Fragment systemu obsługi studenta w uczelni

Kamil Chwetkowski
Adam Polański
Michał Rusinek

Lato 2023

Spis treści

najwyższych w hierarchii projektu, do tych położonych niżej.

c. Testy funkcjonalne

Dodaj materialy z przedmiotów

Logowanie

Skorzystaj z narzędzi dydaktycznych

Sprawdź materiały z przedmiotów

Wstaw oceny + Sprawdź oceny

Rejestracja

d. Testy systemowe + testy akceptacyjne

- Logowanie i rejestracja

Dla Studenta

Dla Wykładowcy

- Materiały z przedmiotów

Sprawdź materiały z przedmiotów

Dodaj materiały z przedmiotów

Usuń ostatni materiał

- Skorzystaj z narzędzi dydaktycznych

- Sprawdź oceny

- Wstaw oceny

- Studenci zapisani do przedmiotu

Dodaj studenta do przedmiotu

Usuń studenta z przedmiotu

- Przedmioty

Dodaj przedmiot

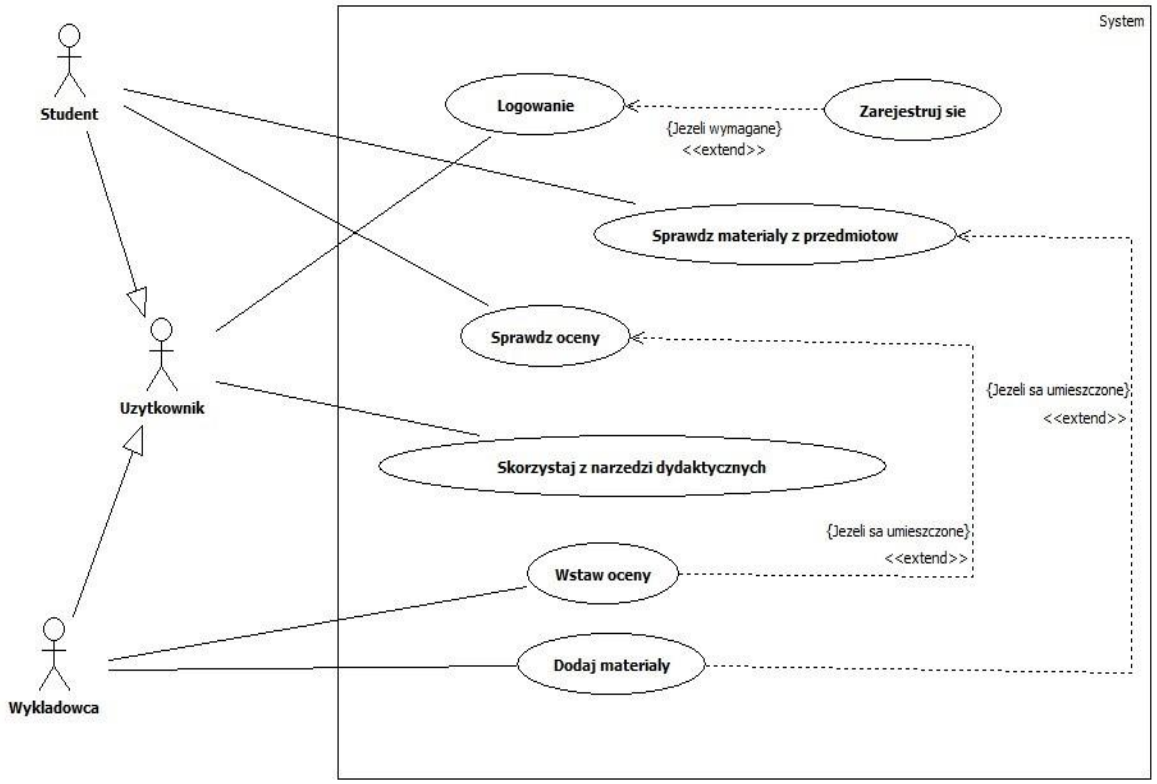
Usuń przedmiot

- Sprawdź dane

Dla Studenta

Dla Wykładowcy

1. Model przypadków użycia dla dziedziny problemu:



Na modelu przypadków użycia znajduje się jeden aktor, który generalizuje zarówno studenta, jak i wykładowcę. Aktor ten nazywa się "Użytkownik". Użytkownik ma możliwość interakcji z systemem za pomocą różnych funkcji, w zależności od swojej roli.

Dla wykładowcy dostępne są dwie funkcje: "Wstaw oceny" oraz "Dodaj materiały". Poprzez funkcję "Wstaw oceny" wykładowca może wprowadzać oceny dla swoich studentów, natomiast funkcja "Dodaj materiały" umożliwia wykładowcy udostępnianie materiałów edukacyjnych, takich jak prezentacje, notatki czy zadania.

Dla studenta dostępne są dwie funkcje: "Sprawdź oceny" oraz "Sprawdź materiały z przedmiotu". Przy użyciu funkcji "Sprawdź oceny", studenci mogą sprawdzać swoje oceny z poszczególnych przedmiotów. Natomiast funkcja "Sprawdź materiały z przedmiotu" pozwala studentom na dostęp do materiałów edukacyjnych związanych z konkretnym przedmiotem.

Wspólnie, zarówno studenci, jak i wykładowcy mają dostęp do logowania do systemu, aby zidentyfikować się jako użytkownicy systemu (lub opcjonalnie rejestracji gdy nie posiadają swojego konta w systemie). Użytkownicy mają również dostęp do funkcji "Skorzystaj z narzędzi dydaktycznych", gdzie mogą korzystać z różnych narzędzi i zasobów oprogramowania udostępnianych przez uczelnię w celach edukacyjnych.

2. Scenariusze i diagramy przepływu

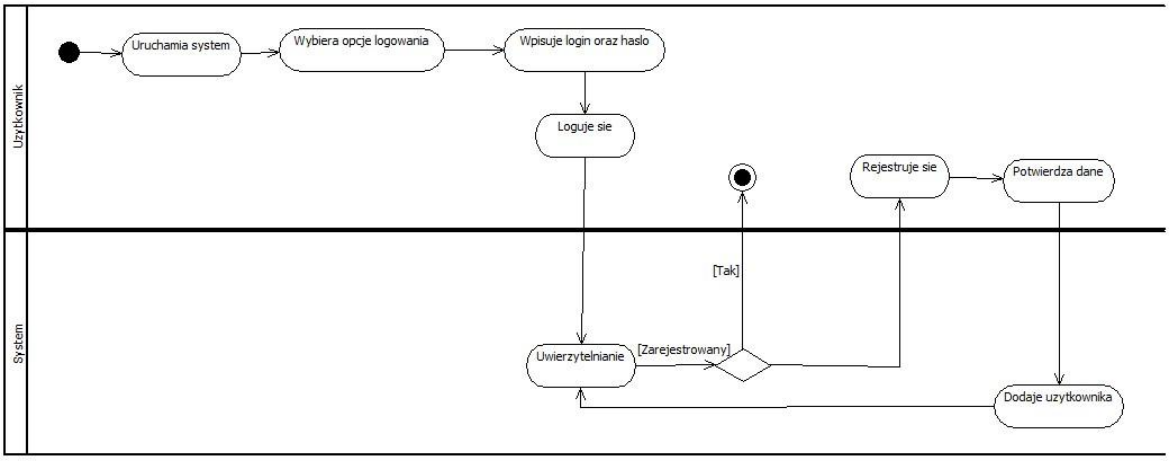
a. Logowanie i rejestracja

Logowanie:

Przypadek użycia	Użytkownik loguje się.
Scenariusz	Zalogowanie się użytkownika do systemu.
Warunki wstępne	Użytkownik jest zarejestrowany.
Niezmienniki	Użytkownik chce się zalogować.
Opis	Użytkownik uruchamia system i przechodzi do ekranu logowania. Użytkownik wybiera opcję "Logowanie dla studenta" lub "Logowanie dla wykładowcy". W polu "Login" wpisuje swoją własną, unikalną nazwę użytkownika, następnie w polu "Hasło" wpisuje hasło do swojego konta. Po wprowadzeniu danych następuje uwierzytelnianie. Użytkownik uzyskuje dostęp do systemu i następuje przekierowanie na stronę główną.
Warunki końcowe	Użytkownik zalogował się poprawnie.
Źródła możliwych błędów	● Wprowadzona nazwa użytkownika nie pasuje do żadnego konta zarejestrowanego w systemie. ● Wprowadzone hasło jest niezgodne z hasłem przypisanym do konta.

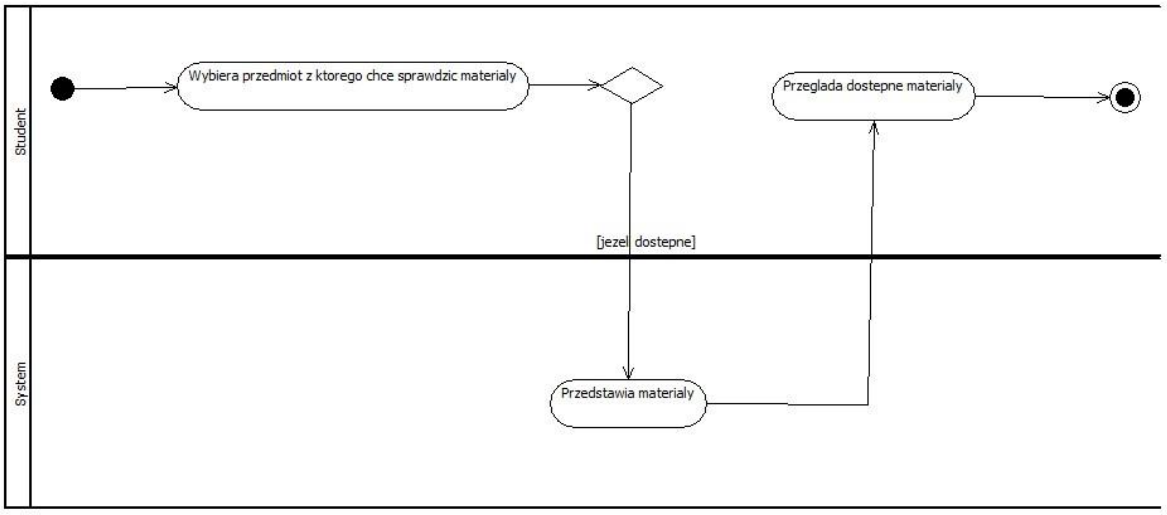
Rejestracja:

Przypadek użycia	Użytkownik rejestruje się w systemie.
Scenariusz	Zarejestrowanie się użytkownika w systemie.
Warunki wstępne	● Użytkownik nie posiada konta w systemie. ● Użytkownik chce się zarejestrować. ● Użytkownik jest studentem i posiada swój własny numer albumu lub użytkownik jest wykładowcą i posiada swój własny identyfikator.
Niezmienniki	Użytkownik chce się zarejestrować.
Opis	Użytkownik uruchamia system i przechodzi do ekranu rejestracji. Użytkownik wybiera opcję "Rejestracja dla studenta" lub "Rejestracja dla wykładowcy". Jeżeli użytkownik wybrał pierwszą z opcji, wyświetlane jest pole "Numer albumu" w którym użytkownik wpisuje swój numer albumu. Jeżeli użytkownik wybrał drugą opcję, wyświetlane jest pole "Identyfikator", w którym użytkownik wpisuje swój identyfikator. W polu "Login" użytkownik wpisuje swoją własną, unikalną nazwę użytkownika, następnie w polu "Hasło" wpisuje hasło do swojego konta. Po wprowadzeniu danych system sprawdza, czy podany login jest unikalny i czy hasło spełnia wymogi. Utworzone zostaje nowe konto w systemie. Użytkownik zostaje zarejestrowany, a następnie automatycznie zalogowany na nowo utworzone konto, uzyskuje dostęp do systemu i następuje przekierowanie na stronę główną.
Warunki końcowe	Użytkownik zarejestrował się poprawnie. W systemie zostaje utworzone nowe konto.
Źródła możliwych błędów	● Wprowadzona nazwa użytkownika już istnieje w systemie. ● Wprowadzone hasło jest niezgodne z ustalonymi wymaganiami.


Uzytkownik
System
Uruchamia system
Wybiera opcje logowania
Wpisuje login oraz haslo
Loguje sie
Rejestruje sie
Potwierdza dane
[Tak]
Uwierzytelnianie
[Zarejestrowany]
Dodaje uzytkownika

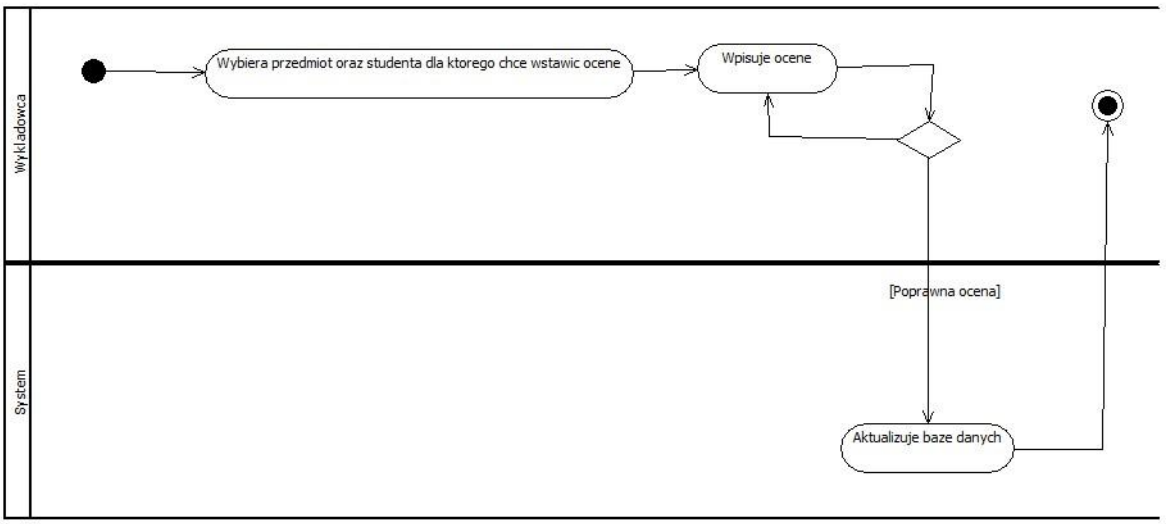
b. Sprawdź materiały z przedmiotów

Przypadek użycia	Użytkownik sprawdza materiały z przedmiotu.
Scenariusz	Przedstawienie materiałów z przedmiotu.
Warunki wstępne	• Użytkownik jest studentem. • Użytkownik jest zalogowany. • Użytkownik jest zapisany na przedmiot.
Niezmienniki	Użytkownik chce sprawdzić materiały dydaktyczne z przedmiotu.
Opis	Użytkownik uruchamia system i loguje się. Przechodzi do sekcji "przedmioty", dokonuje selekcji przedmiotu i otwiera sekcję materiały. System przedstawia użytkownikowi dostępne materiały z wybranego przedmiotu.
Warunki końcowe	Użytkownikowi zostają przedstawione materiały z przedmiotu.
Źródła możliwych błędów	Brak dodanych materiałów z przedmiotu.



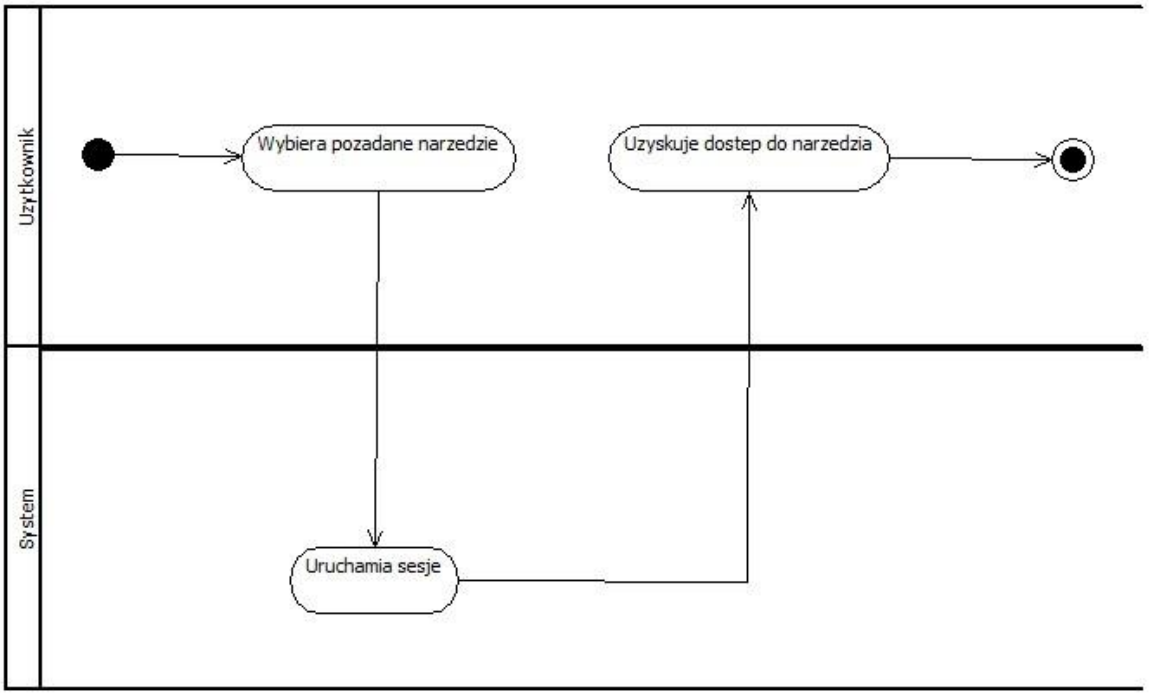
c. Sprawdź oceny

Przypadek użycia	Użytkownik sprawdza oceny z przedmiotu.
Scenariusz	Przedstawienie ocen użytkownika.
Warunki wstępne	• Użytkownik jest studentem. • Użytkownik jest zalogowany. • Użytkownik jest zapisany na przedmiot.
Niezmienniki	Użytkownik chce sprawdzić oceny.
Opis	Użytkownik uruchamia system i loguje się. Przechodzi do sekcji "przedmioty", dokonuje selekcji przedmiotu i otwiera sekcję "oceny". System przedstawia użytkownikowi jego oceny z wybranego przedmiotu.
Warunki końcowe	Użytkownikowi zostają przedstawione oceny.
Źródła możliwych błędów	Brak wpisanych ocen z przedmiotu.



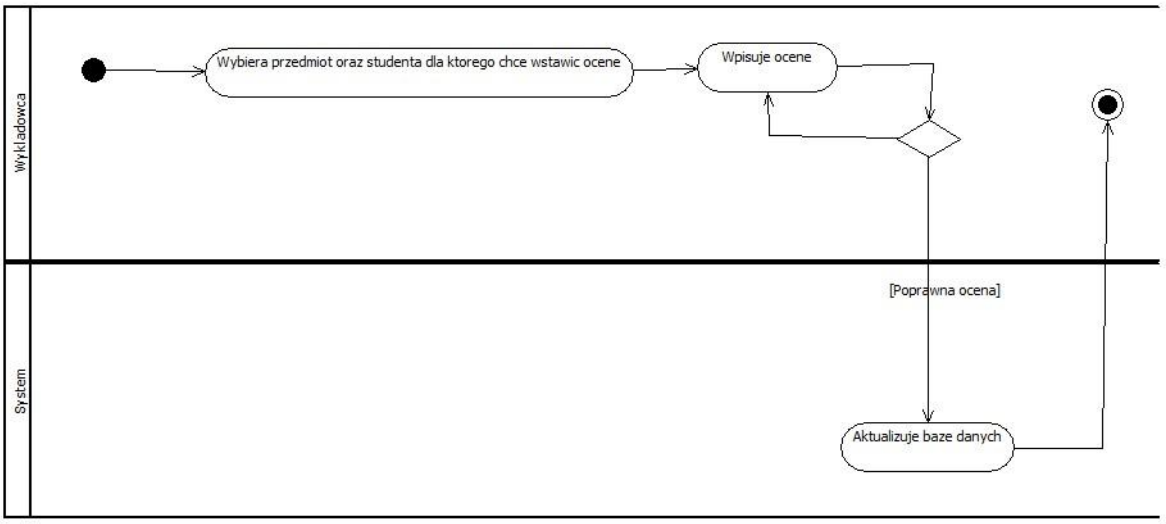
d. Skorzystaj z narzędzi dydaktycznych

Przypadek użycia	Użytkownik korzysta z narzędzi dydaktycznych
Scenariusz	Użytkownik otwiera repozytorium narzędzi.
Warunki wstępne	Użytkownik jest zalogowany.
Niezmienniki	Repozytorium zawiera wystarczający zestaw narzędzi dydaktycznych.
Opis	Użytkownik uruchamia system i loguje się. Wybiera sekcję "narzędzia dydaktyczne" i wybiera pożądane narzędzie. Następuje przekierowanie i system uruchamia sesję w wybranym programie. Po skorzystaniu z wybranego narzędzia, użytkownik opuszcza środowisko, a sesja się zamyka.
Warunki końcowe	Użytkownik uzyskał dostęp do wybranego narzędzia.
Źródła możliwych błędów	Ograniczone zasoby repozytorium



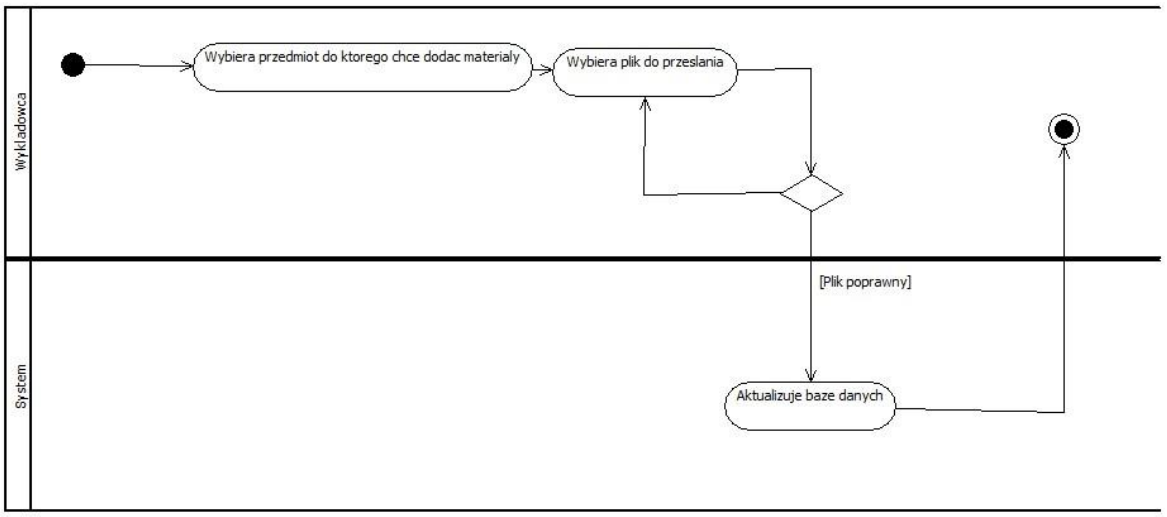
e. Wstaw oceny

Przypadek użycia	Użytkownik wstawia do systemu oceny.
Scenariusz	Użytkownik otwiera sekcję wstawiania ocen.
Warunki wstępne	● Użytkownikiem jest wykładowca. ● Użytkownik jest zalogowany. ● Użytkownik ma dostęp do przedmiotu.
Niezmienniki	Użytkownik posiada dostęp do przedmiotu.
Opis	Użytkownik uruchamia system i loguje się. Przechodzi do sekcji "przedmioty", dokonuje selekcji przedmiotu i otwiera sekcję "wstaw oceny", która umożliwia wprowadzanie ocen dla studentów. System prezentuje listę studentów zarejestrowanych na wybrany przedmiot. Użytkownik dokonuje wyboru konkretnego studenta spośród listy dostępnych opcji. Po wybraniu studenta, użytkownik wpisuje oceny dla tego studenta, zgodnie z określonymi kryteriami oceniania. System aktualizuje bazę danych.
Warunki końcowe	Użytkownik pomyślnie wprowadził oceny do systemu dla wybranego przedmiotu i wybranych studentów.
Źródła możliwych błędów	Nieprawidłowe wprowadzenie ocen, takie jak niepoprawny format lub wartości ocen.

