
Bazy Danych - Projekt

Etap: 3

Autor sprawozdania: Michał Dziedziak 263901, Michał Zapała 263935

Imię i Nazwisko prowadzącego kurs: Dr inż. Marcin Łopuszyński

Dzień i godzina zajęć: Wtorek, 17:05 - 18:45

Spis treści

1	Makieta interfejsu graficznego	3
2	Diagram klas	9
3	Wdrożenie aplikacji	10
3.1	Okna startowe	11
3.2	Okna klienta	13
3.3	Okna pracownika	14
4	Testowanie aplikacji	19
4.1	Testy jednostkowe	19
4.2	Testy GUI	25

Spis tabel

Spis rysunków

1	Makieta widoku powitalnego.	3
2	Makieta widoku logowania.	3
3	Makieta widoku rejestracji.	4
4	Makieta widoku klienta.	5
5	Makieta widoku pracownika.	6
6	Makieta widoku dodawania zasobu.	7
7	Makieta widoku zarządzania kopiami zasobu.	7
8	Makieta widoku akceptacji prośby o wypożyczenie.	8
9	Makieta widoku przedłużenia prośby o wypożyczenie.	8
10	Diagram klas dla Front-End'u.	9
11	Diagram klas dla Back-End'u.	10
12	Okno powitalne.	11
13	Okno logowania.	11
14	Okno rejestracji.	12

15	Okno przeglądania zbioru biblioteki.	13
16	Okno przeglądania pożyczonych pozycji.	13
17	Okno przeglądania próśb o wypożyczenie.	14
18	Okno przeglądania zasobów biblioteki.	14
19	Okno zarządzania egzemplarzami zasobu.	15
20	Okno dodawania nowego zasobu.	15
21	Okno przeglądania wszystkich próśb o wypożyczenie.	16
22	Okno przeglądania nowych próśb o wypożyczenie.	16
23	Okno akceptacji próśby o wypożyczenie.	17
24	Okno przeglądania próśb o przedłużenie wypożyczenia.	17
25	Okno akceptacji próśby o przedłużenie wypożyczenia.	18
26	Okno przeglądania wypożyczonych egzemplarzy (oczekujących na zwrot).	18
27	Test 1: Próba wypożyczenia zasobu, dla którego nie ma dostępnych kopii.	25
28	Test 2: Próba przedłużenia wypożyczenia egzemplarza, który nie jest wypożyczony przez użytkownika.	26
29	Test 3: Próba usunięcia zasobu, którego egzemplarz jest aktualnie wypożyczony klientowi.	26
30	Test 4: Próba dodania zasobu bez uzupełnienia pól formularza.	27
31	Test 5: Próba usunięcia próśby o wypożyczenie dla niezwróconej kopii.	27
32	Test 6: Próba dodanie zasobu o nieznanym typie.	28
33	Test 7: Próba dodanie zasobu o niepoprawnej dacie wydania.	28
34	Test 8: Próba zalogowania bez podania danych.	29
35	Test 9: Próba usunięcia wypożyczonej kopii zasobu.	29
36	Test 10: Próba rejestracji konta bez uzupełnionego formularza.	30

1 Makieta interfejsu graficznego



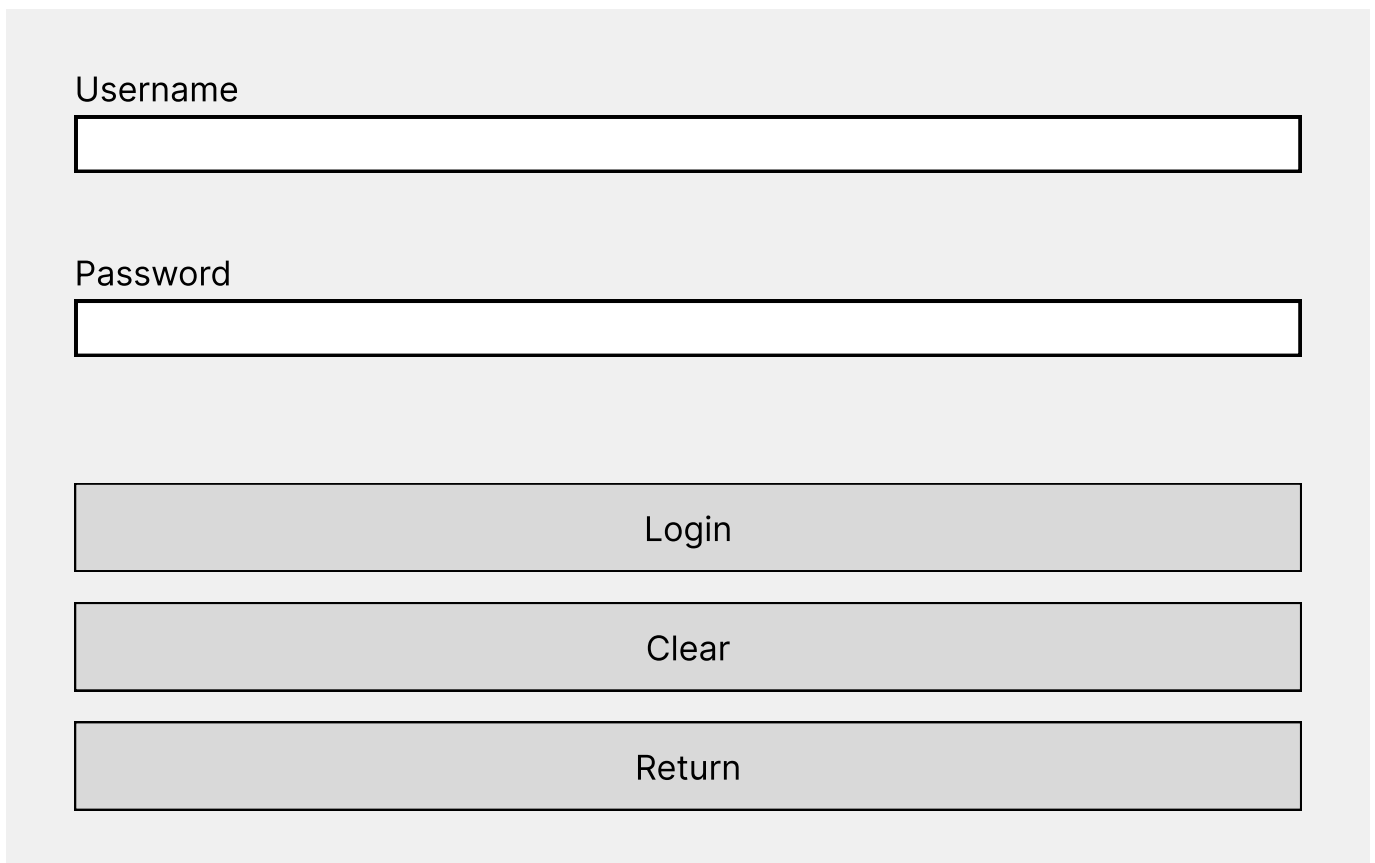
A mockup of a library welcome screen. The background is a light gray. In the center, the word "Library" is written in a large, bold, black sans-serif font. Below the title, there are two rectangular buttons with black borders and light gray backgrounds. The left button contains the text "Add New Copy" and the right button contains the text "Sign Up".

Library

Add New Copy

Sign Up

Rysunek 1: Makieta widoku powitalnego.



A mockup of a library login screen. The background is a light gray. It features two input fields for "Username" and "Password", each with a black border. Below the input fields are three rectangular buttons with black borders and light gray backgrounds, labeled "Login", "Clear", and "Return" from top to bottom.

Username

Password

Login

Clear

Return

Rysunek 2: Makieta widoku logowania.

First Name

Last Name

Username

Password

Confirm Password

Key (optional)

Register

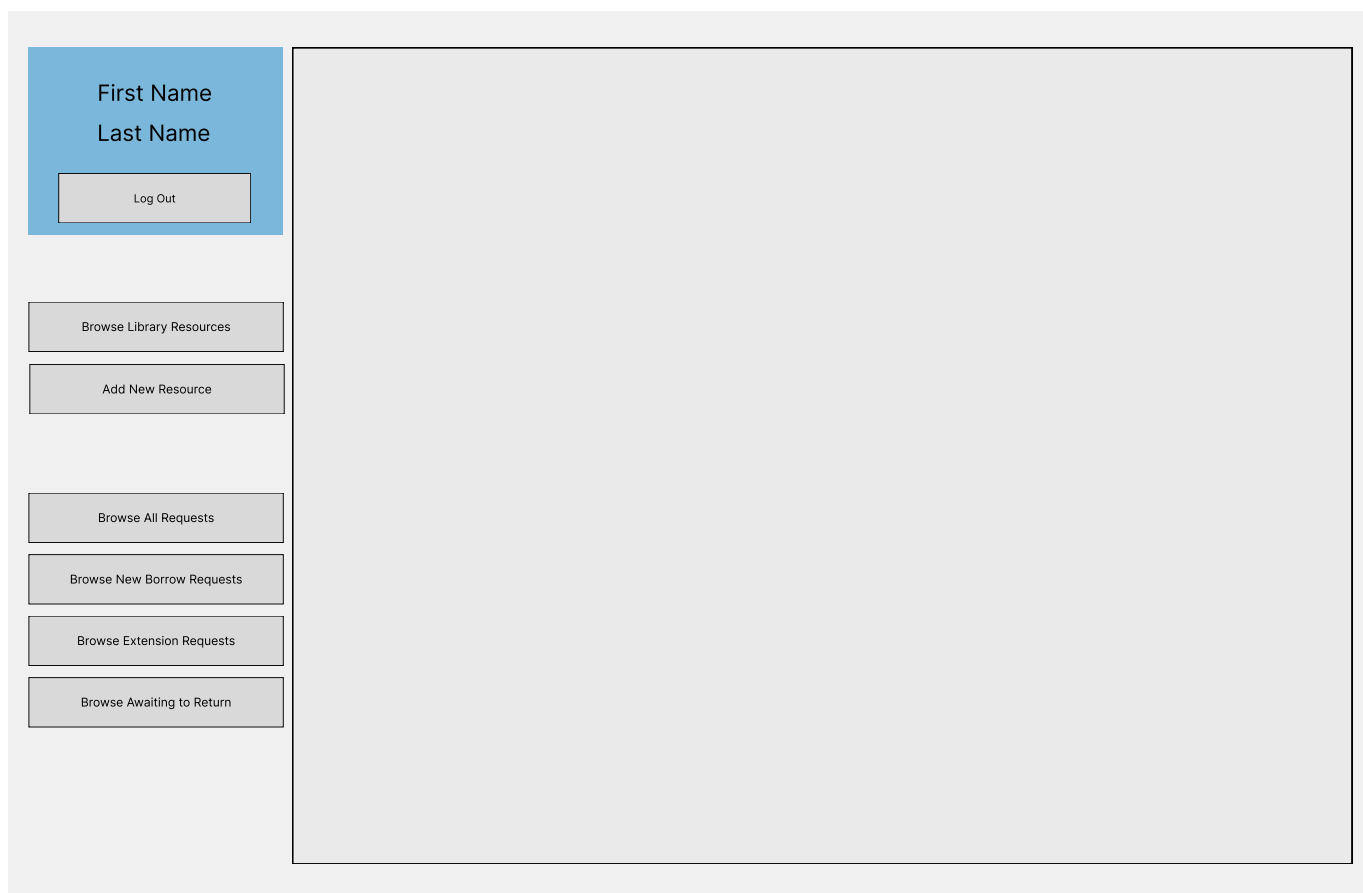
Clear

Return

Rysunek 3: Makieta widoku rejestracji.

