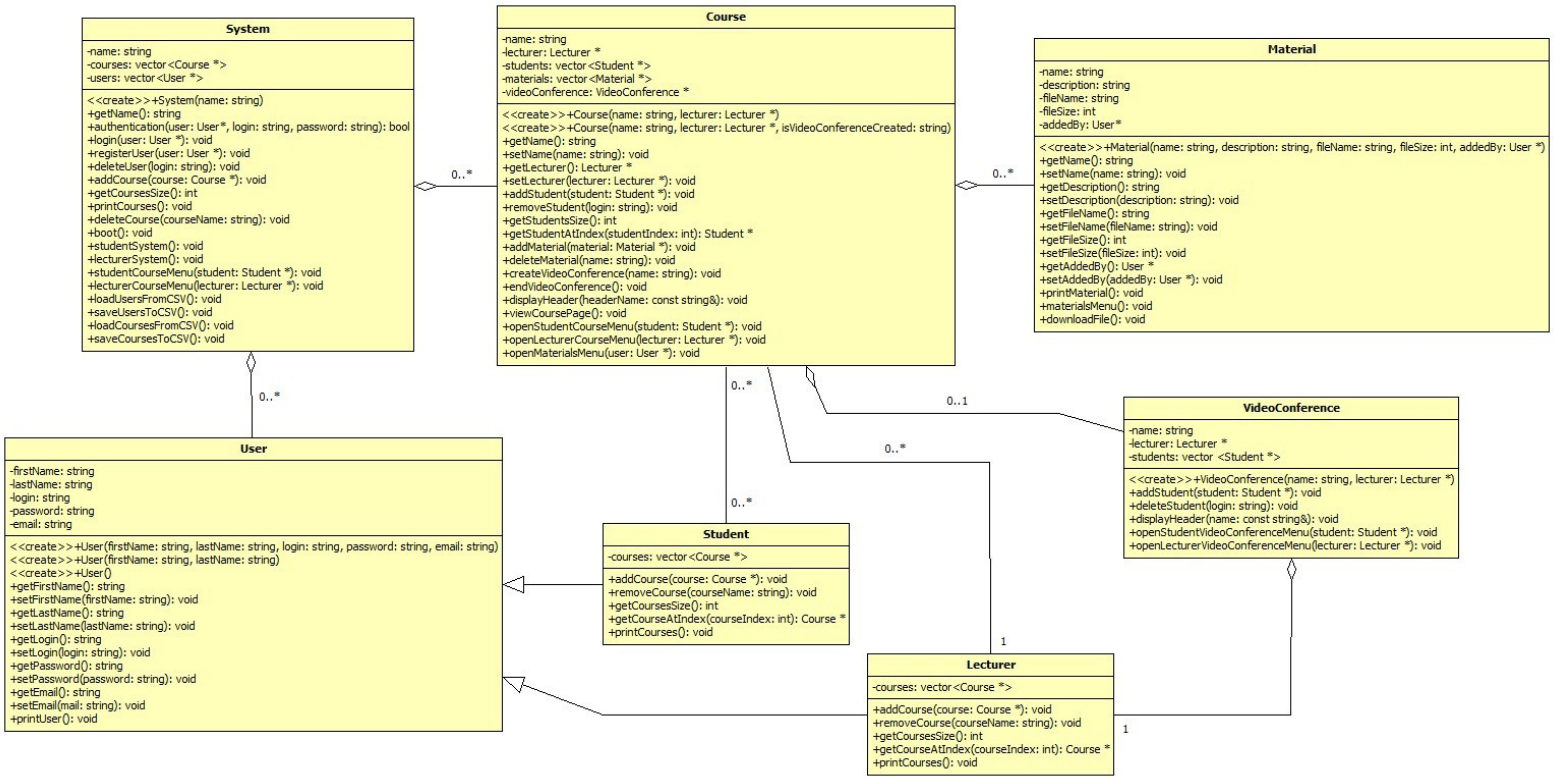
System Obsługi Studiów

Autorzy
- Tomasz Wnuk
- Bartosz Szynkaruk
- Mikołaj Hasiec

Model przypadków użycia



Model logiczny



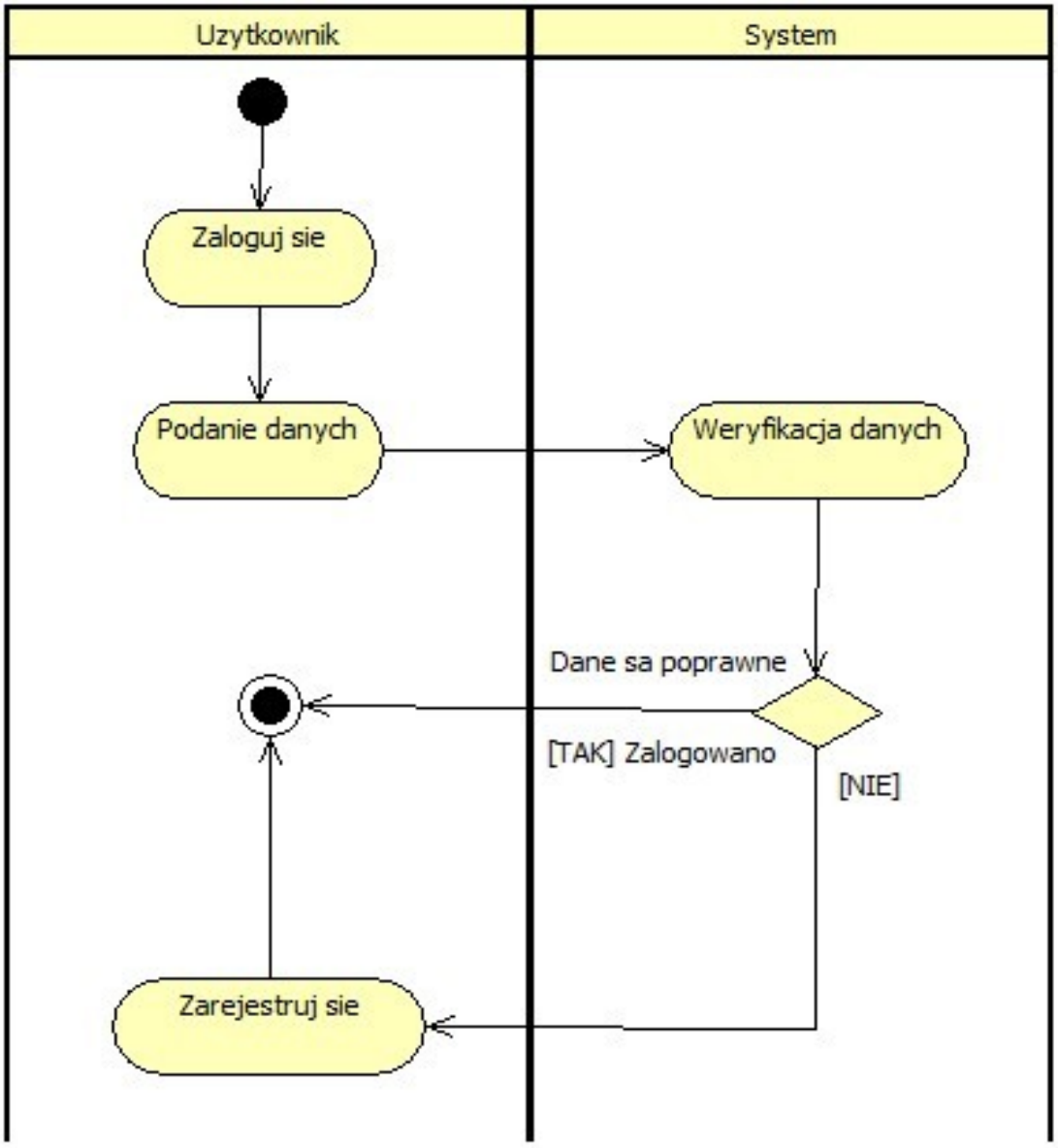
Przypadki użycia

Logowanie do systemu

- Scenariusz

Przypadek użycia	Logowanie do systemu.
Scenariusz	Prawidłowe podanie danych logowania do systemu.
Warunki wstępne	Użytkownik posiada konto w systemie.
Niezmienniki	Użytkownik chce zalogować się do systemu.
Opis	Użytkownik wpisuje swoje dane logowania. System weryfikuje dane.
Warunki końcowe	Użytkownik zostaje zalogowany i przeniesiony na stronę główną.

- Diagram Przepływu

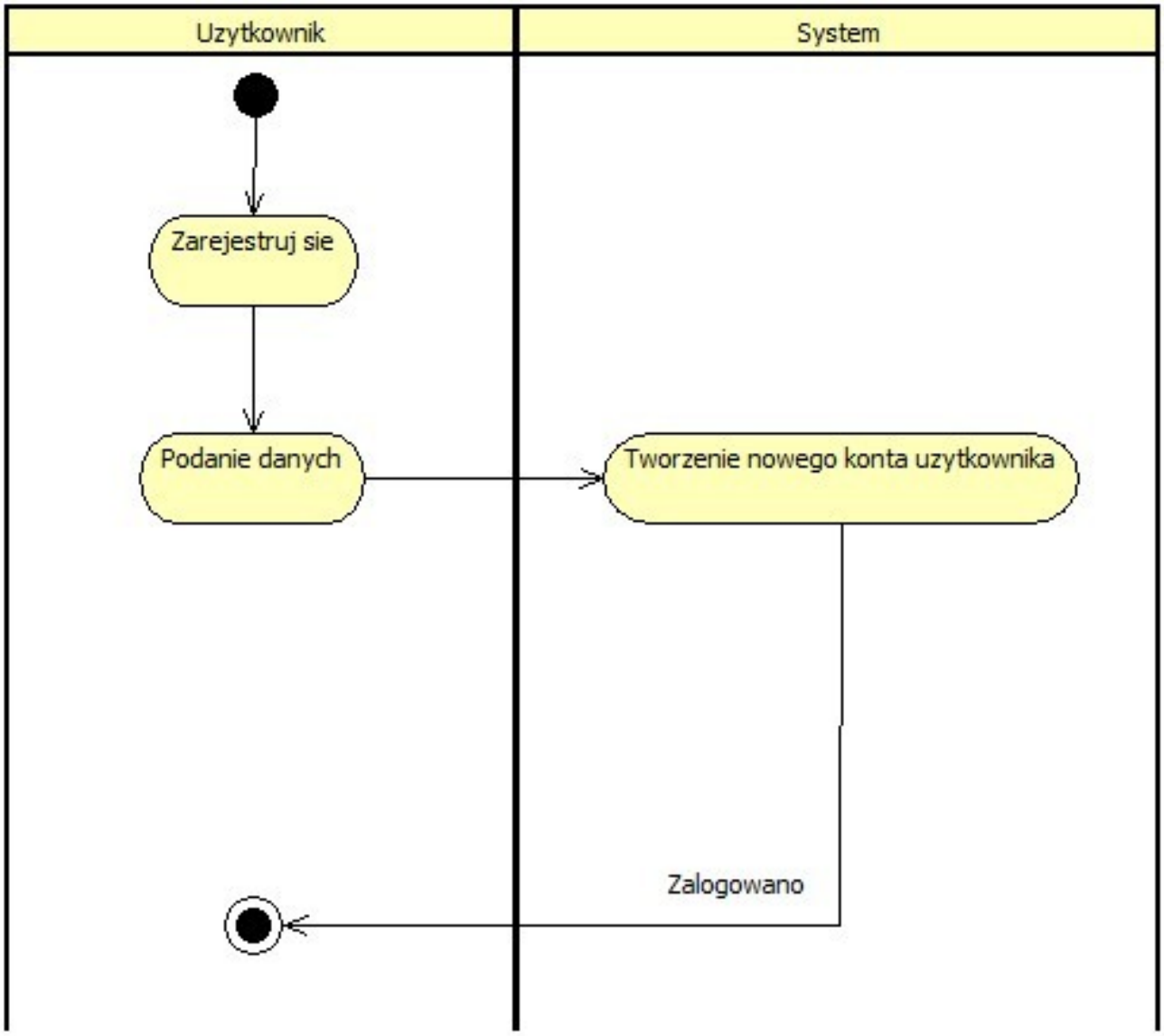


Rejestracja

- Scenariusz

Przypadek użycia	Rejestracja.
Scenariusz	Podanie danych używanych do rejestracji.
Warunki wstępne	Użytkownik nie posiada konta w systemie, konto o podanych danych nie istnieje.
Niezmienniki	Użytkownik chce utworzyć konto.
Opis	Użytkownik otwiera stronę rejestracji i wpisuje dane. System sprawdza czy konto o takiej nazwie nie istnieje i tworzy konto użytkownika.
Warunki końcowe	Użytkownik zostaje automatycznie zalogowany do systemu i przeniesiony na stronę główną.

- Diagram przepływu



Strona główna

- Scenariusz

Przypadek użycia	Strona główna.
Scenariusz	Użytkownik jest przekierowywany na stronę główną.
Warunki wstępne	Pomyślne zalogowanie do systemu.
Niezmienniki	Użytkownik chce skorzystać z funkcji systemu.
Opis	Pomyślne zalogowanie i przeniesienie użytkownika na stronę główną.
Warunki końcowe	Użytkownik może wybrać funkcjonalność systemu.

- Diagram przepływu

Nie posiada diagramu przepływu, ponieważ przypadek jest trywialny.

Otwórz kurs przedmiotu

- Scenariusz

Przypadek użycia	Otwórz kurs przedmiotu.
Scenariusz	Użytkownik wybiera kurs ze strony kursów.
Warunki wstępne	Użytkownik jest zalogowany i jest na stronie kursów.
Niezmienniki	Użytkownik chce otworzyć stronę kursu przedmiotu.
Opis	Użytkownik wybiera kurs z listy. System przekierowuje go na stronę wybranego kursu.
Warunki końcowe	Użytkownik zostaje przekierowany na stronę kursu.

- Diagram przepływu

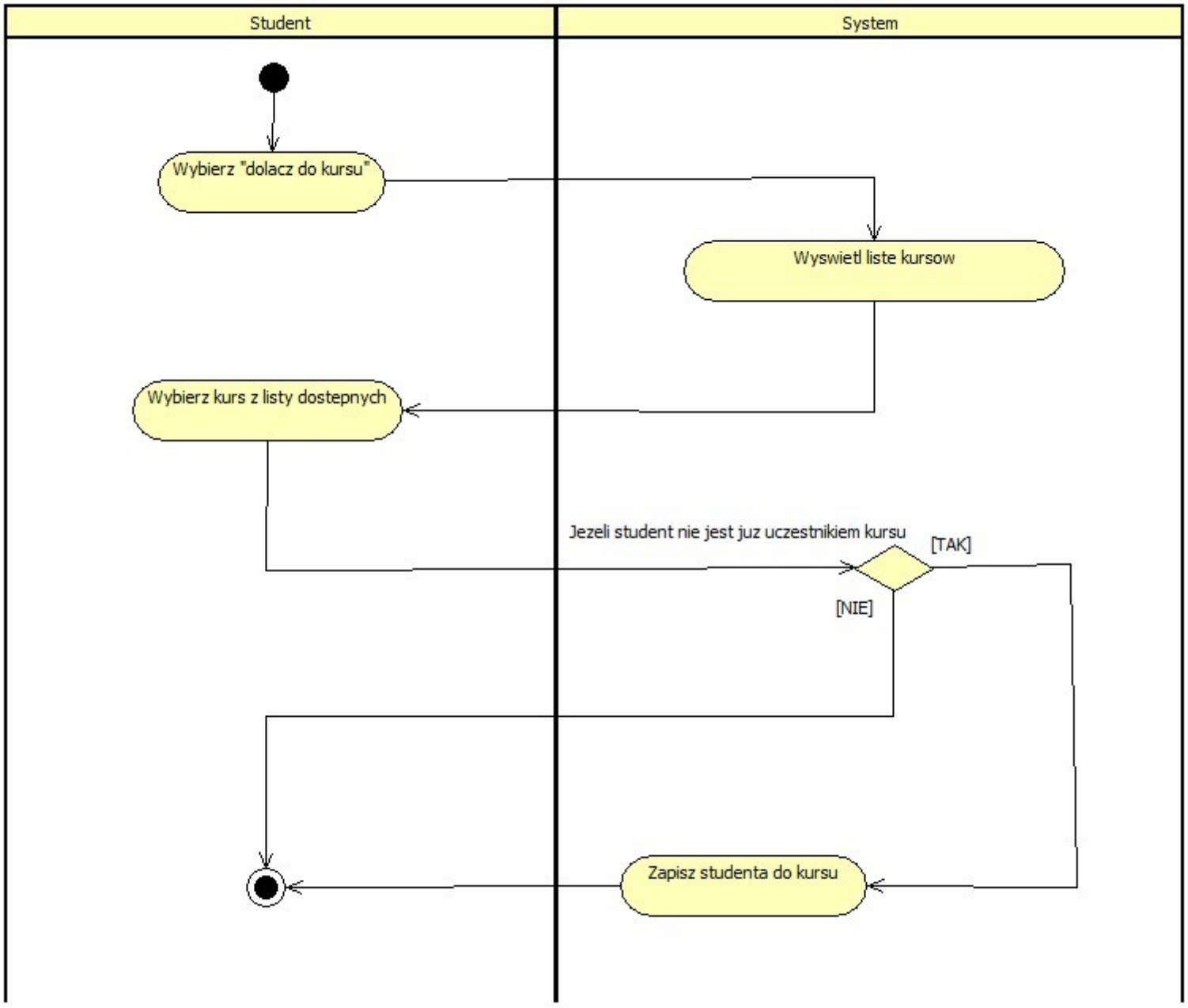
Nie posiada diagramu przepływu, ponieważ przypadek jest trywialny.

Dołącz do kursu

- Scenariusz

Przypadek użycia	Dołącz do kursu.
Scenariusz	Student dołącza do kursu.
Warunki wstępne	Student jest zalogowany. Istnieje kurs, do którego można dołączyć.
Niezmienniki	Student chce dołączyć do kursu.
Opis	Student wybiera kurs do którego chce dołączyć.
Warunki końcowe	Student zostaje uczestnikiem kursu.