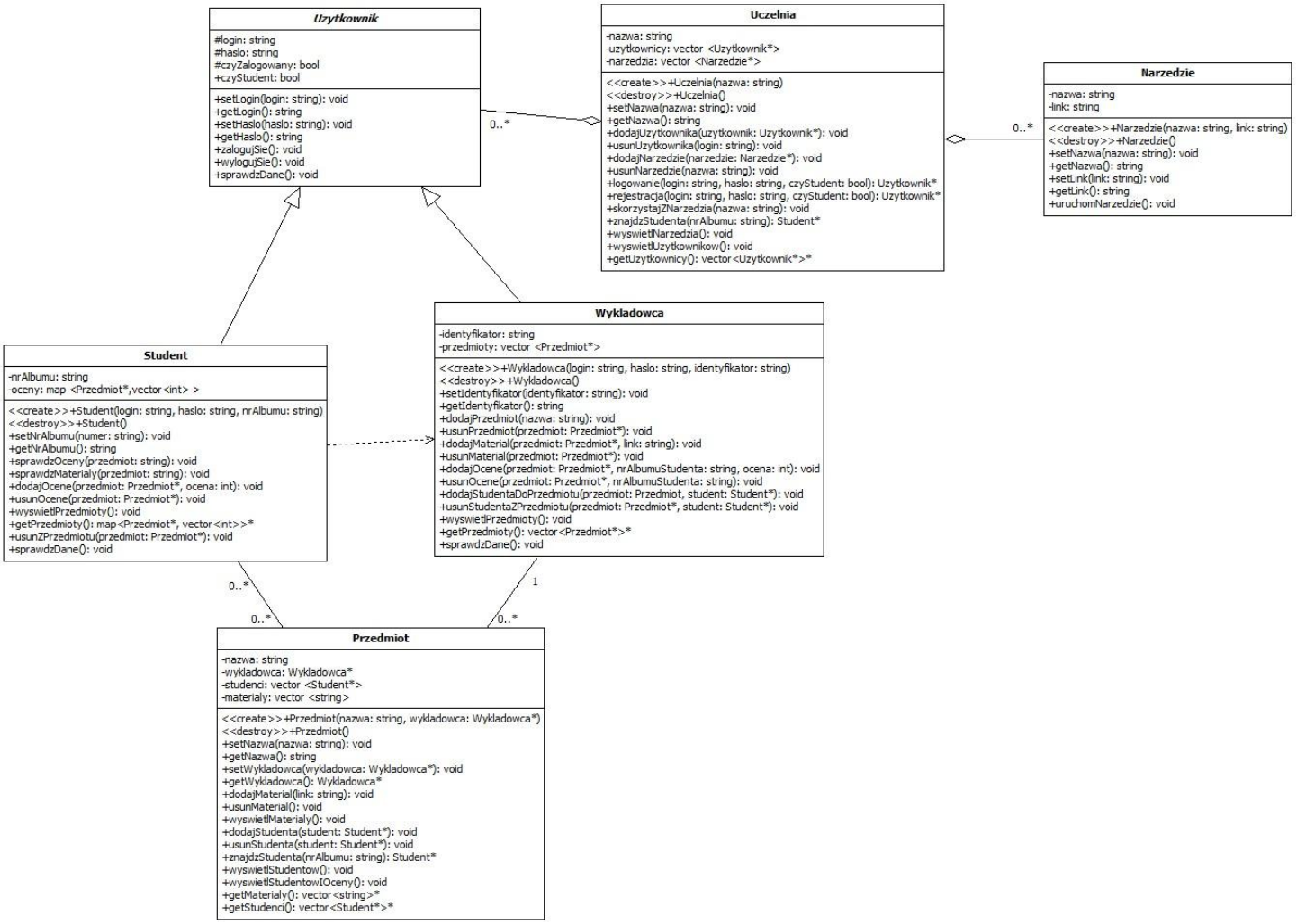


f. Dodaj materiały z przedmiotów

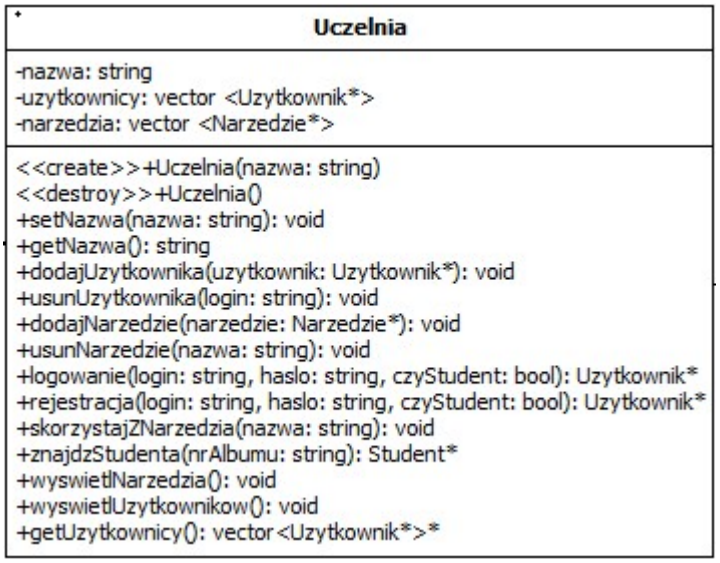
Przypadek użycia	Użytkownik wstawia do wskazanego przedmiotu materiały.
Scenariusz	Użytkownik otwiera sekcję wstawiania materiałów.
Warunki wstępne	• Użytkownikiem jest wykładowca. • Użytkownik jest zalogowany. • Użytkownik ma dostęp do wybranego przedmiotu.
Niezmienniki	Użytkownik dodaje materiały.
Opis	Użytkownik uruchamia system i loguje się. Przechodzi do sekcji "przedmioty", dokonuje selekcji przedmiotu i otwiera sekcję "dodaj materiały", która umożliwia zamieszczanie materiałów dla studentów i przesyła plik. Jeśli przesłany plik jest poprawny, system zapisuje materiał w bazie danych, przypisując go do odpowiedniego przedmiotu.
Warunki końcowe	Użytkownik pomyślnie dodał materiały do systemu dla wybranego przedmiotu.
Źródła możliwych błędów	Konflikt nazw plików.



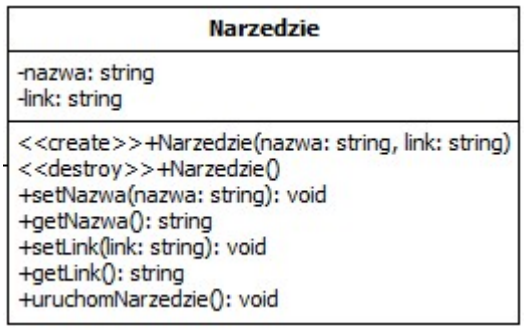
3. Model logiczny dziedziny problemu



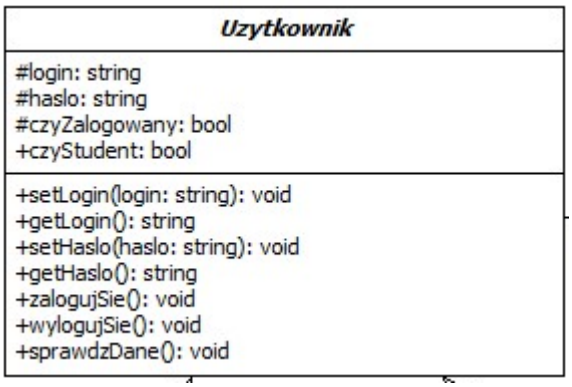
Klasa Uczelnia przechowuje nazwę uczelni, wektor użytkowników oraz wektor narzędzi. W metodach znajdziemy dostęp do dodawania/usuwania użytkowników i narzędzi, logowanie z opcjonalną rejestracją, opcję skorzystania z narzędzi dydaktycznych.



Klasa Narzędzie odpowiada za obsługę narzędzi dydaktycznych. Przechowuje nazwę i link do określonego narzędzia. Posiada również funkcję do symbolicznego uruchomienia narzędzia.

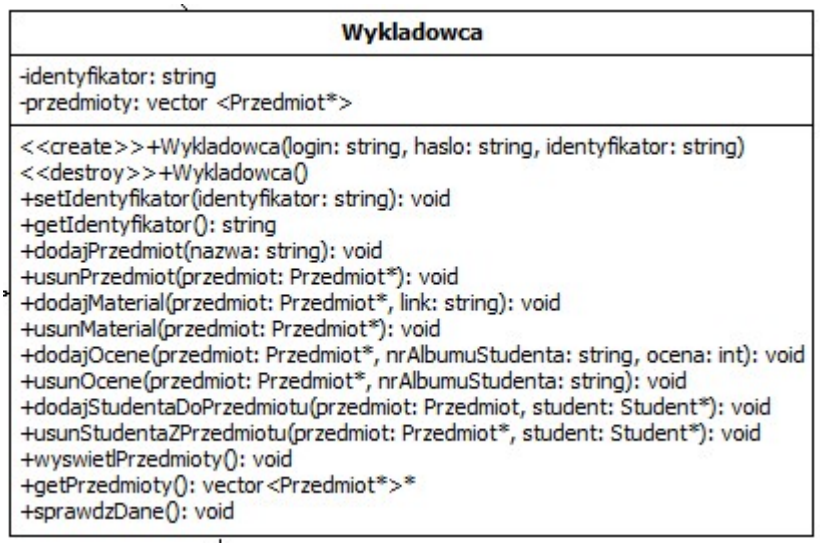


Klasa Użytkownik jest klasą abstrakcyjną, uogólnieniem klasy **Student** oraz **Wykładowca**. Odpowiada za logowanie i wylogowanie z systemu. W atrybutach znajdziemy unikalny login, wybrane hasło oraz zmienne pomocnicze do operacji na użytkowniku.

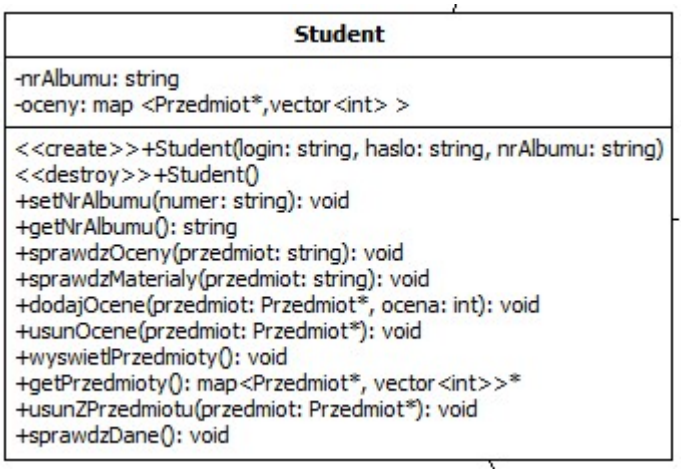


Klasa Wykładowca w atrybutach posiada unikalny identyfikator i wektor przedmiotów które prowadzi wykładowca oraz zawiera metody którym wpływa na system. Obejmuje to operowanie na przedmiotach - zarządzanie:

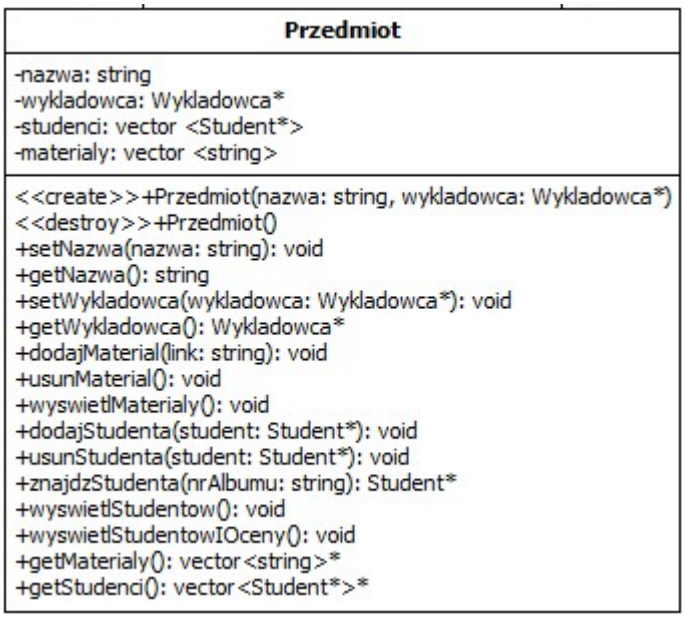
- materiałami
- ocenami
- uczestnikami



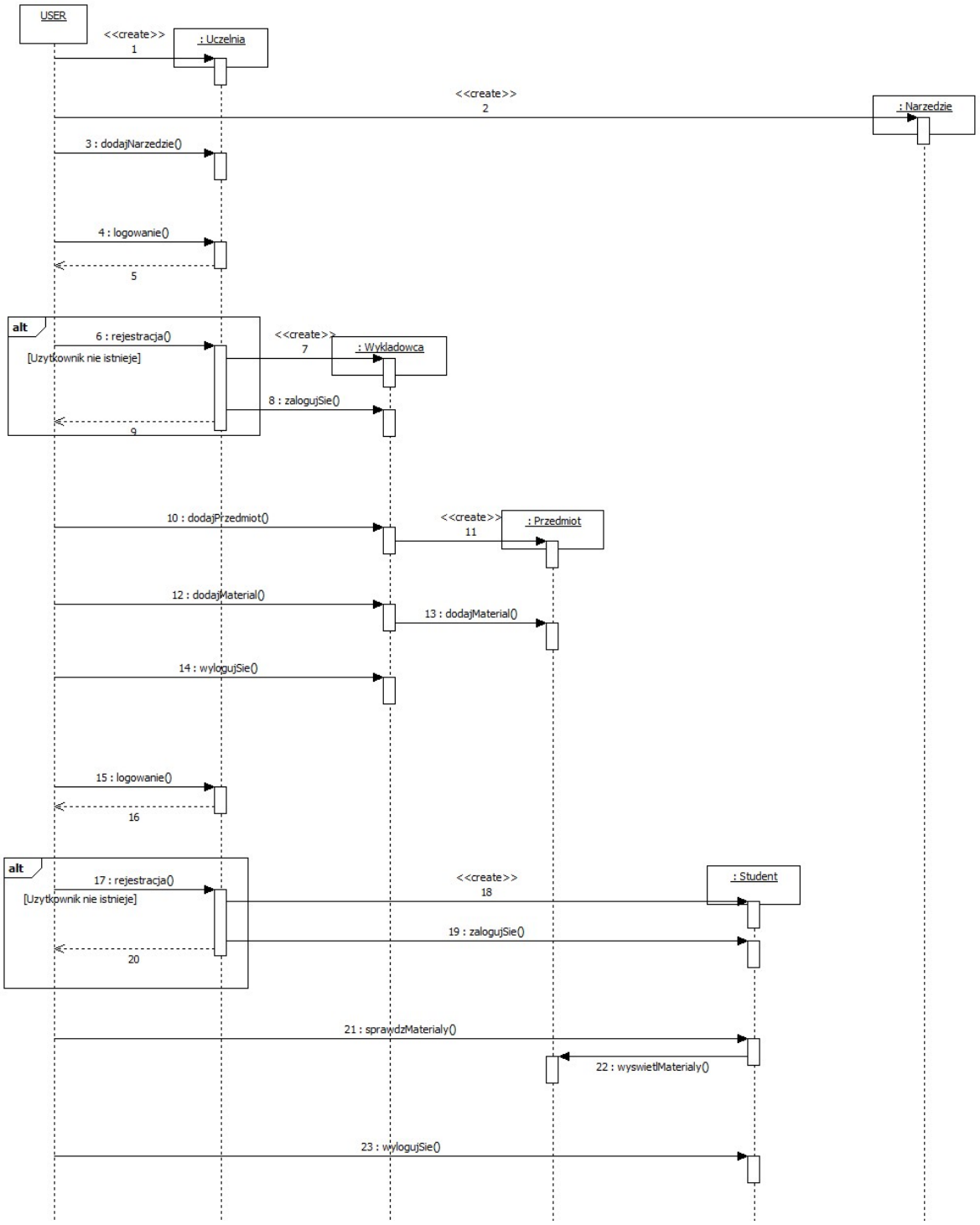
Klasa Student przechowuje metody dostępne dla studenta którymi ma wgląd w przedmiot którego jest uczestnikiem. Znajdują się tu dodatkowo metody pomocnicze zwracające wskaźnik potrzebny do operacji na studencie. Każdy student posiada również swój unikalny numer albumu i mapę ocen skojarzoną z przedmiotami w których uczestniczy



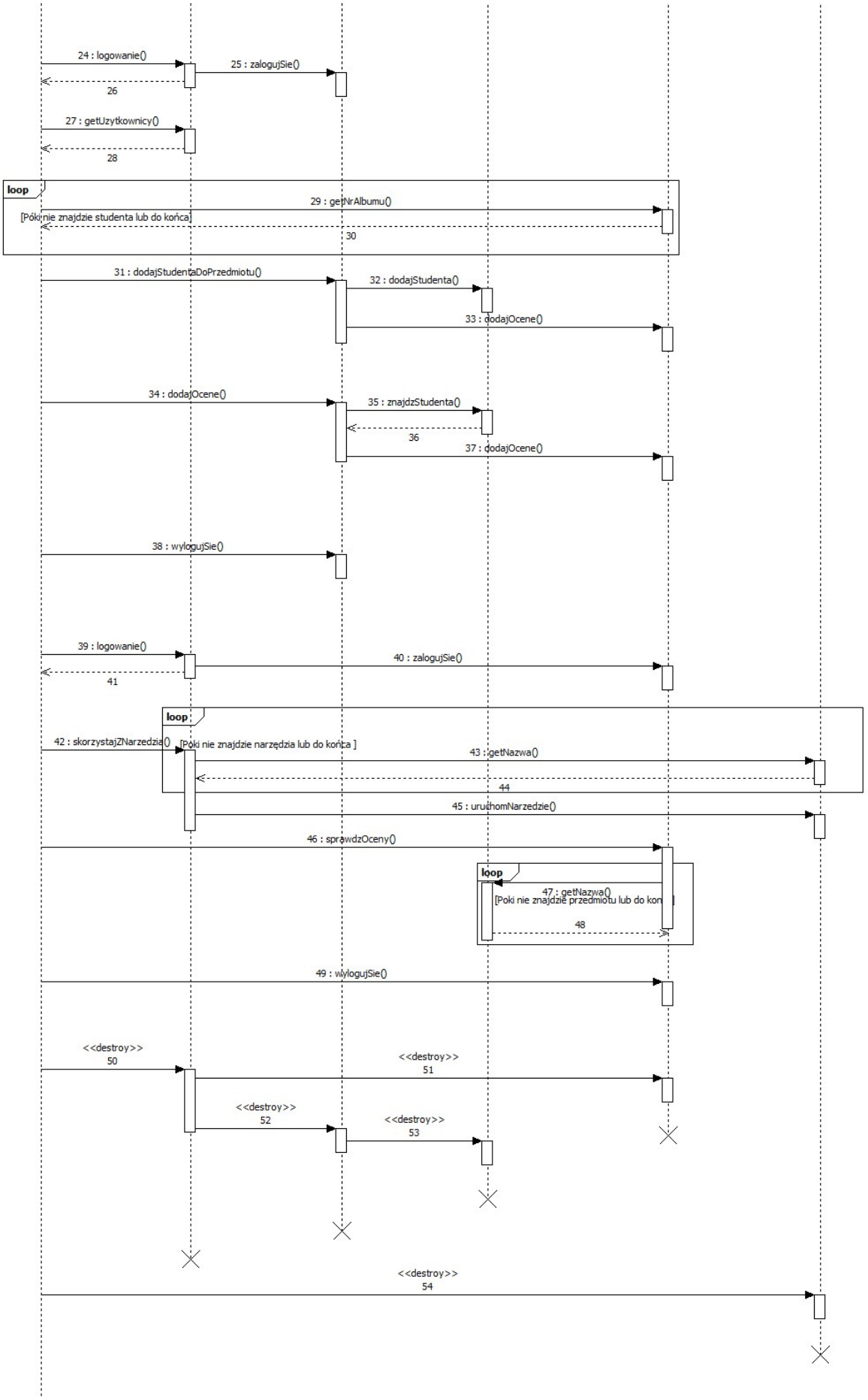
Klasa Przedmiot opisuje prowadzone zajęcia na uczelni. Każdy przedmiot posiada swoją nazwę, prowadzącego wykładowcę, przypisanych do niego studentów oraz dostępne materiały dydaktyczne. Metody pozwalają na edycję powyższych punktów.



4. Diagram sekwencji



...


24 : logowanie()
25 : zalogujSie()
26
27 : getUzytkownicy()
28
loop
[Póki nie znajdzie studenta lub do końca]
29 : getNrAlbumu()
30
31 : dodajStudentaDoPrzedmiotu()
32 : dodajStudenta()
33 : dodajOcene()
34 : dodajOcene()
35 : znajdzStudenta()
36
37 : dodajOcene()
38 : wylogujSie()
39 : logowanie()
40 : zalogujSie()
41
loop
42 : skorzystajZNarzedzia()
[Poki nie znajdzie narzędzia lub do końca]
43 : getNazwa()
44
45 : uruchomNarzedzie()
46 : sprawdzOceny()
loop
47 : getNazwa()
[Poki nie znajdzie przedmiotu lub do kon
48
49 : wylogujSie()
<<destroy>>
50
<<destroy>>
51
<<destroy>>
52
<<destroy>>
53
<<destroy>>
54

Na diagramie sekwencji przedstawiono przykładowy, podstawowy cykl użytkowania programu. Sekwencja rozpoczyna się od uruchomienia systemu i stworzenia wymaganych obiektów.