Browse Library Resources		First Name Last Name	Log Out
Borrowed By Me	My Requests		

Rysunek 4: Makieta widoku klienta.



Rysunek 5: Makieta widoku pracownika.

Title

Author

Year Published

Resource Type

Add Resource Button

Rysunek 6: Makieta widoku dodawania zasobu.

ResourceID:

Title:

Author:

Year Published:

Type:

Add New Copy

Return

Rysunek 7: Makieta widoku zarządzania kopiami zasobu.

Copy ID:

◀

January 2024

▶

Accept

Rysunek 8: Makieta widoku akceptacji prośby o wypożyczenie.

Select Date

◀

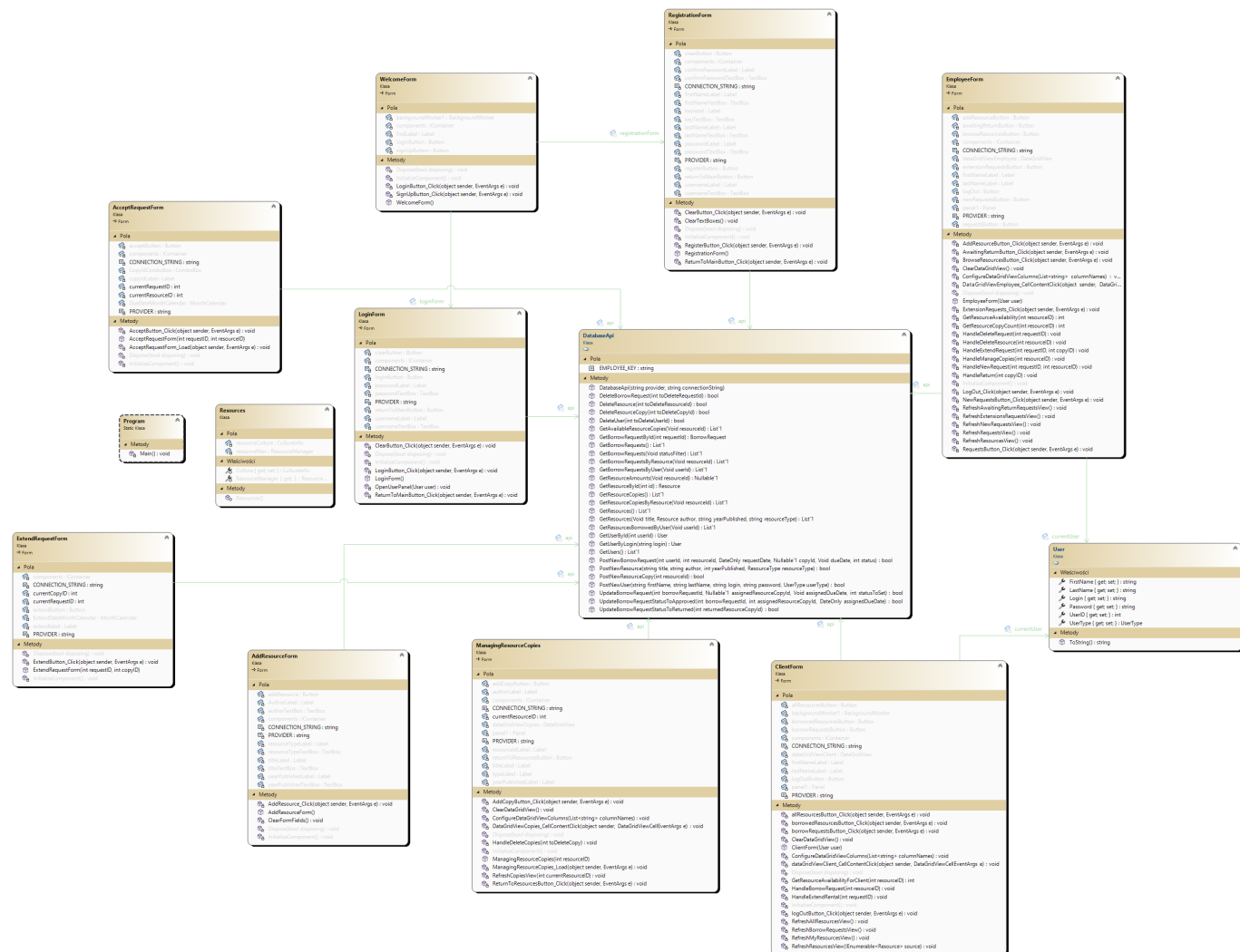
January 2024

▶

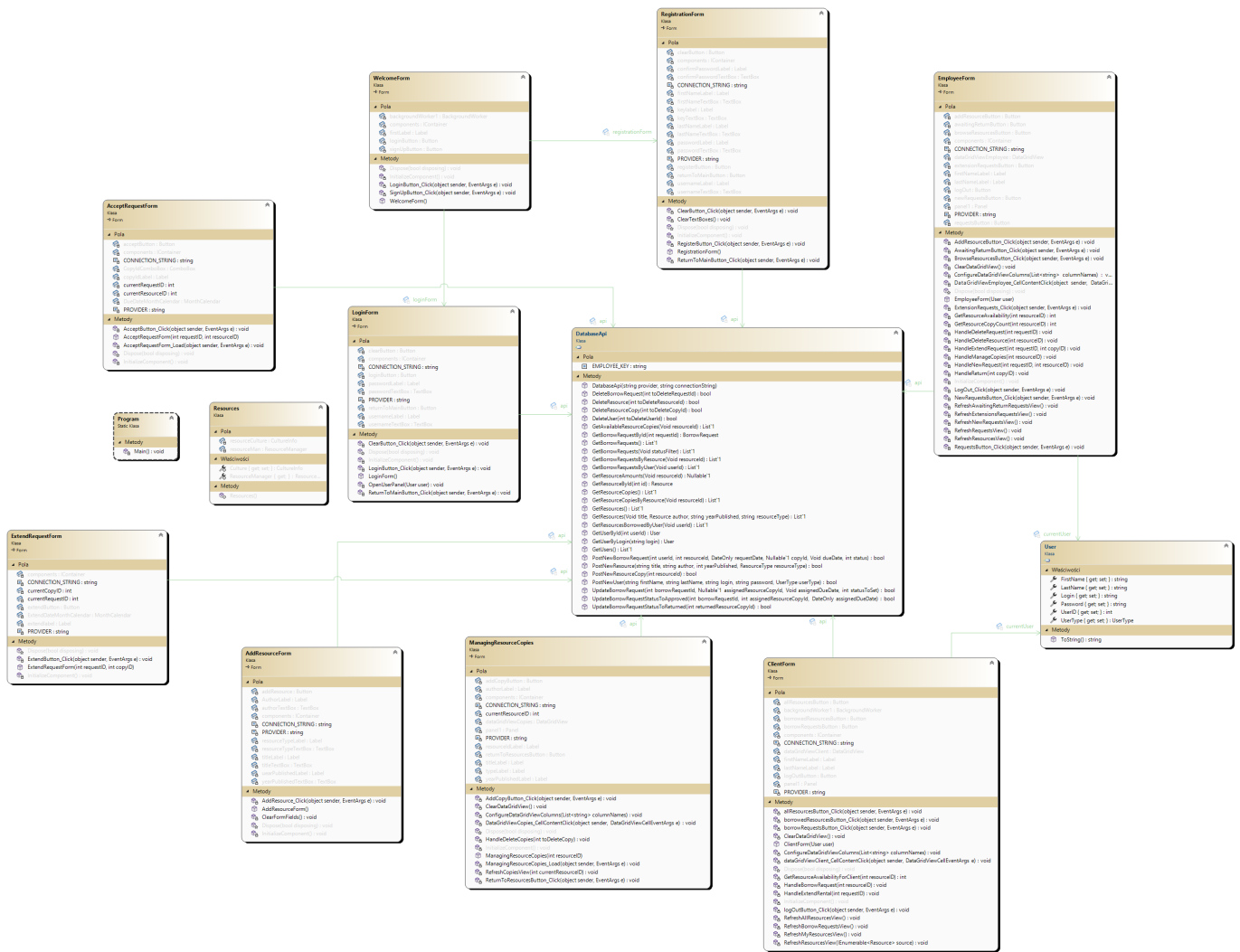
Extend

Rysunek 9: Makieta widoku przedłużenia prośby o wypożyczenie.

2 Diagram klas



Rysunek 10: Diagram klas dla Front-End'u.

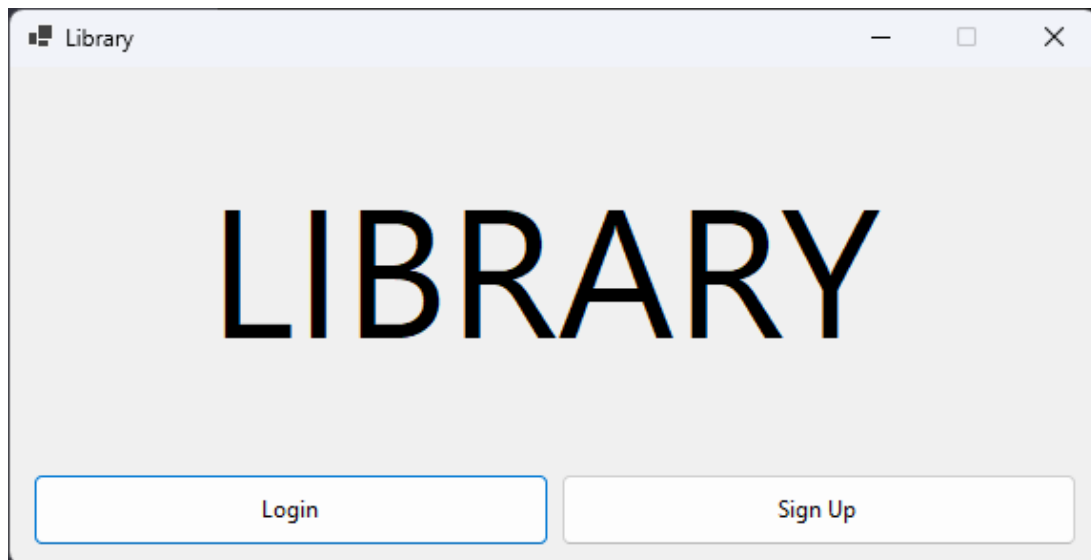


Rysunek 11: Diagram klas dla Back-End'u.