- Diagram przepływu

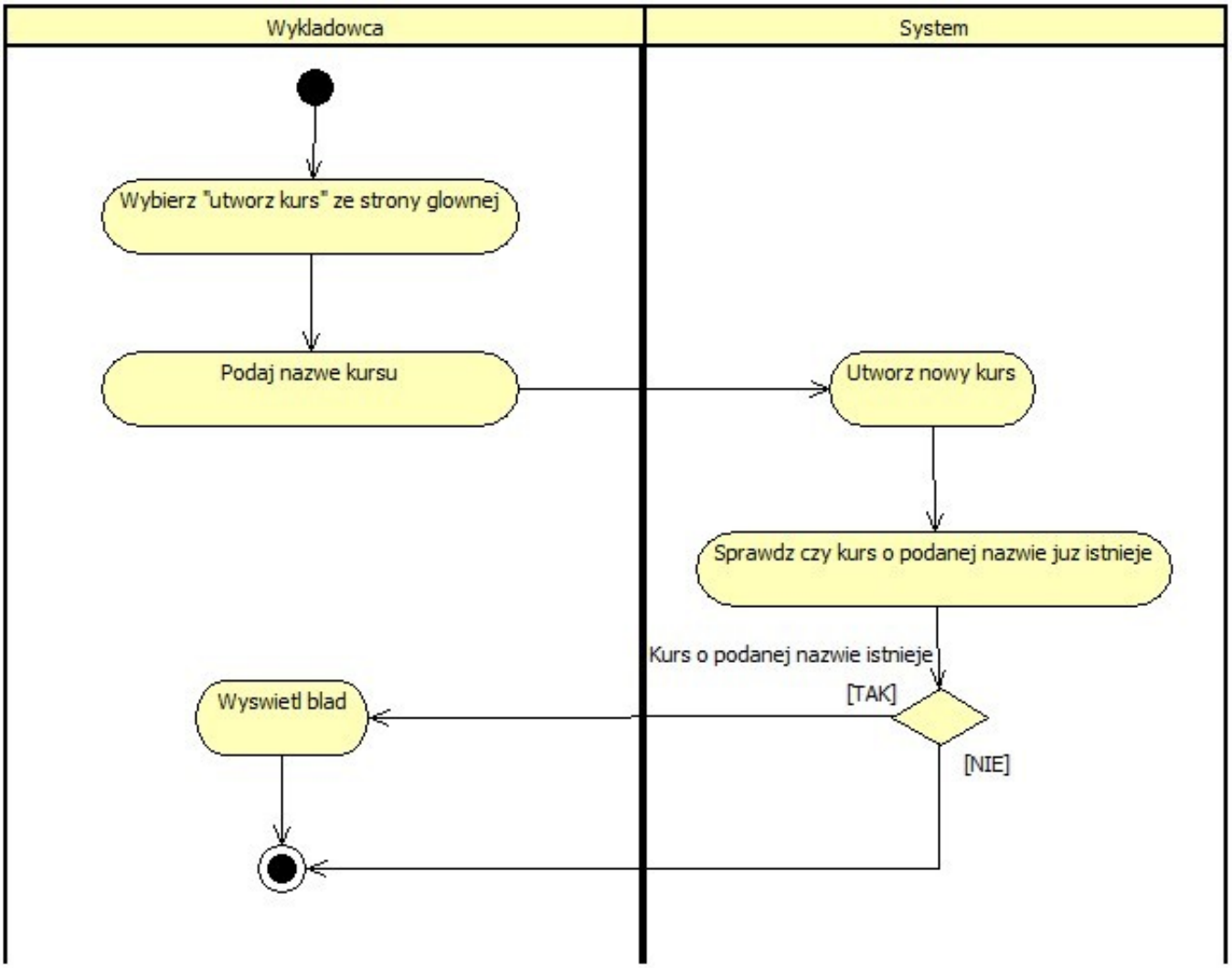


Utwórz kurs

- Scenariusz

Przypadek użycia	Utwórz kurs.
Scenariusz	Prowadzący tworzy kurs przedmiotu.
Warunki wstępne	Prowadzący jest zalogowany.
Niezmienniki	Prowadzący chce utworzyć kurs przedmiotu.
Opis	Prowadzący tworzy nowy kurs i podaje jego nazwę. System sprawdza poprawność nazwy.
Warunki końcowe	Nowy kurs zostaje utworzony.

- Diagram przepływu

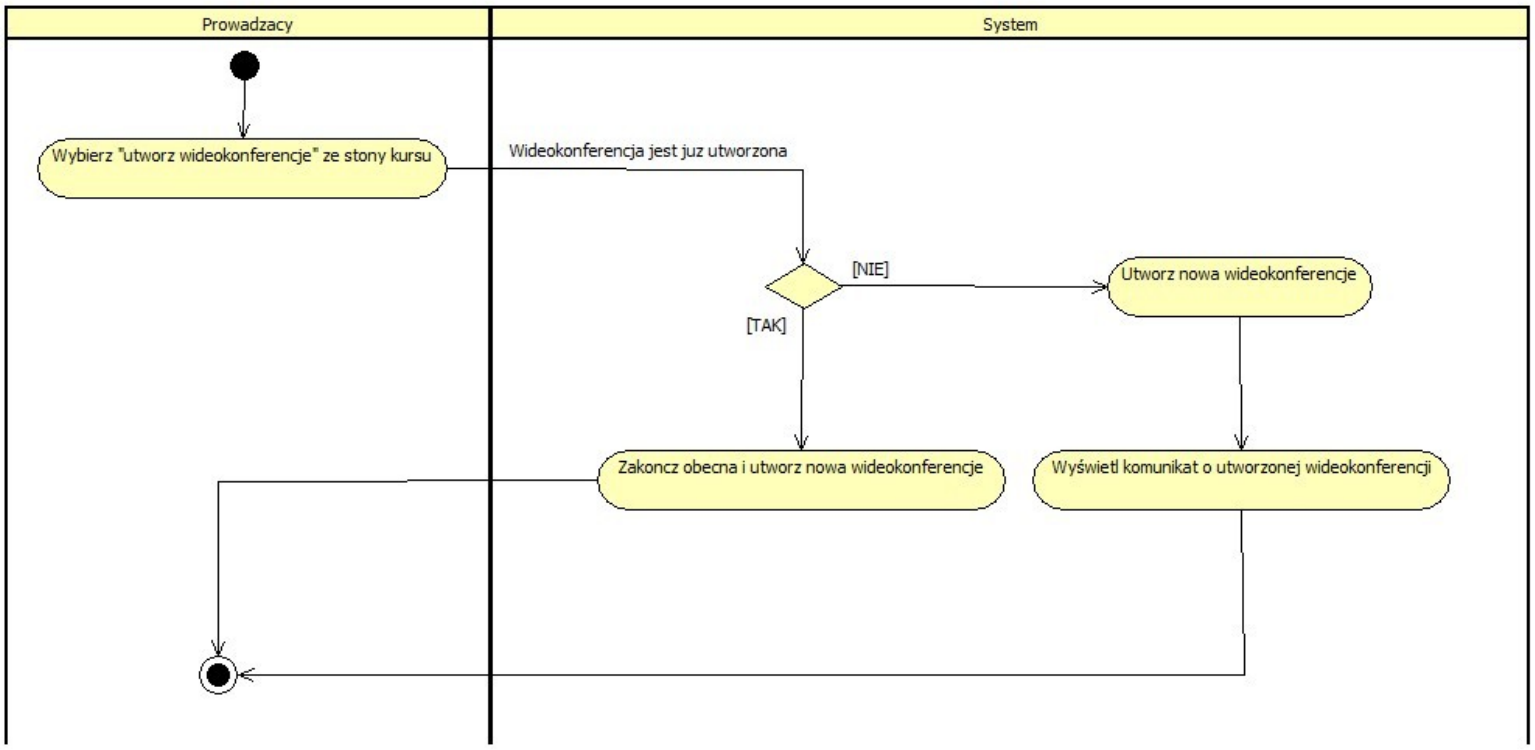


Utwórz wideokonferencje

- Scenariusz

Przypadek użycia	Utwórz wideokonferencję.
Scenariusz	Prowadzący tworzy wideokonferencję.
Warunki wstępne	Prowadzący jest zalogowany, wideokonferencja nie istnieje.
Niezmienniki	Prowadzący chce utworzyć wideokonferencję.
Opis	Prowadzący tworzy wideokonferencję.
Warunki końcowe	Wideokonferencja została utworzona.

- Diagram przepływu

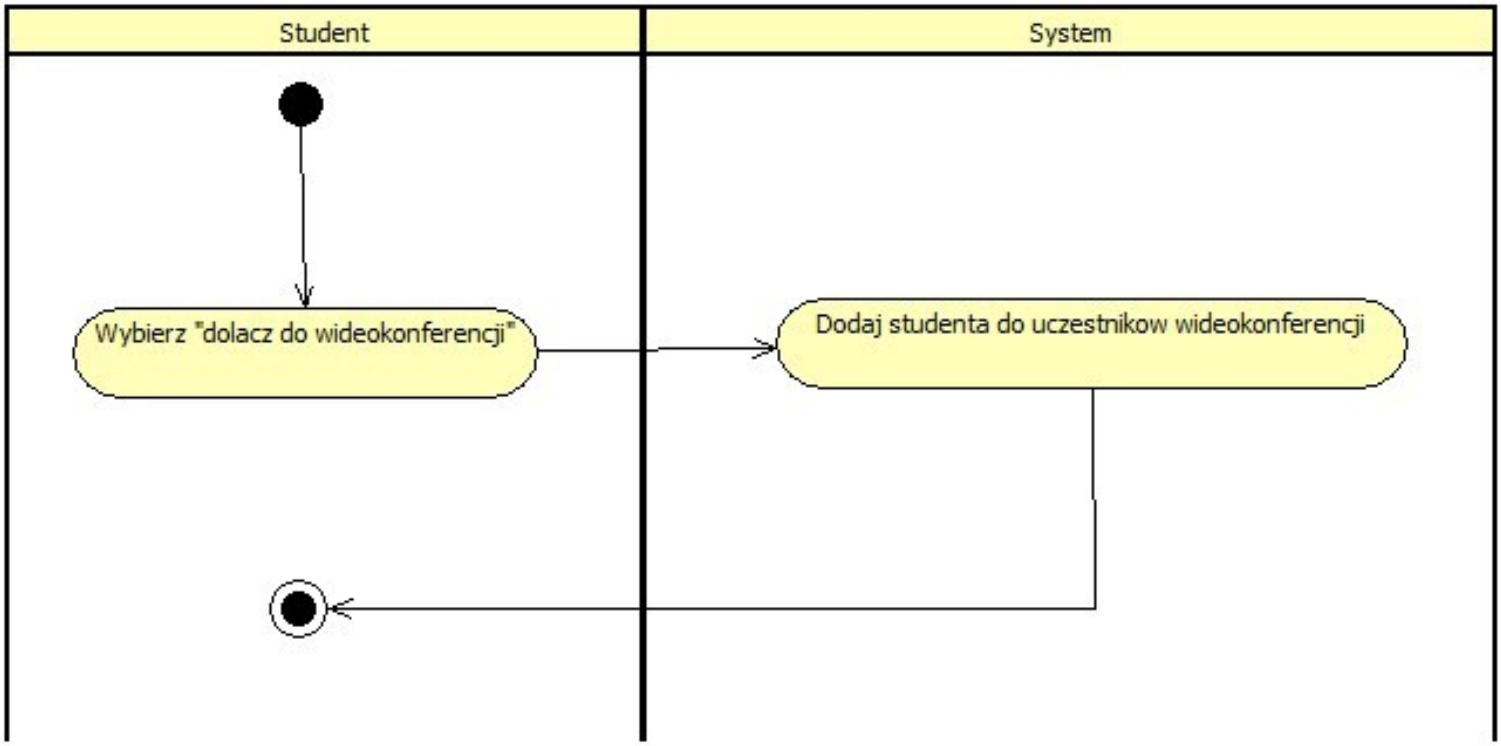


Dołącz do wideokonferencji

- Scenariusz

Przypadek użycia	Dołącz do wideokonferencji.
Scenariusz	Student dołącza do istniejącej wideokonferencji.
Warunki wstępne	Student jest zalogowany, wideokonferencja istnieje, student jest uczestnikiem kursu.
Niezmienniki	Student chce dołączyć do wideokonferencji.
Opis	Student dołącza do wideokonferencji.
Warunki końcowe	Student jest uczestnikiem wideokonferencji.

- Diagram przepływu

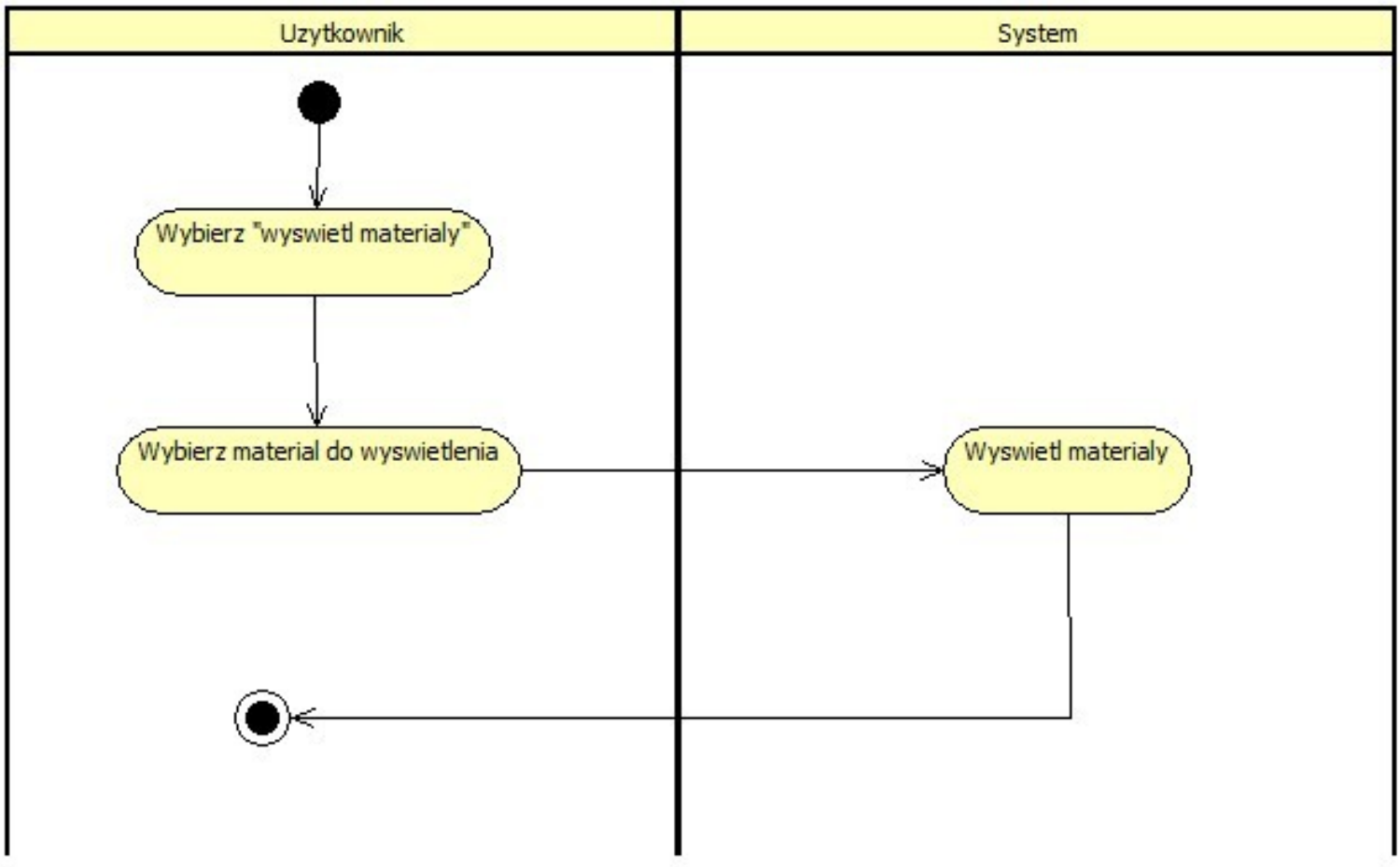


Wyświetl materiały

- Scenariusz

Przypadek użycia	Wyświetl materiały.
Scenariusz	Użytkownik wyświetla materiały.
Warunki wstępne	Użytkownik jest zalogowany, użytkownik jest uczestnikiem lub prowadzącym kursu, istnieją materiały możliwe do wyświetlenia.
Niezmienniki	Użytkownik chce wyświetlić materiały.
Opis	System przenosi użytkownika na stronę materiałów.
Warunki końcowe	Użytkownik zostaje przeniesiony na stronę materiałów.

- Diagram przepływu

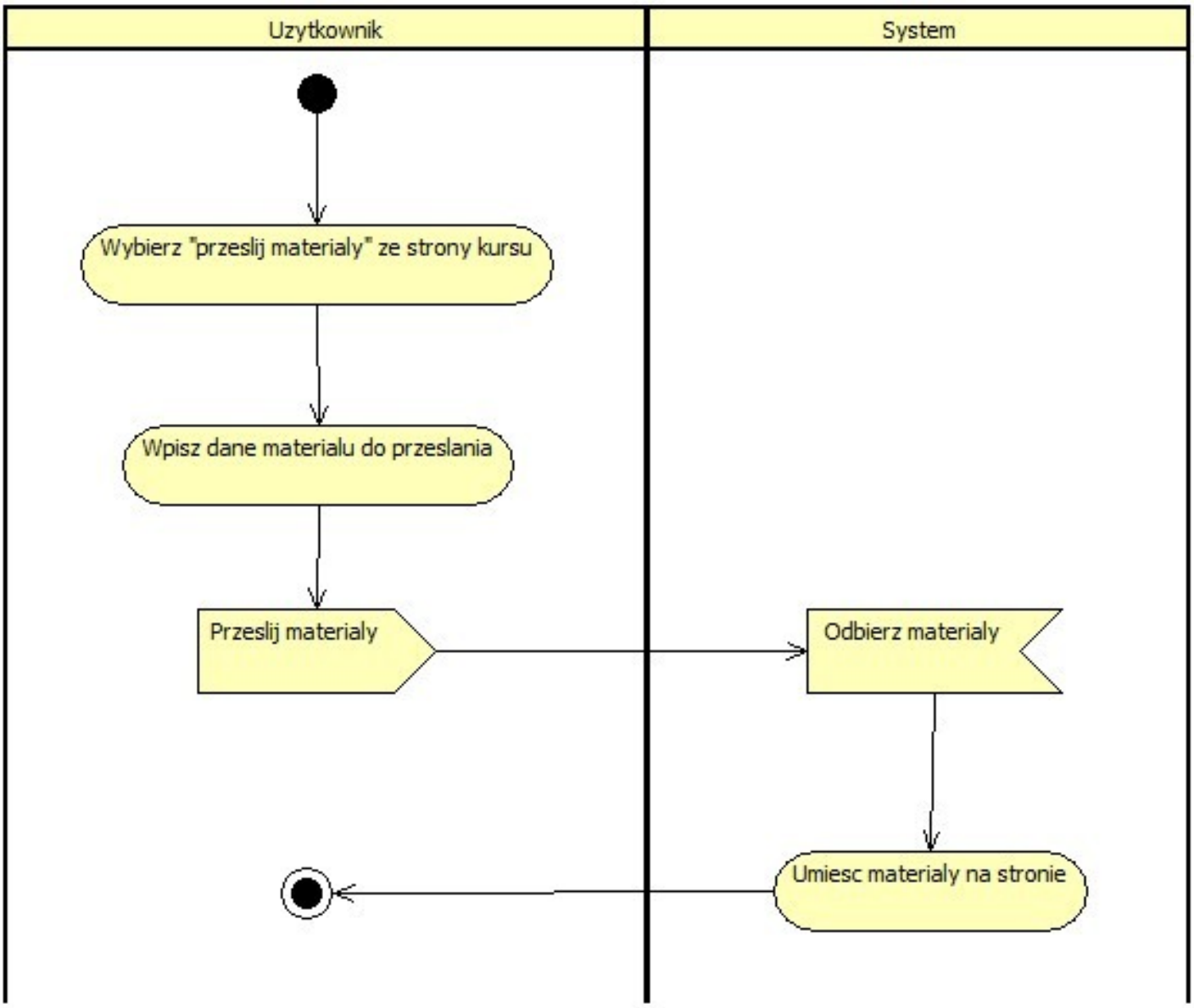


Prześlij materiały

- Scenariusz

Przypadek użycia	Prześlij materiały.
Scenariusz	Użytkownik przesyła materiały na stronę kursu.
Warunki wstępne	Użytkownik jest zalogowany, użytkownik jest uczestnikiem lub prowadzącym kursu.
Niezmienniki	Użytkownik chce przesłać materiały na stronę kursu.
Opis	Użytkownik przesyła materiały na stronę kursu. Użytkownik nadaje nazwę i opis przesłanym materiałom.
Warunki końcowe	Materiały zostają dodane na stronę kursu.

- Diagram przepływu



Pobierz materiały

- Scenariusz

Przypadek użycia	Pobierz materiały.
Scenariusz	Użytkownik pobiera materiały.
Warunki wstępne	Użytkownik jest zalogowany, użytkownik jest uczestnikiem lub prowadzącym kursu, istnieją materiały dostępne do pobrania.
Niezmienniki	Użytkownik chce pobrać materiały ze strony kursu.
Opis	Użytkownik pobiera materiały.
Warunki końcowe	Materiały zostają pobrane na dysk użytkownika.