Następnie następuje próba zalogowania i ewentualna rejestracja wykładowcy. W trakcie swojej sesji tworzy on nowy przedmiot i dodaje do niego materiały, po czym wylogowuje się.

Następuje zmiana użytkownika systemu i tym razem loguje się (rejestruje) student. Po uzyskaniu dostępu, sprawdza on dostępne materiały i kończy swoją sesję.

W dalszej sekwencji, dostęp do systemu przejmuje ponownie wykładowca, który dodaje studenta do swojego przedmiotu - program najpierw pobiera listę studentów, po czym sprawdza czy szukany student jest w bazie. Po dodaniu studenta, wykładowca wystawia mu ocenę i wylogowuje się.

Na koniec działania systemu kontrola znowu przechodzi do studenta, który korzysta z narzędzi dydaktycznych i sprawdza swoje oceny.

Po wylogowaniu studenta, program kończy działanie, a wszystkie obiekty pozostałe w pamięci są usunięte.

5. Implementacja

a. Narzedzie.h

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Untitled
//  @ File Name : Narzedzie.h
//  @ Date : 24.05.2023
//  @ Author :
//
//


#if !defined(_NARZEDZIE_H)
#define _NARZEDZIE_H

#include <iostream>
#include <string>

using std::cout, std::cin;
using std::string;

class Narzedzie {
public:
	Narzedzie(string nazwa, string link);
	~Narzedzie();
	void setNazwa(string nazwa);
	string getNazwa();
	void setLink(string link);
	string getLink();
	void uruchomNarzedzie();
private:
	string nazwa;
	string link;
};

#endif  //_NARZEDZIE_H
```

b. Narzedzie.cpp

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Untitled
//  @ File Name : Narzedzie.cpp
//  @ Date : 24.05.2023
//  @ Author :
//
//


#include "Narzedzie.h"

Narzedzie::Narzedzie(string nazwa, string link) {
    cout << "Stworzono obiekt klasy " << "Narzedzie!\n";
    this->nazwa = nazwa;
    this->link = link;
}

Narzedzie::~Narzedzie() {
    cout << "Usunieto obiekt klasy " << "Narzedzie!\n";
}

void Narzedzie::setNazwa(string nazwa) {
    this->nazwa = nazwa;
}

string Narzedzie::getNazwa() {
    return this->nazwa;
}

void Narzedzie::setLink(string link) {
    this->link = link;
}

string Narzedzie::getLink() {
    return this->link;
}

void Narzedzie::uruchomNarzedzie() {
    cout << "Uruchomiono narzedzie: " << nazwa << "\n";
    cout << "Link: " << link << "\n";
```

```
}
```

c. Przedmiot.h

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Untitled
//  @ File Name : Przedmiot.h
//  @ Date : 24.05.2023
//  @ Author :
//
//


#if !defined(_PRZEDMIOT_H)
#define _PRZEDMIOT_H

#include <iostream>
#include <string>
#include <vector>

using std::cout, std::cin;
using std::string;
using std::vector;

#include "Wykladowca.h"
#include "Student.h"

class Student;
class Wykladowca;

class Przedmiot {
public:
      Przedmiot(string nazwa, Wykladowca* wykladowca);
      ~Przedmiot();
      void setNazwa(string nazwa);
      string getNazwa();
      void setWykladowca(Wykladowca* wykladowca);
      Wykladowca* getWykladowca();
      void dodajMaterial(string link);
      void usunMaterial();
      void wyswietlMaterialy();
      void dodajStudenta(Student* student);
      void usunStudenta(Student* student);
      Student* znajdzStudenta(string nrAlbumu);
```

```
	void wyswietlStudentow();
	void wyswietlStudentowIOceny(string przedmiot);
	vector <string>* getMaterialy();
	vector <Student*>* getStudenci();
private:
	string nazwa;
	Wykladowca* wykladowca;
	vector <Student*> studenci;
	vector <string> materialy;
};

#endif  //_PRZEDMIOT_H
```

d. Przedmiot.cpp

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Untitled
//  @ File Name : Przedmiot.cpp
//  @ Date : 24.05.2023
//  @ Author :
//
//


#include "Przedmiot.h"

Przedmiot::Przedmiot(string nazwa, Wykladowca* wykladowca) {
    cout << "Stworzono obiekt klasy " << "Przedmiot!\n";
    this->nazwa = nazwa;
    this->wykladowca = wykladowca;
}

Przedmiot::~Przedmiot() {
    cout << "Usunieto obiekt klasy " << "Przedmiot!\n";
}

void Przedmiot::setNazwa(string nazwa) {
    this->nazwa = nazwa;
}

string Przedmiot::getNazwa() {
    return this->nazwa;
}

void Przedmiot::setWykladowca(Wykladowca* wykladowca) {
    this->wykladowca = wykladowca;
}

Wykladowca* Przedmiot::getWykladowca() {
    return this->wykladowca;
}

void Przedmiot::dodajMaterial(string link) {
    materialy.push_back(link);
    cout<<"Pomyslnie dodano material!\n";
```

```
}

void Przedmiot::usunMaterial() {
    if(!materialy.empty()) {
        materialy.pop_back();
        cout<<"Pomyslnie usunieto ostatni material!\n";
        return;
    }
    cout << "Brak materialow w bazie!\n";
}

void Przedmiot::wyswietlMaterialy() {
    for(auto x = materialy.begin(); x != materialy.end(); ++x){
        cout<<"\t> "<<(*x)<<"\n";
    }
}

void Przedmiot::dodajStudenta(Student* student) {
    studenci.push_back(student);
}

void Przedmiot::usunStudenta(Student* student) {
    for(auto x = studenci.begin(); x != studenci.end(); ++x){
        if((*x)->getNrAlbumu() == student->getNrAlbumu()){
            studenci.erase(x);
            return;
        }
    }
}

Student* Przedmiot::znajdzStudenta(string nrAlbumu){
    for(auto x = studenci.begin(); x != studenci.end(); ++x){
        if((*x)->getNrAlbumu() == nrAlbumu){
            return *x;
        }
    }
    cout<<"Nie ma takiego studenta!\n";
    return NULL;
}

void Przedmiot::wyswietlStudentow() {
    for(auto x = studenci.begin(); x != studenci.end(); ++x){
        cout<<"\t> nrAlbumu: "<<(*x)->getNrAlbumu()<<", Login:
"<<(*x)->getLogin()<<"\n";
    }
}
```

```
void Przedmiot::wyswietlStudentowIOceny(string przedmiot) {
    for(auto x = studenci.begin(); x != studenci.end(); ++x){
        cout<<"\t> nrAlbumu: "<<(*x)->getNrAlbumu()<<", Login:
"<<(*x)->getLogin()<<", Oceny: ";
        (*x)->sprawdzOceny((przedmiot));
        cout<<"\n";
    }
}

vector <string>* Przedmiot::getMaterialy(){
    return &materialy;
}

vector <Student*>* Przedmiot::getStudenci(){
    return &studenci;
}
```

e. Student.h

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Untitled
//  @ File Name : Student.h
//  @ Date : 24.05.2023
//  @ Author :
//
//


#if !defined(_STUDENT_H)
#define _STUDENT_H

#include <iostream>
#include <string>
#include <map>
#include <vector>

using std::cout, std::cin;
using std::string;
using std::map;

#include "Uzytkownik.h"
#include "Przedmiot.h"

class Przedmiot;

class Student : public Uzytkownik {
public:
	Student(string login, string haslo, string nrAlbumu);
	~Student();
	void setNrAlbumu(string numer);
	string getNrAlbumu();
	void sprawdzOceny(string przedmiot);
	void sprawdzMaterialy(string przedmiot);
	void dodajOcene(Przedmiot* przedmiot, int ocena);
	void usunOcene(Przedmiot* przedmiot);
	void wyswietlPrzedmioty();
	map<Przedmiot*, vector<int>>* getPrzedmioty();
	void usunZPrzedmiotu(Przedmiot* przedmiot);
	void sprawdzDane();
```

```
private:
	string nrAlbumu;
	map <Przedmiot*,vector<int> > oceny;
};

#endif  //_STUDENT_H
```

f. Student.cpp

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Untitled
//  @ File Name : Student.cpp
//  @ Date : 24.05.2023
//  @ Author :
//
//


#include "Student.h"

Student::Student(string login, string haslo, string nrAlbumu) {
    cout << "Stworzono obiekt klasy " << "Student!\n";
    this->login = login;
    this->haslo = haslo;
    this->nrAlbumu = nrAlbumu;
    this->czyStudent = true;
}

Student::~Student() {
    cout << "Usunieto obiekt klasy " << "Student!\n";

}

void Student::setNrAlbumu(string numer) {
    this->nrAlbumu = numer;
}

string Student::getNrAlbumu() {
    return this->nrAlbumu;
}

void Student::sprawdzOceny(string przedmiot) {
    bool brakOcen = true;
    for(auto it = oceny.begin(); it != oceny.end(); ++it){
        if((*it).first->getNazwa() == przedmiot){
            for(auto x = (*it).second.begin(); x!= (*it).second.end();
++x){
                cout << (*x) << " ";
                brakOcen = false;
```

```
            }
            if(brakOcen) {
                cout<<"Brak ocen z przedmiotu";
            }
            return;
        }
    }
    cout << "Nie znaleziono przedmiotu!\n";
}

void Student::sprawdzMaterialy(string przedmiot) {
    for(auto it = oceny.begin(); it != oceny.end(); ++it){
        if((*it).first->getNazwa() == przedmiot){
            cout << "Materialy z przedmiotu " << przedmiot << ":\n";
            (*it).first->wyswietlMaterialy();
            cout << "\n";
            return;
        }
    }
    cout << "Nie znaleziono przedmiotu!\n";
}

void Student::dodajOcene(Przedmiot* przedmiot, int ocena) {
    if(ocena != -1){
        oceny[przedmiot].push_back(ocena);
    } else {
        vector <int> tmp;
        oceny[przedmiot] = tmp;
    }
}

void Student::usunOcene(Przedmiot* przedmiot) {
    if(!oceny[przedmiot].empty()) {
        oceny[przedmiot].pop_back();
        cout<<"Pomyslnie usunieto ostatnia ocene!\n";
    } else {
        cout<<"Brak ocen do usuniecia!\n";
    }
}

void Student::wyswietlPrzedmioty(){
    for(auto x: oceny){
        cout <<"\t> "<<(x.first)->getNazwa() << "\n";
    }
}
```

```
map<Przedmiot*, vector<int>>* Student::getPrzedmioty(){
    return &oceny;
}

void Student::usunZPrzedmiotu(Przedmiot* przedmiot) {
    oceny[przedmiot].clear();
    oceny.erase(przedmiot);
}

void Student::sprawdzDane() {
    cout<<"Witaj studencie!\n";
    cout<<"Login: "<<login<<"\n";
    cout<<"Nr. albumu: "<<nrAlbumu<<"\n";
}
```

g. Uczelnia.h

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Untitled
//  @ File Name : Uczelnia.h
//  @ Date : 24.05.2023
//  @ Author :
//
//


#if !defined(_UCZELNIA_H)
#define _UCZELNIA_H

#include <iostream>
#include <string>
#include <vector>

using std::cout, std::cin;
using std::string;

#include "Uzytkownik.h"
#include "Student.h"
#include "Narzedzie.h"

class Uczelnia {
public:
	Uczelnia(string nazwa);
	~Uczelnia();
	void setNazwa(string nazwa);
	string getNazwa();
	void dodajUzytkownika(Uzytkownik* uzytkownik);
	void usunUzytkownika(string login);
	void dodajNarzedzie(Narzedzie* narzedzie);
	void usunNarzedzie(string nazwa);
	Uzytkownik* logowanie(string login, string haslo, bool
czyStudent);
	Uzytkownik* rejestracja(string login, string haslo, bool
czyStudent);
	void skorzystajZNarzedzia(string nazwa);
	Student* znajdzStudenta(string nrAlbumu);
	void wyswietlNarzedzia();
```

```
	void wyswietlUzytkownikow();
	vector <Uzytkownik*>* getUzytkownicy();
private:
	string nazwa;
	vector <Uzytkownik*> uzytkownicy;
	vector <Narzedzie*> narzedzia;
};

#endif  //_UCZELNIA_H
```

h. Uczelnia.cpp

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Untitled
//  @ File Name : Uczelnia.cpp
//  @ Date : 24.05.2023
//  @ Author :
//
//

#include "Uczelnia.h"

Uczelnia::Uczelnia(string nazwa) {
    cout << "Stworzono obiekt klasy " << "Uczelnia!\n";
    this->nazwa = nazwa;
}

Uczelnia::~Uczelnia() {
    for(auto x = uzytkownicy.begin(); x != uzytkownicy.end(); ++x){
        if((*x)->czyStudent) delete (Student*)*x;
        else delete (Wykladowca*)*x;
    }

    cout << "Usunieto obiekt klasy " << "Uczelnia!\n";
}

void Uczelnia::setNazwa(string nazwa) {
    this->nazwa = nazwa;
}

string Uczelnia::getNazwa() {
    return this->nazwa;
}

void Uczelnia::dodajUzytkownika(Uzytkownik* uzytkownik) {
    this->uzytkownicy.push_back(uzytkownik);
}

void Uczelnia::usunUzytkownika(string login) {
    for(auto x = uzytkownicy.begin(); x != uzytkownicy.end(); ++x){
        if((*x)->getLogin() == login){
            delete *x;
```

```
            uzytkownicy.erase(x);
            return;
        }
    }
}

void Uczelnia::dodajNarzedzie(Narzedzie* narzedzie) {
    this->narzedzia.push_back(narzedzie);
}

void Uczelnia::usunNarzedzie(string nazwa) {
    for(auto x = narzedzia.begin(); x != narzedzia.end(); ++x){
        if((*x)->getNazwa() == nazwa){
            narzedzia.erase(x);
            return;
        }
    }
}

Uzytkownik* Uczelnia::logowanie(string login, string haslo, bool
czyStudent) {
    if(uzytkownicy.empty()){
        cout << "Brak uzytkownikow!\n";
        return NULL;
    }

    for(auto x = uzytkownicy.begin(); x != uzytkownicy.end(); ++x){
        if((*x)->getLogin() == login && (*x)->czyStudent == czyStudent){
            if((*x)->getHaslo() == haslo){
                (*x)->zalogujSie();
                return *x;
            }
            cout << "Nieprawidlowe haslo!\n";
            return NULL;
        }
    }

    cout << "Nie znaleziono uzytkownika z takim loginem!\n";
    return NULL;
}

Uzytkownik* Uczelnia::rejestracja(string login, string haslo, bool
czyStudent) {
    for(auto x = uzytkownicy.begin(); x != uzytkownicy.end(); ++x){
        if((*x)->getLogin() == login){
            cout << "Konto z podanym loginem juz istnieje!\n";
```

```
            return NULL;
        }
    }

    if(czyStudent){
        cout << "Podaj numer albumu: ";
        string nrAlbumu;
        cin >> nrAlbumu;
        Student* student = new Student(login, haslo, nrAlbumu);
        student->zalogujSie();
        uzytkownicy.push_back(student);
        return student;
    } else {
        cout << "Podaj identyfikator: ";
        string identyfikator;
        cin >> identyfikator;
        Wykladowca* wykladowca = new Wykladowca(login, haslo,
identyfikator);
        wykladowca->zalogujSie();
        uzytkownicy.push_back(wykladowca);
        return wykladowca;
    }
}

void Uczelnia::skorzystajZNarzedzia(string nazwa) {
    for(auto x = narzedzia.begin(); x != narzedzia.end(); ++x){
        if((*x)->getNazwa() == nazwa){
            (*x)->uruchomNarzedzie();
            return;
        }
    }
    cout << "Nie znaleziono narzedzia!\nPowrot do ekranu
poczatkowego\n";
}

Student* Uczelnia::znajdzStudenta(string nrAlbumu){
    for(auto x = uzytkownicy.begin(); x != uzytkownicy.end(); ++x){
        if((*x)->czyStudent){
            Student* tmp = (Student*)(*x);
            if(tmp->getNrAlbumu() == nrAlbumu){
                return tmp;
            }
        }
    }

    cout << "Nie znaleziono studenta z takim numerem albumu!\n";
```

```
    return nullptr;
}

void Uczelnia::wyswietlNarzedzia(){
    for(auto x = narzedzia.begin(); x != narzedzia.end(); ++x){
        cout << "\t> " << (*x)->getNazwa() << " (" << (*x)->getLink() <<
")\n";
     }
}

void Uczelnia::wyswietlUzytkownikow(){
    for(auto x = uzytkownicy.begin(); x != uzytkownicy.end(); ++x){
        if((*x)->czyStudent) {
            cout<<"\t> STUDENT - nr Albumu:
"<<((Student*)(*x))->getNrAlbumu()<<", Login: "<<(*x)->getLogin()<<"\n";
        } else {
            cout<<"\t> WYKLADOWCA - nr Identyfikatora:
"<<((Wykladowca*)(*x))->getIdentyfikator()<<", Login:
"<<(*x)->getLogin()<<"\n";
        }
     }
}

vector <Uzytkownik*>* Uczelnia::getUzytkownicy() {
    return &uzytkownicy;
}
```

i. Uzytkownik.h

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Untitled
//  @ File Name : Uzytkownik.h
//  @ Date : 24.05.2023
//  @ Author :
//
//


#if !defined(_UZYTKOWNIK_H)
#define _UZYTKOWNIK_H

#include <iostream>
#include <string>

using std::cout, std::cin;
using std::string;

class Uzytkownik {
public:
	void setLogin(string login);
	string getLogin();
	void setHaslo(string haslo);
	string getHaslo();
	void zalogujSie();
	void wylogujSie();
	virtual void sprawdzDane() = 0;
	bool czyStudent;
protected:
	string login;
	string haslo;
	bool czyZalogowany;
};

#endif  //_UZYTKOWNIK_H
```

j. Uzytkownik.cpp

```cpp
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Untitled
//  @ File Name : Uzytkownik.cpp
//  @ Date : 24.05.2023
//  @ Author :
//
//


#include "Uzytkownik.h"

void Uzytkownik::setLogin(string login) {
    this->login = login;
}

string Uzytkownik::getLogin() {
    return this->login;
}

void Uzytkownik::setHaslo(string haslo) {
    this->haslo = haslo;
}

string Uzytkownik::getHaslo() {
    return this->haslo;
}

void Uzytkownik::zalogujSie() {
    this->czyZalogowany = true;
    cout << "Zalogowano!\n";
}

void Uzytkownik::wylogujSie() {
    this->czyZalogowany = false;
    cout << "Wylogowano!\n";
}
```

k. Wykladowca.h

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Untitled
//  @ File Name : Wykladowca.h
//  @ Date : 24.05.2023
//  @ Author :
//
//


#if !defined(_WYKLADOWCA_H)
#define _WYKLADOWCA_H

#include <iostream>
#include <string>
#include <vector>

using std::cout, std::cin;
using std::string;
using std::vector;

#include "Uzytkownik.h"
#include "Przedmiot.h"
#include "Student.h"

class Student;
class Przedmiot;

class Wykladowca : public Uzytkownik {
public:
	Wykladowca(string login, string haslo, string identyfikator);
	~Wykladowca();
	void setIdentyfikator(string identyfikator);
	string getIdentyfikator();
	void dodajPrzedmiot(string nazwa);
	void usunPrzedmiot(Przedmiot* przedmiot);
	void dodajMaterial(Przedmiot* przedmiot, string link);
	void usunMaterial(Przedmiot* przedmiot);
	void dodajOcene(Przedmiot* przedmiot, string nrAlbumuStudenta, int
ocena);
	void usunOcene(Przedmiot* przedmiot, string nrAlbumuStudenta);
```

```
	void dodajStudentaDoPrzedmiotu(Przedmiot* przedmiot, Student*
student);
	void usunStudentaZPrzedmiotu(Przedmiot* przedmiot, Student*
student);
	void wyswietlPrzedmioty();
	vector <Przedmiot*>* getPrzedmioty();
	void sprawdzDane();
private:
	string identyfikator;
	vector <Przedmiot*> przedmioty;
};

#endif  //_WYKLADOWCA_H
```

l. Wykladowca.cpp

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Untitled
//  @ File Name : Wykladowca.cpp
//  @ Date : 24.05.2023
//  @ Author :
//
//


#include "Wykladowca.h"

Wykladowca::Wykladowca(string login, string haslo, string identyfikator)
{
    cout << "Stworzono obiekt klasy " << "Wykladowca!\n";
    this->login = login;
    this->haslo = haslo;
    this->identyfikator = identyfikator;
    this->czyStudent = false;
}

Wykladowca::~Wykladowca() {
    for(auto x = przedmioty.begin(); x != przedmioty.end(); x++){
        delete (*x);
    }

    cout << "Usunieto obiekt klasy " << "Wykladowca!\n";
}

void Wykladowca::setIdentyfikator(string identyfikator) {
    this->identyfikator = identyfikator;
}

string Wykladowca::getIdentyfikator() {
    return this->identyfikator;
}

void Wykladowca::dodajPrzedmiot(string nazwa) {
    Przedmiot* przedmiot = new Przedmiot(nazwa, this);
    this->przedmioty.push_back(przedmiot);
```

```
}

void Wykladowca::usunPrzedmiot(Przedmiot* przedmiot) {
    for(auto student: *(przedmiot->getStudenci())) {
        usunStudentaZPrzedmiotu(przedmiot,student);
    }
    for(auto x=przedmioty.begin();x!=przedmioty.end();++x) {
        if((*x) == przedmiot) {
            przedmioty.erase(x);
            break;
        }
    }

    delete przedmiot;
    cout<<"Pomyslnie usunieto przedmiot!\n";
}

void Wykladowca::dodajMaterial(Przedmiot* przedmiot, string link) {
    przedmiot->dodajMaterial(link);
}

void Wykladowca::usunMaterial(Przedmiot* przedmiot) {
    przedmiot->usunMaterial();
}

void Wykladowca::dodajOcene(Przedmiot* przedmiot, string nrAlbumu, int
ocena) {
    Student* stud = przedmiot->znajdzStudenta(nrAlbumu);
    if(stud != nullptr){
        stud->dodajOcene(przedmiot, ocena);
    }
    cout<<"Pomyslnie dodano ocene "<<ocena<<" studentowi o nr. albumu
"<<nrAlbumu<<"!\n";
}

void Wykladowca::usunOcene(Przedmiot* przedmiot, string nrAlbumu) {
    Student* stud = przedmiot->znajdzStudenta(nrAlbumu);
    if(stud != nullptr){
        stud->usunOcene(przedmiot);
    }
}

void Wykladowca::dodajStudentaDoPrzedmiotu(Przedmiot* przedmiot,
Student* student){
    przedmiot->dodajStudenta(student);
    student->dodajOcene(przedmiot, -1);
```