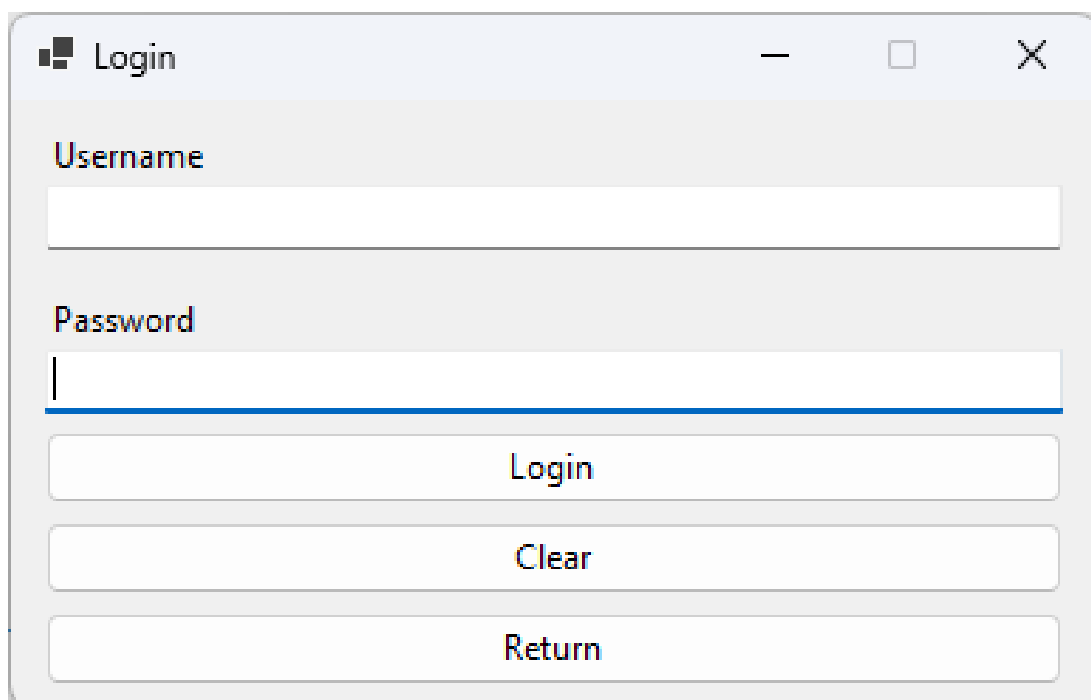
3 Wdrożenie aplikacji

//screeny

3.1 Okna startowe



Rysunek 12: Okno powitalne.



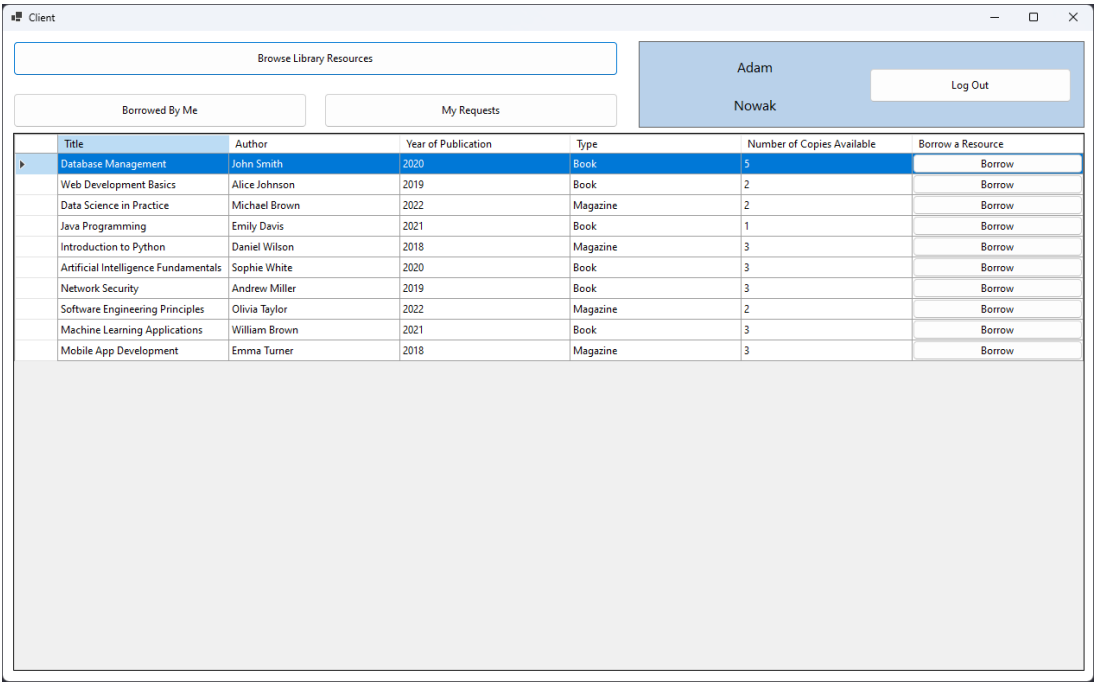
Rysunek 13: Okno logowania.

The image shows a graphical user interface for a registration window titled "Register". The window has a standard title bar with a minimize button, a maximize button (disabled), and a close button. The main area contains six text input fields, each preceded by a label: "First Name", "Last Name", "Username", "Password", "Confirm Password", and "Key (optional)". Below the input fields are three buttons: "Regsiter" (with a blue border), "Clear", and "Return".

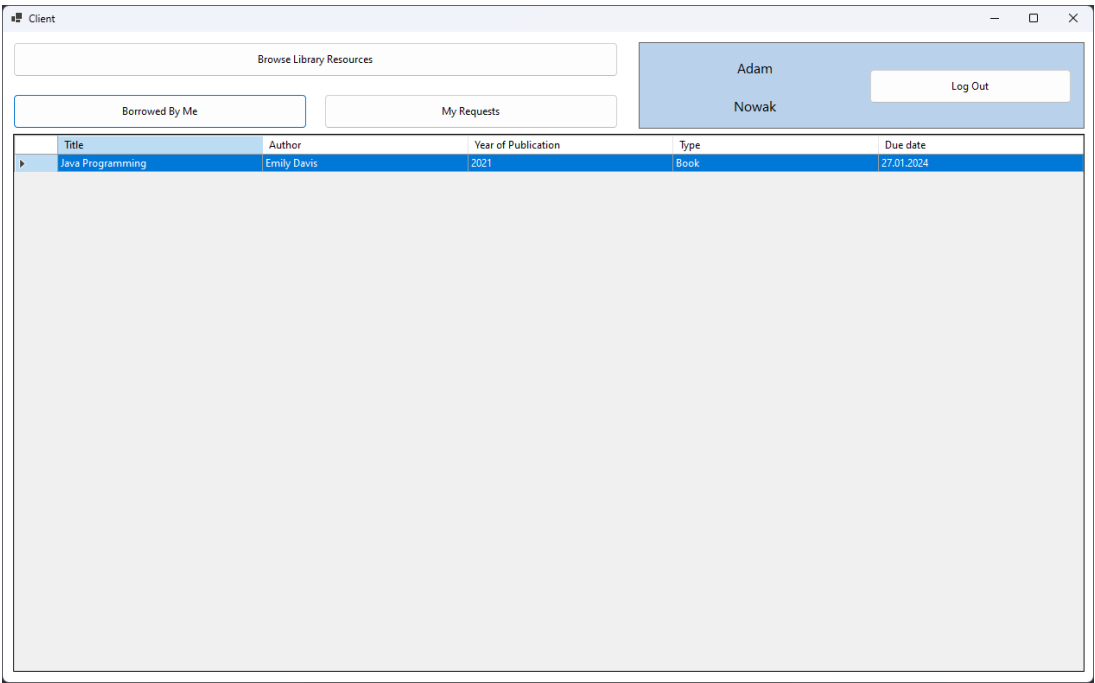
Field Label	Field Type
First Name	Text Input
Last Name	Text Input
Username	Text Input
Password	Text Input
Confirm Password	Text Input
Key (optional)	Text Input
Regsiter	Submit Button
Clear	Reset Button
Return	Cancel Button

Rysunek 14: Okno rejestracji.

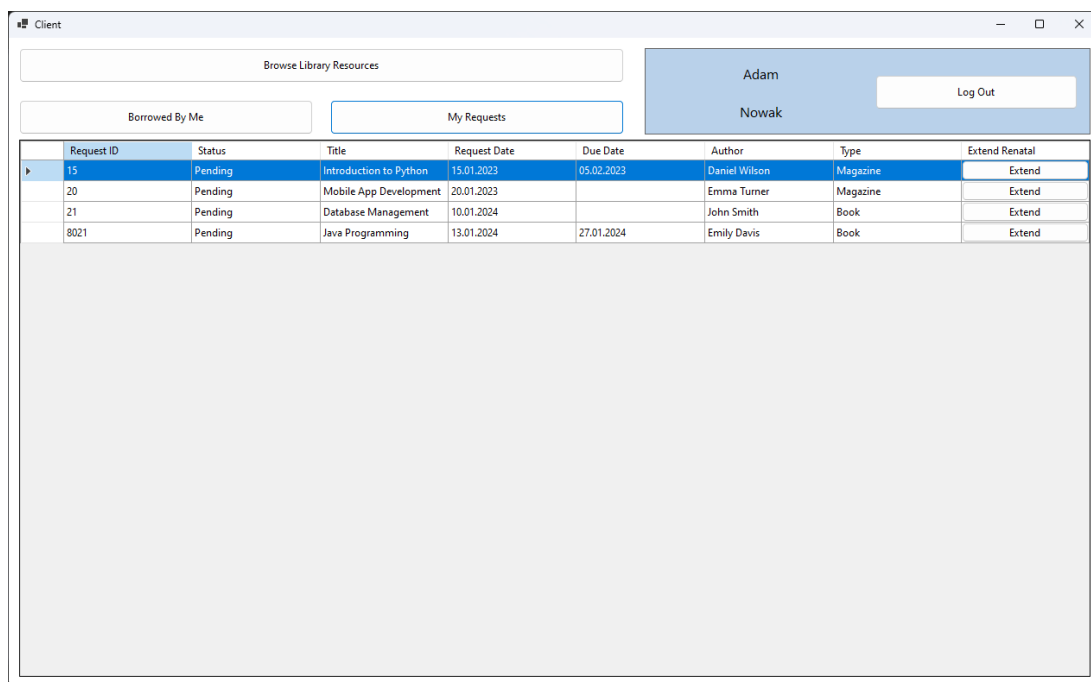
3.2 Okna klienta



Rysunek 15: Okno przeglądania zbioru biblioteki.