- Diagram przepływu

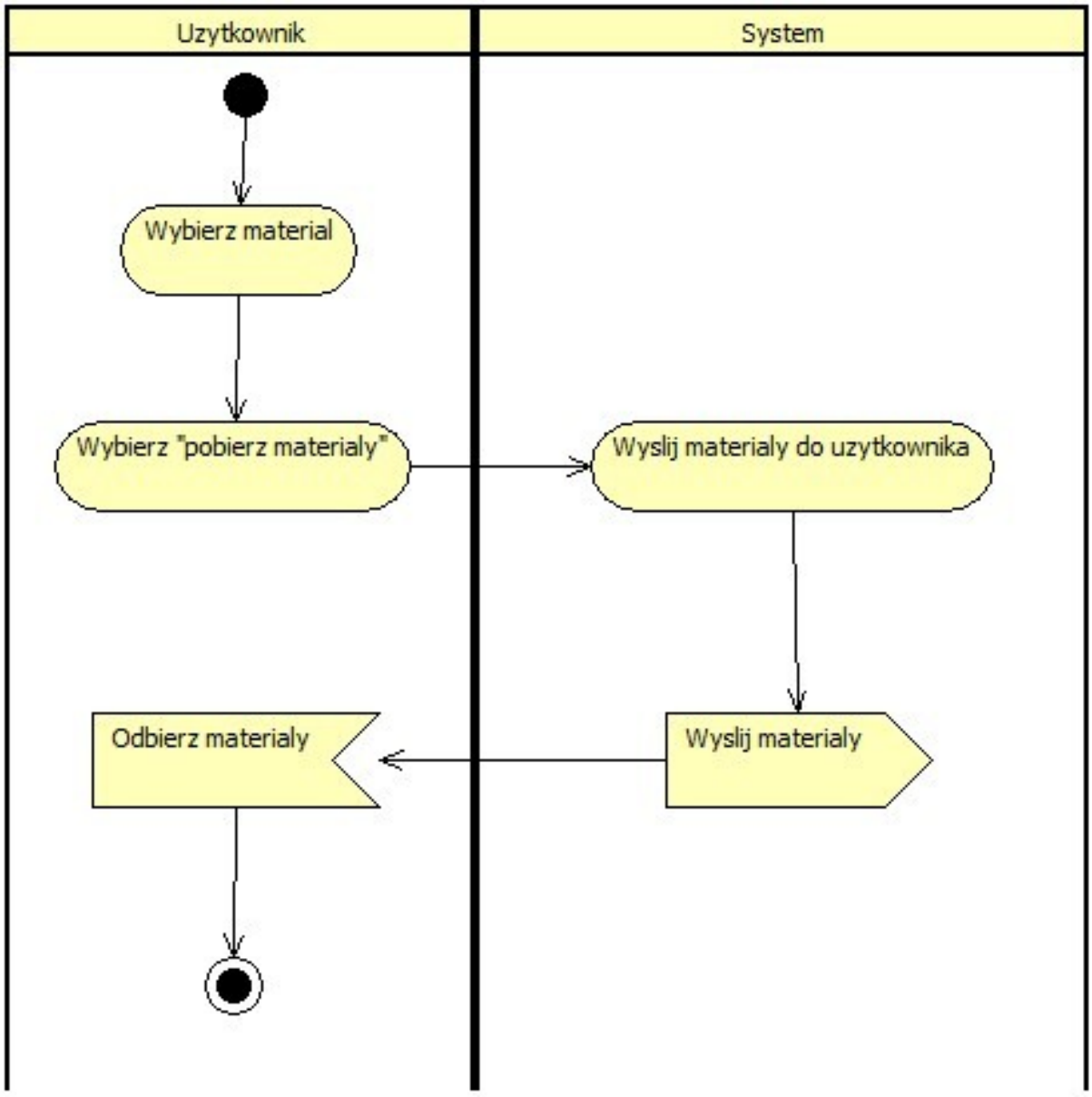
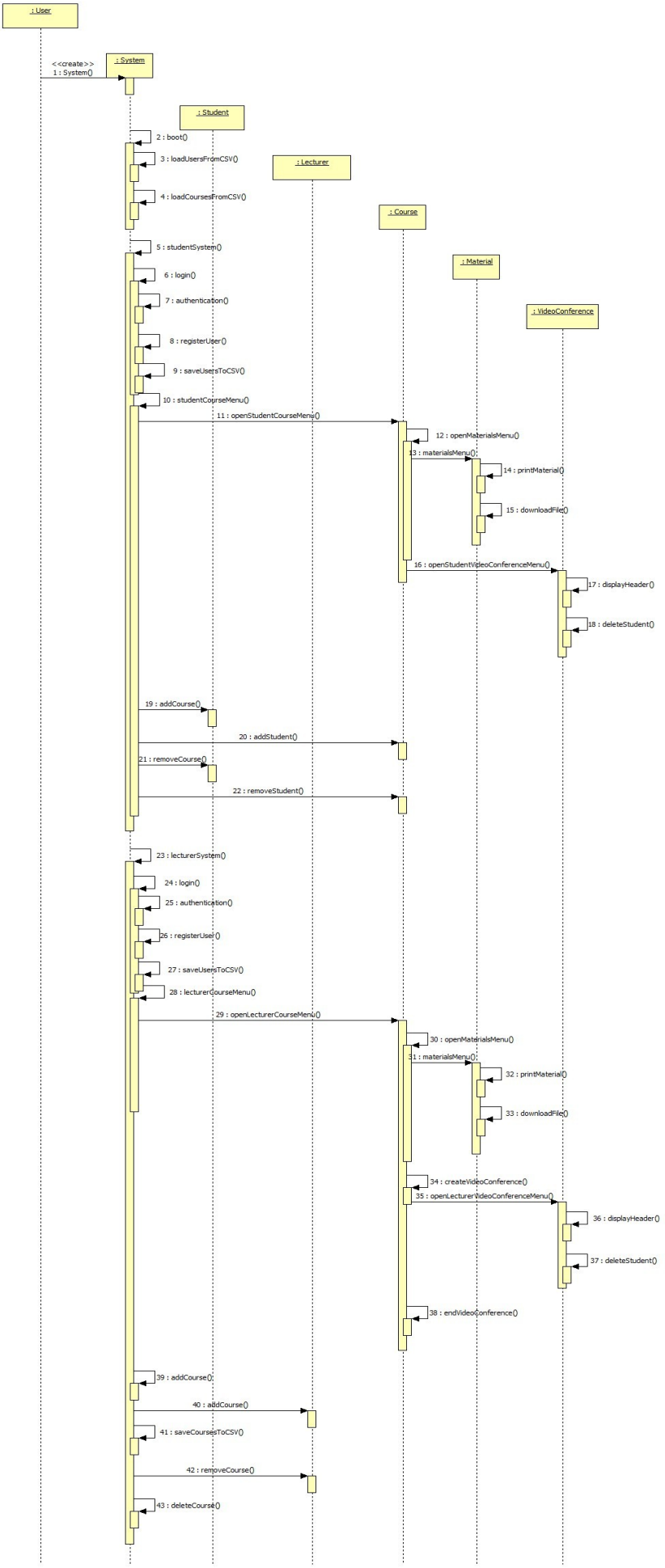


Diagram sekwencji



Kod programu

- Main.cpp

```cpp
//
//
//  @ Project : System Obsługi Studiów
//  @ File Name : main.cpp
//  @ Date : 10.06.2023
//  @ Author : Tomasz Wnuk, Bartosz Szynkaruk, Mikołaj Hasiec
//
//

// Deklaracja zależności i bibliotek
#include "System.h"

// Funkcja main
int main() {
    // Utworzenie obiektu systemu
    System system = System("System Obslugi Studiow");
    // Uruchomienie systemu
    system.boot();

    // Zmienna przechowująca wybór użytkownika
    std::string userInput;

    // Wyświetl opcje
    std::cout << "Wybierz opcje:" << "\n";
    std::cout << "1. System Obslugi Studiow dla Studentow" << "\n";
    std::cout << "2. System Obslugi Studiow dla Wykladowcow" << "\n";

    // Pętla wyboru wersji systemu
    while(true) {
        // Pobierz wybór użytkownika
        std::cin >> userInput;
        // Jeżeli użytkownik wybrał opcję 1
        if(userInput == "1") {
            // Wywołaj metodę systemu studenta
            system.studentSystem();
            // Jeżeli użytkownik wybrał opcję 2
        } else if(userInput == "2") {
            // Wywołaj metodę systemu wykładowcy
            system.lecturerSystem();
            // Jeżeli użytkownik wybrał inną opcję
        } else {
            std::cout << "Niepoprawna opcja!" << "\n";
        }
    }

    return 0;
}
```

- User.h

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : System Obsługi Studiów
//  @ File Name : User.h
//  @ Date : 10.06.2023
//  @ Author : Tomasz Wnuk, Bartosz Szynkaruk, Mikołaj Hasiec
//
//

#if !defined(_USER_H)
#define _USER_H

// Deklaracja zależności i bibliotek
#include <iostream>
#include <string>

// Deklaracja klasy User
class User {
public:
   User(std::string firstName, std::string lastName, std::string login, std::string password, std::string email);  //
Konstruktor klasy User z polami firstName, lastName, login, password i email
   User(std::string firstName, std::string lastName);  // Konstruktor klasy User z polami firstName i lastName
   User();  // Pusty konstruktor klasy User
   std::string getFirstName();  // Akcesor pola firstName
   void setFirstName(std::string firstName);  // Mutator pola firstName
   std::string getLastName();  // Akcesor pola lastName
   void setLastName(std::string lastName);  // Mutator pola lastName
   std::string getLogin();  // Akcesor pola login
   void setLogin(std::string login);  // Mutator pola login
   std::string getPassword();  // Akcesor pola password
   void setPassword(std::string password);  // Mutator pola password
   std::string getEmail();  // Akcesor pola email
   void setEmail(std::string email);  // Mutator pola email
   void printUser();  // Metoda wyświetlająca dane użytkownika
private:
   std::string firstName;  // Imię użytkownika
   std::string lastName;  // Nazwisko użytkownika
   std::string login;  // Login użytkownika
   std::string password;  // Hasło użytkownika
   std::string email;  // Adres email użytkownika
};

#endif  //_USER_H
```

- User.cpp

```cpp
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : System Obsługi Studiów
//  @ File Name : User.cpp
//  @ Date : 10.06.2023
//  @ Author : Tomasz Wnuk, Bartosz Szynkaruk, Mikołaj Hasiec
//
//

// Deklaracja zależności i bibliotek
#include <iostream>
#include "User.h"

// Konstruktor klasy User z polami firstName, lastName, login, password i email
User::User(std::string firstName, std::string lastName, std::string login, std::string password, std::string email) {
    this->firstName = firstName;
    this->lastName = lastName;
    this->login = login;
    this->password = password;
    this->email = email;
}

// Konstruktor klasy User z polami firstName i lastName
User::User(std::string firstName, std::string lastName) {
    this->firstName = firstName;
    this->lastName = lastName;
}

// Pusty konstruktor klasy User
User::User() {}

// Akcesor pola firstName
std::string User::getFirstName() {
    return firstName;
}

// Mutator pola firstName
void User::setFirstName(std::string firstName) {
    this->firstName = firstName;
}

// Akcesor pola lastName
std::string User::getLastName() {
    return lastName;
}

// Mutator pola lastName
void User::setLastName(std::string lastName) {
    this->lastName = lastName;
}

// Akcesor pola login
std::string User::getLogin() {
    return login;
}

// Mutator pola login
void User::setLogin(std::string login) {
```

```
    this->login = login;
}

// Akcesor pola password
std::string User::getPassword() {
    return password;
}

// Mutator pola password
void User::setPassword(std::string password) {
    this->password = password;
}

// Akcesor pola email
std::string User::getEmail() {
    return email;
}

// Mutator pola email
void User::setEmail(std::string email) {
    this->email = email;
}

// Metoda wyświetlająca dane użytkownika
void User::printUser() {
    // Utwórz zmienne do wyświetlenia nagłówka danych użytkownika
    const std::string headerName = "Uzytkownik";  // Nazwa nagłówka
    const int totalWidth = 90;  // Szerokość całego wyświetlanego napisu
    const int nameWidth = headerName.length();  // Szerokość nazwy kursu
    const int paddingWidth = (totalWidth - nameWidth) / 2;  // Szerokość wypełnienia
    std::string hiddenUserPassword = getPassword();  // Utwórz ukrytą kopię hasła użytkownika

    // Zamień każdy znak hasła na znak '*'
    hiddenUserPassword.replace(0, hiddenUserPassword.length(), hiddenUserPassword.length(), '*');

    // Wyświetl nagłówek danych użytkownika
    std::cout <<
"[=============================================================================================
====]\n";
    std::cout << "|
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Uzytkownik~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\
n";
    std::cout << "|
=============================================================================================
===|\n";
    // Wyświetl dane użytkownika
    std::cout << "| Login: " << getLogin() << "\n";  // Wyświetl login użytkownika
    std::cout << "| Haslo: " << hiddenUserPassword << "\n";  // Wyświetl ukryte hasło użytkownika
    std::cout << "| Imie: " << getFirstName() << "\n";  // Wyświetl imię użytkownika
    std::cout << "| Nazwisko: " << getLastName() << "\n";  // Wyświetl nazwisko użytkownika
    std::cout << "| Email: " << getEmail() << "\n";  // Wyświetl email użytkownika
    std::cout <<
"[=============================================================================================
====]\n";
}
```

- Lecturer.h

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : System Obsługi Studiów
//  @ File Name : Lecturer.h
//  @ Date : 10.06.2023
//  @ Author : Tomasz Wnuk, Bartosz Szynkaruk, Mikołaj Hasiec
//
//

#if !defined(_LECTURER_H)
#define _LECTURER_H

// Deklaracja zależności i bibliotek
#include <string>
#include <vector>
#include "User.h"
#include "Course.h"

// Deklaracja klas
class Course;

// Deklaracja klasy Lecturer dziedziczącej po klasie User
class Lecturer : public User {
   using User::User;  // Dziedziczenie konstruktorów klasy User
public:
   void addCourse(Course * course);  // Metoda dodająca kurs do wektora kursów wykładowcy
   void removeCourse(std::string courseName);  // Metoda usuwająca kurs z wektora kursów wykładowcy
   int getCoursesSize();  // Akcesor rozmiaru wektora kursów wykładowcy
   Course * getCourseAtIndex(int courseIndex);  // Metoda zwracająca kurs z wektora kursów wykładowcy o podanym
indeksie
   void printCourses();  // Metoda wyświetlająca kursy wykładowcy
private:
   std::vector <Course *> courses;  // Wektor kursów
};

#endif  //_LECTURER_H
```

- Lecturer.cpp

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : System Obsługi Studiów
//  @ File Name : Lecturer.cpp
//  @ Date : 10.06.2023
//  @ Author : Tomasz Wnuk, Bartosz Szynkaruk, Mikołaj Hasiec
//
//


// Deklaracja zależności i bibliotek
#include "Lecturer.h"

// Metoda dodająca kurs do wektora kursów wykładowcy
void Lecturer::addCourse(Course * course) {
    // Dodanie kursu do wektora kursów
    courses.push_back(course);
}

// Metoda usuwająca kurs z wektora kursów wykładowcy
void Lecturer::removeCourse(std::string courseName) {
    // Przeszukaj wektor kursów
    for(int i = 0; i < courses.size(); i++) {
        // Jeżeli name kursu jest taka sama jak podana name
        if(courses[i]->getName() == courseName) {
            // Usuń kurs z wektora kursów
            courses.erase(courses.begin() + i);
        }
    }
}

// Akcesor rozmiaru wektora kursów wykładowcy
int Lecturer::getCoursesSize() {
    // Zwróć rozmiar wektora kursów
    return courses.size();
}

// Metoda zwracająca kurs z wektora kursów wykładowcy o podanym indeksie
Course * Lecturer::getCourseAtIndex(int courseIndex) {
    // Zwróć kursu o podanym indeksie
    return courses[courseIndex];
}

// Metoda wyświetlająca kursy wykładowcy
void Lecturer::printCourses() {
    // Wyświetl nagłówek kursów wykładowcy
    std::cout <<
"[=============================================================================================
====]\n";
    std::cout << "|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Twoje
Kursy~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\n";
    std::cout <<
"[=============================================================================================
====]\n";
    // Przeszukaj wektor kursów
    for(int i = 0; i < courses.size(); i++) {
        // Wyświetl nazwę kursu
        std::cout << "| " << i + 1 << ". " << courses[i]->getName() << "\n";
    }
}
```

- Student.h

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : System Obsługi Studiów
//  @ File Name : Student.h
//  @ Date : 10.06.2023
//  @ Author : Tomasz Wnuk, Bartosz Szynkaruk, Mikołaj Hasiec
//
//

#if !defined(_STUDENT_H)
#define _STUDENT_H

// Deklaracja zależności i bibliotek
#include <string>
#include <vector>
#include "User.h"
#include "Course.h"

// Deklaracja klas
class Course;

// Deklaracja klasy Student dziedziczącej po klasie User
class Student : public User {
   using User::User;  // Dziedziczenie konstruktorów klasy User
public:
   void addCourse(Course * course);  // Metoda dodająca kurs do wektora kursów studenta
   void removeCourse(std::string courseName);  // Metoda usuwająca kurs z wektora kursów studenta
   int getCoursesSize();  // Akcesor rozmiaru wektora kursów studenta
   Course * getCourseAtIndex(int courseIndex); // Metoda zwracająca kurs z wektora kursów studenta o podanym
indeksie
   void printCourses();  // Metoda wyświetlająca kursy studenta
private:
   std::vector <Course *> courses;  // Wektor kursów
};

#endif  //_STUDENT_H
```

- Student.cpp

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : System Obsługi Studiów
//  @ File Name : Student.cpp
//  @ Date : 10.06.2023
//  @ Author : Tomasz Wnuk, Bartosz Szynkaruk, Mikołaj Hasiec
//
//

// Deklaracja zależności i bibliotek
#include "Student.h"

// Metoda dodająca kurs do wektora kursów studenta
void Student::addCourse(Course * course) {
   // Dodanie kursu do wektora kursów
   courses.push_back(course);
}

// Metoda usuwająca kurs z wektora kursów studenta
void Student::removeCourse(std::string courseName) {
   // Przeszukajwektor kursów
   for(int i = 0; i < courses.size(); i++) {
      // Jeśli name kursu jest równa podanej nazwie
      if(courses[i]->getName() == courseName) {
         // Usuń kurs z wektora kursów
         courses.erase(courses.begin() + i);
      }
   }
}

// Akcesor rozmiaru wektora kursów studenta
int Student::getCoursesSize() {
   // Zwróć rozmiar wektora kursów
   return courses.size();
}

// Metoda zwracająca kurs z wektora kursów studenta o podanym indeksie
Course * Student::getCourseAtIndex(int courseIndex) {
   // Zwróć kurs o podanym indeksie
   return courses[courseIndex];
}