```
    cout<<"Pomyslnie dodano studenta do przedmiotu!\n";
}

void Wykladowca::usunStudentaZPrzedmiotu(Przedmiot* przedmiot, Student*
student){
    student->usunZPrzedmiotu(przedmiot);
    przedmiot->usunStudenta(student);
    cout<<"Pomyslnie usunieto studenta z przedmiotu!\n";

}

void Wykladowca::wyswietlPrzedmioty() {
    int i=0;
    for(auto x = przedmioty.begin(); x != przedmioty.end(); x++){
        cout<<"\t> "<<(*x)->getNazwa()<<"\n";
        ++i;
    }
    if(i==0) {
        cout<<"Brak prowadzonych przedmiotow w bazie danych!\n";
    }
}

vector <Przedmiot*>* Wykladowca::getPrzedmioty(){
    return &przedmioty;
}

void Wykladowca::sprawdzDane() {
    cout<<"Witaj wykladowco!\n";
    cout<<"Login: "<<login<<"\n";
    cout<<"Identyfikator: "<<identyfikator<<"\n";
}
```

m.main.cpp

```
#include <iostream>
#include <windows.h>
#include <conio.h>

#include "Uczelnia.h"
#include "Uzytkownik.h"
#include "Student.h"
#include "Wykladowca.h"
#include "Przedmiot.h"

using std::cout, std::cin;
using std::string;
using std::vector;


int main()
{
    Uczelnia* uczelnia = new Uczelnia("UMCS");

    Narzedzie* excel = new Narzedzie("Excel", "www.excel.com");
    Narzedzie* word = new Narzedzie("Word", "www.word.com");

    uczelnia->dodajNarzedzie(excel);
    uczelnia->dodajNarzedzie(word);

    bool EXIT_FLAG = false;
    bool error = false;

    Uzytkownik* user = NULL;

    char option = 0;

    string login, haslo;
    string wybor;

    bool czyStudent = false;

    cout << "Witaj na stronie uczelni " << uczelnia->getNazwa() <<
"!\n";
    system("pause");
    system("cls");
```

```
    while(!EXIT_FLAG){
        cout<<"\nS T R O N A   D O M O W A";
        if(user!=NULL) {
            if(user->czyStudent) {
                cout<<"   -   P A N E L   S T U D E N T A";
            } else {
                cout<<"   -   P A N E L   W Y K L A D O W C Y";
            }
        }
        cout<<"\n\n-----------------------------\n\n";
        cout << "Wybierz opcje:\n";
        if(user == NULL){                       // WYLOGOWANY

            cout << "1 - Zaloguj sie\n2 - Zarejestruj sie\n9 - Wyjdz z
programu\n\n";
            cout<<"-----------------------------\n\n";
            cin >> option;
            system("cls");

            switch(option){
                case '1':{                  // LOGOWANIE
                    cout<<"\nL O G O W A N I E\n\n";
                    cout<<"-----------------------------\n\n";
                    cout<<"1 - Logowanie dla studenta\n2 - Logowanie dla
wykladowcy\n\n";
                    cout<<"-----------------------------\n\n";
                    cin >> option;
                    system("cls");
                    switch(option){
                        case '1':{
                            cout<<"\nL O G O W A N I E   D L A   S T U D
E N T A\n\n";
                            czyStudent = true;
                            break;
                        }
                        case '2':{
                            cout<<"\nL O G O W A N I E   D L A   W Y K L
A D O W C Y\n\n";
                            czyStudent = false;
                            break;
                        }
                        default: {
                            cout<<"Brak opcji o kodzie "<<option<<"\n";
                            error = true;
                            system("pause");
```

```
                        break;
                    }
                }
                cout<<"------------------------------\n\n";
                if(error) {
                    error = false;
                    break;
                }
                cout<<"Podaj login: ";
                cin >> login;
                cout<<"Podaj haslo: ";
                cin >> haslo;
                cout<<"\n------------------------------\n\n";

                user = uczelnia->logowanie(login, haslo,
czyStudent);
                if(user != NULL){
                    break;
                } else {
                    cout << "Czy chcesz sie zarejestrowac?\n1 -
Tak\n2 - Nie\n\n";
                    cout<<"------------------------------\n\n";
                    cin >> option;
                    cout<<"\n";
                    if(option == '1'){
                        cout<<"Przejscie do rejestracji...\n";
                        system("pause");
                        system("cls");
                    } else if (option == '2'){
                        system("cls");
                        cout<<"Powrot do ekranu
poczatkowego...\n\n";
                        system("pause");
                        break;
                    } else {
                        system("cls");
                        cout<<"Bledna opcja, powrot do ekranu
poczatkowego...\n\n";
                        system("pause");
                        break;
                    }
                }
            }
            case '2':{ // REJESTRACJA
                cout<<"\nR E J E S T R A C J A\n\n";
                cout<<"------------------------------\n\n";
```

```
                    cout<<"1 - Rejestracja dla studenta\n2 - Rejestracja
dla wykladowcy\n\n";
                    cout<<"------------------------------\n\n";
                    cin >> option;
                    system("cls");
                    switch(option){
                        case '1':{
                            cout<<"\nR E J E S T R A C J A   D L A   S T
U D E N T A\n\n";
                            czyStudent = true;
                            break;
                        }
                        case '2':{
                            cout<<"\nR E J E S T R A C J A   D L A   W Y
K L A D O W C Y\n\n";
                            czyStudent = false;
                            break;
                        }
                        default: {
                            cout<<"Brak opcji o kodzie
"<<option<<"\n\n";
                            error = true;
                            system("pause");
                            break;
                        }
                    }
                    cout<<"------------------------------\n\n";
                    if(error) {
                        error = false;
                        break;
                    }

                    cout<<"Podaj login: ";
                    cin >> login;
                    cout<<"Podaj haslo: ";
                    cin >> haslo;
                    user = uczelnia->rejestracja(login, haslo,
czyStudent);
                    if(user == NULL){
                        system("pause");
                        system("cls");
                        continue;
                    }
                    system("pause");
                    break;
                }
```

```
                case '9':{
                    EXIT_FLAG = true;
                    cout << "Wyjscie z programu...\n";
                    system("pause");
                    break;
                }
                default: {
                    cout<<"Brak opcji o kodzie "<<option<<"\n";
                    system("pause");
                    break;
                }
            }

        } else if (user->czyStudent){           // STUDENT
            cout << "0 - Wyloguj sie\n1 - Narzedzia dydaktyczne\n2 -
Przedmioty\n3 - Wyswietl profil\n9 - Wyjdz z programu\n\n";
            cout<<"------------------------------\n\n";
            cin >> option;
            system("cls");
            switch(option){
                case '0':{
                    user->wylogujSie();
                    user = NULL;
                    system("pause");
                    break;
                }
                case '1':{
                    cout<<"\nN A R Z E D Z I A   D Y D A K T Y C Z N
E\n\n";
                    cout<<"------------------------------\n\n";
                    cout<<"Dostepne narzedzia dydaktyczne:\n";
                    uczelnia->wyswietlNarzedzia();
                    cout<<"\n------------------------------\n\n";
                    cout << "Wpisz nazwe wybranego oprogramowania: ";
                    cin >> wybor;
                    cout<<"\n------------------------------\n\n";
                    uczelnia->skorzystajZNarzedzia(wybor);
                    cout<<"\n";
                    system("pause");
                    break;
                }
                case '2':{
                    cout<<"\nP R Z E D M I O T Y\n\n";
                    cout<<"------------------------------\n\n";
                    map<Przedmiot*, vector<int>>* s_przedmioty =
((Student*)user)->getPrzedmioty();
```

```
                    if(s_przedmioty->empty()){
                        cout << "Nie znaleziono przedmiotow!\n";
                        system("pause");
                        break;
                    } else {
                        cout << "Wybierz przedmiot:\n";
                        for(auto x: *s_przedmioty){
                            cout << "\t> " << x.first->getNazwa() <<
"\n";
                        }
                        cout<<"\n------------------------------\n\n";
                        cin >> wybor;
                        Przedmiot* przedmiot = NULL;
                        vector<int> oceny;
                        for(auto x: *s_przedmioty){
                            if(x.first->getNazwa() == wybor){
                                przedmiot = x.first;
                                oceny = x.second;
                                break;
                            }
                        }
                        if(przedmiot == NULL){
                            cout << "Nie znaleziono przedmiotu o takiej
nazwie!\n\n";
                            system("pause");
                            break;
                        }
                        system("cls");
                        cout << "\nPrzedmiot: " << przedmiot->getNazwa()
<< "\n\n";
                        cout<<"------------------------------\n\n";
                        cout << "1 - Materialy\n2 - Oceny\n\n";
                        cout<<"------------------------------\n\n";
                        char choice;
                        cin >> choice;
                        system("cls");
                        switch(choice) {
                            case '1': {
                                cout<<"\nM A T E R I A L Y\n\n";

cout<<"------------------------------\n\n";
                                cout<<"Linki:\n";
                                przedmiot->wyswietlMaterialy();
                                cout<<"\n";
                                system("pause");
                                break;
```

```
                            }
                            case '2': {
                                cout<<"\nO C E N Y\n\n";

cout<<"-----------------------------\n\n";
                                cout << "Oceny z przedmiotu " <<
przedmiot->getNazwa() << ": ";
                                for(auto it = oceny.begin(); it !=
oceny.end(); ++it){

                                    cout << *it;
                                    if(it != oceny.end()-1){
                                        cout << ", ";
                                    }
                                }
                                cout << "\n\n";
                                system("pause");
                                break;
                            }
                            default: {
                                cout<<"Brak opcji o kodzie
"<<option<<"\n";

                                system("pause");
                                break;
                            }
                        }
                    break;
                    }
                }
                case '3': {
                    system("cls");
                    cout<<"\nP R O F I L\n\n";
                    cout<<"-----------------------------\n\n";
                    ((Student*)user)->sprawdzDane();
                    cout<<"\n";
                    system("pause");
                    break;
                }
                case '9':{
                    EXIT_FLAG = true;
                    cout << "Wyjscie z programu...\n";
                    system("pause");
                    break;
                }
                default: {
                    cout<<"Brak opcji o kodzie "<<option<<"\n";
                    system("pause");
```

```
                    break;
                }
            }

        } else {                                    // WYKLADOWCA
            cout << "0 - Wyloguj sie\n1 - Narzedzia dydaktyczne\n2 -
Przedmioty\n3 - Wyswietl profil\n9 - Wyjdz z programu\n\n";
            cout<<"------------------------------\n\n";
            cin >> option;
            system("cls");
            switch(option){
                case '0':{
                    user->wylogujSie();
                    user = NULL;
                    system("pause");
                    break;
                }
                case '1':{
                    cout<<"\nN A R Z E D Z I A   D Y D A K T Y C Z N
E\n\n";
                    cout<<"------------------------------\n\n";
                    cout<<"Dostepne oprogramowanie:\n";
                    uczelnia->wyswietlNarzedzia();
                    cout<<"\n------------------------------\n\n";
                    cout << "Wpisz nazwe wybranego oprogramowania: ";
                    cin >> wybor;
                    cout<<"\n------------------------------\n\n";
                    uczelnia->skorzystajZNarzedzia(wybor);
                    cout<<"\n";
                    system("pause");
                    break;
                }
                case '2': {
                    cout<<"\nP R Z E D M I O T Y\n\n";
                    cout<<"------------------------------\n\n";
                    if(((Wykladowca*)user)->getPrzedmioty()->empty()){
                        cout << "Nie znaleziono przedmiotow!\n";
                    } else {
                        cout<<"Prowadzone przedmioty:\n";
                        ((Wykladowca*)user)->wyswietlPrzedmioty();
                    }
                    cout<<"\n------------------------------\n\n";

                    cout << "Wybierz przedmiot, lub wpisz nazwe dla
nowego przedmiotu:\n";
                    cin >> wybor;
```

```
                    system("cls");

                    Przedmiot* przedmiot = NULL;
                    for(auto x: *((Wykladowca*)user)->getPrzedmioty()){
                        if(x->getNazwa() == wybor){
                            przedmiot = x;
                            break;
                        }
                    }

                    if(przedmiot == NULL){
                        ((Wykladowca*)user)->dodajPrzedmiot(wybor);
                        cout << "Dodano nowy przedmiot \"" << wybor <<
"\"\n";
                        przedmiot =
((Wykladowca*)user)->getPrzedmioty()->back();
                        system("pause");
                    }

                    system("cls");
                    cout << "\nPrzedmiot: " << przedmiot->getNazwa() <<
"\n\n";
                    cout<<"------------------------------\n\n";
                    cout<<"1 - Dodaj materialy\n2 - Usun ostatni
material\n3 - Dodaj ocene\n4 - Usun ostatnia ocene\n5 - Dodaj studenta
do przedmiotu\n6 - Usun studenta z przedmiotu\n7 - Usun przedmiot\n0 -
Powrot do ekranu domowego\n\n";
                    cout<<"------------------------------\n\n";
                    cin>>option;
                    system("cls");
                    switch(option) {
                        case '0': {
                            break;
                        }
                        case '1': {
                            cout<<"\nD O D A J   M A T E R I A L Y\n\n";
                            cout<<"------------------------------\n\n";
                            cout<<"Materialy do przedmiotu:\n";
                            przedmiot->wyswietlMaterialy();

cout<<"\n------------------------------\n\n";
                            cout<<"Podaj link do materialu: ";
                            string link;
                            cin>>link;

((Wykladowca*)user)->dodajMaterial(przedmiot,link);
```

```
                            break;
                        }
                        case '2': {
                            cout<<"\nU S U N   O S T A T N I   M A T E R
I A L\n\n";
                            cout<<"------------------------------\n\n";
                            cout<<"Materialy do przedmiotu:\n";
                            przedmiot->wyswietlMaterialy();

cout<<"\n------------------------------\n\n";

((Wykladowca*)user)->usunMaterial(przedmiot);
                            break;
                        }
                        case '3': {
                            cout<<"\nD O D A J   O C E N E\n\n";
                            cout<<"------------------------------\n\n";
                            cout<<"Studenci zapisani na przedmiot i ich
oceny:\n";

przedmiot->wyswietlStudentowIOceny(przedmiot->getNazwa());

cout<<"\n------------------------------\n\n";
                            int ocena;
                            string nrAlbumu;
                            cout<<"Podaj numer albumu studenta: ";
                            cin>>nrAlbumu;
                            bool err = true;
                            for(Student* x:
*(przedmiot->getStudenci())){
                                if(x->getNrAlbumu() == nrAlbumu){
                                    err = false;
                                    break;
                                }
                            }
                            if(err) {
                                cout << "Nie ma takiego studenta!\n";
                                break;
                            }

                            cout<<"Podaj ocene: ";
                            cin>>ocena;
                            while(ocena < 2 || ocena > 5){
                                cout << "Zla ocena, wpisz poprawna: ";
                                cin>>ocena;
                            }
```

```
((Wykladowca*)user)->dodajOcene(przedmiot,nrAlbumu,ocena);
                            break;
                        }
                        case '4': {
                            cout<<"\nU S U N  O S T A T N I A  O C E N
E\n\n";
                            cout<<"------------------------------\n\n";
                            cout<<"Studenci zapisani na przedmiot i ich
oceny:\n";

przedmiot->wyswietlStudentowIOceny(przedmiot->getNazwa());

cout<<"\n------------------------------\n\n";
                            string nrAlbumu;
                            cout<<"Podaj numer albumu studenta: ";
                            cin>>nrAlbumu;
                            bool err = true;
                            for(Student* x:
*(przedmiot->getStudenci())){
                                if(x->getNrAlbumu() == nrAlbumu){
                                    err = false;
                                    break;
                                }
                            }
                            if(err) {
                                cout << "Nie ma takiego studenta!\n";
                                break;
                            }

((Wykladowca*)user)->usunOcene(przedmiot,nrAlbumu);
                            break;
                        }
                        case '5': {
                            cout<<"\nD O D A J   S T U D E N T A   D O
P R Z E D M I O T U\n\n";
                            cout<<"------------------------------\n\n";
                            cout<<"Studenci zapisani na
"<<przedmiot->getNazwa()<<":\n";
                            przedmiot->wyswietlStudentow();

cout<<"\n------------------------------\n\n";
                            cout<<"Uzytkownicy zarejestrowani w
systemie:\n";
                            uczelnia->wyswietlUzytkownikow();
```

```
cout<<"\n------------------------------\n\n";
                            string nrAlbumu;
                            cout<<"Podaj numer albumu studenta: ";
                            cin>>nrAlbumu;
                            bool err = true;
                            for(Uzytkownik* x:
*(uczelnia->getUzytkownicy())){
                                if(x->czyStudent) {
                                    if(((Student*)x)->getNrAlbumu() ==
nrAlbumu){
                                        err = false;
                                        break;
                                    }
                                }
                            }
                            if(err) {
                                cout << "Nie ma takiego studenta!\n";
                                break;
                            }

                            for(Student* x:
*(przedmiot->getStudenci())){
                                if(x->getNrAlbumu() == nrAlbumu){
                                    cout<<"Student jest juz zapisany na
ten przedmiot!\n";
                                    err = true;
                                    break;
                                }
                            }
                            if(err) {
                                break;
                            }


((Wykladowca*)user)->dodajStudentaDoPrzedmiotu(przedmiot,uczelnia->znajd
zStudenta(nrAlbumu));
                            break;
                        }
                        case '6': {
                            cout<<"\nU S U N   S T U D E N T A   Z   P R
Z E D M I O T U\n\n";
                            cout<<"------------------------------\n\n";
                            cout<<"Studenci zapisani na
"<<przedmiot->getNazwa()<<":\n";
                            przedmiot->wyswietlStudentow();
```

```
cout<<"\n------------------------------\n\n";
                            string nrAlbumu;
                            cout<<"Podaj numer albumu studenta: ";
                            cin>>nrAlbumu;
                            bool err = true;
                            for(Student* x:
*(przedmiot->getStudenci())){
                                if(x->getNrAlbumu() == nrAlbumu){
                                    err = false;
                                    break;
                                }
                            }
                            if(err) {
                                cout << "Nie ma takiego studenta
zapisanego na ten przedmiot!\n";
                                break;
                            }

((Wykladowca*)user)->usunStudentaZPrzedmiotu(przedmiot,uczelnia->znajdzS
tudenta(nrAlbumu));
                            break;
                        }
                        case '7': {
                            cout<<"\nU S U N   P R Z E D M I O T\n\n";
                            cout<<"------------------------------\n\n";
                            string conf;
                            cout<<"Potwierdz usuniecie przedmiotu
"<<przedmiot->getNazwa()<<" komenda \"USUN\": ";
                            cin>>conf;
                            if(conf == "USUN") {

((Wykladowca*)user)->usunPrzedmiot(przedmiot);
                            } else {
                                cout<<"Niepoprawna opcja, powrot do
ekranu glownego...\n\n";
                            }
                            break;
                        }
                        default: {
                            cout<<"Brak opcji o kodzie "<<option<<"\n";
                            break;
                        }
                    }
                    system("pause");
                    break;
                }
```

```
                case '3': {
                    system("cls");
                    cout<<"\nP R O F I L\n\n";
                    cout<<"------------------------------\n\n";
                    ((Wykladowca*)user)->sprawdzDane();
                    cout<<"\n";
                    system("pause");
                    break;
                }
                case '9':{
                    EXIT_FLAG = true;
                    cout << "Wyjscie z programu...\n";
                    system("pause");
                    break;
                }
                default: {
                    cout<<"Brak opcji o kodzie "<<option<<"\n";
                    system("pause");
                    break;
                }
            }
        }
        system("cls");
    }

    // Destruktor klasy Uczelnia, usuwa wszystkie obiekty typu
Uzytkownik (Student, Wykladowca)
    // Destruktor klasy Wykladowca usuwa rowniez wszystkie obiekty typu
Przedmiot
    // W rezultacie destruktor klasy Uczelnia usuwa wszystkie obiekty
sciśle powiazane z obiektem uczelnia.

    delete excel;
    delete word;
    delete uczelnia;

    return 0;
}
```

6. Testy

a. Testy jednostkowe

Testowane były poszczególne metody i parametry wszystkich klas.

Testy Narzędzie

```
#include <cassert>
#include <iostream>
#include "../Narzedzie.h"

void testuj_nazwe(){
    Narzedzie narzedzie("nazwa", "link");
    std::cout << "\nTestuje nazwe narzedzia: ";
    assert(narzedzie.getNazwa() == "nazwa");
    std::cout << "Ok\n\n";
}

void testuj_link(){
    Narzedzie narzedzie("nazwa", "link");
    std::cout << "\nTestuje link narzedzia: ";
    assert(narzedzie.getLink() == "link");
    std::cout << "Ok\n\n";
}

int main(){
    testuj_nazwe();
    testuj_link();
    return 0;
}
```

Testy Przedmiot

```cpp
#include <cassert>
#include <iostream>
#include "../Przedmiot.h"

void testuj_nazwe(){
    Przedmiot przedmiot("nazwa", nullptr);
    std::cout << "\nTestuje nazwe przedmiotu: ";
    assert(przedmiot.getNazwa() == "nazwa");
    std::cout << "Ok\n\n";
}

void testuj_materialy(){
    Przedmiot przedmiot("nazwa", nullptr);

    std::cout << "\nTestuje materialy przedmiotu: \n\n";
    przedmiot.dodajMaterial("link1");
    przedmiot.dodajMaterial("link2");
    przedmiot.dodajMaterial("link3");

    std::vector<std::string>* materialy = przedmiot.getMaterialy();
    assert(materialy->size() == 3);
    std::cout << "> Rozmiar: " << materialy->size() << "\n";
    assert((*materialy)[0] == "link1");
    std::cout << "> Link 1: " << (*materialy)[0] << "\n";
    assert((*materialy)[1] == "link2");
    std::cout << "> Link 2: " << (*materialy)[1] << "\n";
    assert((*materialy)[2] == "link3");
    std::cout << "> Link 3: " << (*materialy)[2] << "\n";
    std::cout << "Ok\n\n";

}

int main(){
    testuj_nazwe();
    testuj_materialy();
    return 0;
}
```

Testy Student

```
#include <cassert>
#include <iostream>
#include "../Student.h"

void testuj_dane_konta(){
    Student student("Login", "Haslo", "123456");
    std::cout << "\nTestuje dane konta: \n\n";

    std::cout << "> Czy student: " << student.czyStudent << "\n";
    assert(student.czyStudent == true);

    std::cout << "> Login: " << student.getLogin() << "\n";
    assert(student.getLogin() == "Login");

    std::cout << "> Haslo: " << student.getHaslo() << "\n";
    assert(student.getHaslo() == "Haslo");

    std::cout << "> Numer albumu: " << student.getNrAlbumu() << "\n";
    assert(student.getNrAlbumu() == "123456");

    std::cout << "Ok\n\n";
}

int main(){
    testuj_dane_konta();

    return 0;
}
```

Testy Uczelnia

```
#include <cassert>
#include <iostream>
#include "../Uczelnia.h"

void testuj(){
    Uczelnia uczelnia("nazwa");
    std::cout << "\nTestuje nazwe uczelni: ";
    assert(uczelnia.getNazwa() == "nazwa");
    std::cout << "Ok\n\n";
}

int main(){
    testuj();

    return 0;
}
```

Testy Wykładowca

```
#include <cassert>
#include <iostream>
#include "../Wykladowca.h"

void testuj_dane_konta(){
    Wykladowca wykladowca("Login", "Haslo", "123456");
    std::cout << "\nTestuje dane konta: \n\n";

    std::cout << "> Czy student: " << wykladowca.czyStudent << "\n";
    assert(wykladowca.czyStudent == false);

    std::cout << "> Login: " << wykladowca.getLogin() << "\n";
    assert(wykladowca.getLogin() == "Login");

    std::cout << "> Haslo: " << wykladowca.getHaslo() << "\n";
    assert(wykladowca.getHaslo() == "Haslo");

    std::cout << "> Numer identyfikator: " <<
wykladowca.getIdentyfikator() << "\n";
    assert(wykladowca.getIdentyfikator() == "123456");

    std::cout << "Ok\n\n";
}

int main(){
    testuj_dane_konta();

    return 0;
}
```

b. Testy integracyjne

Testowane były interakcje między klasami i ich poprawność.

Bottom-up - integracja z dołu - testowane są po kolei komponenty od najniższych w hierarchii projektu, do tych położonych wyżej.