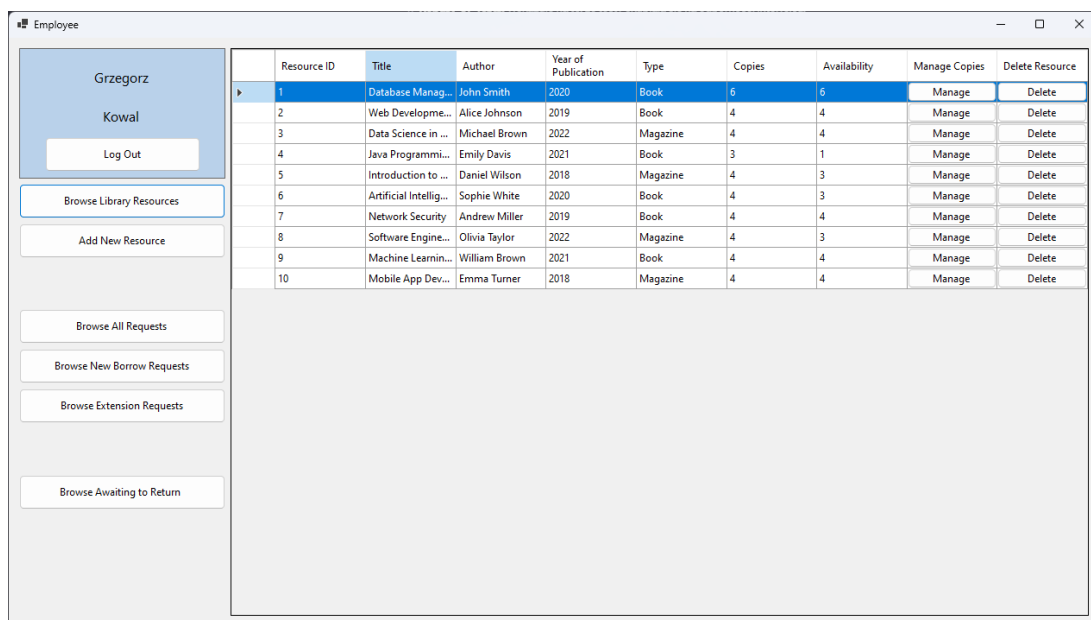


Rysunek 16: Okno przeglądania pożyczonych pozycji.



Rysunek 17: Okno przeglądania próśb o wypożyczenie.

3.3 Okna pracownika



Rysunek 18: Okno przeglądania zasobów biblioteki.

Resource Copies

Resource ID: 1

Title: Database Management

Author: John Smith

Year Published: 2020

Type: Book

Add New Copy

Return

CopyID	Status	Due Date	Delete Copy
1	Ready		Delete
2	Ready		Delete
12	Ready		Delete
22	Ready		Delete
32	Ready		Delete
11042	Ready		Delete

Rysunek 19: Okno zarządzania egzemplarzami zasobu.

Add New Resource

Title

Author

Year Published

Resource Type

Add Resource

Rysunek 20: Okno dodawania nowego zasobu.

Employee							
<div>Grzegorz</div> <div>Kowal</div> <div>Log Out</div> <div>Browse Library Resources</div> <div>Add New Resource</div> <div>Browse All Requests</div> <div>Browse New Borrow Requests</div> <div>Browse Extension Requests</div> <div>Browse Awaiting to Return</div>							
Resource ID	Copy ID	ResourceTitle	Request Date	Due Date	Status	Delete Request	
21		Database Management	10.01.2024		Pending	Delete	
11	11	Database Management	11.01.2023	26.01.2023	Returned	Delete	
12		Web Development Ba...	12.01.2023		Pending	Delete	
2		Web Development Ba...	02.01.2023		Pending	Delete	
13		Data Science in Practice	13.01.2023		Pending	Delete	
3		Data Science in Practice	03.01.2023		Pending	Delete	
8021	15	Java Programming	13.01.2024	27.01.2024	Pending	Delete	
14	13	Java Programming	14.01.2023	28.01.2023	Approved	Delete	
4	4	Java Programming	04.01.2023	15.01.2024	Returned	Delete	
15	14	Introduction to Python	15.01.2023	05.02.2023	Pending	Delete	
16	15	Artificial Intelligence F...	16.01.2023	30.01.2023	Returned	Delete	
6	6	Artificial Intelligence F...	06.01.2023	21.01.2023	Approved	Delete	
17	1	Network Security	17.01.2023	30.08.2023	Returned	Delete	
7		Network Security	07.01.2023		Pending	Delete	
18		Software Engineering ...	18.01.2023		Pending	Delete	
8	11	Software Engineering ...	08.01.2023	26.01.2024	Approved	Delete	
19	17	Machine Learning Ap...	19.01.2023	01.02.2023	Returned	Delete	
9		Machine Learning Ap...	09.01.2023		Pending	Delete	
20		Mobile App Develop...	20.01.2023		Pending	Delete	
10	10	Mobile App Develop...	10.01.2023	25.01.2023	Returned	Delete	

Rysunek 21: Okno przeglądania wszystkich próśb o wypożyczenie.

Employee					
<div>Grzegorz</div> <div>Kowal</div> <div>Log Out</div> <div>Browse Library Resources</div> <div>Add New Resource</div> <div>Browse All Requests</div> <div>Browse New Borrow Requests</div> <div>Browse Extension Requests</div> <div>Browse Awaiting to Return</div>					
Resource ID	Title	Request Date	Status	Consider Borrow Request	
2	Web Development Basics	02.01.2023	Pending	Consider	
3	Data Science in Practice	03.01.2023	Pending	Consider	
7	Network Security	07.01.2023	Pending	Consider	
9	Machine Learning Applications	09.01.2023	Pending	Consider	
12	Web Development Basics	12.01.2023	Pending	Consider	
13	Data Science in Practice	13.01.2023	Pending	Consider	
18	Software Engineering Principles	18.01.2023	Pending	Consider	
20	Mobile App Development	20.01.2023	Pending	Consider	
21	Database Management	10.01.2024	Pending	Consider	

Rysunek 22: Okno przeglądania nowych próśb o wypożyczenie.

Accept New Request

Copy ID: 3

styczeń 2024

pon.	wt.	śr.	czw.	pt.	sob.	niedz.
25	26	27	28	29	30	31
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

Dziś: 16.01.2024

Accept

Rysunek 23: Okno akceptacji prośby o wypożyczenie.

Employee

Grzegorz Kowal

Log Out

Browse Library Resources

Add New Resource

Browse All Requests

Browse New Borrow Requests

Browse Extension Requests

Browse Awaiting to Return

Resource ID	Copy ID	Title	Request Date	Due Date	Status	Consider Extension
15	14	Introduction to Python	15.01.2023	05.02.2023	Pending	Consider
8021	15	Java Programming	13.01.2024	27.01.2024	Pending	Consider

Rysunek 24: Okno przeglądania próśb o przedłużenie wypożyczenia.

Extend Request

Select Date

styczeń 2024

pon.	wt.	śr.	czw.	pt.	sob.	niedz.
25	26	27	28	29	30	31
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

Dziś: 16.01.2024

Extend

Rysunek 25: Okno akceptacji prośby o przedłużenie wypożyczenia.

Employee

Grzegorz Kowal

Log Out

Browse Library Resources

Add New Resource

Browse All Requests

Browse New Borrow Requests

Browse Extension Requests

Browse Awaiting to Return

Resource ID	Copy ID	Title	Request Date	Due Date	Status	Return a Copy
6	6	Artificial Intelligence F...	06.01.2023	21.01.2023	Approved	Return
8	11	Software Engineering ...	08.01.2023	26.01.2024	Approved	Return
14	13	Java Programming	14.01.2023	28.01.2023	Approved	Return

Rysunek 26: Okno przeglądania wypożyczonych egzemplarzy (oczekujących na zwrot).