// Metoda wyświetlająca kursy studenta
void Student::printCourses() {
   // Wyświetl nagłówek kursów studenta
   std::cout <<
"[===========================================================================
====]\n";
   std::cout << "|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Twoje
Kursy~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\n";
   std::cout <<
"[===========================================================================
====]\n";
   // Przeszukaj wektor kursów
   for(int i = 0; i < courses.size(); i++) {
      // Wyświetl nazwę kursu
      std::cout << "| " << i + 1 << ". " << courses[i]->getName() << "\n";
   }
}
```

- System.h

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : System Obsługi Studiów
//  @ File Name : System.h
//  @ Date : 10.06.2023
//  @ Author : Tomasz Wnuk, Bartosz Szynkaruk, Mikołaj Hasiec
//
//

#if !defined(_SYSTEM_H)
#define _SYSTEM_H

// Deklaracja zależności i bibliotek
#include <string>
#include <vector>
#include "User.h"
#include "Course.h"

// Deklaracja klasy User
class System {
public:
   System(std::string name);  // Konstruktor klasy System
   std::string getName();  // Akcesor pola name
   bool authentication(User * user, std::string login, std::string password);  // Metoda autoryzująca użytkownika
   void login(User * user);  // Metoda logująca użytkownika
   void registerUser(User * user);  // Metoda rejestrująca użytkownika
   void deleteUser(std::string login);  // Metoda usuwająca użytkownika z wektora users
   void addCourse(Course * course);  // Metoda dodająca kurs do wektora kursów
   int getCoursesSize();  // Akcesor rozmiaru wektora kursów
   void printCourses();  // Metoda wyświetlająca kursy
   void deleteCourse(std::string courseName);  // Metoda usuwająca kurs z systemu
   void boot();  // Metoda uruchamiająca system
   void studentSystem();  // Metoda uruchamiająca system w wersji studenckiej
   void lecturerSystem();  // Metoda uruchamiająca system w werjsi wykładowcy
   void studentCourseMenu(Student * student);  // Metoda otwierająca menu kursów dla studenta
   void lecturerCourseMenu(Lecturer * lecturer);  // Metoda otwierająca menu kursów dla wykładowcy
   void loadUsersFromCSV(const std::string& fileName);  // Metoda wczytująca użytkowników z pliku CSV
   void saveUsersToCSV(const std::string& fileName);  // Metoda zapisująca użytkowników do pliku CSV
   void loadCoursesFromCSV(const std::string& fileName);  // Metoda wczytująca kursy z pliku CSV
   void saveCoursesToCSV(const std::string& fileName);  // Metoda zapisująca kursy do pliku CSV
private:
   std::string name;  // Nazwa systemu
   std::vector <Course *> courses;  // Wektor kursów
   std::vector <User *> users;  // Wektor użytkowników
};

#endif  //_SYSTEM_H
```

- System.cpp

```cpp
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : System Obsługi Studiów
//  @ File Name : System.cpp
//  @ Date : 10.06.2023
//  @ Author : Tomasz Wnuk, Bartosz Szynkaruk, Mikołaj Hasiec
//
//

// Deklaracja zależności i bibliotek
#include <iostream>
#include <fstream>
#include <sstream>
#include "System.h"

// Logo systemu
const std::string logo = R"(
  ____         _                ____  _       _          ____ _           _ _
 / ___|_   _ __| |_ ___ _ __ ___ / ___\| |__ __| |_ _   _ / ___| | ___ _____ (_)_____ __ __
 \___ \ || (_-<  _/ -_) ' \  | (_) | '_ (_-< | || / _` | | \___ \  _| || / _` | / _ \ V  V /
 |____/\_, /__/\__\___|_|_|_|  \___/|_.__/__/_|\_,_\__, |_| |____/\__|\_,_\__,_|_\___/\_/\_/
       |__/                                         |___/
)";

// Ścieżka do pliku csv z kursami
const std::string coursesFilePath = "courses.csv";

// Ścieżka do pliku csv z użytkownikami
const std::string usersFilePath = "users.csv";

// Konstruktor klasy System
System::System(std::string name) {
    this->name = name;
}

// Akcesor nazwy systemu
std::string System::getName() {
    return name;
}

// Metoda autoryzująca użytkownika
bool System::authentication(User * user, std::string login, std::string password) {
    // Dla każdego użytkownika w wektorze users
    for(int i = 0; i < users.size(); i++) {
        // Jeżeli login i hasło użytkownika zgadzają się ze znalezionym użytkownikiem
        if(users[i]->getLogin() == login && users[i]->getPassword() == password) {
            // Ustaw dane użytkownika
            user->setLogin(users[i]->getLogin());  // Ustaw login użytkownika
            user->setPassword(users[i]->getPassword());  // Ustaw hasło użytkownika
            user->setFirstName(users[i]->getFirstName());  // Ustaw imię użytkownika
            user->setLastName(users[i]->getLastName());  // Ustaw nazwisko użytkownika
            user->setEmail(users[i]->getEmail());  // Ustaw email użytkownika

            // Zwróć prawdę
            return true;
        }
    }
    // Jeżeli nie znaleziono użytkownika, zwróć fałsz
```

```
    return false;
}

// Metoda logująca użytkownika
void System::login(User * user) {
    // Zadeklaruj zmienne przechowujące login i hasło użytkownika
    std::string loginUzytkownika;
    std::string hasloUzytkownika;

    // Wyświetl nagłówek logowania
    std::cout <<
"[=============================================================================================
====]\n";
    std::cout << "|
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Logowanie~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\
n";
    std::cout <<
"[=============================================================================================
====]\n";

    // Zaloguj użytkownika
    std::cout << "| Login: ";         // Wyświetl informacje o podaniu loginu
    std::cin >> loginUzytkownika;   // Pobierz login od użytkownika

    std::cout << "| Haslo: ";         // Wyświetl informacje o podaniu hasła
    std::cin >> hasloUzytkownika;   // Pobierz hasło od użytkownika

    // Jeżeli autoryzacja użytkownika przebiegła pomyślnie
    if(authentication(user, loginUzytkownika, hasloUzytkownika)) {
        // Wyświetl nagłówek o zalogowaniu
        std::cout <<
"[=============================================================================================
====]\n";
        std::cout << "|
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Zalogowano~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\
n";
        std::cout <<
"[=============================================================================================
====]\n";
        // Zakończ działanie metody logowania
        return;
    // W przeciwnym wypadku
    } else {
        // Wyświetl informacje o niepoprawnym loginie lub haśle
        std::cout << "Niepoprawny login lub haslo!" << "\n";
        // Wywołaj metodę rejestracji użytkownika
        registerUser(user);
    }
}

// Metoda rejestrująca użytkownika
void System::registerUser(User * user) {
    // Utwórz zmienną przechowującą wybór użytkownika
    std::string userInput;

    // Wyświetl nagłówek rejestracji
    std::cout <<
"[=============================================================================================
====]\n";
    std::cout << "|
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Rejestracja~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\n";
    std::cout <<
"[=============================================================================================
```

```
====]\n";

    // Zarejestruj użytkownika
    std::cout << "| Login: ";          // Wyświetl informacje o podaniu loginu
    std::cin >> userInput;          // Pobierz login od użytkownika
    user->setLogin(userInput);      // Ustaw login użytkownika

    std::cout << "| Haslo: ";          // Wyświetl informacje o podaniu hasła
    std::cin >> userInput;          // Pobierz hasło od użytkownika
    user->setPassword(userInput);   // Ustaw hasło użytkownika

    std::cout << "| Imie: ";          // Wyświetl informacje o podaniu imienia
    std::cin >> userInput;          // Pobierz imię od użytkownika
    user->setFirstName(userInput);  // Ustaw imię użytkownika

    std::cout << "| Nazwisko: ";      // Wyświetl informacje o podaniu nazwiska
    std::cin >> userInput;          // Pobierz nazwisko od użytkownika
    user->setLastName(userInput);   // Ustaw nazwisko użytkownika

    std::cout << "| Email: ";         // Wyświetl informacje o podaniu emaila
    std::cin >> userInput;          // Pobierz email od użytkownika
    user->setEmail(userInput);      // Ustaw email użytkownika

    // Dodaj użytkownika do wektora użytkowników
    users.push_back(user);
    // Zapisz użytkownika do pliku użytkowników w formacie CSV
    saveUsersToCSV(usersFilePath);
}

// Metoda usuwająca użytkownika z systemu
void System::deleteUser(std::string login) {
    // Przeszukaj wektor użytkowników
    for(int i = 0; i < users.size(); i++) {
        // Jeżeli login użytkownika jest równy loginowi podanemu jako argument
        if(users[i]->getLogin() == login) {
            // Usuń użytkownika z wektora użytkowników
            users.erase(users.begin() + i);
        }
    }
}

// Metoda dodająca kurs do wektora kursów
void System::addCourse(Course * course) {
    // Dodaj kurs do wektora kursów
    courses.push_back(course);
}

// Akcesor rozmiaru wektora kursów
int System::getCoursesSize() {
    // Zwróć rozmiar wektora kursów
    return courses.size();
}

// Metoda wyświetlająca kursy
void System::printCourses() {
    // Wyświetl nazwy kursów
    std::cout <<
"[==========================================================================
====]\n";
    std::cout << "|
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Kursy~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\
n";
    std::cout <<
```

```
"[=============================================================================
====]\n";

    // Dla każdego kursu w wektorze kursów
    for(int i = 0; i < courses.size(); i++) {
        // Wyświetl numer kursu i nazwę kursu
        std::cout << "| " << i + 1 << ". " << courses[i]->getName() << "\n";
    }
}

// Metoda usuwająca kurs z systemu
void System::deleteCourse(std::string courseName) {
    // Przeszuakj wektor kursów
    for(int i = 0; i < courses.size(); i++) {
        // Jeżeli nazwa kursu jest równa nazwie kursu podanej jako argument
        if(courses[i]->getName() == courseName) {
            // Usuń kurs z wektora kursów
            courses.erase(courses.begin() + i);
        }
    }
}

// Metoda wczytująca użytkowników z pliku CSV
void System::loadUsersFromCSV(const std::string& fileName) {
    // Utwórz zmienne do odczytu danych z pliku
    std::ifstream file(fileName);   // Utwórz strumień plikowy
    std::string line;               // Utwórz zmienną przechowującą linię z pliku

    // Jeżeli nie udało się otworzyć pliku
    if(!file.is_open()) {
        // Wyświetl informacje o błędzie
        std::cout << "Blad podczas otwierania pliku: " << fileName << std::endl;
        // Zakończ działanie metody
        return;
    }

    // Dla każdej linii w pliku
    while(std::getline(file, line)) {
        // Utwórz zmienne przechowujące dane użytkownika
        std::string imie, nazwisko, login, haslo, email;
        // Pobierz linię z pliku
        std::stringstream ss(line);
        // Jeżeli udało się pobrać dane użytkownika z podanym formatowaniem
        if((std::getline(ss, imie, ' ') &&
            std::getline(ss, nazwisko, ',') &&
            std::getline(ss, login, ',')) &&
            std::getline(ss, haslo, ',') &&
            std::getline(ss, email, ',')) {
                // Dodaj użytkownika do wektora użytkowników
                users.push_back(new User(imie, nazwisko, login, haslo, email));
        }
    }
    // Zamknij plik
    file.close();
}

// Metoda zapisująca użytkowników do pliku CSV
void System::saveUsersToCSV(const std::string& fileName) {
    // Otwórz plik do zapisu
    std::ofstream file(fileName);

    // Jeżeli nie udało się otworzyć pliku
    if (!file.is_open()) {
```

```
        // Wyświetl informacje o błędzie
        std::cout << "Blad podczas otwierania pliku: " << fileName << std::endl;
        // Zakończ działanie metody
        return;
    }

    // Dla każdego użytkownika w wektorze użytkowników
    for(const auto& user : users) {
        // Zapisz dane użytkownika do pliku w formacie CSV
        file << user->getFirstName() << " " << user->getLastName() << "," << user->getLogin() << "," << user->getPassword() << "," << user->getEmail() << ",\n";
    }

    // Zamknij plik
    file.close();
}

// Metoda wczytująca kursy z pliku CSV
void System::loadCoursesFromCSV(const std::string& fileName) {
    // Utwórz zmienne do odczytu danych z pliku
    std::vector<std::string> participants;  // Utwórz wektor uczestników kursu
    std::ifstream file(fileName);  // Utwórz strumień plikowy
    Course * newCourse;  // Utwórz wskaźnik na nowy kurs
    std::string line;  // Utwórz zmienną przechowującą linię z pliku

    // Jeżeli nie udało się otworzyć pliku
    if(!file.is_open()) {
        // Wyświetl informacje o błędzie
        std::cout << "Blad podczas otwierania pliku: " << fileName << std::endl;
        // Zakończ działanie metody
        return;
    }

    // Dla każdej linii w pliku
    while(std::getline(file, line)) {
        // Utwórz zmienne przechowujące dane kursu
        std::string courseName, lecturerFirstName, lecturerLastName, isVideoConferenceCreated;
        // Pobierz linię z pliku
        std::stringstream ss(line);
        // Jeżeli udało się pobrać dane kursu z podanym formatowaniem
        if(std::getline(ss, courseName, ',') && std::getline(ss, lecturerFirstName, ' ') &&
           std::getline(ss, lecturerLastName, ',') && std::getline(ss, isVideoConferenceCreated, ',')) {
            // Utwórz nowy kurs
            newCourse = new Course(courseName, new Lecturer(lecturerFirstName, lecturerLastName), isVideoConferenceCreated);
            // Dodaj kurs do wektora kursów
            courses.push_back(newCourse);

            // Odczytaj uczestników kursu
            std::string participant;
            while(std::getline(ss, participant, ',')) {
                // Usuń początkową i końcową spację z imienia i nazwiska uczestnika
                participant = participant.substr(0, participant.length());

                // Podziel imię i nazwisko uczestnika
                std::istringstream participantISS(participant);
                std::string participantFirstName, participantLastName;
                // Jeżeli udało się pobrać imię i nazwisko uczestnika
                if(std::getline(participantISS, participantFirstName, ' ') &&
                   std::getline(participantISS, participantLastName, ',')) {
                    // Dodaj uczestnika do wektora uczestników kursu
                    newCourse->addStudent(new Student(participantFirstName, participantLastName));
                }
```

```
            }
        }
    }
    // Zamknij plik
    file.close();
}

// Metoda zapisująca kursy do pliku CSV
void System::saveCoursesToCSV(const std::string& fileName) {
    // Otwórz plik do zapisu
    std::ofstream file(fileName);

    // Jeżeli nie udało się otworzyć pliku
    if(!file.is_open()) {
        // Wyświetl informacje o błędzie
        std::cout << "Blad podczas otwierania pliku: " << fileName << std::endl;
        // Zakończ działanie metody
        return;
    }

    // Dla każdego kursu w wektorze kursów
    for(int i = 0; i < courses.size(); i++) {
        // Zapisz dane kursu do pliku w formacie CSV
        file << courses[i]->getName() << "," << courses[i]->getLecturer()->getFirstName() << " " << courses[i]-
>getLecturer()->getLastName() << "," << 0 << ",";
        for(int j = 0; j < courses[i]->getStudentsSize(); j++) {
            // Zapisz dane uczestników kursu do pliku w formacie CSV
            file << courses[i]->getStudentAtIndex(j)->getFirstName() << " " << courses[i]->getStudentAtIndex(j)-
>getLastName() << ",";
        }
        // Dodaj znak nowej linii do pliku CSV
        file << "\n";
    }

    // Zamknij plik
    file.close();
}

// Metoda uruchamiająca system
void System::boot() {
    // Wczytaj dane z plików CSV
    loadUsersFromCSV(usersFilePath);  // Wczytaj użytkowników
    loadCoursesFromCSV(coursesFilePath);  // Wczytaj kursy

    // Wyświetl logo systemu
    std::cout <<
"================================================================================================
========\n";
    std::cout <<
"================================================================================================
========\n";
    std::cout <<
"================================================================================================
========\n";
    std::cout << logo << "\n";
    std::cout <<
"================================================================================================
========\n";
    std::cout <<
"================================================================================================
========\n";
    std::cout <<
"================================================================================================
```

```
========\n";
}

// Metoda uruchamiająca system w wersji studenckiej
void System::studentSystem() {
    // Utwórz zmienną przechowującą dane użytkownika
    std::string userInput;

    // Utwórz nowego studenta
    Student * student = new Student();

    // Wywołaj metodę logowania
    login(student);

    // Pętla do wyboru opcji systemu studenta
    while(true) {
        // Wyświetl stronę główną systemu studenta
        std::cout <<
"[===========================================================================
====]\n";
        std::cout << "|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Strona
glowna~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\n";
        std::cout << "|
===========================================================================
===|\n";
        std::cout << "| 1. Kursy                                                              |\n";
        std::cout << "| 2. Wyswietl swoj profil                                               |\n";
        std::cout << "| 3. Wyloguj sie                                                        |\n";
        std::cout <<
"[===========================================================================
====]\n";

        // Pobierz wybór użytkownika
        std::cin >> userInput;

        // Wykonaj akcję w zależności od wyboru użytkownika
        // Jeżeli użytkownik wybrał opcję 1
        if(userInput == "1") {
            // Wywołaj metodę menu kursów studenta
            studentCourseMenu(student);
        // Jeżeli użytkownik wybrał opcję 2
        } else if(userInput == "2") {
            // Wyświetl profil użytkownika
            student->printUser();
        // Jeżeli użytkownik wybrał opcję 3
        } else if(userInput == "3") {
            // Zakończ działanie programu
            exit(0);
        // W przeciwnym wypadku
        } else {
            // Wyświetl komunikat o niepoprawnym wyborze
            std::cout << "Niepoprawna opcja!\n";
        }
    }
}

// Metoda uruchamiająca system w werjsi wykładowcy
void System::lecturerSystem() {
    // Utwórz zmienną przechowującą dane użytkownika
    std::string userInput;

    // Utwórz nowego wykładowcę
    Lecturer * lecturer = new Lecturer();
```

```
    // Wywołaj metodę logowania
    login(lecturer);

    // Pętla do wyboru opcji systemu wykładowcy
    while(true) {
        // Wyświetl stronę główną systemu wykładowcy
        std::cout <<
"[======================================================================================
====]\n";
        std::cout << "|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Strona
glowna~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\n";
        std::cout << "|
=======================================================================================
===|\n";
        std::cout << "| 1. Kursy                                                                              |\n";
        std::cout << "| 2. Wyswietl swoj profil                                                               |\n";
        std::cout << "| 3. Wyloguj sie                                                                        |\n";
        std::cout <<
"[======================================================================================
====]\n";

        // Pobierz wybór użytkownika
        std::cin >> userInput;

        // Wykonaj akcję w zależności od wyboru użytkownika
        // Jeżeli użytkownik wybrał opcję 1
        if(userInput == "1") {
            // Wywołaj metodę menu kursów studenta
            lecturerCourseMenu(lecturer);
        // Jeżeli użytkownik wybrał opcję 2
        } else if(userInput == "2") {
            // Wyświetl profil użytkownika
            lecturer->printUser();
        // Jeżeli użytkownik wybrał opcję 3
        } else if(userInput == "3") {
            // Zakończ działanie programu
            exit(0);
        // W przeciwnym wypadku
        } else {
            // Wyświetl komunikat o niepoprawnym wyborze
            std::cout << "Niepoprawna opcja!" << "\n";
        }
    }
}

// Metoda otwierająca menu kursów dla studenta
void System::studentCourseMenu(Student * student) {
    // Utwórz zmienną przechowującą dane użytkownika
    std::string userInput;

    // Pętla do wyboru opcji systemu studenta
    while(true) {
        // Wyświetl menu kursów studenta
        std::cout <<
"[======================================================================================
====]\n";
        std::cout << "|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Menu
Kursow~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\n";
        std::cout << "|
=======================================================================================
===|\n";
        std::cout << "| 1. Wybierz ze swoich kursow                                                           |\n";
```

```
        std::cout << "| 2. Zapisz sie do kursu                                    |\n";
        std::cout << "| 3. Wypisz sie z kursu                                     |\n";
        std::cout << "| 4. Wyswietl swoje kursy                                     |\n";
        std::cout << "| 5. Wyswietl wszystkie kursy                                   |\n";
        std::cout << "| 6. Wroc                                                |\n";
        std::cout <<
"[=============================================================================
====]\n";

        // Pobierz wybór użytkownika
        std::cin >> userInput;

        // Wykonaj akcję w zależności od wyboru użytkownika
        // Jeżeli użytkownik wybrał opcję 1
        if(userInput == "1") {
            // Wyświetl kursy studenta
            student->printCourses();
            // Wyświetl opcję powrotu
            std::cout << "| " << student->getCoursesSize() + 1 << ". Wroc" << "\n";
            std::cout <<
"[=============================================================================
====]\n";

            std::cout << "Wybierz kurs: ";  // Wyświetl komunikat o wyborze kursu
            std::cin >> userInput;  // Pobierz wybór użytkownika

            // Jeżeli wybrany kurs istnieje
            if(std::stoi(userInput) - 1 < student->getCoursesSize()) {
                // Wywołaj menu kursu studenta o podanym indeksie
                student->getCourseAtIndex(std::stoi(userInput) - 1)->openStudentCourseMenu(student);
            // W przeciwnym wypadku
            } else if(std::stoi(userInput) - 1  == student->getCoursesSize()) {
                // Wyświetl komunikat o opuszczeniu menu kursów
                std::cout << "Wyszedles z menu kursow!" << "\n";
            // W przeciwnym wypadku
            } else {
                // Wyświetl komunikat o niepoprawnym wyborze
                std::cout << "Podany kurs nie istnieje!" << "\n";
            }
        // Jeżeli użytkownik wybrał opcję 2
        } else if(userInput == "2") {
            // Wyświetl wszystkie kursy
            printCourses();
            // Wyświetl opcję powrotu
            std::cout << "| " << getCoursesSize() + 1 << ". Wroc" << "\n";
            std::cout <<
"[=============================================================================
====]\n";

            // Wyświetl komunikat o wyborze kursu
            std::cout << "Wybierz kurs: ";
            // Pobierz wybór użytkownika
            std::cin >> userInput;

            // Jeżeli wybrany kurs istnieje
            if(std::stoi(userInput) - 1 < getCoursesSize()) {
                // Przejdź przez wszystkie kursy studenta
                for(int i = 0; i < student->getCoursesSize(); i++) {
                    // Jeżeli student jest już zapisany na dany kurs
                    if(student->getCourseAtIndex(i)->getName() == courses[std::stoi(userInput) - 1]->getName()) {
                        // Wyświetl komunikat o tym, że student jest już zapisany na ten kurs
                        std::cout << "Juz jestes zapisany na ten kurs!" << "\n";
                        // Zakończ pętlę
```

```
                    return;
                }
            }
            // Dodaj kurs do kursów studenta
            student->addCourse(courses[std::stoi(userInput) - 1]);
            // Dodaj studenta do uczestników kursu
            courses[std::stoi(userInput) - 1]->addStudent(student);
            // Zapisz studenta do pliku kursów w formacie CSV
            saveCoursesToCSV(coursesFilePath);
        // Jeżeli użytkownik wybrał opcję powrotu
        } else if(std::stoi(userInput) - 1 == getCoursesSize()) {
            // Wyświetl komunikat o opuszczeniu menu kursów
            std::cout << "Wyszedles z menu kursow!" << "\n";
        // W przeciwnym wypadku
        } else {
            // Wyświetl komunikat o niepoprawnym wyborze
            std::cout << "Podany kurs nie istnieje!" << "\n";
        }
    // Jeżeli użytkownik wybrał opcję 3
    } else if(userInput == "3") {
        // Utwórz zmienną przechowującą nazwe kursu do usunięcia
        std::string courseToDeleteName = "";

        // Wyświetl kursy studenta
        student->printCourses();
        // Wyświetl opcję powrotu
        std::cout << "| " << student->getCoursesSize() + 1 << ". Wroc" << "\n";
        std::cout <<
"[=========================================================================================
====]\n";

        // Wyświetl komunikat o wyborze kursu
        std::cout << "Wybierz kurs: ";
        // Pobierz wybór użytkownika
        std::cin >> userInput;

        // Jeżeli wybrany kurs istnieje
        if(std::stoi(userInput) - 1 < student->getCoursesSize()) {
            // Przejdz przez wszystkie kursy wykładowcy
            for(int i = 0; i < student->getCoursesSize(); i++) {
                // Jeżeli nazwa kursu o podanym indeksie jest taka sama jak nazwa kursu o podanym indeksie
                if(student->getCourseAtIndex(i)->getName() == student->getCourseAtIndex(std::stoi(userInput) - 1)-
>getName()) {
                    // Ustaw zmienną przechowującą nazwe kursu do usunięcia na nazwę kursu o podanym indeksie
                    std::string courseToDeleteName = student->getCourseAtIndex(i)->getName();
                    // Przejdz przez wszystkie kursy systemu
                    for(int j = 0; j < courses.size(); j++) {
                        // Jeżeli nazwa kursu o podanym indeksie jest taka sama jak nazwa kursu do usunięcia
                        if(courses[j]->getName() == courseToDeleteName) {
                            // Usuń studenta z listy uczestników kursu
                            courses[j]->removeStudent(student->getLogin());
                            // Usuń kurs z listy kursów studenta
                            student->removeCourse(courseToDeleteName);
                            // Wyświetl komunikat wyjścia z kursu
                            std::cout << "Wyszedles z kursu " << courses[j]->getName() << "!" << "\n";
                        }
                    }
                }
            }
        // Jeżeli użytkownik wybrał opcję powrotu
        } else if(std::stoi(userInput) - 1 == student->getCoursesSize()) {
            // Wyświetl komunikat o opuszczeniu menu kursów
            std::cout << "Wyszedles z menu kursow!" << "\n";
```

```
            } else {
                // Wyświetl komunikat o niepoprawnym wyborze
                std::cout << "Podany kurs nie istnieje!" << "\n";
            }
        // Jeżeli użytkownik wybrał opcję 4
        } else if(userInput == "4") {
            // Wyświetl kursy studenta
            student->printCourses();
        // Jeżeli użytkownik wybrał opcję 5
        } else if(userInput == "5") {
            // Wyświetl wszystkie kursy
            printCourses();
        // Jeżeli użytkownik wybrał opcję 6
        } else if(userInput == "6") {
            // Wyświetl komunikat o opuszczeniu menu kursów
            std::cout << "Wyszedles z menu kursow!" << "\n";
            // Zakończ pętlę
            break;
        // W przeciwnym wypadku
        } else {
            // Wyświetl komunikat o niepoprawnym wyborze
            std::cout << "Niepoprawna opcja!" << "\n";
        }
    }
}

// Metoda otwierająca menu kursów dla wykładowcy
void System::lecturerCourseMenu(Lecturer * lecturer) {
    // Utwórz zmienną przechowującą dane użytkownika
    std::string userInput;

    // Pętla do wyboru opcji systemu wykładowcy
    while(true) {
        // Wyświetl menu kursów wykładowcy
        std::cout <<
"[===================================================================================
====]\n";
        std::cout << "|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Menu
Kursow~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\n";
        std::cout << "|
====================================================================================
===|\n";
        std::cout << "| 1. Wybierz ze swoich kursow                                        |\n";
        std::cout << "| 2. Utworz kurs                                                     |\n";
        std::cout << "| 3. Zamknij kurs                                                    |\n";
        std::cout << "| 4. Wyswietl swoje kursy                                            |\n";
        std::cout << "| 5. Wyswietl wszystkie kursy                                        |\n";
        std::cout << "| 6. Wroc                                                            |\n";
        std::cout <<
"[===================================================================================
====]\n";

        // Pobierz wybór użytkownika
        std::cin >> userInput;

        // Wykonaj akcję w zależności od wyboru użytkownika
        // Jeżeli użytkownik wybrał opcję 1
        if(userInput == "1") {
            // Wyświetl kursy wykładowcy
            lecturer->printCourses();
            // Wyświetl opcję powrotu
            std::cout << "| " << lecturer->getCoursesSize() + 1 << ". Wroc" << "\n";
            std::cout <<
```

```
"[==========================================================================================]\n";

        std::cout << "Wybierz kurs: ";  // Wyświetl komunikat o wyborze kursu
        std::cin >> userInput;  // Pobierz wybór użytkownika

        // Jeżeli wybrany kurs istnieje
        if(std::stoi(userInput) - 1 < lecturer->getCoursesSize()) {
            // Wywołaj menu kursu wykładowcy o podanym indeksie
            lecturer->getCourseAtIndex(std::stoi(userInput) - 1)->openLecturerCourseMenu(lecturer);
        // Jeżeli użytkownik wybrał opcję powrotu
        } else if(std::stoi(userInput) - 1 == lecturer->getCoursesSize()) {
            // Wyświetl komunikat o opuszczeniu menu kursów
            std::cout << "Wyszedles z menu kursow!" << "\n";
        // W przeciwnym wypadku
        } else {
            // Wyświetl komunikat o niepoprawnym wyborze
            std::cout << "Podany kurs nie istnieje!" << "\n";
        }
    // Jeżeli użytkownik wybrał opcję 2
    } else if(userInput == "2") {
        // Utwórz zmienną przechowującą linie tekstu
        std::string inputLine;
        // Zresetuj zmienną przechowującą dane użytkownika
        userInput = "";

        // Wyświetl komunikat o podaniu nazwy kursu
        std::cout << "Podaj nazwe kursu: ";
        // Pobierz nazwę kursu
        while(std::getline(std::cin, inputLine)) {
            // Dodaj linię tekstu do zmiennej przechowującej dane użytkownika
            userInput += inputLine;
            if(inputLine.find(' ') != std::string::npos) {
                // Jeżeli w linii tekstu znajduje się spacja, zakończ pobieranie
                break;
            }
        }

        // Przejdź przez wszystkie kursy
        for(int i = 0; i < courses.size(); i++) {
            // Jeżeli nazwa kursu jest taka sama jak nazwa kursu do utworzenia
            if(courses[i]->getName() == userInput) {
                // Wyświetl komunikat o istnieniu kursu
                std::cout << "Kurs o podanej nazwie juz istnieje!" << "\n";
                // Wyjdź z pętli
                return;
            }
        }

        // Utwórz nowy kurs o podanej nazwie i wykładowcy
        Course * newCourse = new Course(userInput, lecturer);
        // Dodaj kurs do listy kursów systemu
        addCourse(newCourse);
        // Dodaj kurs do listy kursów wykładowcy
        lecturer->addCourse(newCourse);
        // Zapisz kurs do pliku kursów w formacie CSV
        saveCoursesToCSV(coursesFilePath);
        // Wyświetl komunikat o utworzeniu kursu
        std::cout << "Kurs zostal utworzony!" << "\n";

    // Jeżeli użytkownik wybrał opcję 3
    } else if(userInput == "3") {
        // Utwórz zmienną przechowującą nazwe kursu do usunięcia
```

```
            std::string courseToDeleteName = "";

            // Wyświetl kursy wykładowcy
            lecturer->printCourses();
            // Wyświetl opcję powrotu
            std::cout << "| " << lecturer->getCoursesSize() + 1 << ". Wroc" << "\n";
            std::cout <<
"[=============================================================================================
====]\n";

            // Wyświetl komunikat o wyborze kursu
            std::cout << "Wybierz kurs: ";
            // Pobierz wybór użytkownika
            std::cin >> userInput;

            // Jeżeli wybrany kurs istnieje
            if(std::stoi(userInput) - 1 < lecturer->getCoursesSize()) {
                // Przejdz przez wszystkie kursy wykładowcy
                for(int i = 0; i < lecturer->getCoursesSize(); i++) {
                    // Jeżeli nazwa kursu o podanym indeksie jest taka sama jak nazwa kursu o podanym indeksie
                    if(lecturer->getCourseAtIndex(i)->getName() == lecturer->getCourseAtIndex(std::stoi(userInput) - 1)-
>getName()) {
                        // Ustaw zmienną przechowującą nazwe kursu do usunięcia na nazwę kursu o podanym indeksie
                        std::string courseToDeleteName = lecturer->getCourseAtIndex(i)->getName();
                        // Przejdz przez wszystkie kursy systemu
                        for(int j = 0; j < courses.size(); j++) {
                            // Jeżeli nazwa kursu o podanym indeksie jest taka sama jak nazwa kursu do usunięcia
                            if(courses[j]->getName() == courseToDeleteName) {
                                // Usuń kurs z listy kursów systemu
                                courses.erase(courses.begin() + j);
                                // Zapisz kursy do pliku kursów w formacie CSV
                                saveCoursesToCSV(coursesFilePath);
                                // Usuń kurs z listy kursów wykładowcy
                                lecturer->removeCourse(courseToDeleteName);
                                // Wyświetl komunikat o usunięciu kursu
                                std::cout << "Kurs " << courseToDeleteName << " zostal zamkniety!" << "\n";
                            }
                        }
                    }
                }
            // Jeżeli użytkownik wybrał opcję powrotu
            } else if(std::stoi(userInput) - 1 == lecturer->getCoursesSize()) {
                // Wyświetl komunikat o opuszczeniu menu kursów
                std::cout << "Wyszedles z menu kursow!" << "\n";
            } else {
                // Wyświetl komunikat o niepoprawnym wyborze
                std::cout << "Podany kurs nie istnieje!" << "\n";
            }
        // Jeżeli użytkownik wybrał opcję 4
        } else if(userInput == "4") {
            // Wyświetl kursy wykładowcy
            lecturer->printCourses();
        // Jeżeli użytkownik wybrał opcję 5
        } else if(userInput == "5") {
            // Wyświetl wszystkie kursy
            printCourses();
        // Jeżeli użytkownik wybrał opcję 6
        } else if(userInput == "6") {
            // Wyświetl komunikat o opuszczeniu menu kursów
            std::cout << "Wyszedles z menu kursow!" << "\n";
            // Wyjdź z pętli
            break;
        // W przeciwnym wypadku
```

```
        } else {
            // Wyświetl komunikat o niepoprawnym wyborze
            std::cout << "Niepoprawna opcja!" << "\n";
        }
    }
}
```