```
/*
    Testy integracyjne - bottom-up:

    - test_NarzedzieUczelnia() - testuje dodawanie i usuwanie narzedzi w
klasie Uczelnia
    - test_PrzedmiotWykladowca() - testuje dodawanie i usuwanie
przedmiotow w klasie Wykladowca
    - test_WykladowcaPrzedmiot() - testuje wykladowce w klasie Przedmiot

*/

#include <iostream>
#include <cassert>

#include "../Uczelnia.h"
#include "../Student.h"
#include "../Wykladowca.h"
#include "../Narzedzie.h"
#include "../Uzytkownik.h"
#include "../Przedmiot.h"

void test_NarzedzieUczelnia(){
    std::cout << "\nTestuje narzedzia w klasie Uczelnia:\n\n";

    Uczelnia* uczelnia = new Uczelnia("Nazwa");
    Narzedzie narzedzie_1("n1", "l1");
    Narzedzie narzedzie_2("n2", "l2");
    Narzedzie narzedzie_3("n3", "l3");

    uczelnia->dodajNarzedzie(&narzedzie_1);
    uczelnia->dodajNarzedzie(&narzedzie_2);

    uczelnia->wyswietlNarzedzia();
    std::cout << "\t---\n";

    uczelnia->usunNarzedzie("n1");
    uczelnia->usunNarzedzie("n2");
    uczelnia->dodajNarzedzie(&narzedzie_3);
```

```
    uczelnia->wyswietlNarzedzia();
    std::cout << "\t---\n";

    uczelnia->usunNarzedzie("n3");

    uczelnia->wyswietlNarzedzia();
    std::cout << "\t---\n";

    delete uczelnia;
}

void test_PrzedmiotWykladowca(){
    std::cout << "\nTestuje przedmioty w klasie Wykladowca:\n\n";

    Wykladowca* wykladowca = new Wykladowca("login", "haslo", "123456");

    wykladowca->dodajPrzedmiot("p1");
    wykladowca->dodajPrzedmiot("p2");

    std::cout << "Ilosc przedmiotow: " <<
wykladowca->getPrzedmioty()->size() << "\n";
    assert(wykladowca->getPrzedmioty()->size() == 2);
    std::cout << "Ok\n\n";

    std::cout << "Przedmiot 1: " <<
wykladowca->getPrzedmioty()->at(0)->getNazwa() << "\n";
    assert(wykladowca->getPrzedmioty()->at(0)->getNazwa() == "p1");
    std::cout << "Ok\n\n";

    std::cout << "Przedmiot 2: " <<
wykladowca->getPrzedmioty()->at(1)->getNazwa() << "\n";
    assert(wykladowca->getPrzedmioty()->at(1)->getNazwa() == "p2");
    std::cout << "Ok\n\n";


    wykladowca->usunPrzedmiot(wykladowca->getPrzedmioty()->at(0));

    std::cout << "Ilosc przedmiotow: " <<
wykladowca->getPrzedmioty()->size() << "\n";
    assert(wykladowca->getPrzedmioty()->size() == 1);
    std::cout << "Ok\n\n";

    std::cout << "Przedmiot 1: " <<
wykladowca->getPrzedmioty()->at(0)->getNazwa() << "\n";
    assert(wykladowca->getPrzedmioty()->at(0)->getNazwa() == "p2");
    std::cout << "Ok\n\n";
```

```
    wykladowca->dodajPrzedmiot("p3");

    std::cout << "Ilosc przedmiotow: " <<
wykladowca->getPrzedmioty()->size() << "\n";
    assert(wykladowca->getPrzedmioty()->size() == 2);
    std::cout << "Ok\n\n";

    std::cout << "Przedmiot 1: " <<
wykladowca->getPrzedmioty()->at(0)->getNazwa() << "\n";
    assert(wykladowca->getPrzedmioty()->at(0)->getNazwa() == "p2");
    std::cout << "Ok\n\n";

    std::cout << "Przedmiot 2: " <<
wykladowca->getPrzedmioty()->at(1)->getNazwa() << "\n";
    assert(wykladowca->getPrzedmioty()->at(1)->getNazwa() == "p3");
    std::cout << "Ok\n\n";


    delete wykladowca;
}

void test_WykladowcaPrzedmiot(){
    std::cout << "\nTestuje wykladowce w klasie Przedmiot:\n\n";

    Wykladowca* wykladowca = new Wykladowca("login", "haslo", "123456");
    Przedmiot* przedmiot = new Przedmiot("nazwa", wykladowca);

    std::cout << "Wykladowca - login: " <<
przedmiot->getWykladowca()->getLogin() << "\n";
    assert(przedmiot->getWykladowca()->getLogin() == "login");
    std::cout << "Ok\n\n";

    std::cout << "Wykladowca - haslo: " <<
przedmiot->getWykladowca()->getHaslo() << "\n";
    assert(przedmiot->getWykladowca()->getHaslo() == "haslo");
    std::cout << "Ok\n\n";

    std::cout << "Wykladowca - identyfikator: " <<
przedmiot->getWykladowca()->getIdentyfikator() << "\n";
    assert(przedmiot->getWykladowca()->getIdentyfikator() == "123456");
    std::cout << "Ok\n\n";

    delete przedmiot;
    delete wykladowca;
}
```

```cpp
int main(){
    std::cout << "\nTesty integracyjne - bottom-up:\n\n";

    std::cout << "test_NarzedzieUczelnia:\n";
    test_NarzedzieUczelnia();
    std::cout << "\n\n\n";

    std::cout << "test_PrzedmiotWykladowca:\n";
    test_PrzedmiotWykladowca();
    std::cout << "\n\n\n";

    std::cout << "test_WykladowcaPrzedmiot:\n";
    test_WykladowcaPrzedmiot();

    return 0;
}
```

Top-down - integracja z góry- testowane są po kolei komponenty od najwyższych w hierarchii projektu, do tych położonych niżej.

```
/*
    Testy integracyjne - top-down:

    - test_UzytkownicyUczelnia() - testuje dodawanie i usuwanie
uzytkownikow w klasie Uczelnia
    - test_PrzedmiotyWykladowcyUczelnia() - testuje dodawanie i usuwanie
przedmiotow wykladowcy w klasie Uczelnia

*/

#include <iostream>
#include <cassert>

#include "../Uczelnia.h"
#include "../Student.h"
#include "../Wykladowca.h"
#include "../Narzedzie.h"
#include "../Uzytkownik.h"
#include "../Przedmiot.h"

void test_UzytkownicyUczelnia(){
    std::cout << "\nTestuje uzytkownikow w klasie Uczelnia:\n\n";

    Uczelnia* uczelnia = new Uczelnia("Nazwa");
    Uzytkownik* uzytkownik_1 = new Student("login_1", "haslo_1", "123");
    Uzytkownik* uzytkownik_2 = new Student("login_2", "haslo_2", "456");
    Uzytkownik* uzytkownik_3 = new Wykladowca("login_3", "haslo_3",
"789");

    uczelnia->dodajUzytkownika(uzytkownik_1);
    uczelnia->dodajUzytkownika(uzytkownik_2);
    uczelnia->dodajUzytkownika(uzytkownik_3);

    std::cout << "Ilosc uzytkownikow: " <<
uczelnia->getUzytkownicy()->size() << "\n";
    assert(uczelnia->getUzytkownicy()->size() == 3);
    std::cout << "Ok\n\n";

    std::cout << "Uzytkownik 1 - czy jest studentem: " <<
uzytkownik_1->czyStudent << "\n";
    assert(uzytkownik_1->czyStudent == true);
    std::cout << "Ok\n\n";
```

```
    std::cout << "Uzytkownik 2 - czy jest studentem: " <<
uzytkownik_2->czyStudent << "\n";
    assert(uzytkownik_2->czyStudent == true);
    std::cout << "Ok\n\n";

    std::cout << "Uzytkownik 3 - czy jest studentem: " <<
uzytkownik_3->czyStudent << "\n";
    assert(uzytkownik_3->czyStudent == false);
    std::cout << "Ok\n\n";


    std::cout << "Usuwam uzytkownika 2\n";
    uczelnia->usunUzytkownika("login_2");
    std::cout << "Ilosc uzytkownikow: " <<
uczelnia->getUzytkownicy()->size() << "\n";
    assert(uczelnia->getUzytkownicy()->size() == 2);
    std::cout << "Ok\n\n";

    std::cout << "Znajdz studenta o numerze albumu '123':\n";
    assert(uczelnia->znajdzStudenta("123") == uzytkownik_1);
    std::cout << "Ok\n\n";

    std::cout << "Znajdz studenta o numerze albumu '456':\n";
    assert(uczelnia->znajdzStudenta("456") == nullptr);
    std::cout << "Ok\n\n";

    std::cout << "Znajdz studenta o numerze albumu '789':\n";
    assert(uczelnia->znajdzStudenta("789") == nullptr);
    std::cout << "Ok\n\n";


    std::cout << "Uzytkownik 1:\n\t - login: " <<
uzytkownik_1->getLogin() << "\n\t - haslo: " << uzytkownik_1->getHaslo()
<< "\n\t - numer albumu: " << ((Student*)uzytkownik_1)->getNrAlbumu() <<
"\n";
    assert(uzytkownik_1->getLogin() == "login_1" &&
uzytkownik_1->getHaslo() == "haslo_1" &&
((Student*)uzytkownik_1)->getNrAlbumu() == "123");
    std::cout << "Ok\n\n";

    std::cout << "Uzytkownik 3:\n\t - login: " <<
uzytkownik_3->getLogin() << "\n\t - haslo: " << uzytkownik_3->getHaslo()
<< "\n\t - numer albumu: " <<
((Wykladowca*)uzytkownik_3)->getIdentyfikator() << "\n";
    assert(uzytkownik_3->getLogin() == "login_3" &&
uzytkownik_3->getHaslo() == "haslo_3" &&
```

```
((Wykladowca*)uzytkownik_3)->getIdentyfikator() == "789");
    std::cout << "Ok\n\n";

    delete uczelnia;
}

void test_PrzedmiotyWykladowcyUczelnia(){
    std::cout << "\nTestuje przedmioty i wykladowcow w klasie
Uczelnia:\n\n";

    Uczelnia* uczelnia = new Uczelnia("Nazwa");
    Uzytkownik* uzytkownik_1 = new Wykladowca("login_1", "haslo_1",
"123");
    Uzytkownik* uzytkownik_2 = new Wykladowca("login_2", "haslo_2",
"456");

    ((Wykladowca*)uzytkownik_1)->dodajPrzedmiot("Przedmiot 1");
    ((Wykladowca*)uzytkownik_1)->dodajPrzedmiot("Przedmiot 2");
    ((Wykladowca*)uzytkownik_2)->dodajPrzedmiot("Przedmiot 3");


((Wykladowca*)uzytkownik_1)->dodajMaterial(((Wykladowca*)uzytkownik_1)->
getPrzedmioty()->at(0), "Material 1");

((Wykladowca*)uzytkownik_1)->dodajMaterial(((Wykladowca*)uzytkownik_1)->
getPrzedmioty()->at(0), "Material 2");

((Wykladowca*)uzytkownik_1)->dodajMaterial(((Wykladowca*)uzytkownik_1)->
getPrzedmioty()->at(1), "Material 3");

((Wykladowca*)uzytkownik_2)->dodajMaterial(((Wykladowca*)uzytkownik_2)->
getPrzedmioty()->at(0), "Material 4");

    uczelnia->dodajUzytkownika(uzytkownik_1);
    uczelnia->dodajUzytkownika(uzytkownik_2);

    std::cout << "Uzytkownik 1 - ilosc przedmiotow: " <<
((Wykladowca*)uzytkownik_1)->getPrzedmioty()->size() << "\n";
    assert(((Wykladowca*)uzytkownik_1)->getPrzedmioty()->size() == 2);
    std::cout << "Ok\n\n";

    std::cout << "Uzytkownik 2 - ilosc przedmiotow: " <<
((Wykladowca*)uzytkownik_2)->getPrzedmioty()->size() << "\n";
    assert(((Wykladowca*)uzytkownik_2)->getPrzedmioty()->size() == 1);
    std::cout << "Ok\n\n";
```

```
    std::cout << "Uzytkownik 1 - ilosc materialow w przedmiocie 1: " <<
((Wykladowca*)uzytkownik_1)->getPrzedmioty()->at(0)->getMaterialy()->siz
e() << "\n";

assert(((Wykladowca*)uzytkownik_1)->getPrzedmioty()->at(0)->getMaterialy
()->size() == 2);
    std::cout << "Ok\n\n";

    std::cout << "Uzytkownik 1 - ilosc materialow w przedmiocie 2: " <<
((Wykladowca*)uzytkownik_1)->getPrzedmioty()->at(1)->getMaterialy()->siz
e() << "\n";

assert(((Wykladowca*)uzytkownik_1)->getPrzedmioty()->at(1)->getMaterialy
()->size() == 1);
    std::cout << "Ok\n\n";

    std::cout << "Uzytkownik 2 - ilosc materialow w przedmiocie 1: " <<
((Wykladowca*)uzytkownik_2)->getPrzedmioty()->at(0)->getMaterialy()->siz
e() << "\n";

assert(((Wykladowca*)uzytkownik_2)->getPrzedmioty()->at(0)->getMaterialy
()->size() == 1);
    std::cout << "Ok\n\n";

    std::cout << "Uzytkownik 1 - material 1 w przedmiocie 1: " <<
((Wykladowca*)uzytkownik_1)->getPrzedmioty()->at(0)->getMaterialy()->at(
0) << "\n";

assert(((Wykladowca*)uzytkownik_1)->getPrzedmioty()->at(0)->getMaterialy
()->at(0) == "Material 1");
    std::cout << "Ok\n\n";

    std::cout << "Uzytkownik 1 - material 2 w przedmiocie 1: " <<
((Wykladowca*)uzytkownik_1)->getPrzedmioty()->at(0)->getMaterialy()->at(
1) << "\n";

assert(((Wykladowca*)uzytkownik_1)->getPrzedmioty()->at(0)->getMaterialy
()->at(1) == "Material 2");
    std::cout << "Ok\n\n";

    std::cout << "Uzytkownik 1 - material 1 w przedmiocie 2: " <<
((Wykladowca*)uzytkownik_1)->getPrzedmioty()->at(1)->getMaterialy()->at(
0) << "\n";

assert(((Wykladowca*)uzytkownik_1)->getPrzedmioty()->at(1)->getMaterialy
()->at(0) == "Material 3");
```

```
    std::cout << "Ok\n\n";

    std::cout << "Uzytkownik 2 - material 1 w przedmiocie 3: " <<
((Wykladowca*)uzytkownik_2)->getPrzedmioty()->at(0)->getMaterialy()->at(
0) << "\n";

assert(((Wykladowca*)uzytkownik_2)->getPrzedmioty()->at(0)->getMaterialy
()->at(0) == "Material 4");
    std::cout << "Ok\n\n";


((Wykladowca*)uzytkownik_1)->usunPrzedmiot(((Wykladowca*)uzytkownik_1)->
getPrzedmioty()->at(0));

((Wykladowca*)uzytkownik_2)->usunPrzedmiot(((Wykladowca*)uzytkownik_2)->
getPrzedmioty()->at(0));

    std::cout << "Uzytkownik 1 - ilosc przedmiotow: " <<
((Wykladowca*)uzytkownik_1)->getPrzedmioty()->size() << "\n";
    assert(((Wykladowca*)uzytkownik_1)->getPrzedmioty()->size() == 1);
    std::cout << "Ok\n\n";

    std::cout << "Uzytkownik 2 - ilosc przedmiotow: " <<
((Wykladowca*)uzytkownik_2)->getPrzedmioty()->size() << "\n";
    assert(((Wykladowca*)uzytkownik_2)->getPrzedmioty()->size() == 0);
    std::cout << "Ok\n\n";

    delete uczelnia;
}

int main(){
    std::cout << "\nTesty integracyjne - top-down:\n\n";

    std::cout << "test_UzytkownicyUczelnia:\n";
    test_UzytkownicyUczelnia();
    std::cout << "\n\n\n";

    std::cout << "test_PrzedmiotyWykladowcyUczelnia:\n";
    test_PrzedmiotyWykladowcyUczelnia();

    return 0;
}
```

c. Testy funkcjonalne

Testowane były oczekiwane funkcjonalności programu - zgodnie z założeniami ze scenariuszy.

Dodaj materialy z przedmiotów

```
#include <iostream>
#include <cassert>

#include "../Narzedzie.h"
#include "../Przedmiot.h"
#include "../Student.h"
#include "../Uczelnia.h"
#include "../Uzytkownik.h"
#include "../Wykladowca.h"

void test(){

	Uczelnia* uczelnia = new Uczelnia("Nazwa");

	uczelnia->dodajUzytkownika(new Wykladowca("login1", "haslo1",
"123"));


((Wykladowca*)uczelnia->getUzytkownicy()->at(0))->dodajPrzedmiot("p_1");

((Wykladowca*)uczelnia->getUzytkownicy()->at(0))->getPrzedmioty()->at(0)
->dodajMaterial("Material 1");
	std::cout << "Utworzony material: " <<
((Wykladowca*)uczelnia->getUzytkownicy()->at(0))->getPrzedmioty()->at(0)
->getMaterialy()->at(0) << "\n";

assert(((Wykladowca*)uczelnia->getUzytkownicy()->at(0))->getPrzedmioty()
->at(0)->getMaterialy()->at(0) == "Material 1");
	std::cout << "Ok\n\n";


((Wykladowca*)uczelnia->getUzytkownicy()->at(0))->dodajPrzedmiot("p_2");

((Wykladowca*)uczelnia->getUzytkownicy()->at(0))->getPrzedmioty()->at(1)
->dodajMaterial("Material 2");
	std::cout << "Utworzony material: " <<
((Wykladowca*)uczelnia->getUzytkownicy()->at(0))->getPrzedmioty()->at(1)
->getMaterialy()->at(0) << "\n";
```

```
assert(((Wykladowca*)uczelnia->getUzytkownicy()->at(0))->getPrzedmioty()
->at(1)->getMaterialy()->at(0) == "Material 2");
	std::cout << "Ok\n\n";


((Wykladowca*)uczelnia->getUzytkownicy()->at(0))->dodajPrzedmiot("p_3");

((Wykladowca*)uczelnia->getUzytkownicy()->at(0))->getPrzedmioty()->at(2)
->dodajMaterial("Material 3");
	std::cout << "Utworzony material: " <<
((Wykladowca*)uczelnia->getUzytkownicy()->at(0))->getPrzedmioty()->at(2)
->getMaterialy()->at(0) << "\n";

assert(((Wykladowca*)uczelnia->getUzytkownicy()->at(0))->getPrzedmioty()
->at(2)->getMaterialy()->at(0) == "Material 3");
	std::cout << "Ok\n\n";


	uczelnia->dodajUzytkownika(new Wykladowca("login2", "haslo2",
"456"));


((Wykladowca*)uczelnia->getUzytkownicy()->at(1))->dodajPrzedmiot("p_4");

((Wykladowca*)uczelnia->getUzytkownicy()->at(1))->getPrzedmioty()->at(0)
->dodajMaterial("Material 4");
	std::cout << "Utworzony material: " <<
((Wykladowca*)uczelnia->getUzytkownicy()->at(1))->getPrzedmioty()->at(0)
->getMaterialy()->at(0) << "\n";

assert(((Wykladowca*)uczelnia->getUzytkownicy()->at(1))->getPrzedmioty()
->at(0)->getMaterialy()->at(0) == "Material 4");
	std::cout << "Ok\n\n";


((Wykladowca*)uczelnia->getUzytkownicy()->at(1))->dodajPrzedmiot("p_5");

((Wykladowca*)uczelnia->getUzytkownicy()->at(1))->getPrzedmioty()->at(1)
->dodajMaterial("Material 5");
	std::cout << "Utworzony material: " <<
((Wykladowca*)uczelnia->getUzytkownicy()->at(1))->getPrzedmioty()->at(1)
->getMaterialy()->at(0) << "\n";

assert(((Wykladowca*)uczelnia->getUzytkownicy()->at(1))->getPrzedmioty()
->at(1)->getMaterialy()->at(0) == "Material 5");
	std::cout << "Ok\n\n";
```

```
((Wykladowca*)uczelnia->getUzytkownicy()->at(1))->getPrzedmioty()->at(1)
->dodajMaterial("Material 6");
	std::cout << "Utworzony material: " <<
((Wykladowca*)uczelnia->getUzytkownicy()->at(1))->getPrzedmioty()->at(1)
->getMaterialy()->at(1) << "\n";

assert(((Wykladowca*)uczelnia->getUzytkownicy()->at(1))->getPrzedmioty()
->at(1)->getMaterialy()->at(1) == "Material 6");
	std::cout << "Ok\n\n";


((Wykladowca*)uczelnia->getUzytkownicy()->at(1))->dodajPrzedmiot("p_6");