4 Testowanie aplikacji

4.1 Testy jednostkowe

```
namespace LibraryDatabaseApiUnitTests;

public class Tests
{
    const string PROVIDER = ".NET Framework Data Provider for SQL Server";
    const string CONNECTION_STRING =
        "Data Source=(localdb)\\MSSQLLocalDB;Initial Catalog=LibraryDataBase;Integrated Security=True";
    private DatabaseApi api = new DatabaseApi(PROVIDER, CONNECTION_STRING);

    [Fact]
    public void GetUserByLoginTest()
    {
        var users = api.GetUsers();
        Assert.NotNull(users);
        Assert.NotEmpty(users);

        var user = api.GetUserByLogin(users[0].Login);
        Assert.NotNull(user);

        Assert.Equal(users[0].UserID, user.UserID);
        Assert.Equal(users[0].FirstName, user.FirstName);
        Assert.Equal(users[0].LastName, user.LastName);
        Assert.Equal(users[0].Login, user.Login);
        Assert.Equal(users[0].Password, user.Password);
        Assert.Equal(users[0].UserType, user.UserType);
    }

    [Fact]
    public void GetUserByIdTest()
    {
        var users = api.GetUsers();
        Assert.NotNull(users);
        Assert.NotEmpty(users);

        var user = api.GetUserById(users[0].UserID);
        Assert.NotNull(user);

        Assert.Equal(users[0].UserID, user.UserID);
        Assert.Equal(users[0].FirstName, user.FirstName);
        Assert.Equal(users[0].LastName, user.LastName);
        Assert.Equal(users[0].Login, user.Login);
        Assert.Equal(users[0].Password, user.Password);
        Assert.Equal(users[0].UserType, user.UserType);
    }

    [Fact]
    public void GetUsersTest()
    {
        var users = api.GetUsers();
        Assert.NotNull(users);
    }

    [Fact]
```

```

public void GetResourceAmountsTest()
{
    var resources = api.GetResources();
    Assert.NotNull(resources);
    Assert.NotEmpty(resources);

    var tmp = api.GetResourceAmounts(resources[0].ResourceID);
    Assert.NotNull(tmp);
    Assert.Equal(tmp.Value.amount, tmp.Value.borrowed + tmp.Value.available);
}

[Fact]
public void GetBorrowRequestsTest()
{
    var allReq = api.GetBorrowRequests();
    Assert.NotNull(allReq);
    Assert.NotEmpty(allReq);

    var returnedReq = api.GetBorrowRequests(Status.Returned);
    var approvedReq = api.GetBorrowRequests(Status.Approved);
    var pendingReq = api.GetBorrowRequests(Status.Pending);

    Assert.NotNull(returnedReq);
    Assert.NotNull(approvedReq);
    Assert.NotNull(pendingReq);

    Assert.Equal(allReq.FindAll(r => r.Status == Status.Returned).Count(), returnedReq.Count());
    Assert.Equal(allReq.FindAll(r => r.Status == Status.Approved).Count(), approvedReq.Count());
    Assert.Equal(allReq.FindAll(r => r.Status == Status.Pending).Count(), pendingReq.Count());

    Assert.True(pendingReq.TrueForAll(r => r.DueDate == null && r.CopyID == null));
    Assert.True(approvedReq.TrueForAll(r => r.DueDate != null && r.CopyID != null));
    Assert.True(returnedReq.TrueForAll(r => r.DueDate != null && r.CopyID != null));
}

[Fact]
public void GetResourceCopiesTest()
{
    var resAll = api.GetResourceCopies();
    Assert.NotNull(resAll);
}

[Fact]
public void GetResourcesTest()
{
    var resAll = api.GetResources();
    Assert.NotNull(resAll);
    Assert.NotEmpty(resAll);

    var fil = api.GetResources(
        resAll[0].Title, resAll[0].Author, resAll[0].YearPublished, resAll[0].ResourceType);
    Assert.NotNull(fil);
    Assert.NotEmpty(fil);

    foreach (var item in fil)
    {
        Assert.Equal(resAll[0].ResourceID, item.ResourceID);
    }
}

```

```

        Assert.Equal(resAll[0].Title, item.Title);
        Assert.Equal(resAll[0].Author, item.Author);
        Assert.Equal(resAll[0].YearPublished, item.YearPublished);
        Assert.Equal(resAll[0].ResourceType, item.ResourceType);
    }
}

[Fact]
public void GetResourcesBorrowedByUserTest()
{
    var users = api.GetUsers();
    Assert.NotNull(users);
    Assert.NotEmpty(users);

    List<Resource>? borrowed = null;
    int userID = 0;
    foreach (var user in users)
    {
        borrowed = api.GetResourcesBorrowedByUser(user.UserID);
        Assert.NotNull(borrowed);
        userID = user.UserID;
        if (borrowed.Count > 0)
        {
            break;
        }
    }
    Assert.NotNull(borrowed);

    var reqAll = api.GetBorrowRequests();
    Assert.NotNull(reqAll);

    var resCopiesAll = api.GetResourceCopies();
    Assert.NotNull(resCopiesAll);

    var customResIds = reqAll.FindAll(
        r => r.UserID == userID && r.Status == Status.Approved).Select(r => r.ResourceID);

    Assert.True(borrowed.TrueForAll(b => customResIds.Contains(b.ResourceID)));
}

[Fact]
public void UpdateBorrowRequestStatusToReturnedTest()
{
    var borrowedReq = api.GetBorrowRequests();
    Assert.NotNull(borrowedReq);
    Assert.NotEmpty(borrowedReq);

    var approvedAll = borrowedReq.FindAll(b => b.Status == Status.Approved);
    Assert.NotNull(approvedAll);

    foreach (var item in approvedAll)
    {
        Assert.NotNull(item.CopyID);
        Assert.True(api.UpdateBorrowRequestStatusToReturned(item.CopyID.Value));
    }
}

```

```

var updatedBorrowedReq = api.GetBorrowRequests();
Assert.NotNull(updatedBorrowedReq);
Assert.True(updatedBorrowedReq.TrueForAll(
    ubr => ubr.Status == Status.Returned || ubr.Status == Status.Pending));

foreach (var item in approvedAll)
{
    Assert.True(api.UpdateBorrowRequest(
        item.RequestID, item.CopyID, item.DueDate, Status.Approved));
}
}

[Fact]
public void UpdateBorrowRequestStatusToApprovedTest()
{
    var borrowedReq = api.GetBorrowRequests();
    Assert.NotNull(borrowedReq);
    Assert.NotEmpty(borrowedReq);

    var pendingAll = borrowedReq.FindAll(b => b.Status == Status.Pending);
    Assert.NotNull(pendingAll);

    var date = new DateOnly(2023, 1, 1);
    foreach (var item in pendingAll)
    {
        Assert.True(api.UpdateBorrowRequestStatusToApproved(item.RequestID, 1, date));
    }

    var updatedBorrowedReq = api.GetBorrowRequests();
    Assert.NotNull(updatedBorrowedReq);
    Assert.True(updatedBorrowedReq.TrueForAll(
        ubr => ubr.Status == Status.Returned || ubr.Status == Status.Approved));

    foreach (var item in pendingAll)
    {
        Assert.True(api.UpdateBorrowRequest(
            item.RequestID, item.CopyID, item.DueDate, item.Status));
    }
}

[Fact]
public void UpdateBorrowRequestTest()
{
    var borrowedReq = api.GetBorrowRequests();
    Assert.NotNull(borrowedReq);
    Assert.NotEmpty(borrowedReq);

    var pendingAll = borrowedReq.FindAll(b => b.Status == Status.Pending);
    Assert.NotNull(pendingAll);
    var approvedAll = borrowedReq.FindAll(b => b.Status == Status.Approved);
    Assert.NotNull(approvedAll);

    var date = new DateOnly(2023, 1, 1);
    foreach (var item in borrowedReq)
    {
        Assert.True(api.UpdateBorrowRequest(item.RequestID, 1, date, Status.Returned));
    }
}

```

```

    }

    var updatedBorrowedReq = api.GetBorrowRequests();
    Assert.NotNull(updatedBorrowedReq);
    Assert.True(updatedBorrowedReq.TrueForAll(ubr => ubr.Status == Status.Returned));
    Assert.True(updatedBorrowedReq.TrueForAll(ubr => ubr.CopyID == 1));
    Assert.True(updatedBorrowedReq.TrueForAll(ubr => ubr.DueDate == date));

    foreach (var item in borrowedReq)
    {
        Assert.True(api.UpdateBorrowRequest(
            item.RequestID, item.CopyID, item.DueDate, item.Status));
    }
}

[Fact]
public void PostDeleteUserTest()
{
    Assert.True(api.PostNewUser(
        "testname", "testlastname", "testlogin", "123", UserType.Employee));

    var user = api.GetUserByLogin("testlogin");
    Assert.NotNull(user);

    var users = api.GetUsers();
    Assert.NotNull(users);

    Assert.True(api.DeleteUser(user.UserID));

    var newUsers = api.GetUsers();
    Assert.NotNull(newUsers);

    Assert.Equal(users.Count(), newUsers.Count() + 1);
}

[Fact]
public void PostDeleteBorrowRequestTest()
{
    var users = api.GetUsers();
    var res = api.GetResources();
    Assert.NotNull(users);
    Assert.NotEmpty(users);
    Assert.NotNull(res);
    Assert.NotEmpty(res);

    var req = api.GetBorrowRequests();
    Assert.NotNull(req);
    Assert.NotEmpty(req);

    Assert.True(api.PostNewBorrowRequest(
        users[0].UserID, res[0].ResourceID, new DateOnly(1900,1,1), null, null, Status.Pending));

    var newReq = api.GetBorrowRequests();
    Assert.NotNull(newReq);

    var selected = newReq.FindAll(e =>
        e.UserID == users[0].UserID &&

```



```

    res[0].ResourceID == e.ResourceID &&
    e.RequestDate == new DateOnly(1900, 1, 1) &&
    e.DueDate == null &&
    e.CopyID == null &&
    e.Status == Status.Pending).Single();

    Assert.Equal(req.Count() + 1, newReq.Count());

    Assert.True(api.DeleteBorrowRequest(selected.RequestID));

    var newNewReq = api.GetBorrowRequests();
    Assert.NotNull(newNewReq);
    Assert.Equal(req.Count(), newNewReq.Count());
}

[Fact]
public void PostDeleteResourceCopyTest()
{
    var res = api.GetResources();
    Assert.NotNull(res);
    Assert.NotEmpty(res);

    var resCop = api.GetResourceCopies();
    Assert.NotNull(resCop);
    Assert.NotEmpty(resCop);

    Assert.True(api.PostNewResourceCopy(res[0].ResourceID));

    var newResCop = api.GetResourceCopies();
    Assert.NotNull(newResCop);

    Assert.Equal(resCop.Count() + 1, newResCop.Count());

    var selected = newResCop.FindAll(s => s.ResourceID == res[0].ResourceID).MaxBy(s => s.CopyID);
    Assert.NotNull(selected);

    Assert.True(api.DeleteResourceCopy(selected.CopyID));

    var newNewResCop = api.GetResourceCopies();
    Assert.NotNull(newNewResCop);
    Assert.Equal(resCop.Count(), newNewResCop.Count());
}

[Fact]
public void PostDeleteResourceTest()
{
    var res = api.GetResources();
    Assert.NotNull(res);
    Assert.NotEmpty(res);

    Assert.True(api.PostNewResource("Testtitle", "testauth", 1000, ResourceType.Magazine));

    var newRes = api.GetResources();
    Assert.NotNull(newRes);

    Assert.Equal(res.Count() + 1, newRes.Count());
}