- Course.h

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : System Obsługi Studiów
//  @ File Name : Course.h
//  @ Date : 10.06.2023
//  @ Author : Tomasz Wnuk, Bartosz Szynkaruk, Mikołaj Hasiec
//
//

#if !defined(_COURSE_H)
#define _COURSE_H

// Deklaracja zależności i bibliotek
#include <string>
#include <vector>
#include "Student.h"
#include "Lecturer.h"
#include "Material.h"
#include "VideoConference.h"

// Deklaracja klas
class Lecturer;
class Student;
class VideoConference;

// Deklaracja klasy Course
class Course {
public:
   Course(std::string name, Lecturer * lecturer);  // Konstruktor klasy Course przyjmujący nazwę kursu i wykładowcę prowadzącego kurs
   Course(std::string name, Lecturer * lecturer, std::string isVideoConferenceCreated);  // Konstruktor klasy Course przyjmujący nazwę kursu, wykładowcę prowadzącego kurs i informację o tym, czy wideokonferencja została utworzona
   std::string getName();  // Akcesor pola name
   void setName(std::string name);  // Mutator pola name
   Lecturer * getLecturer();  // Akcesor pola lecturer
   void setLecturer(Lecturer * lecturer);  // Mutator pola lecturer
   void addStudent(Student * student);  // Metoda dodająca studenta do wektora kursu
   void removeStudent(std::string login);  // Metoda usuwająca studenta z wektora kursu
   int getStudentsSize();  // Akcesor rozmiaru wektora studentów kursu
   Student * getStudentAtIndex(int studentIndex);  // Metoda zwracająca studenta z kursu o podanym indeksie
   void addMaterial(Material * material);  // Metoda dodająca materiał do wektora kursu
   void deleteMaterial(std::string name);  // Metoda usuwająca materiał z wektora kursu
   void createVideoConference(std::string name);  // Metoda tworząca wideokonferencję
   void endVideoConference();  // Metoda kończąca wideokonferencję
   void displayHeader(const std::string& headerName);  // Metoda wyświetlająca nagłówek
   void viewCoursePage();  // Metoda wyświetlająca stronę kursu
   void openStudentCourseMenu(Student * student);  // Metoda otwierająca menu kursu dla studenta
   void openLecturerCourseMenu(Lecturer * lecturer);  // Metoda otwierająca menu kursu dla wykładowcy
   void openMaterialsMenu(User * user);  // Metoda otwierająca menu materiałów
private:
   std::string name;  // Nazwa kursu
   Lecturer * lecturer;  // Wykładowca prowadzący kurs
   std::vector <Student *> students;  // Wektor studentów uczęszczających na kurs
   std::vector <Material *> materials;  // Wektor materiałów przypisanych do kursu
   VideoConference * videoConference;  // Wideokonferencja
};
```

```
#endif  //_COURSE_H
```

- Course.cpp

```cpp
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : System Obsługi Studiów
//  @ File Name : Course.cpp
//  @ Date : 10.06.2023
//  @ Author : Tomasz Wnuk, Bartosz Szynkaruk, Mikołaj Hasiec
//
//

// Deklaracja zależności i bibliotek
#include <iostream>
#include "Course.h"

// Konstruktor klasy Course przyjmujący nazwę kursu i wykładowcę prowadzącego kurs
Course::Course(std::string name, Lecturer * lecturer) {
    this->name = name;
    this->lecturer = lecturer;
}

// Konstruktor klasy Course przyjmujący nazwę kursu, wykładowcę prowadzącego kurs i informację o tym, czy wideokonferencja została utworzona
Course::Course(std::string name, Lecturer * lecturer, std::string isVideoConferenceCreated) {
    this->name = name;
    this->lecturer = lecturer;
    // Jeżeli isVideoConferenceCreated jest podana jako utworzona
    if(isVideoConferenceCreated == "1") {
        // Utwórz wideokonferencję
        createVideoConference(name);
    }
}

// Akcesor pola name
std::string Course::getName() {
    return name;
}

// Mutator pola name
void Course::setName(std::string name) {
    this->name = name;
}

// Akcesor pola lecturer
Lecturer * Course::getLecturer() {
    return lecturer;
}

// Mutator pola lecturer
void Course::setLecturer(Lecturer * lecturer) {
    this->lecturer = lecturer;
}

// Metoda dodająca studenta do wektora kursu
void Course::addStudent(Student * student) {
    // Dodanie studenta do wektora students
    students.push_back(student);
}

// Metoda usuwająca studenta z wektora kursu
```

```cpp
void Course::removeStudent(std::string login) {
    // Przeszukaj wektor students
    for(int i = 0; i < students.size(); i++) {
        // Jeżeli login studenta jest równy loginowi podanemu w argumencie
        if(students[i]->getLogin() == login) {
            // Usuń studenta z wektora
            students.erase(students.begin() + i);
        }
    }
}

// Akcesor rozmiaru wektora studentów kursu
int Course::getStudentsSize(){
    // Zwróć rozmiar wektora students
    return students.size();
}

// Metoda zwracająca studenta z kursu o podanym indeksie
Student * Course::getStudentAtIndex(int studentIndex) {
    // Zwróć studenta o podanym indeksie
    return students[studentIndex];
}

// Metoda dodająca materiał do wektora kursu
void Course::addMaterial(Material *material) {
    // Dodaj materiał do wektora materials
    materials.push_back(material);
}

// Metoda usuwająca materiał z wektora kursu
void Course::deleteMaterial(std::string name) {
    // Przeszukaj wektor materials
    for(int i = 0; i < materials.size(); i++) {
        // Jeżeli name materiału jest równa nazwie podanej w argumencie
        if(materials[i]->getName() == name) {
            // Usuń materiał z wektora
            materials.erase(materials.begin() + i);
        }
    }
}

// Metoda tworząca wideokonferencję
void Course::createVideoConference(std::string name) {
    // Utwórz wideokonferencję
    this->videoConference = new VideoConference(name, lecturer);
}

// Metoda kończąca wideokonferencję
void Course::endVideoConference() {
    // Wyświetl komunikat o zakończeniu wideokonferencji
                    std::cout << "Zakonczono wideokonferencje!" << "\n";
    // Usuń studentów z wideokonferencji
    for(int i = 0; i < students.size(); i++) {
        videoConference->removeStudent(students[i]->getLogin());
    }
    // Usuń wskaźnik na wideokonferencję
    videoConference = nullptr;
}

// Metoda wyświetlająca nagłówek
void Course::displayHeader(const std::string& headerName) {
    const int totalWidth = 90;  // Szerokość całego wyświetlanego napisu
    const int nameWidth = name.length();  // Szerokość nazwy kursu
```

```
    const int paddingWidth = (totalWidth - nameWidth) / 2;  // Szerokość wypełnienia
    // Wyświetl nagłówek
    std::cout <<
"[=============================================================================================
====]\n";
    std::cout << "|" << std::string(paddingWidth, '~') << headerName << std::string(paddingWidth - 1, '~') << "|\n";
    std::cout <<
"[=============================================================================================
====]\n";
}