((Wykladowca*)uczelnia->getUzytkownicy()->at(1))->getPrzedmioty()->at(2)
->dodajMaterial("Material 7");
	std::cout << "Utworzony material: " <<
((Wykladowca*)uczelnia->getUzytkownicy()->at(1))->getPrzedmioty()->at(2)
->getMaterialy()->at(0) << "\n";

assert(((Wykladowca*)uczelnia->getUzytkownicy()->at(1))->getPrzedmioty()
->at(2)->getMaterialy()->at(0) == "Material 7");
	std::cout << "Ok\n\n";

	delete uczelnia;
}

int main() {
	std::cout << "Test: Dodaj materialy z przedmiotow\n\n";

	std::cout << "test():\n\n";
	test();

	return 0;
}
```

Logowanie

```
#include <iostream>
#include <cassert>

#include "../Narzedzie.h"
#include "../Przedmiot.h"
#include "../Student.h"
#include "../Uczelnia.h"
#include "../Uzytkownik.h"
#include "../Wykladowca.h"

void test(){
	Uczelnia* uczelnia = new Uczelnia("Nazwa");


	uczelnia->dodajUzytkownika(new Student("login1", "haslo1",
"123"));
	uczelnia->dodajUzytkownika(new Student("login2", "haslo2",
"456"));
	uczelnia->dodajUzytkownika(new Student("login3", "haslo3",
"789"));
	uczelnia->dodajUzytkownika(new Wykladowca("login4", "haslo4",
"987"));
	uczelnia->dodajUzytkownika(new Wykladowca("login5", "haslo5",
"654"));
	uczelnia->dodajUzytkownika(new Wykladowca("login6", "haslo6",
"321"));


	std::cout << "Logowanie uzytkownika o loginie: " <<
uczelnia->getUzytkownicy()->at(0)->getLogin() << "\n";
	assert(uczelnia->logowanie("login1", "haslo1", true) ==
uczelnia->getUzytkownicy()->at(0));
	std::cout << "OK\n\n";
	uczelnia->getUzytkownicy()->at(0)->wylogujSie();

	std::cout << "Logowanie uzytkownika o loginie: " <<
uczelnia->getUzytkownicy()->at(3)->getLogin() << "\n";
	assert(uczelnia->logowanie("login4", "haslo4", false) ==
uczelnia->getUzytkownicy()->at(3));
	std::cout << "OK\n\n";
	uczelnia->getUzytkownicy()->at(3)->wylogujSie();

	std::cout << "Logowanie uzytkownika o loginie: " <<
uczelnia->getUzytkownicy()->at(5)->getLogin() << "\n";
	assert(uczelnia->logowanie("login6", "haslo6", false) ==
```

```
uczelnia->getUzytkownicy()->at(5));
	std::cout << "OK\n\n";
	uczelnia->getUzytkownicy()->at(5)->wylogujSie();

	std::cout << "Logowanie uzytkownika o loginie: " <<
uczelnia->getUzytkownicy()->at(1)->getLogin() << "\n";
	assert(uczelnia->logowanie("login2", "haslo2", true) ==
uczelnia->getUzytkownicy()->at(1));
	std::cout << "OK\n\n";
	uczelnia->getUzytkownicy()->at(1)->wylogujSie();

	std::cout << "Logowanie uzytkownika o loginie: " <<
uczelnia->getUzytkownicy()->at(4)->getLogin() << "\n";
	assert(uczelnia->logowanie("login5", "haslo5", false) ==
uczelnia->getUzytkownicy()->at(4));
	std::cout << "OK\n\n";
	uczelnia->getUzytkownicy()->at(4)->wylogujSie();

	std::cout << "Logowanie uzytkownika o loginie: " <<
uczelnia->getUzytkownicy()->at(2)->getLogin() << "\n";
	assert(uczelnia->logowanie("login3", "haslo3", true) ==
uczelnia->getUzytkownicy()->at(2));
	std::cout << "OK\n\n";
	uczelnia->getUzytkownicy()->at(2)->wylogujSie();


	delete uczelnia;
}

int main() {
	std::cout << "Test: Logowanie\n\n";

	std::cout << "test():\n\n";
	test();

	return 0;
}
```

Skorzystaj z narzędzi dydaktycznych

```
#include <iostream>
#include <cassert>

#include "../Narzedzie.h"
#include "../Przedmiot.h"
#include "../Student.h"
#include "../Uczelnia.h"
#include "../Uzytkownik.h"
#include "../Wykladowca.h"

void test(){
	Uczelnia* uczelnia = new Uczelnia("Nazwa");

	Narzedzie* word = new Narzedzie("Word", "www.word.com");
	Narzedzie* excel = new Narzedzie("Excel", "www.excel.com");
	Narzedzie* powerpoint = new Narzedzie("PowerPoint",
"www.powerpoint.com");

	uczelnia->dodajNarzedzie(word);
	uczelnia->dodajNarzedzie(excel);
	uczelnia->dodajNarzedzie(powerpoint);

	uczelnia->dodajUzytkownika(new Student("login1", "haslo1",
"123"));
	uczelnia->dodajUzytkownika(new Wykladowca("login2", "haslo2",
"456"));


	Uzytkownik* uzytkownik = uczelnia->logowanie("login1", "haslo1",
true);
	uczelnia->wyswietlNarzedzia();
	uczelnia->skorzystajZNarzedzia("Word");
	uzytkownik->wylogujSie();

	uzytkownik = uczelnia->logowanie("login2", "haslo2", false);
	uczelnia->wyswietlNarzedzia();
	uczelnia->skorzystajZNarzedzia("Excel");
	uzytkownik->wylogujSie();

	uzytkownik = uczelnia->logowanie("login1", "haslo1", true);
	uczelnia->wyswietlNarzedzia();
	uczelnia->skorzystajZNarzedzia("PowerPoint");
	uzytkownik->wylogujSie();
```

```
	delete uczelnia;
}

int main() {
	cout << "Test: Skorzystaj z narzedzi dydaktycznych\n\n";

	cout << "test():\n\n";

	test();

	return 0;
}
```

Sprawdź materiały z przedmiotów

```
#include <iostream>
#include <cassert>

#include "../Narzedzie.h"
#include "../Przedmiot.h"
#include "../Student.h"
#include "../Uczelnia.h"
#include "../Uzytkownik.h"
#include "../Wykladowca.h"

void test(){

	Uczelnia* uczelnia = new Uczelnia("Nazwa");

	uczelnia->dodajUzytkownika(new Wykladowca("login1", "haslo1",
"123"));
	uczelnia->dodajUzytkownika(new Student("login2", "haslo2",
"456"));
	uczelnia->dodajUzytkownika(new Student("login3", "haslo3",
"789"));

	Uzytkownik* uzytkownik = uczelnia->getUzytkownicy()->at(0);

	((Wykladowca*)uzytkownik)->dodajPrzedmiot("Przedmiot1");
	((Wykladowca*)uzytkownik)->dodajPrzedmiot("Przedmiot2");
	((Wykladowca*)uzytkownik)->dodajPrzedmiot("Przedmiot3");


((Wykladowca*)uzytkownik)->dodajMaterial(((Wykladowca*)uzytkownik)->getP
rzedmioty()->at(0), "link 1");

((Wykladowca*)uzytkownik)->dodajMaterial(((Wykladowca*)uzytkownik)->getP
rzedmioty()->at(0), "link 2");

((Wykladowca*)uzytkownik)->dodajMaterial(((Wykladowca*)uzytkownik)->getP
rzedmioty()->at(0), "link 3");


((Wykladowca*)uzytkownik)->dodajMaterial(((Wykladowca*)uzytkownik)->getP
rzedmioty()->at(1), "link 4");

((Wykladowca*)uzytkownik)->dodajMaterial(((Wykladowca*)uzytkownik)->getP
rzedmioty()->at(1), "link 5");

((Wykladowca*)uzytkownik)->dodajMaterial(((Wykladowca*)uzytkownik)->getP
```

```
rzedmioty()->at(1), "link 6");


((Wykladowca*)uzytkownik)->dodajMaterial(((Wykladowca*)uzytkownik)->getP
rzedmioty()->at(2), "link 7");

((Wykladowca*)uzytkownik)->dodajMaterial(((Wykladowca*)uzytkownik)->getP
rzedmioty()->at(2), "link 8");

((Wykladowca*)uzytkownik)->dodajMaterial(((Wykladowca*)uzytkownik)->getP
rzedmioty()->at(2), "link 9");


((Wykladowca*)uzytkownik)->dodajStudentaDoPrzedmiotu(((Wykladowca*)uzytk
ownik)->getPrzedmioty()->at(0),
(Student*)uczelnia->getUzytkownicy()->at(1));

((Wykladowca*)uzytkownik)->dodajStudentaDoPrzedmiotu(((Wykladowca*)uzytk
ownik)->getPrzedmioty()->at(0),
(Student*)uczelnia->getUzytkownicy()->at(2));

((Wykladowca*)uzytkownik)->dodajStudentaDoPrzedmiotu(((Wykladowca*)uzytk
ownik)->getPrzedmioty()->at(1),
(Student*)uczelnia->getUzytkownicy()->at(1));

((Wykladowca*)uzytkownik)->dodajStudentaDoPrzedmiotu(((Wykladowca*)uzytk
ownik)->getPrzedmioty()->at(2),
(Student*)uczelnia->getUzytkownicy()->at(2));


      uzytkownik = uczelnia->logowanie("login2", "haslo2", true);
      ((Student*)uzytkownik)->wyswietlPrzedmioty();
      ((Student*)uzytkownik)->sprawdzMaterialy("Przedmiot1");
      uzytkownik->wylogujSie();

      uzytkownik = uczelnia->logowanie("login3", "haslo3", true);
      ((Student*)uzytkownik)->wyswietlPrzedmioty();
      ((Student*)uzytkownik)->sprawdzMaterialy("Przedmiot1");
      uzytkownik->wylogujSie();

      uzytkownik = uczelnia->logowanie("login2", "haslo2", true);
      ((Student*)uzytkownik)->wyswietlPrzedmioty();
      ((Student*)uzytkownik)->sprawdzMaterialy("Przedmiot2");
      uzytkownik->wylogujSie();

      uzytkownik = uczelnia->logowanie("login3", "haslo3", true);
```

```
	((Student*)uzytkownik)->wyswietlPrzedmioty();
	((Student*)uzytkownik)->sprawdzMaterialy("Przedmiot3");
	uzytkownik->wylogujSie();


	delete uczelnia;
}

int main() {
	std::cout << "Test: Sprawdz materialy z przedmiotow\n\n";

	std::cout << "test():\n\n";
	test();

	return 0;
}
```

Wstaw oceny + Sprawdź oceny

```
#include <iostream>
#include <cassert>

#include "../Narzedzie.h"
#include "../Przedmiot.h"
#include "../Student.h"
#include "../Uczelnia.h"
#include "../Uzytkownik.h"
#include "../Wykladowca.h"

void test(){

	Uczelnia* uczelnia = new Uczelnia("Nazwa");

	uczelnia->dodajUzytkownika(new Wykladowca("login1", "haslo1",
"123"));
	uczelnia->dodajUzytkownika(new Student("login2", "haslo2",
"456"));
	uczelnia->dodajUzytkownika(new Student("login3", "haslo3",
"789"));

	Uzytkownik* uzytkownik = uczelnia->logowanie("login1", "haslo1",
false);

	((Wykladowca*)uzytkownik)->dodajPrzedmiot("Przedmiot1");
	((Wykladowca*)uzytkownik)->dodajPrzedmiot("Przedmiot2");
	((Wykladowca*)uzytkownik)->dodajPrzedmiot("Przedmiot3");


((Wykladowca*)uzytkownik)->dodajStudentaDoPrzedmiotu(((Wykladowca*)uzytk
ownik)->getPrzedmioty()->at(0),
(Student*)uczelnia->getUzytkownicy()->at(1));

((Wykladowca*)uzytkownik)->dodajStudentaDoPrzedmiotu(((Wykladowca*)uzytk
ownik)->getPrzedmioty()->at(0),
(Student*)uczelnia->getUzytkownicy()->at(2));

((Wykladowca*)uzytkownik)->dodajStudentaDoPrzedmiotu(((Wykladowca*)uzytk
ownik)->getPrzedmioty()->at(1),
(Student*)uczelnia->getUzytkownicy()->at(1));

((Wykladowca*)uzytkownik)->dodajStudentaDoPrzedmiotu(((Wykladowca*)uzytk
ownik)->getPrzedmioty()->at(2),
(Student*)uczelnia->getUzytkownicy()->at(2));
```

```
((Wykladowca*)uzytkownik)->dodajOcene(((Wykladowca*)uzytkownik)->getPrze
dmioty()->at(0),
((Student*)uczelnia->getUzytkownicy()->at(1))->getNrAlbumu(), 5);

((Wykladowca*)uzytkownik)->dodajOcene(((Wykladowca*)uzytkownik)->getPrze
dmioty()->at(0),
((Student*)uczelnia->getUzytkownicy()->at(1))->getNrAlbumu(), 2);

((Wykladowca*)uzytkownik)->dodajOcene(((Wykladowca*)uzytkownik)->getPrze
dmioty()->at(0),
((Student*)uczelnia->getUzytkownicy()->at(2))->getNrAlbumu(), 4);

((Wykladowca*)uzytkownik)->dodajOcene(((Wykladowca*)uzytkownik)->getPrze
dmioty()->at(0),
((Student*)uczelnia->getUzytkownicy()->at(2))->getNrAlbumu(), 3);

((Wykladowca*)uzytkownik)->dodajOcene(((Wykladowca*)uzytkownik)->getPrze
dmioty()->at(0),
((Student*)uczelnia->getUzytkownicy()->at(2))->getNrAlbumu(), 2);

((Wykladowca*)uzytkownik)->dodajOcene(((Wykladowca*)uzytkownik)->getPrze
dmioty()->at(1),
((Student*)uczelnia->getUzytkownicy()->at(1))->getNrAlbumu(), 3);

((Wykladowca*)uzytkownik)->dodajOcene(((Wykladowca*)uzytkownik)->getPrze
dmioty()->at(1),
((Student*)uczelnia->getUzytkownicy()->at(1))->getNrAlbumu(), 4);

((Wykladowca*)uzytkownik)->dodajOcene(((Wykladowca*)uzytkownik)->getPrze
dmioty()->at(1),
((Student*)uczelnia->getUzytkownicy()->at(2))->getNrAlbumu(), 5);

((Wykladowca*)uzytkownik)->dodajOcene(((Wykladowca*)uzytkownik)->getPrze
dmioty()->at(2),
((Student*)uczelnia->getUzytkownicy()->at(2))->getNrAlbumu(), 2);

((Wykladowca*)uzytkownik)->dodajOcene(((Wykladowca*)uzytkownik)->getPrze
dmioty()->at(2),
((Student*)uczelnia->getUzytkownicy()->at(1))->getNrAlbumu(), 3);

((Wykladowca*)uzytkownik)->dodajOcene(((Wykladowca*)uzytkownik)->getPrze
dmioty()->at(2),
((Student*)uczelnia->getUzytkownicy()->at(2))->getNrAlbumu(), 4);
      uzytkownik->wylogujSie();
```

```
	uzytkownik = uczelnia->logowanie("login2", "haslo2", true);
	((Student*)uzytkownik)->wyswietlPrzedmioty();
	std::cout << "Oceny: ";
	((Student*)uzytkownik)->sprawdzOceny("Przedmiot1");
	std::cout << "\n";
	std::cout << "Oceny: ";
	((Student*)uzytkownik)->sprawdzOceny("Przedmiot2");
	std::cout << "\n";
	std::cout << "Oceny: ";
	((Student*)uzytkownik)->sprawdzOceny("Przedmiot3");
	std::cout << "\n";
	uzytkownik->wylogujSie();

	uzytkownik = uczelnia->logowanie("login3", "haslo3", true);
	((Student*)uzytkownik)->wyswietlPrzedmioty();
	std::cout << "Oceny: ";
	((Student*)uzytkownik)->sprawdzOceny("Przedmiot1");
	std::cout << "\n";
	std::cout << "Oceny: ";
	((Student*)uzytkownik)->sprawdzOceny("Przedmiot2");
	std::cout << "\n";
	std::cout << "Oceny: ";
	((Student*)uzytkownik)->sprawdzOceny("Przedmiot3");
	std::cout << "\n";
	uzytkownik->wylogujSie();


	delete uczelnia;
}

int main() {
	std::cout << "Test: Wstaw i Sprawdz oceny\n\n";

	std::cout << "test():\n\n";
	test();

	return 0;
}
```

Rejestracja

```
#include <iostream>
#include <cassert>
#include <sstream>

#include "../Narzedzie.h"
#include "../Przedmiot.h"
#include "../Student.h"
#include "../Uczelnia.h"
#include "../Uzytkownik.h"
#include "../Wykladowca.h"

void test(){
	Uczelnia* uczelnia = new Uczelnia("Nazwa");
	Uzytkownik* user;

	std::istringstream input("123\n");
	std::cin.rdbuf(input.rdbuf());
	user = uczelnia->rejestracja("login1", "haslo1", false);
	cout << ((Wykladowca*)user)->getIdentyfikator() << "\n";
	user->wylogujSie();
	cout << "\n\n";

	input.str("321\n");
	std::cin.rdbuf(input.rdbuf());
	user = uczelnia->rejestracja("login2", "haslo2", true);
	cout << ((Student*)user)->getNrAlbumu() << "\n";
	user->wylogujSie();
	cout << "\n\n";

	input.str("456\n");
	std::cin.rdbuf(input.rdbuf());
	user = uczelnia->rejestracja("login3", "haslo3", false);
	cout << ((Wykladowca*)user)->getIdentyfikator() << "\n";
	user->wylogujSie();
	cout << "\n\n";

	input.str("654\n");
	std::cin.rdbuf(input.rdbuf());
	user = uczelnia->rejestracja("login4", "haslo4", true);
	cout << ((Student*)user)->getNrAlbumu() << "\n";
	user->wylogujSie();
	cout << "\n\n";

	input.str("789\n");
	std::cin.rdbuf(input.rdbuf());
```

```
	user = uczelnia->rejestracja("login5", "haslo5", false);
	cout << ((Wykladowca*)user)->getIdentyfikator() << "\n";
	user->wylogujSie();
	cout << "\n\n";

	input.str("321\n");
	std::cin.rdbuf(input.rdbuf());
	user = uczelnia->rejestracja("login6", "haslo6", true);
	cout << ((Student*)user)->getNrAlbumu() << "\n";
	user->wylogujSie();
	cout << "\n\n";


	delete uczelnia;
}

int main() {
	std::cout << "Test: Zarejestruj sie\n\n";

	std::cout << "test():\n\n";
	test();