```

```

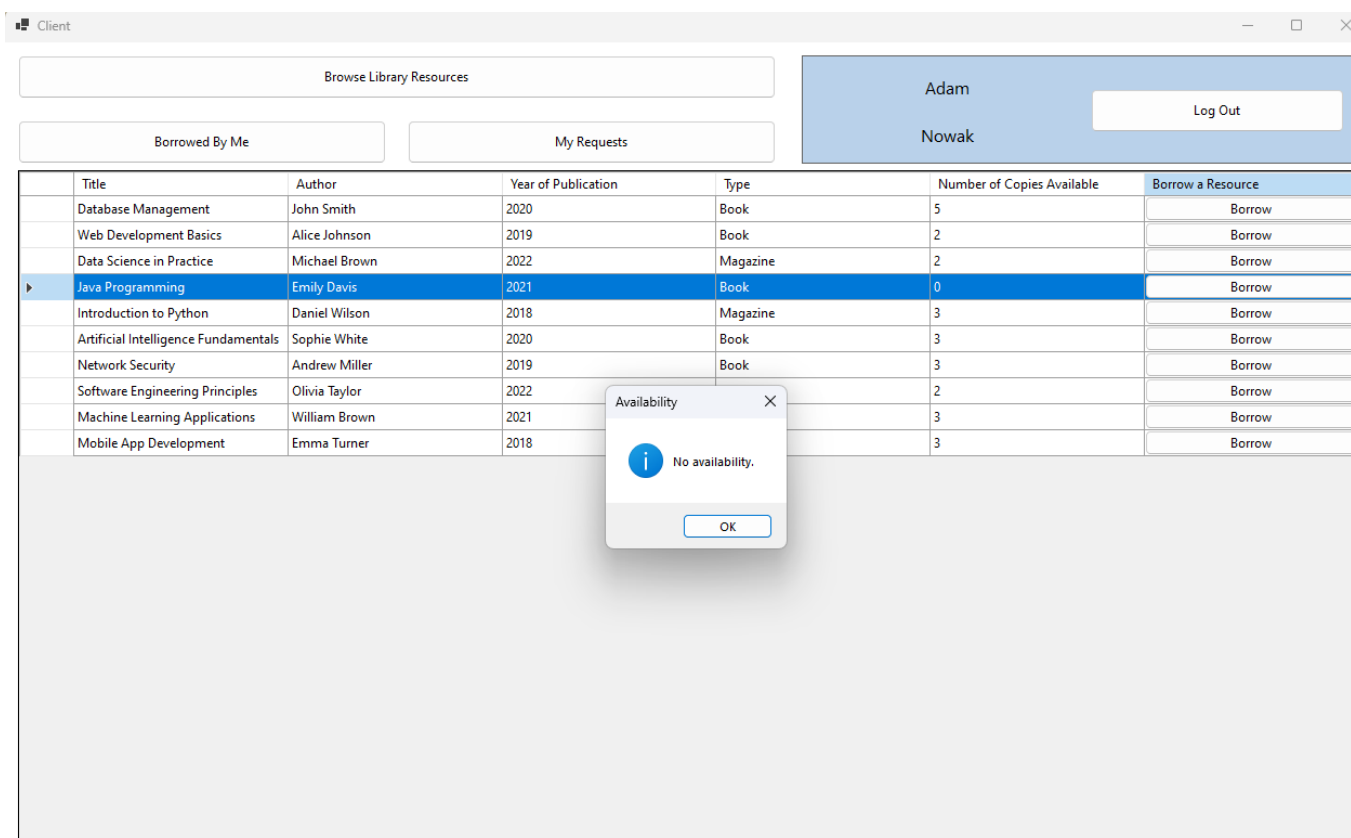
var added = api.GetResources("Testtitle", "testauth", 1000, ResourceType.Magazine);
Assert.NotNull(added);

Assert.True(api.DeleteResource(added[0].ResourceID));

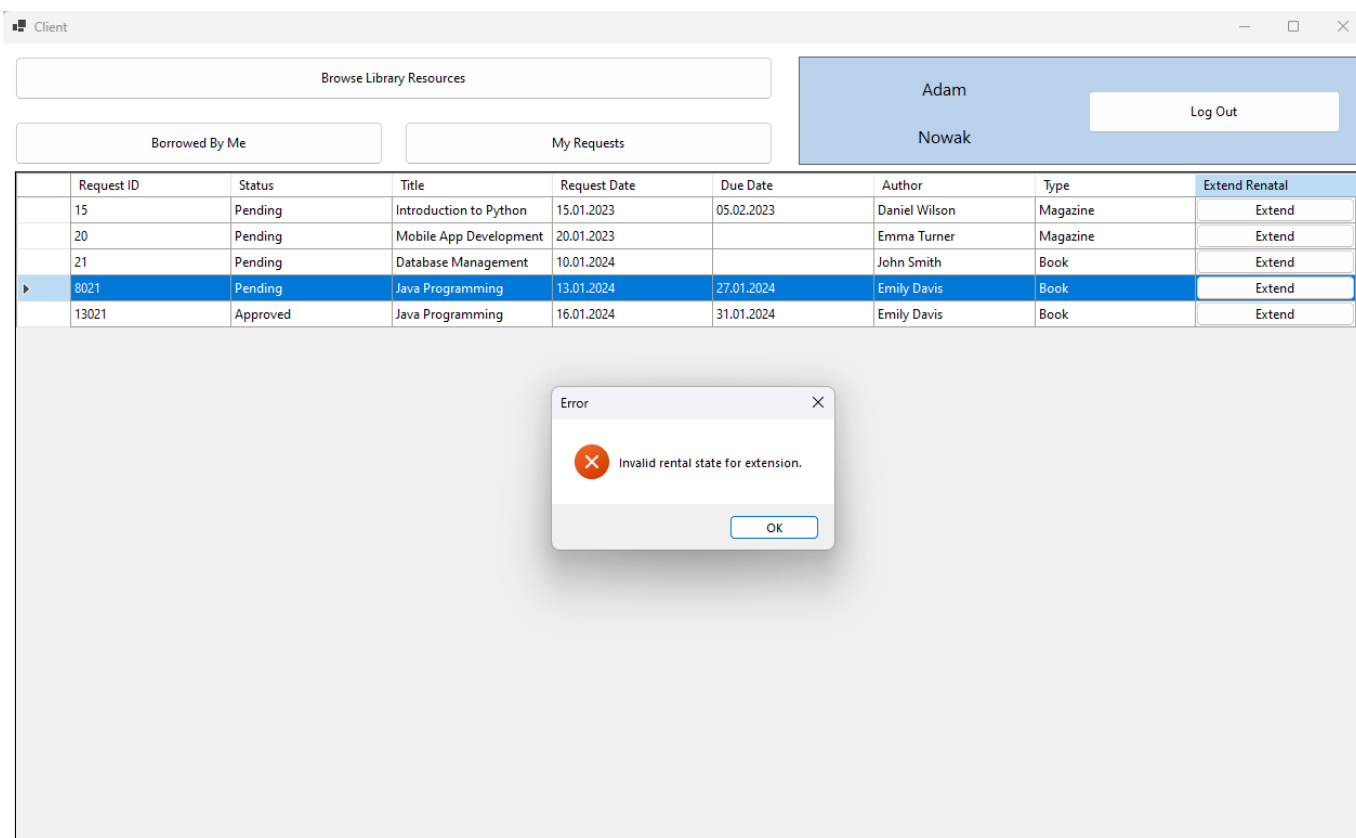
var newNewRes = api.GetResources();
Assert.NotNull(newNewRes);
Assert.Equal(res.Count(), newNewRes.Count());
}
}