// Metoda wyświetlająca stronę kursu
void Course::viewCoursePage() {
    // Wyświetl nagłówek strony głównej kursu
    displayHeader(getName());

    // Wyświetl wykładowcę
                std::cout << "| Wykladowca: " << lecturer->getFirstName() << " " << lecturer->getLastName() << "\n";

    // Wyświetl studentów
                std::cout << "| Uczestnicy: \n";
    // Przeszukaj wektor students
                for(int i = 0; i < students.size(); i++) {
        // Wyświetl numer studenta, imię i nazwisko
                                std::cout << "| " << i + 1 << ". " << students[i]->getFirstName() << " " << students[i]-
>getLastName() << "\n";
                }
}

// Metoda otwierająca menu kursu dla studenta
void Course::openStudentCourseMenu(Student * student) {
    // Utwórz zmienną przechowującą wybór użytkownika
    std::string userInput;

    // Pętla menu kursu
    while(true) {
        // Wyświetl nagłówek strony głównej kursu
        displayHeader(getName());

        // Wyświetl menu kursu dla studenta
        std::cout << "| 1. Wyswietl strone kursu                                    |\n";
        std::cout << "| 2. Otworz materialy                                        |\n";
        std::cout << "| 3. Dolacz do wideokonferencji                                  |\n";
        std::cout << "| 4. Wroc                                                |\n";
        std::cout <<
"[=============================================================================================
====]\n";

        // Pobierz wybór użytkownika
        std::cin >> userInput;

        // Wykonaj akcję w zależności od wyboru użytkownika
        // Jeżeli wybór jest równy 1
        if(userInput == "1") {
            // Wyświetl stronę kursu
            viewCoursePage();
        // Jeżeli wybór jest równy 2
        } else if(userInput == "2") {
            // Otwórz menu materiałów kursu
            openMaterialsMenu(student);
        // Jeżeli wybór jest równy 3
        } else if(userInput == "3") {
            // Jeżeli wideokonferencja istnieje
```

```
            if(videoConference != nullptr) {
                // Dodaj studenta do wideokonferencji
                videoConference->addStudent(student);
                // Otwórz menu wideokonferencji
                videoConference->openStudentVideoConferenceMenu(student);
            // W przeciwnym wypadku
            } else {
                // Wyświetl komunikat o braaku wideokonferencji
                std::cout << "Wideokonferencja nie jest utworzona!" << "\n";
            }
        // Jeżeli wybór jest równy 4
        } else if(userInput == "4") {
            // Wyświetl komunikat o opuszczeniu menu kursu
            std::cout << "Opusciles menu kursu " << getName() << "!" << "\n";
            // Wyjdź z pętli
            break;
        // W przeciwnym wypadku
        } else {
            // Wyświetl komunikat o niepoprawnym wyborze
            std::cout << "Niepoprawna opcja!" << "\n";
        }
    }
}

// Metoda otwierająca menu materiałów kursu dla wykładowcy
void Course::openLecturerCourseMenu(Lecturer * lecturer) {
    // Utwórz zmienną przechowującą wybór użytkownika
    std::string userInput;

    // Pętla menu kursu
    while(true) {
        // Wyświetl nagłówek strony głównej kursu
        displayHeader(getName());

        // Wyświetl menu kursu dla studenta
        std::cout << "| 1. Wyswietl strone kursu                                                    |\n";
        std::cout << "| 2. Otworz materialy                                                        |\n";
        std::cout << "| 3. Dolacz do wideokonferencji                                                  |\n";
        std::cout << "| 4. Wroc                                                                |\n";
        std::cout <<
"[=============================================================================================
====]\n";

        // Pobierz wybór użytkownika
        std::cin >> userInput;

        // Wykonaj akcję w zależności od wyboru użytkownika
        // Jeżeli wybór jest równy 1
        if(userInput == "1") {
            // Wyświetl stronę kursu
            viewCoursePage();
        // Jeżeli wybór to 2
        } else if(userInput == "2") {
            // Otwórz menu materiałów kursu
            openMaterialsMenu(lecturer);
        // Jeżeli wybór to 3
        } else if(userInput == "3") {
            // Jeżeli wideokonferencja istnieje
            if(videoConference == nullptr) {
                // Utwórz wideokonferencję
                createVideoConference(getName());
                // Otwórz menu wideokonferencji
                videoConference->openLecturerVideoConferenceMenu(lecturer);
```

```
            // W przeciwnym wypadku
            } else {
                // Zakończ wideokonferencję
                endVideoConference();
                // Utwórz nową wideokonferencję
                createVideoConference(getName());
                // Otwórz menu wideokonferencji
                videoConference->openLecturerVideoConferenceMenu(lecturer);
            }
            // Jeżeli wybór to 4
        } else if (userInput == "4") {
            // Wyświetl komunikat o opuszczeniu menu kursu
            std::cout << "Opusciles menu kursu " << getName() << "!" << "\n";
            // Wyjdź z pętli
            break;
            // W przeciwnym wypadku
        } else {
            // Wyświetl komunikat o niepoprawnym wyborze
            std::cout << "Niepoprawna opcja!" << "\n";
        }
    }
}

// Metoda otwierająca menu materiałów kursu
void Course::openMaterialsMenu(User * user) {
    // Utwórz zmienną przechowującą wybór użytkownika
    std::string userInput;

    // Pętla menu materiałów kursu
    while(true) {
        // Wyświetl nagłówek menu materiałów kursu
        std::cout <<
"[===================================================================================
====]\n";
        std::cout << "|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Materialy
kursu~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\n";
        std::cout <<
"[===================================================================================
====]\n";

        // Wyświetl menu materiałów kursu
        std::cout << "| 1. Otworz materialy                                                   |\n";
        std::cout << "| 2. Dodaj materialy                                                    |\n";
        std::cout << "| 3. Wroc                                                               |\n";
        std::cout <<
"[===================================================================================
====]\n";

        // Pobierz wybór użytkownika
        std::cin >> userInput;

        // Wykonaj akcję w zależności od wyboru użytkownika
        // Jeżeli wybór to równy 1
        if(userInput == "1") {
            // Jeżeli materiały kursu nie są puste
            if(!materials.empty()) {
                // Wyświetl nagłówek menu materiałów kursu
                std::cout <<
"[===================================================================================
====]\n";
                std::cout << "|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Materialy
Kursu~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\n";
                std::cout << "|
```

```
==========================================================================================
===|\n";

            // Przeszuakj wektor materiałów
            for(int i = 0; i < materials.size(); i++) {
                // Wyświetl numer i nazwę materiału
                std::cout << "| " << i + 1 << ". " << materials.at(i)->getName() << "\n";
            }

            // Wyświetl opcję powrotu
            std::cout << "| " << materials.size() + 1 << ". Wroc" << "\n";
            std::cout <<
"[==========================================================================================
====]\n";

            // Pobierz wybór użytkownika
            std::cin >> userInput;

            // Wykonaj akcję w zależności od wyboru użytkownika
            // Jeżeli wybór jest równy 1
            if(std::stoi(userInput) - 1 < materials.size()) {
                // Otwórz menu materiału
                materials.at(std::stoi(userInput) - 1)->materialsMenu();
            } else if((std::stoi(userInput) - 1) == materials.size()) {
                // Wyświetl komunikat o opuszczeniu menu materiałów
                std::cout << "Opusciles menu materialow!" << "\n";
                // Wyjdź z pętli
                break;
                // W przeciwnym wypadku
            } else {
                // Wyświetl komunikat o niepoprawnym wyborze
                std::cout << "Niepoprawna opcja!" << "\n";
            }
        // Jeżeli materiały kursu są puste
        } else {
            // Wyświetl komunikat o braku materiałów
            std::cout << "Brak materialow!" << "\n";
        }
    // Jeżeli wybór to 2
    } else if(userInput == "2") {
        // Menu dodawania materiału
        // Pobierz dane materiału
        // Utwórz zmienne przechowujące dane materiału
        std::string materialName;          // Nazwa
        std::string materialDescription;   // Opis
        std::string materialFileName;      // Nazwa pliku
        std::string materialFileSize;      // Rozmiar pliku
        User * addedBy = user;             // Materiał dodany przez

        // Pobierz dane materiału
        // Pobierz nazwę
        std::cout << "Podaj nazwe: ";
        std::cin >> materialName;

        // Pobierz opis
        std::cout << "Podaj opis: ";
        std::cin >> materialDescription;

        // Pobierz nazwę pliku
        std::cout << "Podaj nazwe pliku: ";
        std::cin >> materialFileName;

        // Pobierz fileSize pliku
```

```
            std::cout << "Podaj rozmiar pliku[kB]: ";
            std::cin >> materialFileSize;

            // Jeżeli któryś z parametrów jest pusty
            if(materialName.empty() || materialDescription.empty() || materialFileName.empty() || materialFileSize.empty())
{
                // Wyświetl komunikat o niepoprawnych danych
                std::cout << "Niepoprawne dane!" << "\n";
                // Wyjdź z pętli
                break;
            }

            // Dodaj materiał
            materials.push_back(new Material(materialName,materialDescription,
                                    materialFileName, std::stoi(materialFileSize), addedBy));
        // Jeżeli wybór to 2
        } else if(userInput == "3") {
            // Wyświetl komunikat o opuszczeniu menu kursów
            std::cout << "Opusciles menu materialow!" << "\n";
            // Wyjdź z pętli
            break;
        // W przeciwnym wypadku
        } else {
            // Wyświetl komunikat o niepoprawnym wyborze
            std::cout << "Niepoprawna opcja!" << "\n";
        }
    }
}
```

- Material.h

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : System Obsługi Studiów
//  @ File Name : Material.h
//  @ Date : 10.06.2023
//  @ Author : Tomasz Wnuk, Bartosz Szynkaruk, Mikołaj Hasiec
//
//

#if !defined(_MATERIAL_H)
#define _MATERIAL_H

// Deklaracja zależności i bibliotek
#include "User.h"
#include <string>
#include <iostream>
#include <thread>
#include <chrono>

// Deklaracja klasy Material
class Material {
public:
   Material(std::string name, std::string description, std::string fileName, int fileSize, User * addedBy);  // Konstruktor klasy Material
   std::string getName();  // Akcesor pola name
   void setName(std::string name);  // Mutator pola name
   std::string getDescription();  // Akcesor pola description
   void setDescription(std::string description);  // Mutator pola description
   std::string getFileName();  // Akcesor pola fileName
   void setFileName(std::string fileName);  // Mutator pola fileName
   int getFileSize();  // Akcesor pola fileSize
   void setFileSize(int fileSize);  // Mutator pola fileSize
   User * getAddedBy();  // Akcesor pola addedBy
   void setAddedBy(User * addedBy);  // Mutator pola addedBy
   void printMaterial();  // Metoda wypisująca informacje o materiale
   void materialsMenu();  // Metoda otwierająca menu materiału
   void downloadFile();  // Metoda symulująca pobieranie pliku
private:
   std::string name;  // Nazwa materiału
   std::string description;  // Opis materiału
   std::string fileName;  // Nazwa pliku
   int fileSize;  // Rozmiar pliku
   User * addedBy;  // Użytkownik dodający materiał
};

#endif  //_MATERIAL_H
```

- Material.cpp

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : System Obsługi Studiów
//  @ File Name : Material.cpp
//  @ Date : 10.06.2023
//  @ Author : Tomasz Wnuk, Bartosz Szynkaruk, Mikołaj Hasiec
//
//

// Deklaracja zależności i bibliotek
#include "Material.h"

// Konstruktor klasy Material
Material::Material(std::string name, std::string description, std::string fileName, int fileSize, User *addedBy) {
    this->name = name;
    this->description = description;
    this->fileName = fileName;
    this->fileSize = fileSize;
    this->addedBy = addedBy;
}

// Akcesor pola name
std::string Material::getName() {
    return name;
}

// Mutator pola name
void Material::setName(std::string name) {
    this->name = name;
}

// Akcesor pola description
std::string Material::getDescription() {
    return description;
}

// Mutator pola description
void Material::setDescription(std::string description) {
    this->description = description;
}

// Akcesor pola fileName
std::string Material::getFileName() {
    return fileName;
}

// Mutator pola fileName
void Material::setFileName(std::string fileName) {
    this->fileName = fileName;
}

// Akcesor pola fileSize
int Material::getFileSize() {
    return fileSize;
}

// Mutator pola fileSize
void Material::setFileSize(int fileSize) {
```

```cpp
    this->fileSize = fileSize;
}

// Akcesor pola addedBy
User * Material::getAddedBy() {
    return addedBy;
}

// Mutator pola addedBy
void Material::setAddedBy(User * addedBy) {
    this->addedBy = addedBy;
}

// Metoda wyswietlajaca dane materialu
void Material::printMaterial() {
    // Wyswietl dane materialu
    std::cout <<
"[==================================================================================
====]\n";
    std::cout << "|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Dane
Materialu~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\n";
    std::cout << "|
==================================================================================
===|\n";
    std::cout << "| Nazwa: " << this->name << "\n";  // Nazwa materiału
    std::cout << "| Opis: " << this->description << "\n";  // Opis materiału
    std::cout << "| Nazwa pliku: " << this->fileName << "\n";  // Nazwa pliku
    std::cout << "| Rozmiar pliku: " << this->fileSize << "\n";  // Rozmiar pliku
    std::cout << "| Dodany przez: " << this->addedBy->getFirstName() << "\n";  // Dodany przez
    std::cout <<
"[==================================================================================
====]\n";
}

// Metoda wyswietlajaca menu materiału
void Material::materialsMenu() {
    // Utwórz zmienną przechowującą wybór użytkownika
    std::string userInput;

    // Petla wyświetlająca menu materiału
    while(true) {
        // Wyswietl menu materiału
        std::cout <<
"[==================================================================================
====]\n";
        std::cout << "|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Menu
Materialu~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\n";
        std::cout <<
"[==================================================================================
====]\n";
        std::cout << "| 1. Wyswietl dane                                                        |\n";
        std::cout << "| 2. Pobierz plik                                                         |\n";
        std::cout << "| 3. Wroc                                                                 |\n";
        std::cout <<
"[==================================================================================
====]\n";

        // Pobierz wybór użytkownika
        std::cin >> userInput;

        // Sprawdź wybór użytkownika
        // Jeżeli wybór to 1
        if(userInput == "1") {
```

```
            // Wyświetl dane materiału
            printMaterial();
        // Jeżeli wybór to 2
        } else if(userInput == "2") {
            // Pobierz plik
            downloadFile();
        // Jeżeli wybór to 3
        } else if (userInput == "3") {
            // Zakończ pętlę
            break;
        // W przeciwnym wypadku
        } else {
            // Wyswietl komunikat o błędzie
            std::cout << "Niepoprawna opcja!" << "\n";
        }
    }
}

// Metoda symulująca pobieranie pliku
void Material::downloadFile() {
    // Utwórz zmienne przechowujące dane paska postępu
    int barWidth = 88;  // Szerokość paska postępu
    int duration = 100;  // Czas trwania symulacji pobierania pliku
    int total = 10;  // Całkowita ilość iteracji pętli

    // Pętla symulująca pobieranie pliku
    // Iteruj od 0 do total
    for(int i = 0; i < total; ++i) {
        // Wyswietl początek paska postępu
        std::cout << "[";
        // Oblicz postęp
        int progress = (barWidth * i) / total;
        // Iteruj od 0 do barWidth
        for(int j = 0; j < barWidth; ++j) {
            // Jeżeli j jest mniejsze od postępu
            if(j < progress) {
                // Wyswietl "="
                std::cout << "=";
            // Jeżeli j jest równe postępowi
            } else if(j == progress) {
                // Wyswietl ">"
                std::cout << ">";
            // W przeciwnym wypadku
            } else {
                // Wyswietl spacje
                std::cout << " ";
            }
        }
        // Wyswietl koniec paska postępu
        std::cout << "] " << int((float(i) / total) * 100.0) << "%\r";
        // Wymuś wypisanie danych na ekran
        std::cout.flush();
        // Uśpij wątek na czas zmiennej duration w milisekundach
        std::this_thread::sleep_for(std::chrono::milliseconds(duration));
    }
    // Wyswietl koniec paska postępu
    std::cout << "[";
    // Iteruj od 0 do barWidth
    for (int j = 0; j < barWidth; ++j) {
        // Wyswietl zank paska postępu
        std::cout << "=";
    }
    // Wyswietl koniec paska postępu
```

```
    std::cout << "] 100%" << std::endl;
    // Wyswietl komunikat o pobraniu pliku
    std::cout << "Pobrano plik!" << std::endl;
}
```

- VideoConference.h

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : System Obsługi Studiów
//  @ File Name : VideoConference.h
//  @ Date : 10.06.2023
//  @ Author : Tomasz Wnuk, Bartosz Szynkaruk, Mikołaj Hasiec
//
//

#if !defined(_VIDEOCONFERENCE_H)
#define _VIDEOCONFERENCE_H

// Deklaracja zależności i bibliotek
#include <iostream>
#include <string>
#include <vector>
#include "Lecturer.h"
#include "Student.h"

// Deklaracja klas
class Lecturer;
class Student;

// Deklaracja klasy VideoConference
class VideoConference {
public:
   VideoConference(std::string name, Lecturer * lecturer);  // Konstruktor
   void addStudent(Student * student);  // Metoda dodająca studenta do wideokonferencji
   void removeStudent(std::string login);  // Metoda usuwająca studenta z wideokonferencji
   void openStudentVideoConferenceMenu(Student * student);  // Metoda otwierająca menu wideokonferencji dla studenta
   void openLecturerVideoConferenceMenu(Lecturer * lecturer);  // Metoda otwierająca menu wideokonferencji dla wykładowcy
private:
   std::string name;  // Nazwa wideokonferencji
   Lecturer * lecturer;  // Wykładowca prowadzący wideokonferencję
   std::vector <Student *> students;  // Wektor studentów uczestniczących w wideokonferencji
};

#endif  //_VIDEOCONFERENCE_H
```

- VideoConference.cpp

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : System Obsługi Studiów
//  @ File Name : VideoConference.cpp
//  @ Date : 10.06.2023
//  @ Author : Tomasz Wnuk, Bartosz Szynkaruk, Mikołaj Hasiec
//
//

// Deklaracja zależności i bibliotek
#include "VideoConference.h"

// Konstruktor
VideoConference::VideoConference(std::string name, Lecturer * lecturer) {
   this->name = name;
   this->lecturer = lecturer;
}

// Metoda dodająca studenta do konferencji
void VideoConference::addStudent(Student * student) {
   // Dodanie studenta do wektora studentów
   students.push_back(student);
}

// Metoda usuwająca studenta z konferencji
void VideoConference::removeStudent(std::string login) {
   // Przeszukaje wektor studentów
   for(int i = 0; i < students.size(); i++) {
      // Jeżeli login studenta jest równy podanemu loginowi
      if(students[i]->getLogin() == login) {
         // Usuń studenta z wektora
         students.erase(students.begin() + i);
         // Zakończ pętlę
         break;
      }
   }
}

// Metoda otwierająca menu konferencji dla studenta
void VideoConference::openStudentVideoConferenceMenu(Student * student) {
   // Utwórz zmienną przechowującą wybór użytkownika
   std::string userInput;

   // Pętla menu wideokonferencji studenta
   while(true) {
      // Wyświetl menu wideokonferencji studenta
      std::cout << "[=======================================================================================]\n";
      std::cout << "|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Menu Wideokonferencji~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\n";
      std::cout << "[=======================================================================================]\n";
      std::cout << "| 1. Wyswietl uczestnikow                                                                |\n";
      std::cout << "| 2. Wyjdz z wideokonferencji                                                            |\n";
      std::cout << "[=======================================================================================
```

```
====]\n";

        // Pobierz wybór użytkownika
        std::cin >> userInput;