	return 0;
}
```

d. Testy systemowe + testy akceptacyjne

Testowany był cały system - wszystkie funkcjonalności na raz, przy jednej sesji.

- *Logowanie i rejestracja*

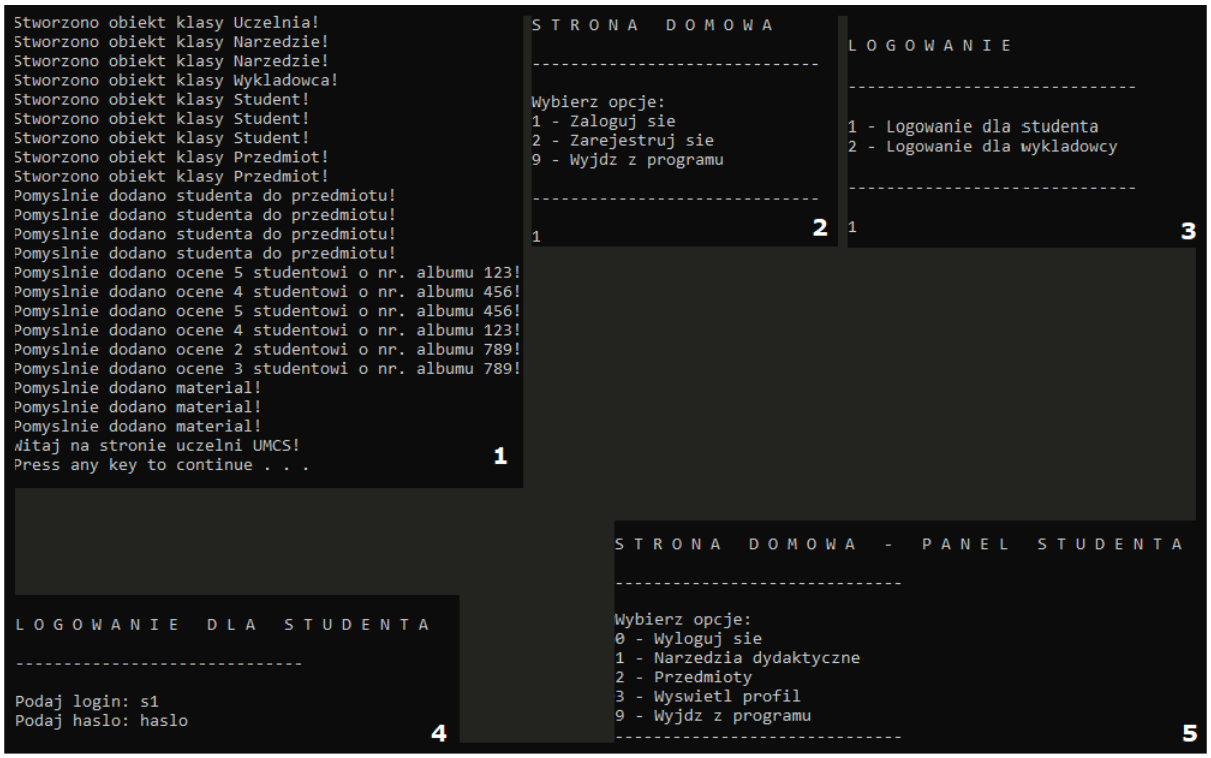
Dla Studenta

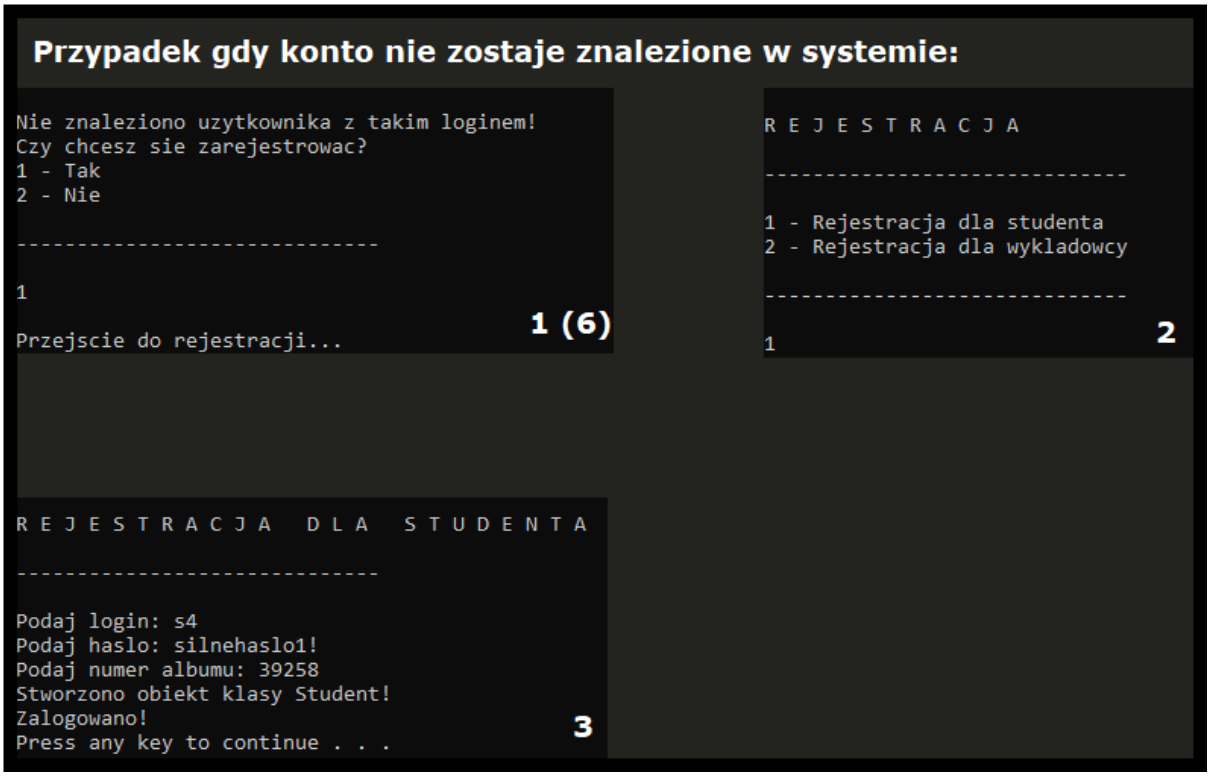
Plan testu:

1. Uruchom system i kliknij dowolny klawisz aby przejść do strony domowej.
2. Wybierz "1" i zatwierdź aby przejść do ekranu logowania.
3. Wpisz "1" i zatwierdź aby wybrać logowanie dla studenta
4. W pole "login" wpisz login, w pole "hasło" wpisz hasło a następnie zatwierdź.
5. Jeśli konto zostało odnalezione w systemie logowanie zakończono pomyślnie.
6. Jeśli konto nie zostało odnalezione w systemie użytkownik przechodzi do ekranu rejestracji.

REJESTRACJA:
1. Wpisz "1", zatwierdź i kliknij dowolny klawisz aby kontynuować.
2. Wpisz "1" i zatwierdź jeśli by wybrać logowanie dla studenta
3. W pole "login" wpisz login, w pole "hasło" wpisz hasło, w pole "numer albumu" wpisz numer albumu a następnie zatwierdź.
4. Kliknij dowolny klawisz aby przejść do strony domowej.

Testowanie:


Stworzono obiekt klasy Uczelnia!
Stworzono obiekt klasy Narzedzie!
Stworzono obiekt klasy Narzedzie!
Stworzono obiekt klasy Wykladowca!
Stworzono obiekt klasy Student!
Stworzono obiekt klasy Student!
Stworzono obiekt klasy Student!
Stworzono obiekt klasy Przedmiot!
Stworzono obiekt klasy Przedmiot!
Pomyslnie dodano studenta do przedmiotu!
Pomyslnie dodano studenta do przedmiotu!
Pomyslnie dodano studenta do przedmiotu!
Pomyslnie dodano studenta do przedmiotu!
Pomyslnie dodano ocene 5 studentowi o nr. albumu 123!
Pomyslnie dodano ocene 4 studentowi o nr. albumu 456!
Pomyslnie dodano ocene 5 studentowi o nr. albumu 456!
Pomyslnie dodano ocene 4 studentowi o nr. albumu 123!
Pomyslnie dodano ocene 2 studentowi o nr. albumu 789!
Pomyslnie dodano ocene 3 studentowi o nr. albumu 789!
Pomyslnie dodano material!
Pomyslnie dodano material!
Pomyslnie dodano material!
Witaj na stronie uczelni UMCS!
Press any key to continue . . .
1
STRONA DOMOWA
Wybierz opcje:
1 - Zaloguj sie
2 - Zarejestruj sie
9 - Wyjdz z programu
1
2
LOGOWANIE
1 - Logowanie dla studenta
2 - Logowanie dla wykladowcy
1
3
LOGOWANIE DLA STUDENTA
Podaj login: s1
Podaj haslo: haslo
4
STRONA DOMOWA - PANEL STUDENTA
Wybierz opcje:
0 - Wyloguj sie
1 - Narzedzia dydaktyczne
2 - Przedmioty
3 - Wyswietl profil
9 - Wyjdz z programu
5



Rezultat testu:
Pomyślnie zalogowano na konto studenta.

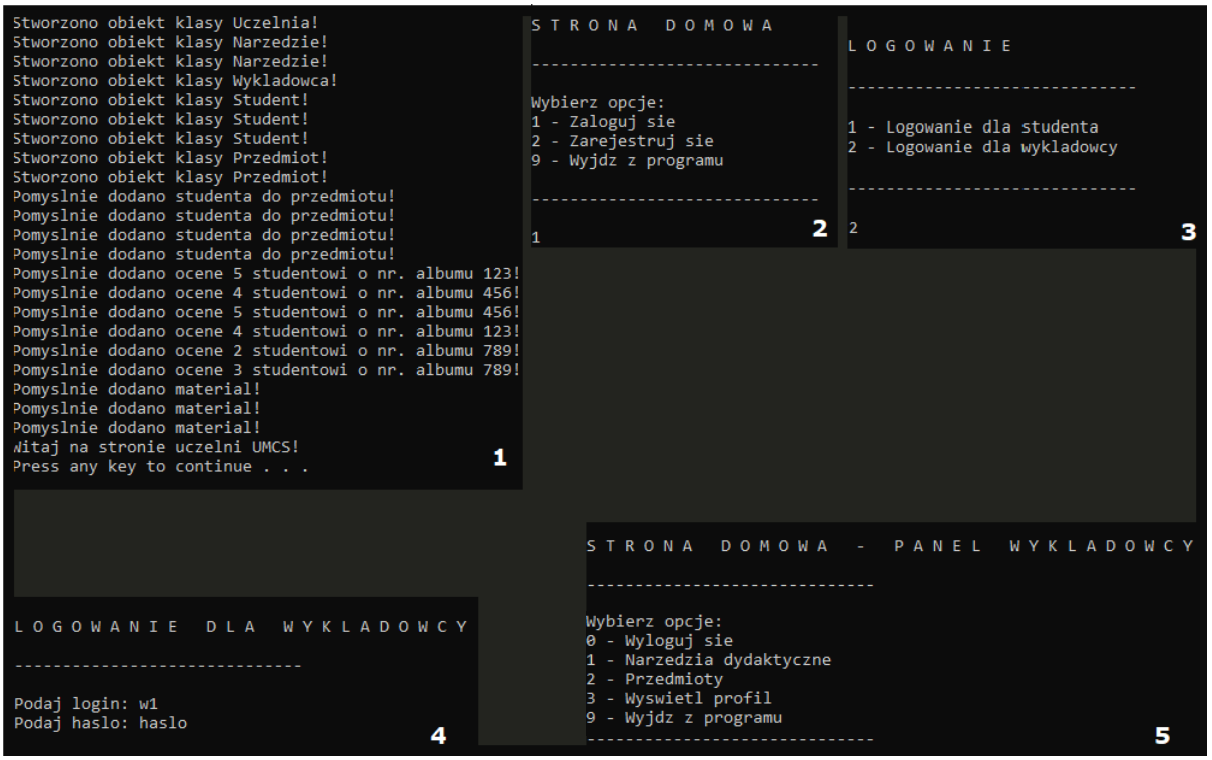
Dla Wykładowcy

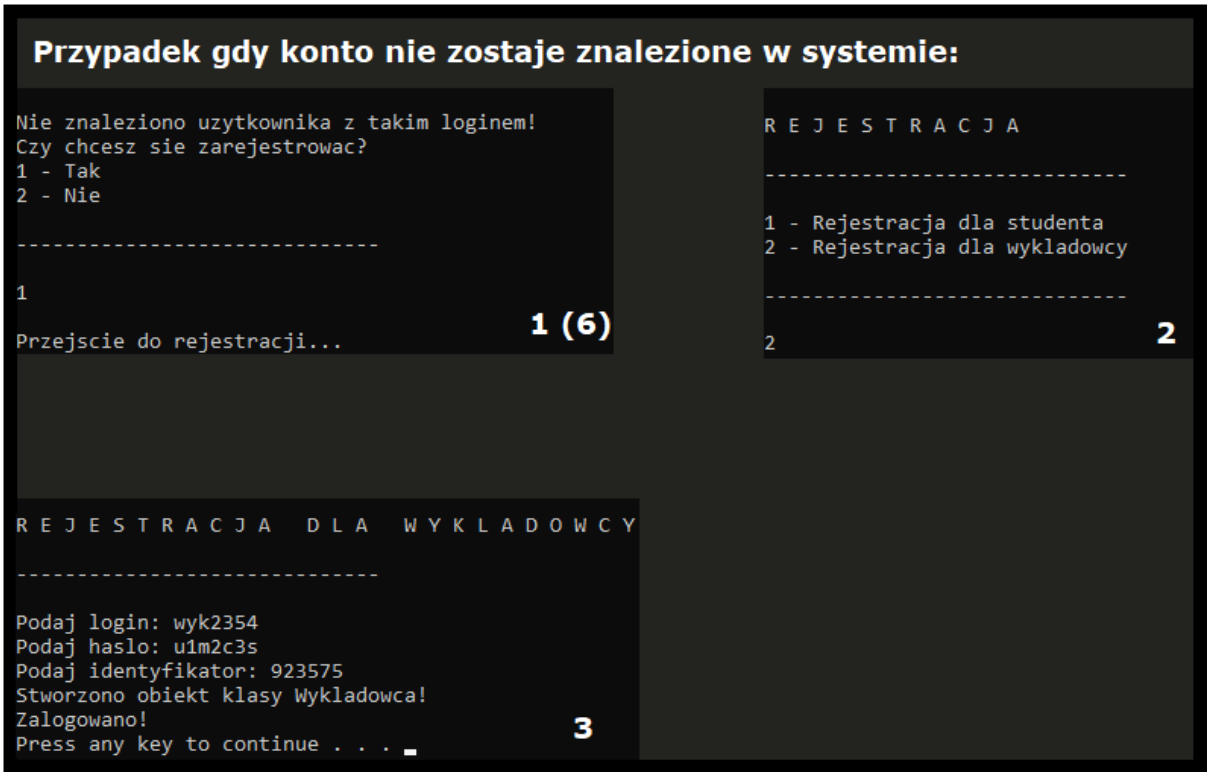
Plan testu:

1. Uruchom system i kliknij dowolny klawisz aby przejść do strony domowej.
2. Wybierz "1" i zatwierdź aby przejść do ekranu logowania.
3. Wpisz "2" i zatwierdź aby wybrać logowanie dla wykładowcy.
4. W pole "login" wpisz login, w pole "hasło" wpisz hasło a następnie zatwierdź.
5. Jeśli konto zostało odnalezione w systemie logowanie zakończono pomyślnie.
6. Jeśli konto nie zostało odnalezione w systemie użytkownik przechodzi do ekranu rejestracji.

REJESTRACJA:
1. Wpisz "1", zatwierdź i kliknij dowolny klawisz aby kontynuować.
2. Wpisz "2" i zatwierdź jeśli by wybrać rejestracje dla wykładowcy.
3. W pole "login" wpisz login, w pole "hasło" wpisz hasło, w pole "identyfikator" wpisz identyfikator a następnie zatwierdź.
4. Kliknij dowolny klawisz aby przejść do strony domowej.

Testowanie:





Rezultat testu:
Pomyślnie zalogowano na konto wykładowcy.

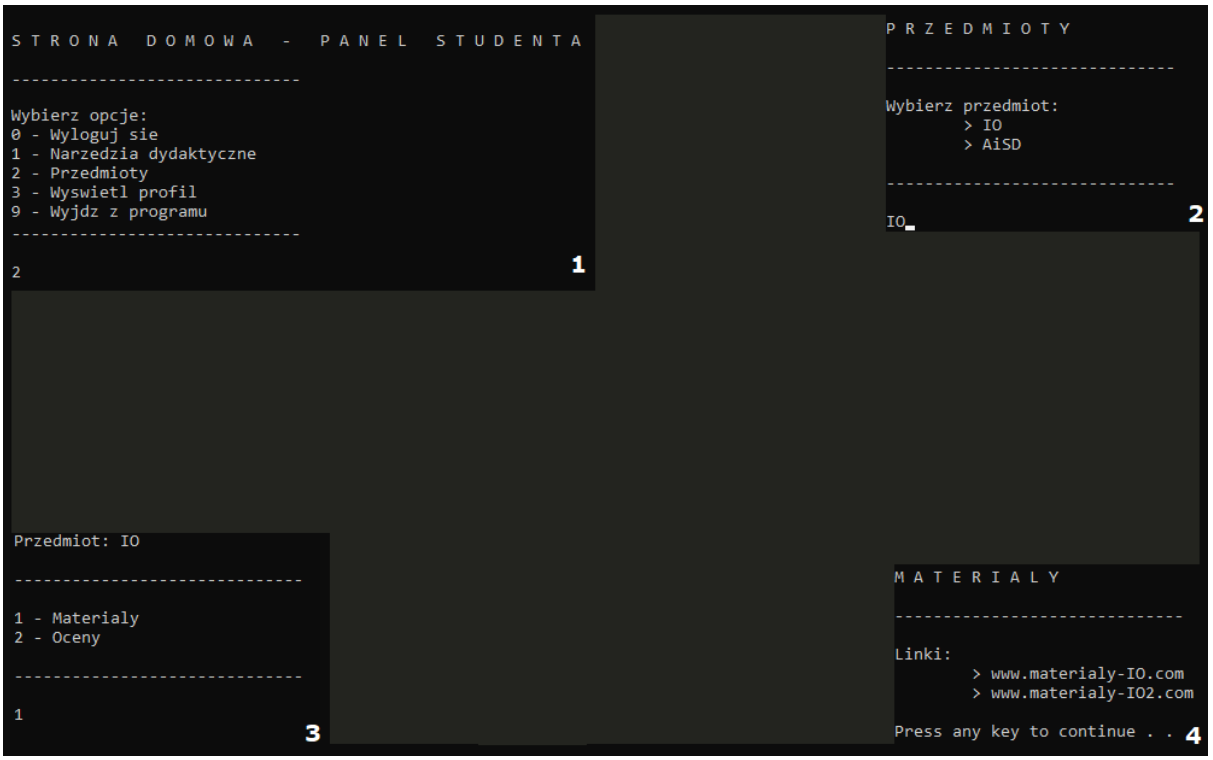
- *Materiały z przedmiotów*

Sprawdź materiały z przedmiotów

Plan testu:

1. Na stronie domowej wpisz "2" aby wybrać przedmioty i zatwierdź.
2. Z listy dostępnych wybierz przedmiot, następnie wpisz jego nazwę i zatwierdź.
3. Wpisz "1" aby wybrać materiały i zatwierdź.
4. Kliknij dowolny klawisz aby przejść do strony domowej.

Testowanie:



Rezultat testu:
Uzyskano dostęp do materiałów.

Dodaj materiały z przedmiotów

Plan testu:
1. Na stronie domowej wpisz "2" aby wybrać przedmioty i zatwierdź.
2. Z listy dostępnych wybierz przedmiot, następnie wpisz jego nazwę i zatwierdź.