```

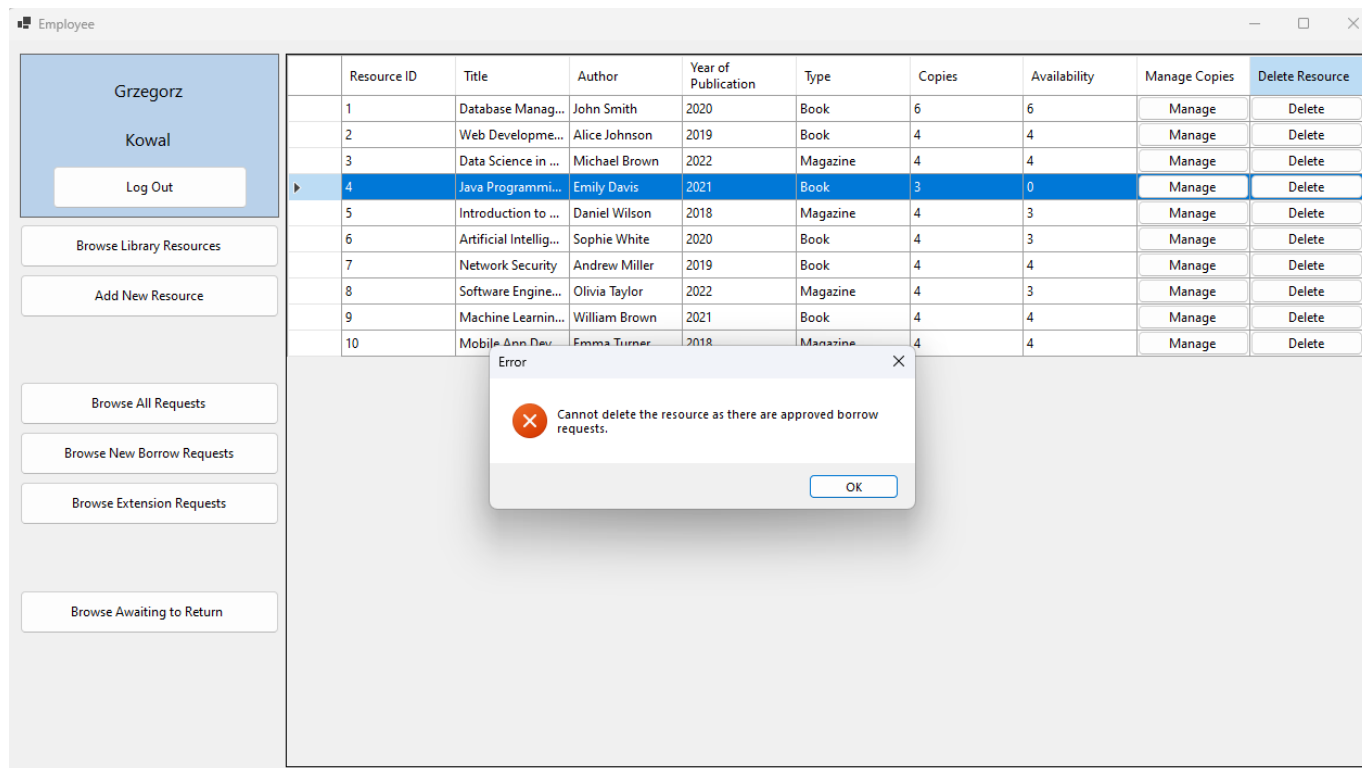
4.2 Testy GUI



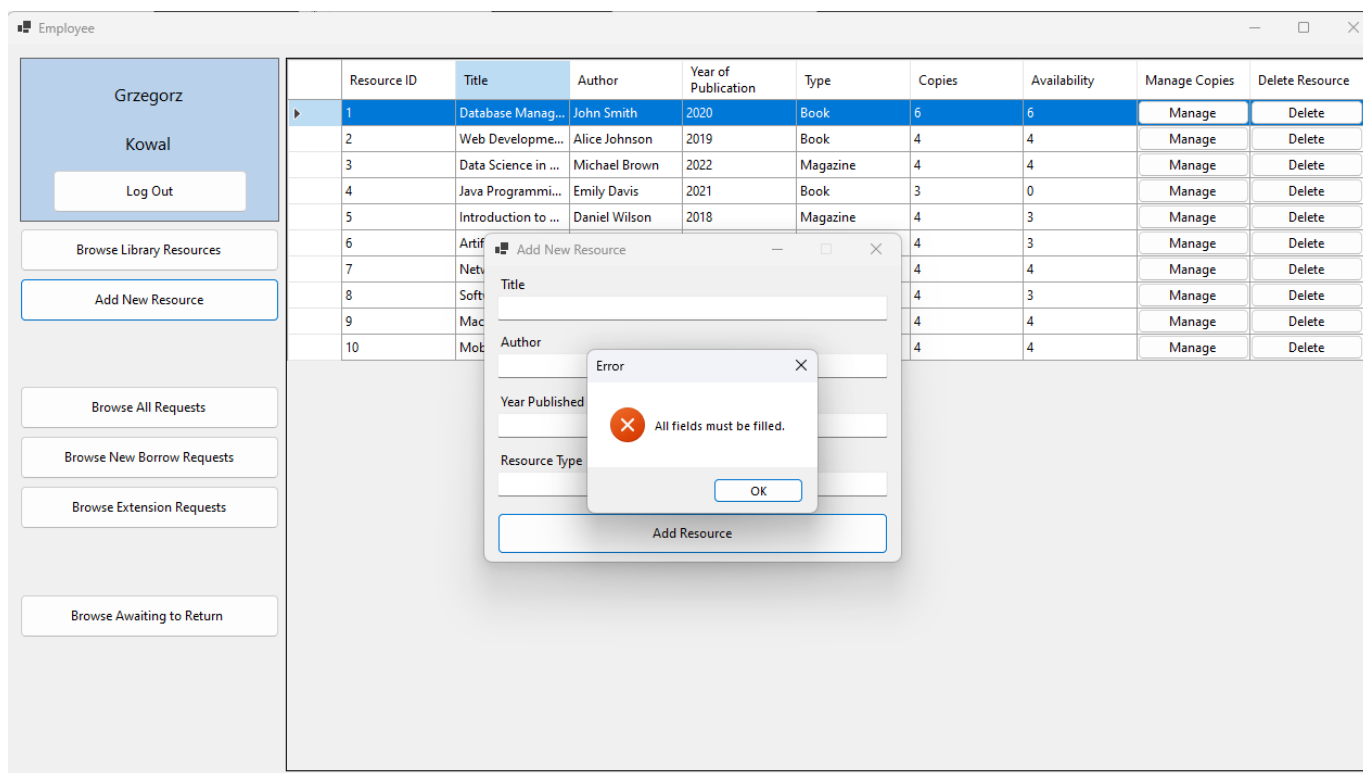
Rysunek 27: Test 1: Próba wypożyczenia zasobu, dla którego nie ma dostępnych kopii



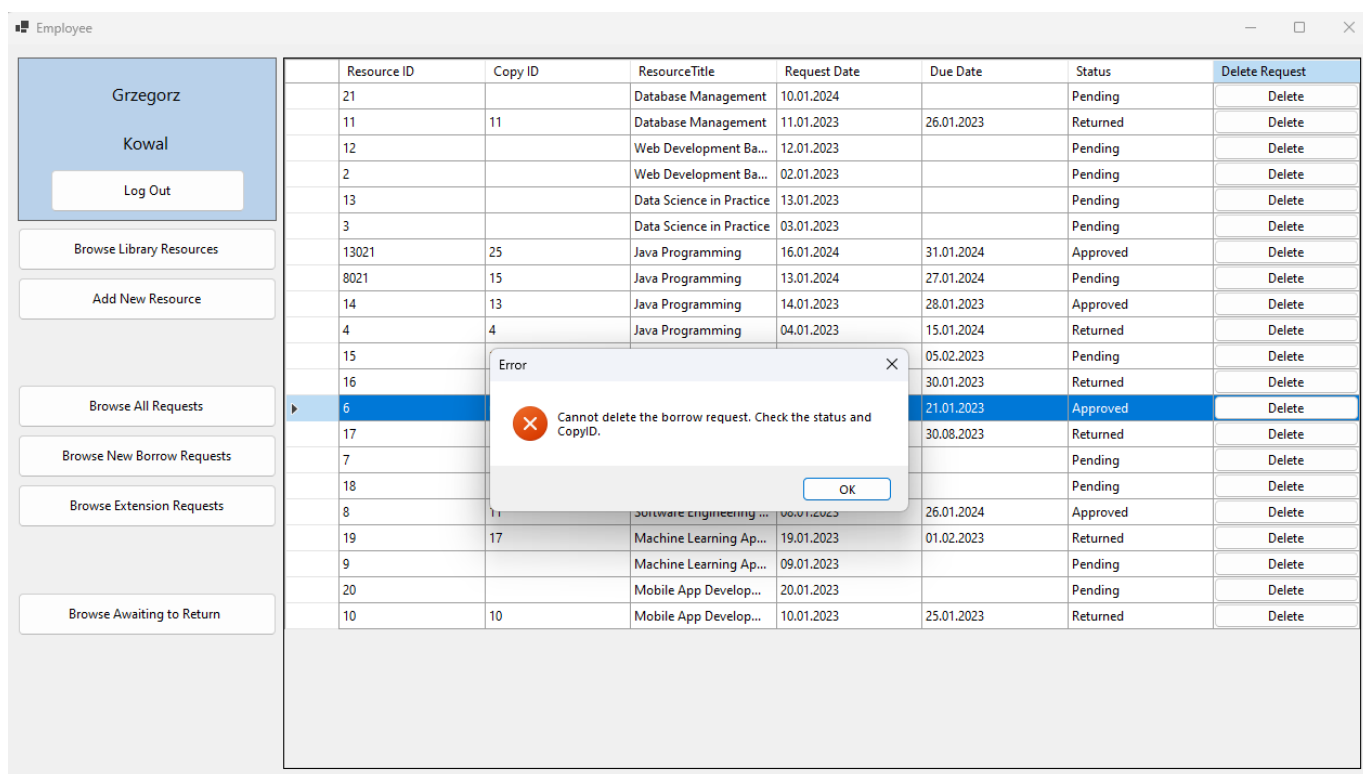
Rysunek 28: Test 2: Próba przedłużenia wypożyczenia egzemplarza, który nie jest wypożyczony przez użytkownika



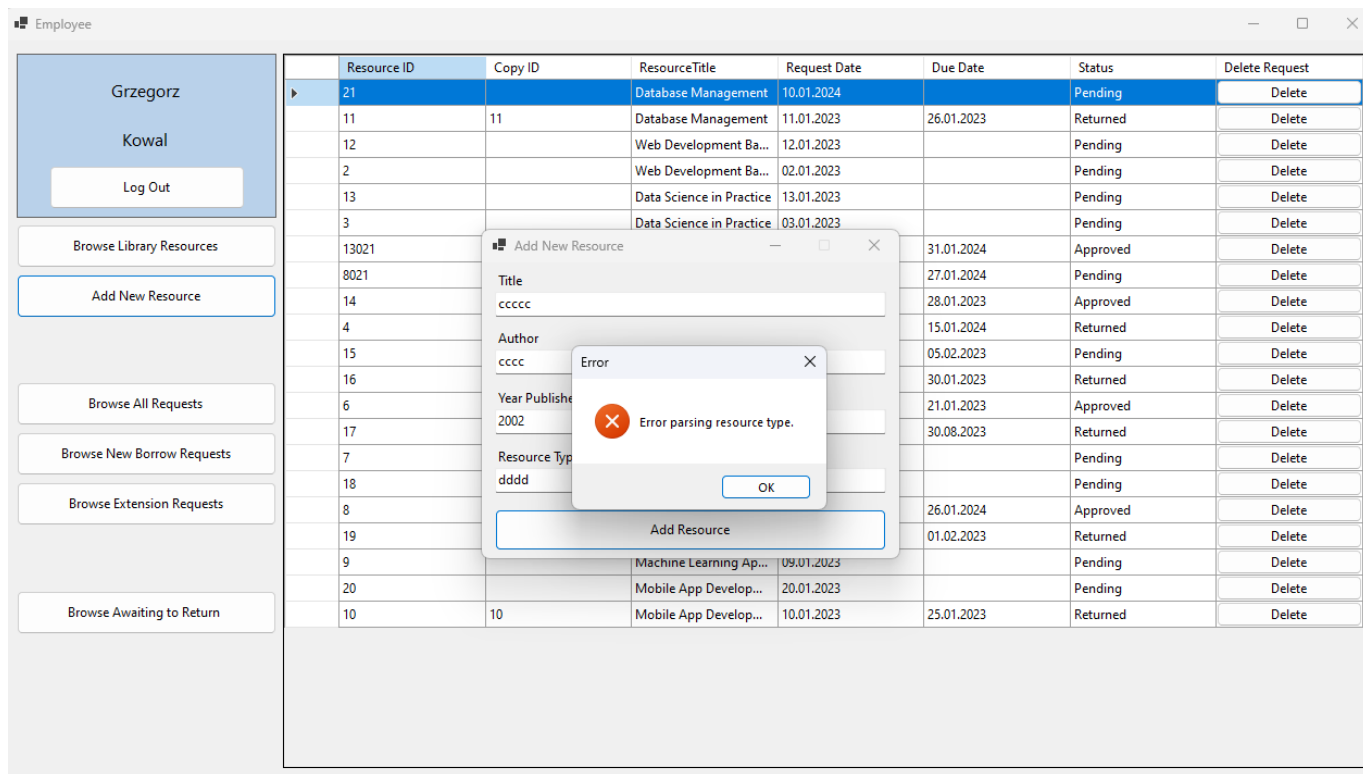
Rysunek 29: Test 3: Próba usunięcia zasobu, którego egzemplarz jest aktualnie wypożyczony klientowi