        // Wykonaj akcję w zależności od wyboru użytkownika
        // Jeżeli wybór to 1
        if(userInput == "1") {
            // Wyświetl nagłówek uczestników wideokonferencji
            std::cout <<
"[==========================================================================
====]\n";
            std::cout << "|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Uczestnicy
Wideokonferencji~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\n";
            std::cout <<
"[==========================================================================
====]\n";

            // Wyświetl wykładowcę
            std::cout << "| Wykladowca: " << lecturer->getFirstName() << " " << lecturer->getLastName() << "\n";

            // Jeżeli wektor studentów nie jest pusty
            if(!students.empty()) {
                // Przeszukaj wektor studentów
                for(int i = 0; i < students.size(); i++) {
                    // Wyświetl imiona i nazwiska studentów w wideokonferencji
                    std::cout << "| " << i + 1 << ". " << students.at(i)->getFirstName() << " " << students.at(i)->getLastName()
<< "\n";
                }
            } else {
                // Wyświetl komunikat o braku uczestników
                std::cout << "Brak uczestnikow!" << "\n";
            }
        // Jeżeli wybór to 2
        } else if(userInput == "2") {
            // Przeszukaj wektor studentów
            for(int i = 0; i < students.size(); i++) {
                // Jeżeli student jest równy studentowi z wektora
                    if(students[i] == student) {
                    // Usuń studenta z wektora
                    removeStudent(student->getLogin());
                    // Wyświetl komunikat o wyjściu z wideokonferencji
                    std::cout << "Wyjscie z wideokonferencji!" << "\n";
                    // Zakończ pętlę
                    break;
                    }
                }
            // Wyjdź z wideokonferencji
            break;
        // W przeciwnym wypadku
        } else {
            // Wyświetl komunikat o niepoprawnym wyborze
            std::cout << "Niepoprawna opcja!" << "\n";
        }
    }
}

// Metoda otwierająca menu konferencji dla wykładowcy
void VideoConference::openLecturerVideoConferenceMenu(Lecturer * lecturer) {
    // Utwórz zmienną przechowującą wybór użytkownika
    std::string userInput;

    // Pętla menu wideokonferencji wykładowcy
```

```
   while(true) {
      // Wyświetl menu wideokonferencji wykładowcy
      std::cout <<
"[=============================================================================================
====]\n";
      std::cout << "|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Menu
Wideokonferencji~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\n";
      std::cout <<
"[=============================================================================================
====]\n";
      std::cout << "| 1. Wyswietl uczestnikow                                                    |\n";
      std::cout << "| 2. Zakoncz wideokonferencje                                                |\n";
      std::cout <<
"[=============================================================================================
====]\n";

      // Pobierz wybór użytkownika
      std::cin >> userInput;

      // Wykonaj akcję w zależności od wyboru użytkownika
      // Jeżeli wybór to 1
      if(userInput == "1") {
         // Wyświetl nagłówek uczestników wideokonferencji
         std::cout <<
"[=============================================================================================
====]\n";
         std::cout << "|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Uczestnicy
Wideokonferencji~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\n";
         std::cout <<
"[=============================================================================================
====]\n";

         // Wyświetl wykładowcę
         std::cout << "| Wykladowca: " << lecturer->getFirstName() << " " << lecturer->getLastName() << "\n";

         // Jeżeli wektor studentów nie jest pusty
         if(!students.empty()) {
            // Przeszukaj wektor studentów
            for(int i = 0; i < students.size(); i++) {
               // Wyświetl imiona i nazwiska studentów w wideokonferencji
               std::cout << "| " << i + 1 << ". " << students.at(i)->getFirstName() << " " << students.at(i)->getLastName()
<< "\n";
            }
         } else {
            // Wyświetl komunikat o braku uczestników
            std::cout << "Brak uczestnikow!" << "\n";
         }
         // Jeżeli wybór to 2
      } else if(userInput == "2") {
         // Przeszukaj wektor studentów
         for(int i = 0; i < students.size(); i++) {
            // Usuń studenta z wektora studentów po zakończeniu wideokonferencji
            removeStudent(students[i]->getLogin());
         }
         // Wyświetl komunikat o zakończeniu wideokonferencji
         std::cout << "Zakonczono wideokonferencje!" << "\n";
         // Wyjdź z wideokonferencji
         break;
      } else {
         // Wyświetl komunikat o niepoprawnym wyborze
         std::cout << "Niepoprawna opcja!" << "\n";
      }
   }
```

```
}
```

Pliki csv

- users.csv

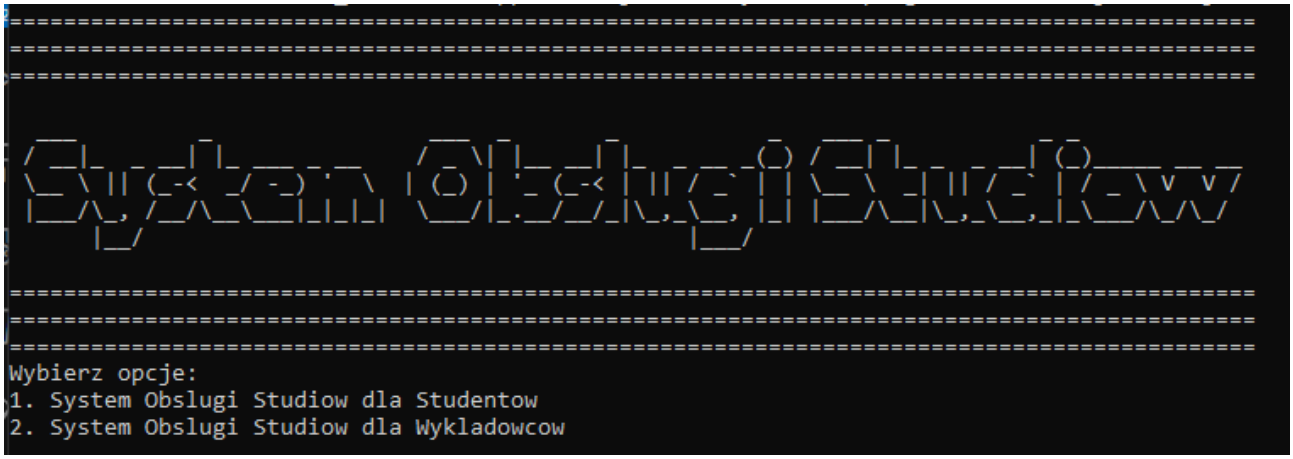
```
admin admin,admin,admin,admin@mail.com,
Kajetan Kowalski,kkowalski,123,kajetan.kowalski@mail.com,
Boleslaw Rutkowski,brutkowski,456,boleslaw.rutkowski@mail.com,
Dorian Krawczyk,dkrawczyk,789,dorian.krawczyk@mail.com,
Krystian Szymanski,kszymanski,123,krystian.szymanski@mail.com,
Aleks Mazur,amazur,456,aleks.mazur@mail.com,
```

- courses.csv

```
Algorytmy i Struktury Danych,Milosz Kaminski,1,Kajetan Kaminski,Roman Adamczyk,Kornel Makowski,Konstanty Sokolowski,Pawel Zalewski,
Inzynieria Oprogramowania,Antoni Wroblewski,1,Remigiusz Jaworski,Michal Kowalczyk,Jerzy Jasinski,Piotr Pawlak,
Podstawy Programowania,Borys Gorski,1,Henryk Pietrzak,Jerzy Kalinowski,Florian Glowacki,
Systemy Operacyjne,Mariusz Sobczak,0,Arkadiusz Gajewski,Eryk Gorecki,
Architektury Systemow Komputerowych,Kajetan Piotrowski,1,Gustaw Cieslak,
```

Testy i opis działania systemu

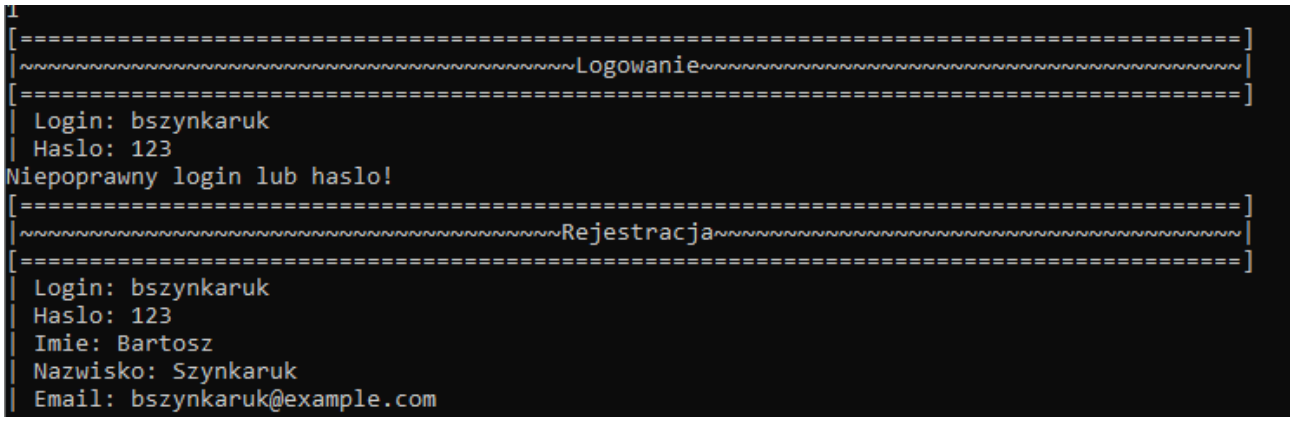
Wybór typu użytkownika



Po uruchomieniu programu wyświetla się wybór typu użytkownika. System działa inaczej w zależności od tego, kto jest zalogowany. Na tym etapie również wczytywane są kursy i użytkownicy z plików .csv.

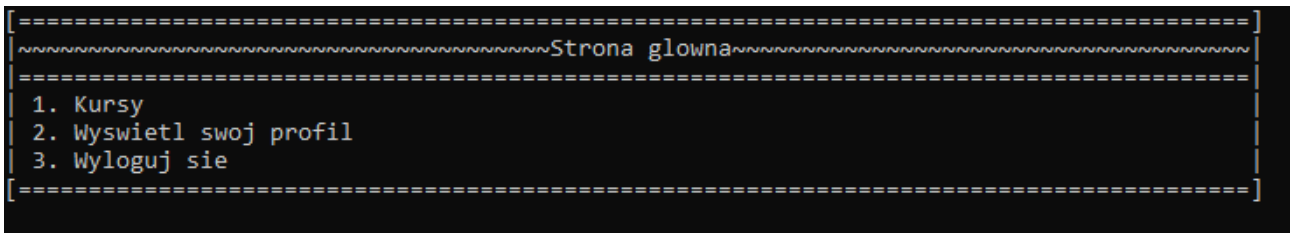
System dla studenta

- Logowanie i rejestracja



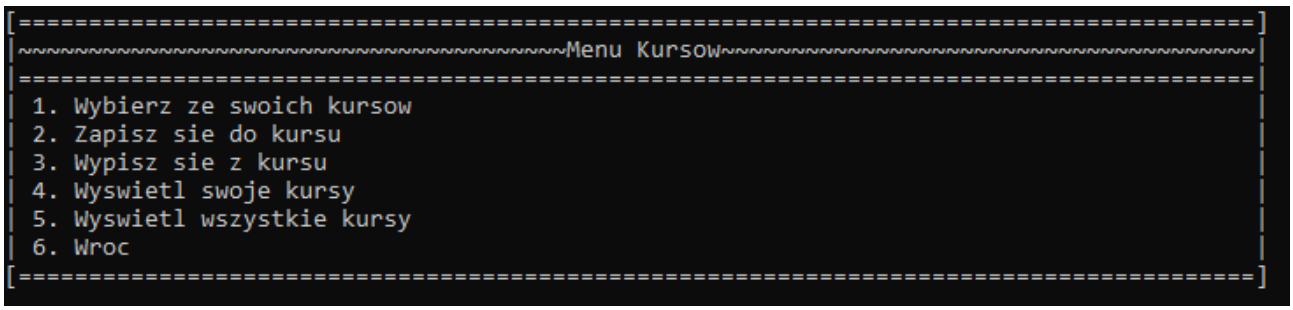
W przypadku podania nieprawidłowych danych system automatycznie przenosi użytkownika na stronę z rejestracją, na której można podać swoje dane. System tworzy wtedy konto użytkownika, loguje go do systemu, przenosi na stronę główną i zapisuje do pliku .csv.

- Strona główna



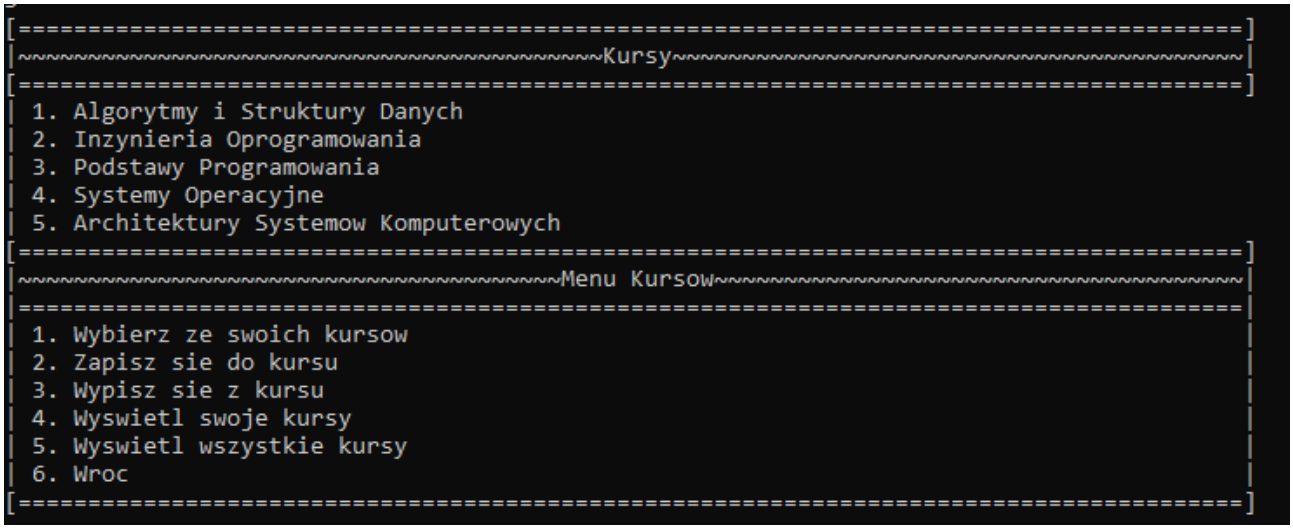
Na stronie głównej użytkownik może wybrać, działanie, jakie chce podjąć.

- Kursy



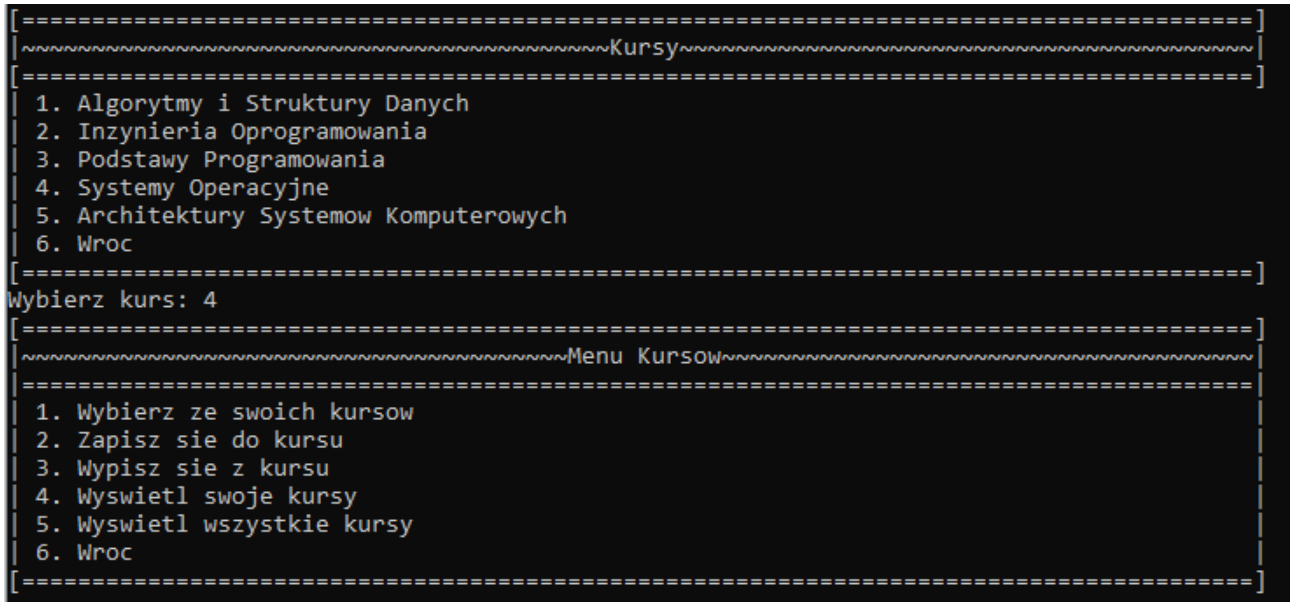
Po wybraniu tej opcji ze strony głównej użytkownik zostaje przeniesiony do menu kursów. W tym menu dostępne jest kilka działań widocznych wyżej.

- Wyświetl wszystkie kursy



Po wybraniu tej opcji system wyświetla użytkownikowi listę kursów.

- Zapisz się do kursu



Po wybraniu “Zapisz się do kursu” wyświetla się lista kursów do których użytkownik może dołączyć. Po wybraniu kursu, użytkownik jest przenoszony na poprzednią stronę.

- Wyświetl swoje kursy

```
[================================================================================]
|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Twoje Kursy~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
[================================================================================]
| 1. Systemy Operacyjne
[================================================================================]
|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Menu Kursow~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
|================================================================================|
| 1. Wybierz ze swoich kursow                                                    |
| 2. Zapisz sie do kursu                                                         |
| 3. Wypisz sie z kursu                                                          |
| 4. Wyswietl swoje kursy                                                        |
| 5. Wyswietl wszystkie kursy                                                    |
| 6. Wroc                                                                        |
[================================================================================]
```

Z powyższego zrzutu ekranu wynika, że udało się pomyślnie zapisać na kurs “Systemy Operacyjne”.

- Wybierz ze swoich kursów

```
[================================================================================]
|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Twoje Kursy~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
[================================================================================]
| 1. Systemy Operacyjne
| 2. Wroc
[================================================================================]
Wybierz kurs: 1
[================================================================================]
|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Systemy Operacyjne~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
[================================================================================]
| 1. Wyswietl strone kursu                                                       |
| 2. Otworz materialy                                                            |
| 3. Dolacz do wideokonferencji                                                  |
| 4. Wroc                                                                        |
[================================================================================]
```

Wybór pierwszego punktu powoduje wyświetlenie listy kursów do których należy użytkownik. Z tej listy można otworzyć stronę wybranego kursu.

- Wyświetl stronę kursu

```
[==============================================================================]
|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Systemy Operacyjne~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
[==============================================================================]
| Wykladowca: Mariusz Sobczak
| Uczestnicy:
| 1. Arkadiusz Gajewski
| 2. Eryk Gorecki
| 3. Bartosz Szynkaruk
[==============================================================================]
|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Systemy Operacyjne~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
[==============================================================================]
| 1. Wyswietl strone kursu                                                     |
| 2. Otworz materialy                                                          |
| 3. Dolacz do wideokonferencji                                                |
| 4. Wroc                                                                      |
[==============================================================================]
```

Ta opcja powoduje wyświetlenie prowadzącego kursu i listy jego uczestników.

- Otwórz materiały

```
[==============================================================================]
|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Materialy kursu~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
[==============================================================================]
| 1. Otworz materialy                                                          |
| 2. Dodaj materialy                                                           |
| 3. Wroc                                                                      |
[==============================================================================]
```

System wyświetla stronę z materiałami.

- Dodaj materiały