3. Wpisz "1" aby wybrać dodawanie materiałów i zatwierdź.
4. Wpisz link do nowego materiału i zatwierdź.
5. Kliknij dowolny klawisz aby przejść do strony domowej.

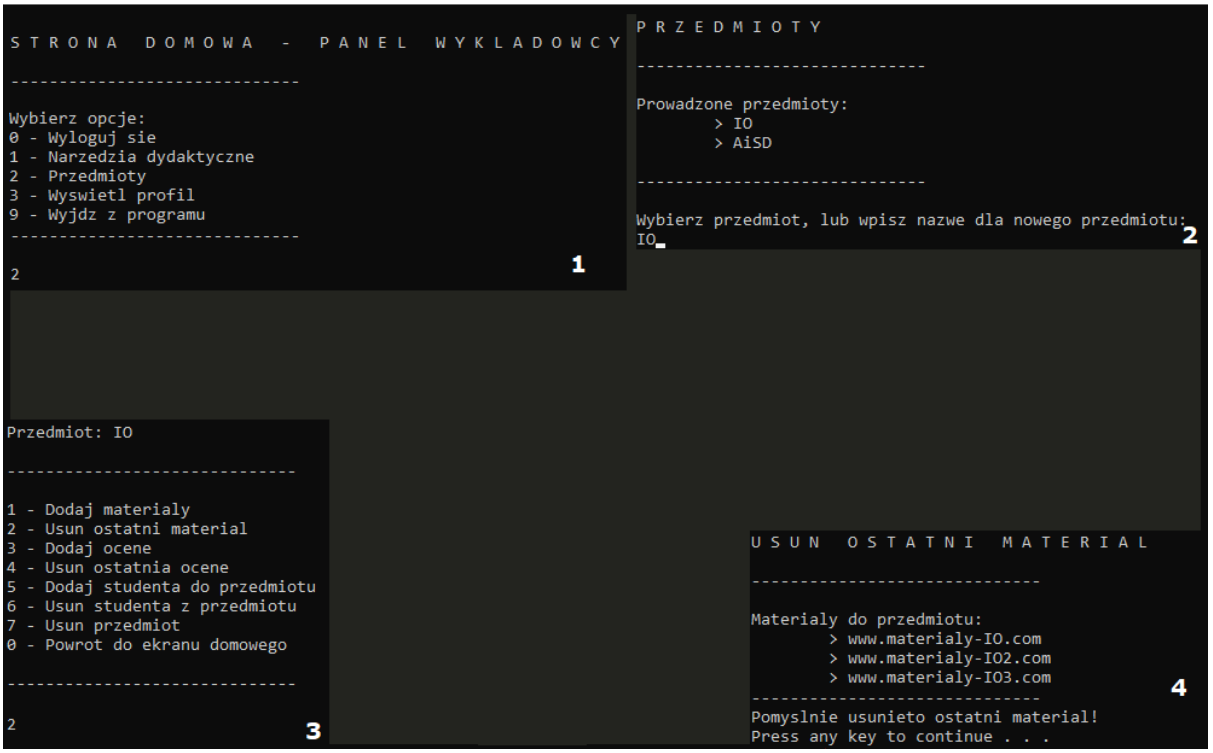
Testowanie:

Rezultat testu:
Dodano materiał.

Usuń ostatni materiał

Plan testu:
1. Na stronie domowej wpisz "2" aby wybrać przedmioty i zatwierdź.
2. Z listy dostępnych wybierz przedmiot, następnie wpisz jego nazwę i zatwierdź.
3. Wpisz "2" aby wybrać dodawanie materiałów i zatwierdź.
4. Kliknij dowolny klawisz aby przejść do strony domowej.

Testowanie:

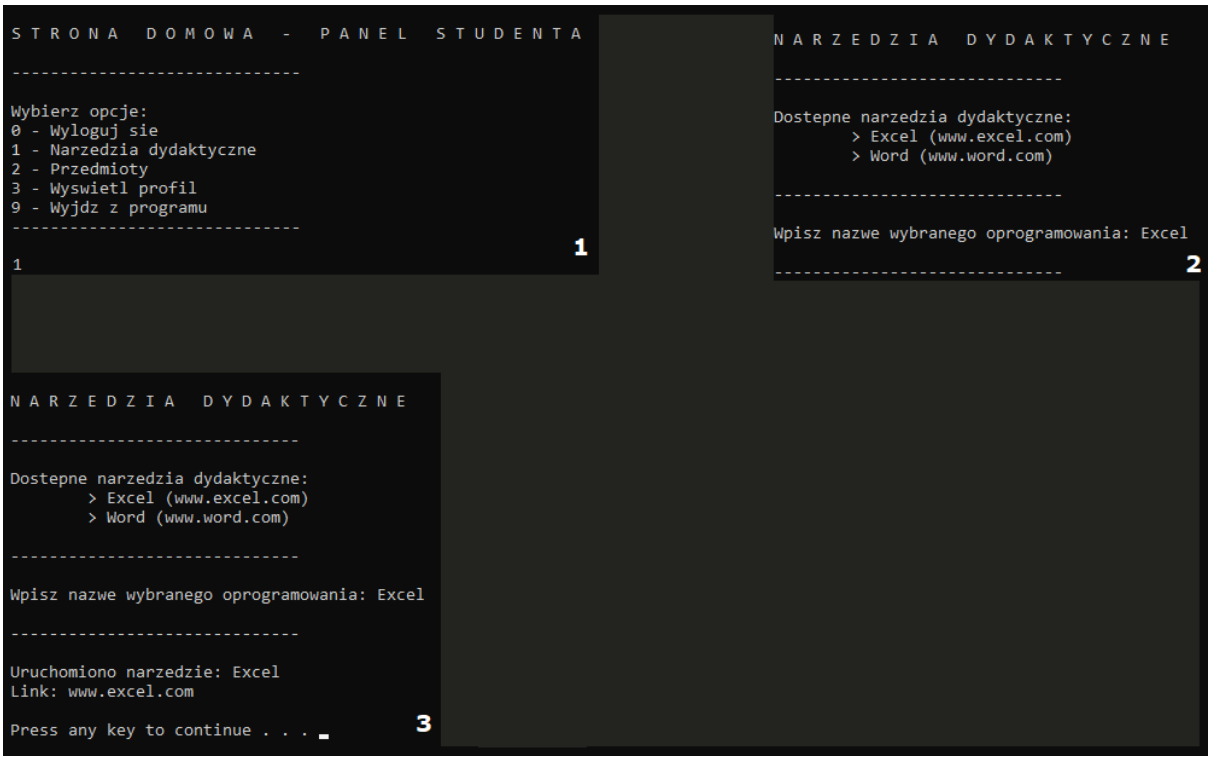


Rezultat testu:
Usunięto ostatni materiał.

- *Skorzystaj z narzędzi dydaktycznych*

Plan testu:
1. Na stronie domowej wpisz "1" aby wybrać narzędzia dydaktyczne i zatwierdź.
2. Z listy dostępnych wybierz narzędzie, następnie wpisz jego nazwę i zatwierdź.
3. Kliknij dowolny klawisz aby przejść do strony domowej.

Testowanie:

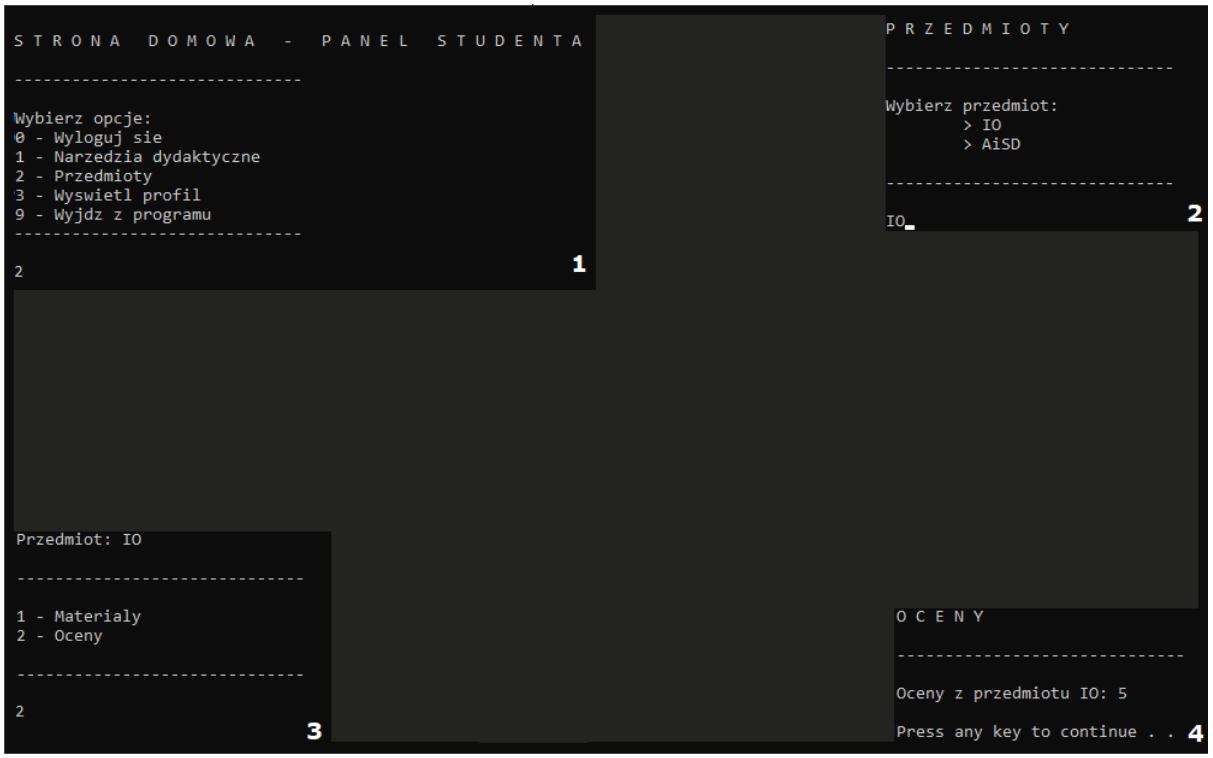


Rezultat testu:
Uzyskano dostęp do narzędzia.

- *Sprawdź oceny*

Plan testu:
1. Na stronie domowej wpisz "2" aby wybrać narzędzia dydaktyczne i zatwierdź.
2. Z listy dostępnych wybierz przedmiot, następnie wpisz jego nazwę i zatwierdź.
3. Wpisz "2" aby wybrać oceny i zatwierdź.
4. Kliknij dowolny klawisz aby przejść do strony domowej.

Testowanie:



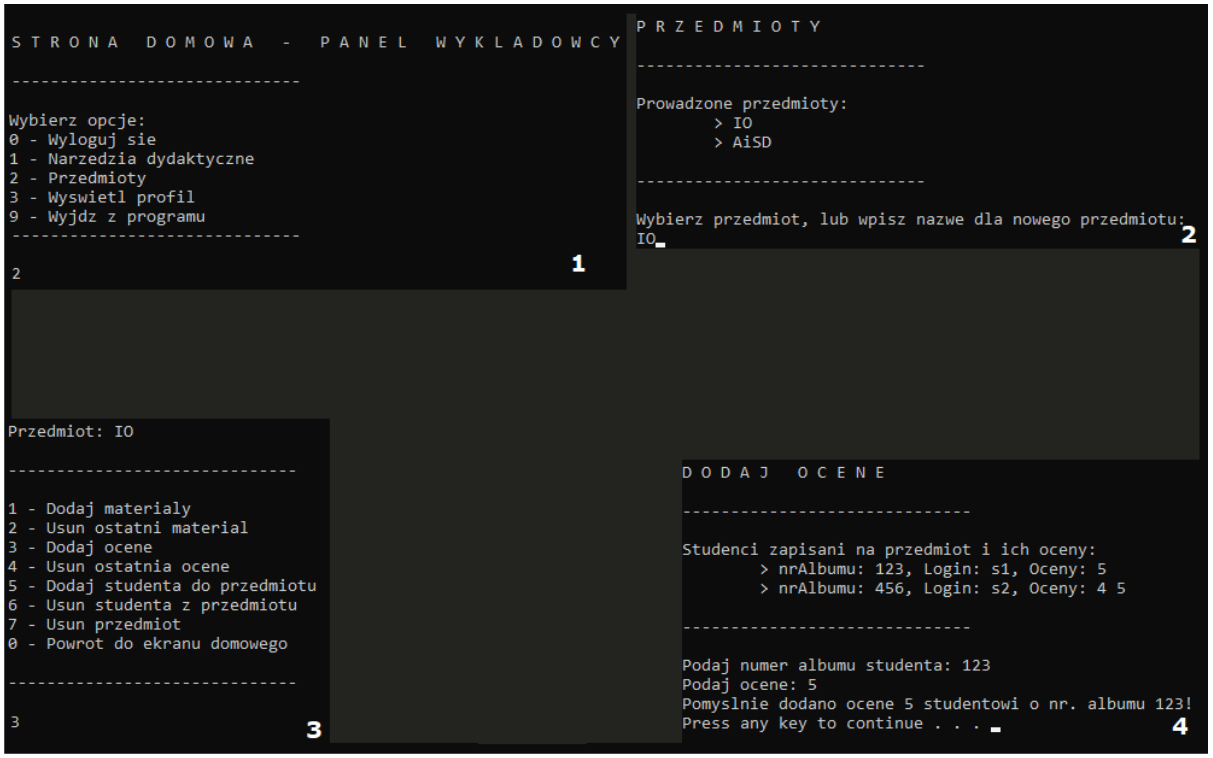
Rezultat testu:
Wyświetlono oceny z przedmiotu.

- *Wstaw oceny*

Plan testu:

1. Na stronie domowej wpisz "2" aby wybrać przedmioty i zatwierdź.
2. Z listy dostępnych wybierz przedmiot, następnie wpisz jego nazwę i zatwierdź.
3. Wpisz "3" aby wybrać dodawanie materiałów i zatwierdź.
4. Wpisz numer albumu studenta i zatwierdź.
5. Wpisz ocenę i zatwierdź.
6. Kliknij dowolny klawisz aby przejść do strony domowej.

Testowanie:



Rezultat testu:

Dodano ocenę dla wybranego studenta.

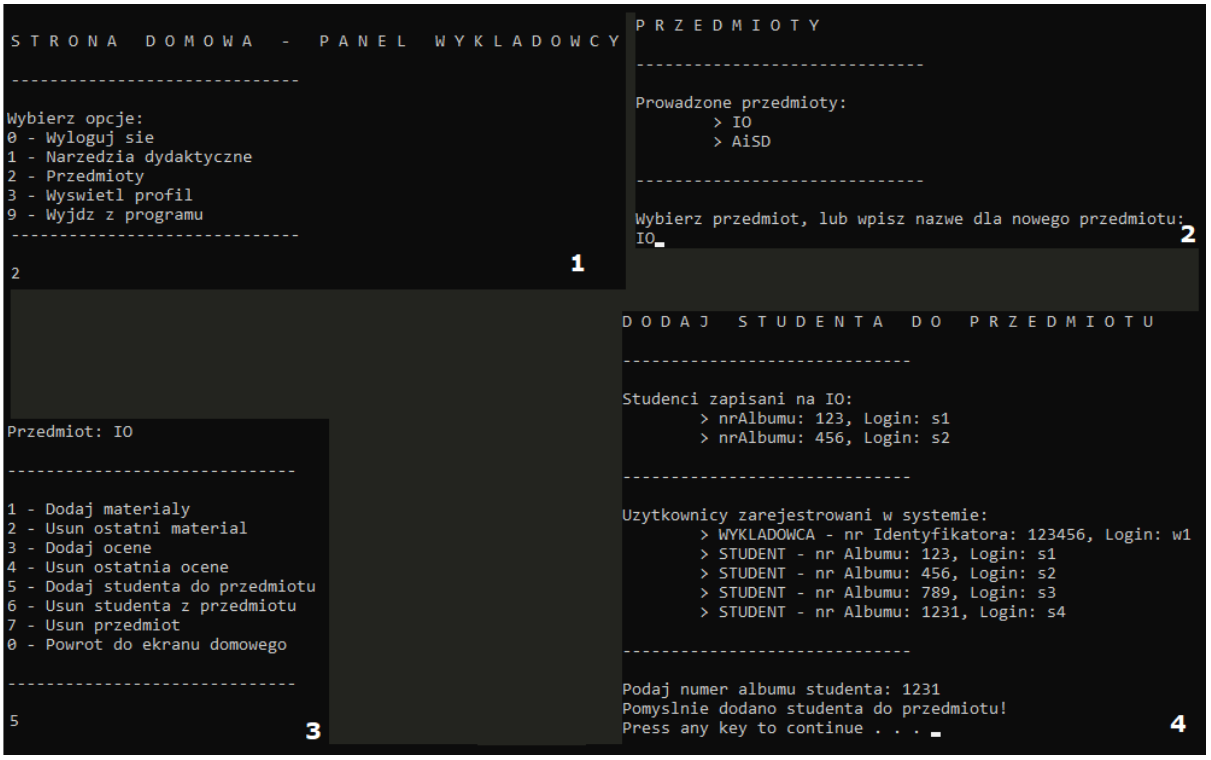
- *Studenci zapisani do przedmiotu*

Dodaj studenta do przedmiotu

Plan testu:

1. Na stronie domowej wpisz "2" aby wybrać przedmioty i zatwierdź.
2. Z listy dostępnych wybierz przedmiot, następnie wpisz jego nazwę i zatwierdź.
3. Wpisz "5" aby wybrać dodawanie studenta do przedmiotu i zatwierdź.
4. Wpisz numer albumu i zatwierdź.
5. Kliknij dowolny klawisz aby przejść do strony domowej.

Testowanie:



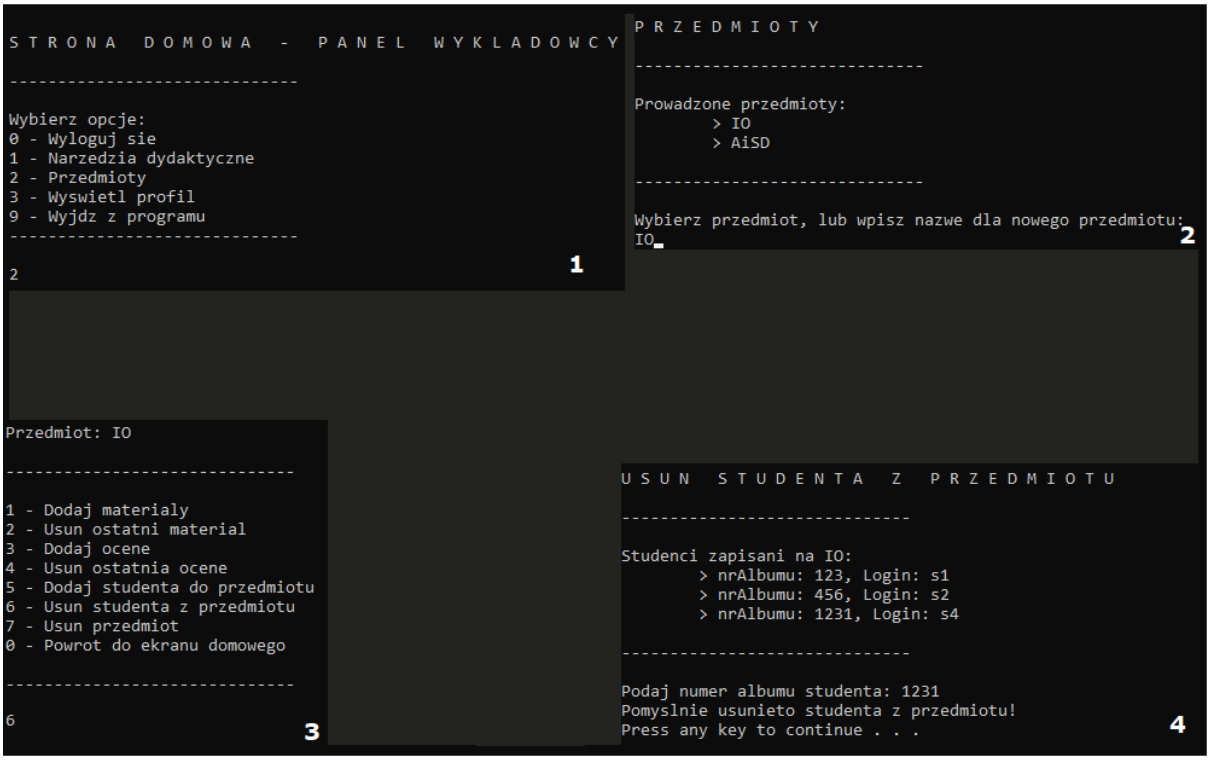
Rezultat testu:
Dodano studenta.

Usuń studenta z przedmiotu

Plan testu:

1. Na stronie domowej wpisz "2" aby wybrać przedmioty i zatwierdź.
2. Z listy dostępnych wybierz przedmiot, następnie wpisz jego nazwę i zatwierdź.
3. Wpisz "6" aby wybrać usuwanie studenta do przedmiotu i zatwierdź.
4. Wpisz numer albumu i zatwierdź.
5. Kliknij dowolny klawisz aby przejść do strony domowej.

Testowanie:



Rezultat testu:
Usunięto studenta.

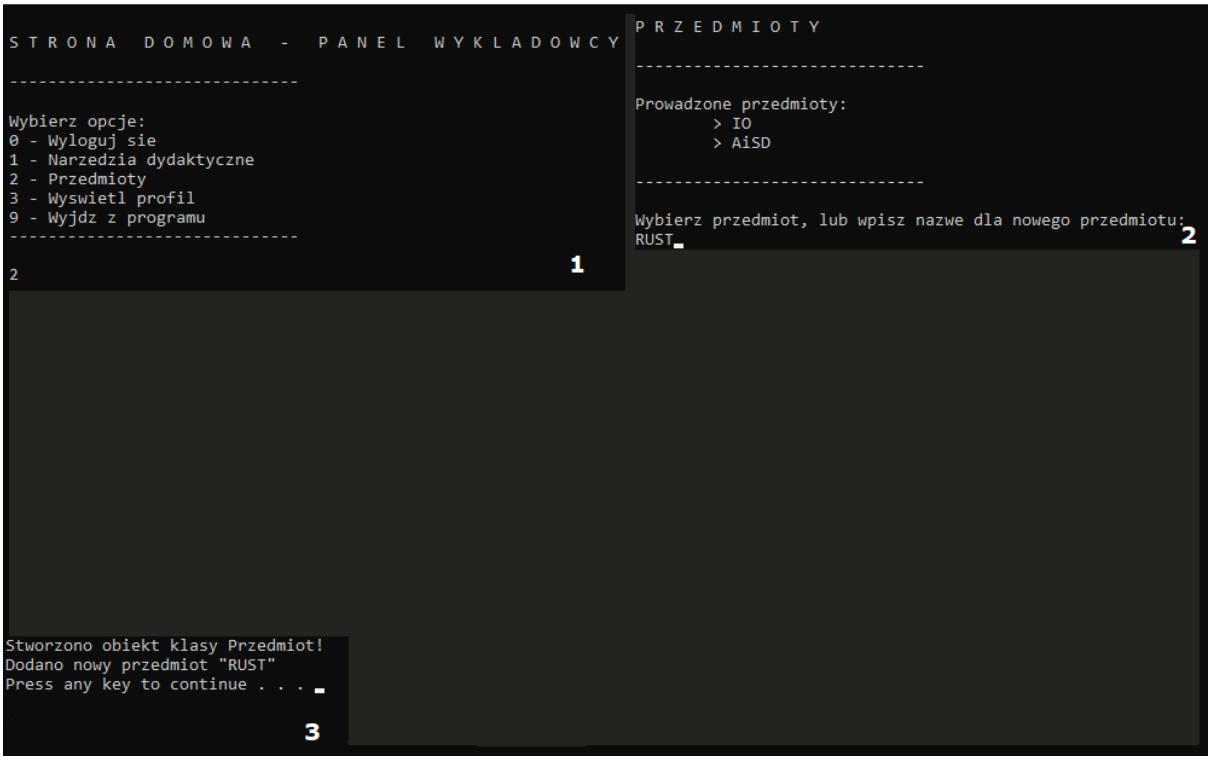
- *Przedmioty*

Dodaj przedmiot

Plan testu:

1. Na stronie domowej wpisz "2" aby wybrać przedmioty i zatwierdź.
2. Wpisz nazwę przedmiotu i zatwierdź.
3. Kliknij dowolny klawisz aby przejść do strony domowej.

Testowanie:



Rezultat testu:

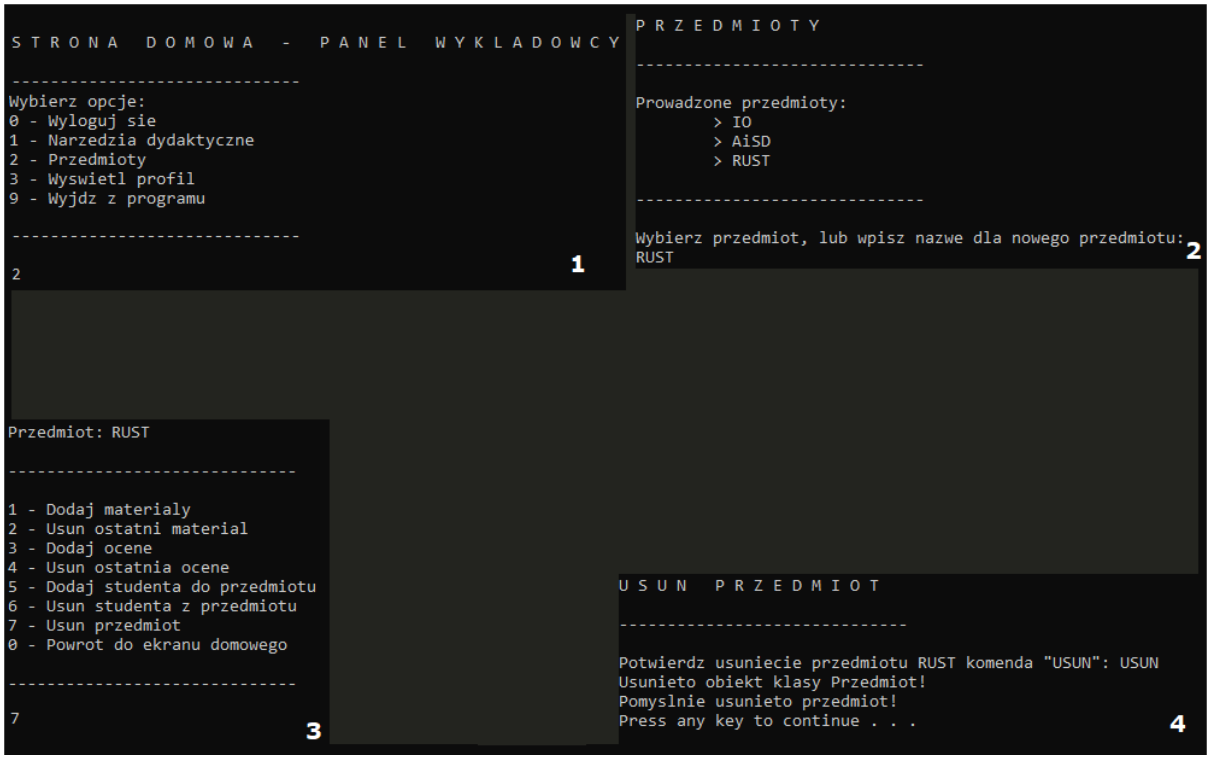
Dodano przedmiot.

Usuń przedmiot

Plan testu:

1. Na stronie domowej wpisz "2" aby wybrać przedmioty i zatwierdź.
2. Z listy dostępnych wybierz przedmiot, następnie wpisz jego nazwę i zatwierdź.
3. Wpisz "7" aby usunąć przedmiot i zatwierdź.
4. Wpisz “USUN” aby potwierdzić.
5. Kliknij dowolny klawisz aby przejść do strony domowej.

Testowanie:



Rezultat testu:
Usunięto przedmiot.

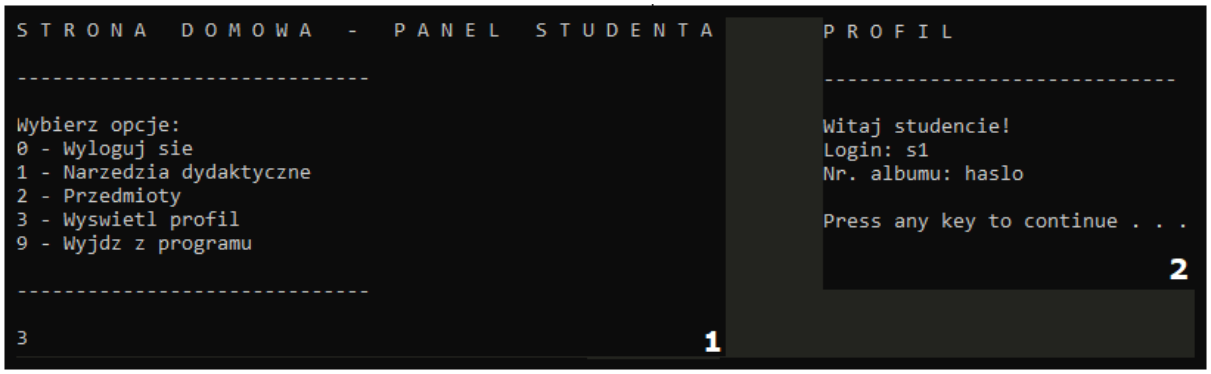
- *Sprawdź dane*

Dla Studenta

Plan testu:

1. Na stronie domowej wpisz "3" aby wybrać przedmioty i zatwierdź.
2. Kliknij dowolny klawisz aby przejść do strony domowej.

Testowanie:



Rezultat testu:
Wyświetlono profil.

Dla Wykładowcy

Plan testu:

1. Na stronie domowej wpisz "3" aby wybrać przedmioty i zatwierdź.
2. Kliknij dowolny klawisz aby przejść do strony domowej.

Testowanie:

Rezultat testu:

Wyświetlono profil.