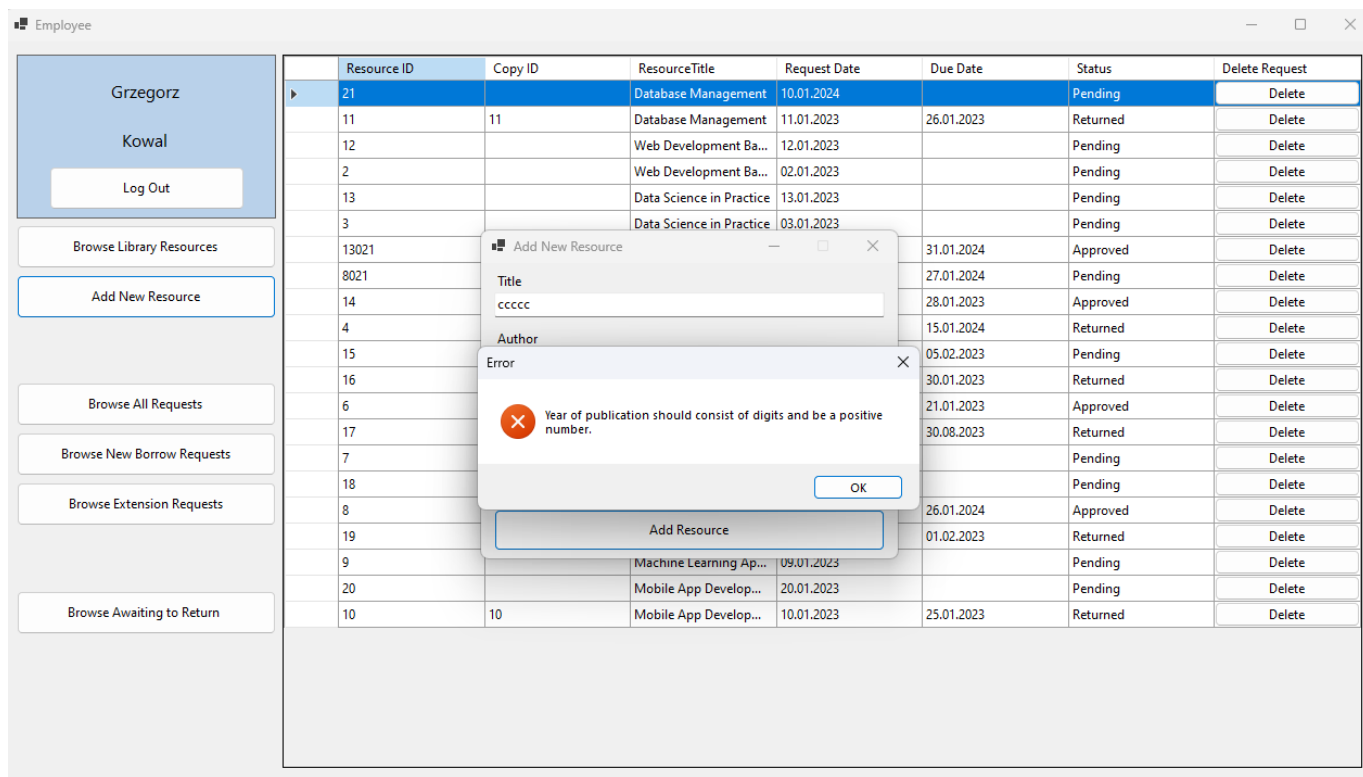
Rysunek 30: Test 4: Próba dodania zasobu bez uzupełnienia pól formularza



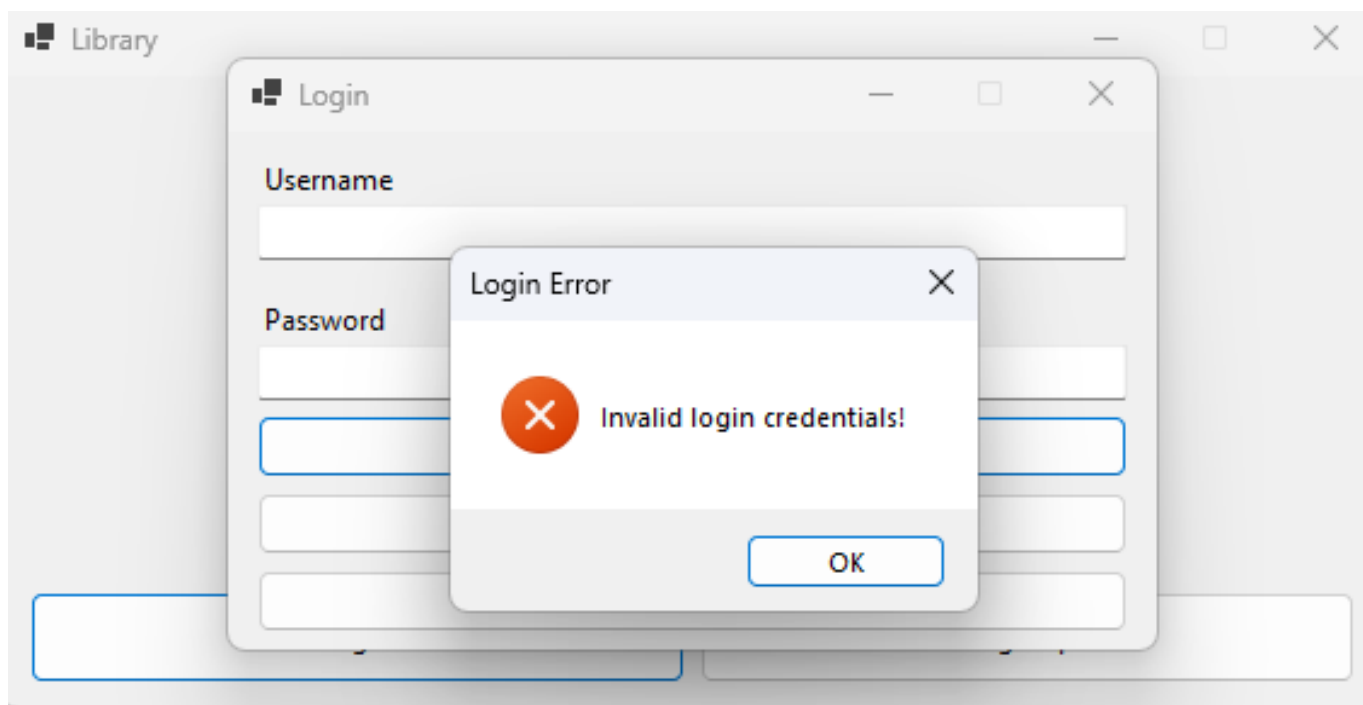
Rysunek 31: Test 5: Próba usunięcie prośby o wypożyczenie dla niezwróconej kopii.



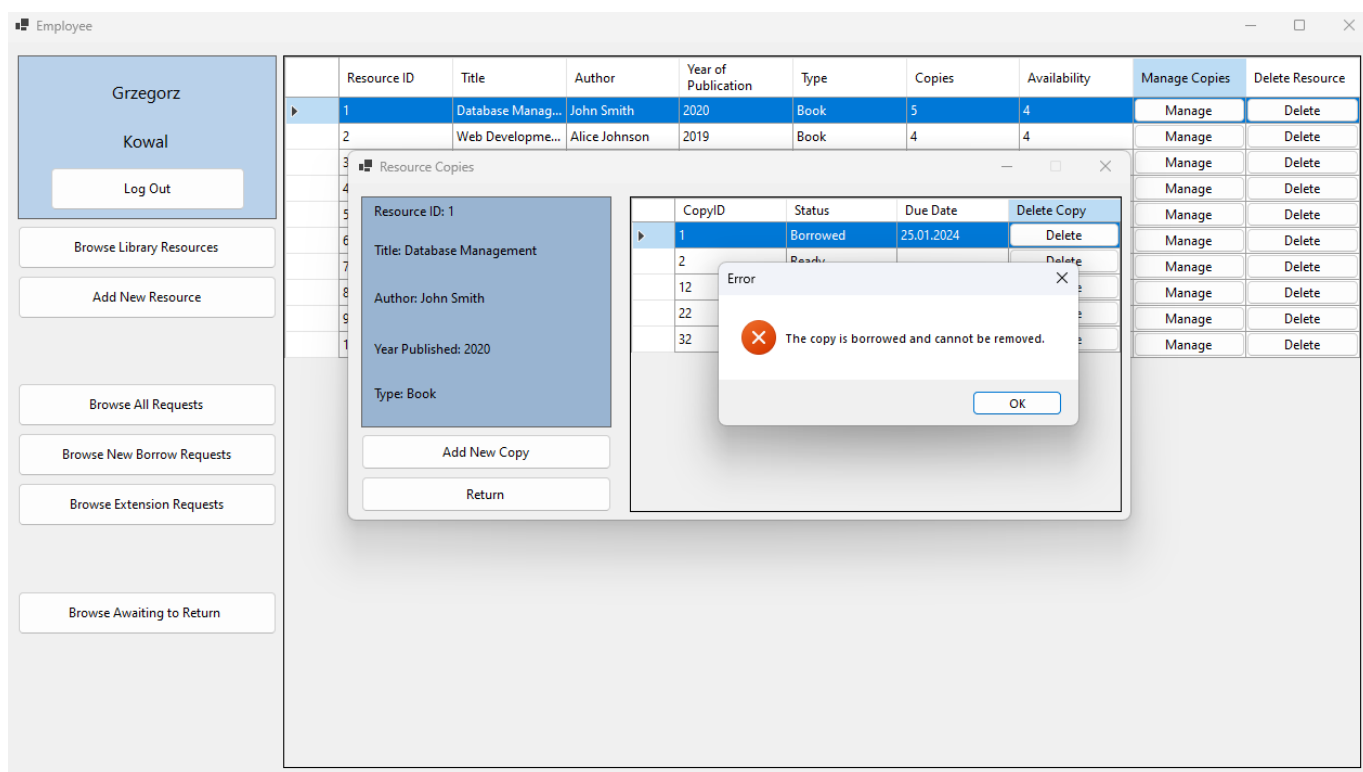
Rysunek 32: Test 6: Próba dodanie zasobu o nieznanym typie.



Rysunek 33: Test 7: Próba dodanie zasobu o niepoprawnej dacie wydania.



Rysunek 34: Test 8: Próba zalogowania bez podania danych



Rysunek 35: Test 9: Próba usunięcia wypożyczonej kopii zasobu.

The image shows a web application window titled "Register". It contains a registration form with the following fields: "First Name", "Last Name", "Username", "Password", "Confirm Password", and "Key (optional)". Each field is represented by a white text input box. Below these fields are three buttons: "Register", "Clear", and "Return". An error dialog box is overlaid on the form. The dialog box has a title bar that says "Error" and a close button (X). Inside the dialog, there is a red circular icon with a white "X" and the text "Fill in all required fields." at the bottom right of the dialog is an "OK" button.

Rysunek 36: Test 10: Próba rejestracji konta bez uzupełnionego formularza.