```
Podaj nazwe: notatki.txt
Podaj opis: Notatki
Podaj nazwe pliku: notatki.txt
Podaj rozmiar pliku[kB]: 10
[==============================================================================]
|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Materialy kursu~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
[==============================================================================]
| 1. Otworz materialy                                                          |
| 2. Dodaj materialy                                                           |
| 3. Wroc                                                                      |
[==============================================================================]
```

System prosi użytkownika o podanie danych do przesyłanego materiału, a następnie umieszcza go na stronie materiałów kursu.

- Otwórz materiały

```
[==================================================================================]
|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Materialy kursu~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
[==================================================================================]
| 1. Otworz materialy                                                              |
| 2. Dodaj materialy                                                               |
| 3. Wroc                                                                          |
[==================================================================================]
1
[==================================================================================]
|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Materialy Kursu~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
|==================================================================================|
| 1. notatki.txt
| 2. Wroc
[==================================================================================]
1
[==================================================================================]
|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Menu Materialu~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
[==================================================================================]
| 1. Wyswietl dane                                                                 |
| 2. Pobierz plik                                                                  |
| 3. Wroc                                                                          |
[==================================================================================]
```

Po wybraniu tej opcji system wyświetla przesłane materiały. Po wybraniu interesującego nas materiału można wyświetlić jego dane i go pobrać.

- Wyświetl dane

```
[==================================================================================]
|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Dane Materialu~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
|==================================================================================|
| Nazwa: Notatki
| Opis: Notatki
| Nazwa pliku: notatki.txt
| Rozmiar pliku: 10
| Dodany przez: Bartosz
[==================================================================================]
[==================================================================================]
|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Menu Materialu~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
[==================================================================================]
| 1. Wyswietl dane                                                                 |
| 2. Pobierz plik                                                                  |
| 3. Wroc                                                                          |
[==================================================================================]
```

System wyświetla dane materiału, takie jak nazwa, opis, rozmiar, a następnie przenosi użytkownika na poprzednią stronę.

- Pobierz plik

```
2
[================================================================================]
Pobrano plik!
[================================================================================]
|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Menu Materialu~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
[================================================================================]
| 1. Wyswietl dane                                                               |
| 2. Pobierz plik                                                                |
| 3. Wroc                                                                        |
[================================================================================]
```

Wybranie tej opcji powoduje pobranie pliku na dysk użytkownika.

- Dołącz do wideokonferencji

```
[================================================================================]
|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Systemy Operacyjne~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
[================================================================================]
| 1. Wyswietl strone kursu                                                       |
| 2. Otworz materialy                                                            |
| 3. Dolacz do wideokonferencji                                                  |
| 4. Wroc                                                                        |
[================================================================================]
3
Wideokonferencja nie jest utworzona!
[================================================================================]
|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Systemy Operacyjne~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
[================================================================================]
| 1. Wyswietl strone kursu                                                       |
| 2. Otworz materialy                                                            |
| 3. Dolacz do wideokonferencji                                                  |
| 4. Wroc                                                                        |
[================================================================================]
```

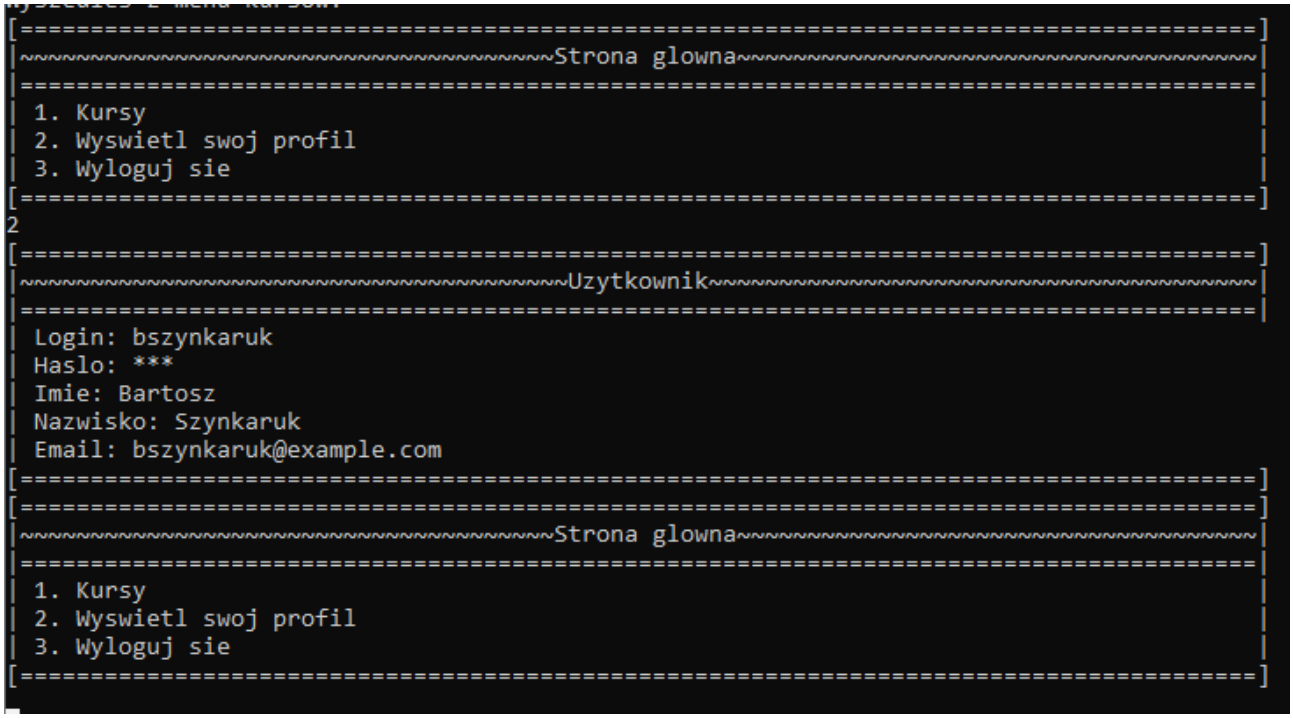
Ze strony kursu można również dołączyć do wideokonferencji, która może zostać utworzona tylko przez prowadzącego kursu.

- Wypisz się z kursu

```
Opusciles menu kursu Systemy Operacyjne!
[========================================================================================]
|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Menu Kursow~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
|========================================================================================|
| 1. Wybierz ze swoich kursow                                                            |
| 2. Zapisz sie do kursu                                                                 |
| 3. Wypisz sie z kursu                                                                  |
| 4. Wyswietl swoje kursy                                                                |
| 5. Wyswietl wszystkie kursy                                                            |
| 6. Wroc                                                                                |
[========================================================================================]
3
[========================================================================================]
|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Twoje Kursy~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
[========================================================================================]
| 1. Systemy Operacyjne
| 2. Wroc
[========================================================================================]
Wybierz kurs: 1
Wyszedles z kursu Systemy Operacyjne!
[========================================================================================]
|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Menu Kursow~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
|========================================================================================|
| 1. Wybierz ze swoich kursow                                                            |
| 2. Zapisz sie do kursu                                                                 |
| 3. Wypisz sie z kursu                                                                  |
| 4. Wyswietl swoje kursy                                                                |
| 5. Wyswietl wszystkie kursy                                                            |
| 6. Wroc                                                                                |
[========================================================================================]
```

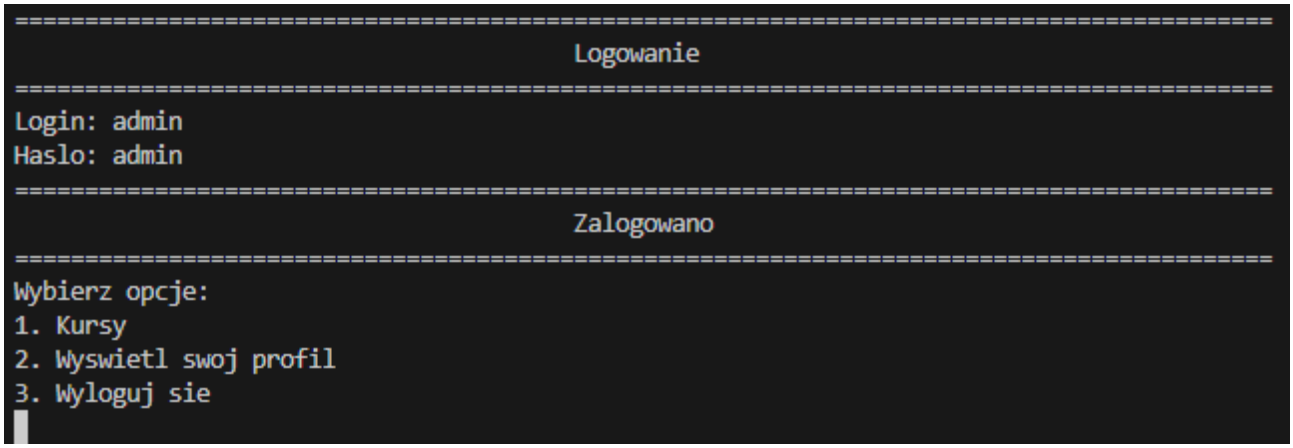
Po powrocie do menu kursów można również wypisać się z kursu. Z powyższego zrzutu ekranu wynika, że udało się pomyślnie wypisać z kursu “Systemy operacyjne”.

- Wyświetl swój profil



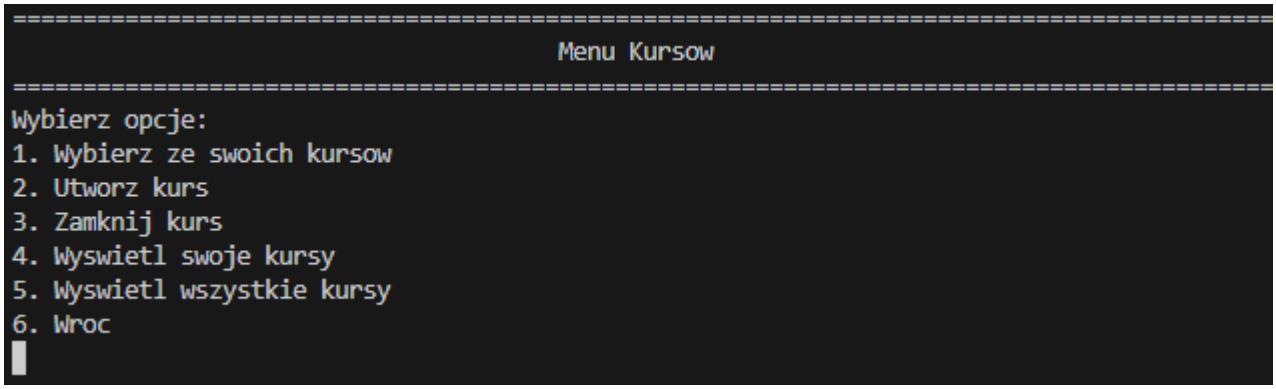
Ze strony głównej można również wyświetlić swój profil użytkownika. Znajdują się w nim informacje podane podczas rejestracji do systemu.

System dla prowadzącego



Po zalogowaniu do systemu zostajemy przeniesieni na tą samą stronę główną.

- Menu kursów



Po wybraniu pierwszej opcji zostajemy przekierowani na menu kursów widoczne powyżej.

- Utwórz kurs

```
================================================================================
                                Menu Kursow
================================================================================
Wybierz opcje:
1. Wybierz ze swoich kursow
2. Utworz kurs
3. Zamknij kurs
4. Wyswietl swoje kursy
5. Wyswietl wszystkie kursy
6. Wroc
2
Podaj nazwe kursu: Ekonomia
Kurs zostal utworzony!
```

Po wybraniu drugiej opcji podajemy nazwę, jaką chcemy nadać naszemu kursowi po czym kurs ten zostaje dodany do pliku csv.

- Wybierz ze swoich kursów

```
================================================================================
                                Twoje Kursy
================================================================================
1. Ekonomia
2. Wroc
Wybierz kurs: 1
================================================================================
                                Ekonomia
================================================================================
Wybierz opcje:
1. Wyswietl strone kursu
2. Otworz materialy
3. Utworz wideokonferencje
4. Wroc
```

Po wybraniu tej opcji i odpowiedniego utworzonego przez prowadzącego kursu możemy wykonać działania takie, jakie są widoczne powyżej.

- Wyświetl stronę kursu

```
1
================================================================================
                                Ekonomia
================================================================================
Lecturer: admin admin
Uczestnicy:
```

Opcja ta pokazuje nam, kto dany kurs stworzył oraz kto w nim uczestniczy.

- Utwórz wideokonferencję

```
================================================================================
                          Wideokonferencja - Ekonomia
================================================================================
Wybierz opcje:
1. Wyswietl uczestnikow
2. Zakoncz wideokonferencje
```

Po wybraniu tej opcji prowadzący jest w stanie zobaczyć uczestników wideokonferencji oraz jest w stanie ją zakończyć.

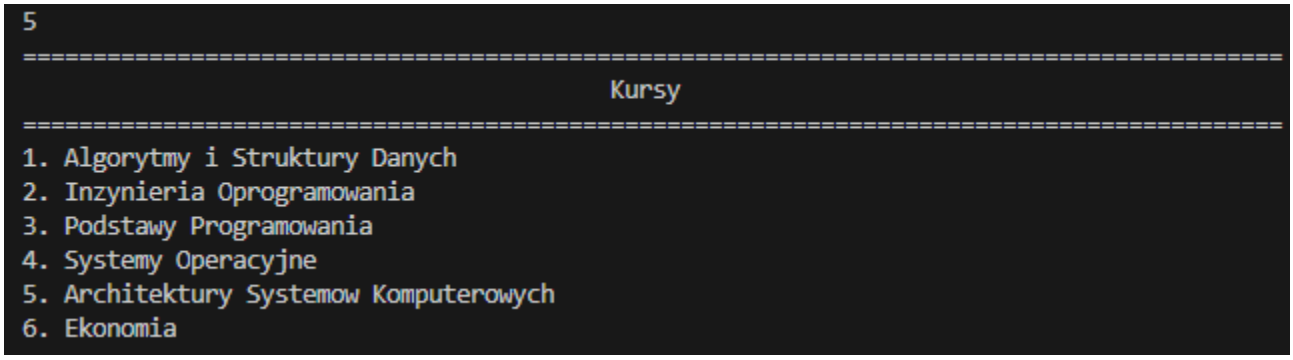
```
================================================================================
                          Wideokonferencja - Ekonomia
================================================================================
Wybierz opcje:
1. Wyswietl uczestnikow
2. Zakoncz wideokonferencje
2
Zakonczono wideokonferencje!
```

- Wyświetl swoje kursy

```
================================================================================
                                  Menu Kursow
================================================================================
Wybierz opcje:
1. Wybierz ze swoich kursow
2. Utworz kurs
3. Zamknij kurs
4. Wyswietl swoje kursy
5. Wyswietl wszystkie kursy
6. Wroc
4
================================================================================
                                  Twoje Kursy
================================================================================
1. Ekonomia
```

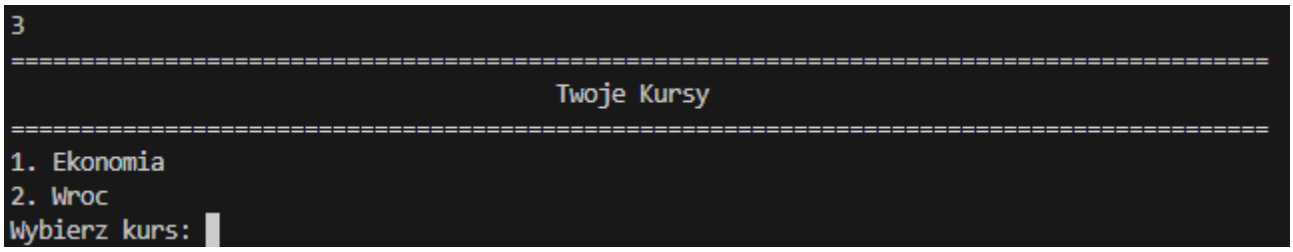
Po powrocie do menu kursów i wybraniu opcji czwartej zostają wyświetlone kursy utworzone przez prowadzącego.

- Wyświetl wszystkie kursy

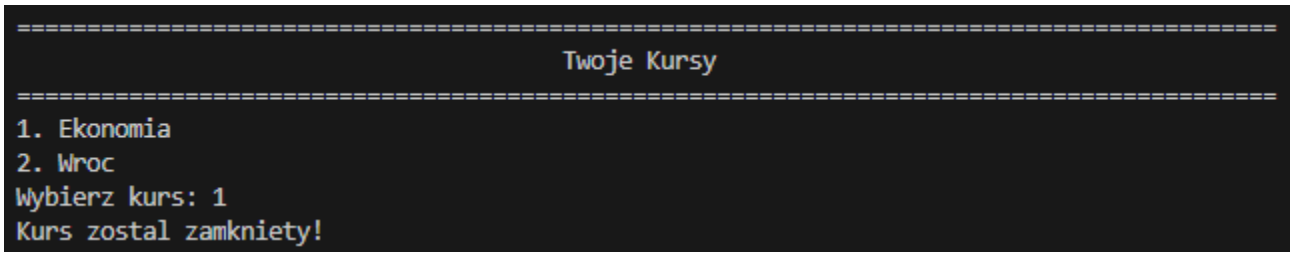


Po wybraniu tej opcji system wyświetla listę kursów oraz kursy utworzone przez prowadzącego.

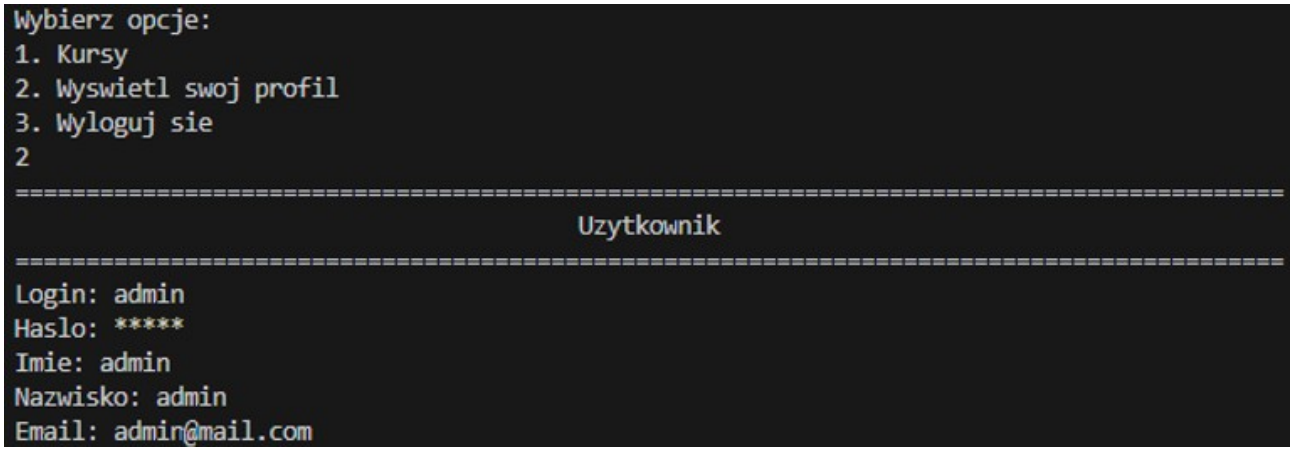
- Zamknij kurs



Po wybraniu tej opcji prowadzącemu zostaje wyświetlona lista utworzonych przez niego kursów, który po wybraniu odpowiedniego zostaje usunięty.

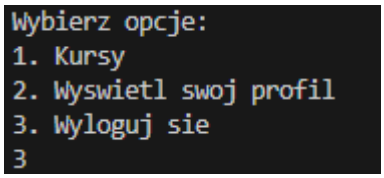


- Wyświetl swój profil



Po wybraniu tej opcji zostają wyświetlone informacje o prowadzącym.

- Wyloguj się



Po wybraniu tej opcji system wylogowuje użytkownika oraz kończy działanie.