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

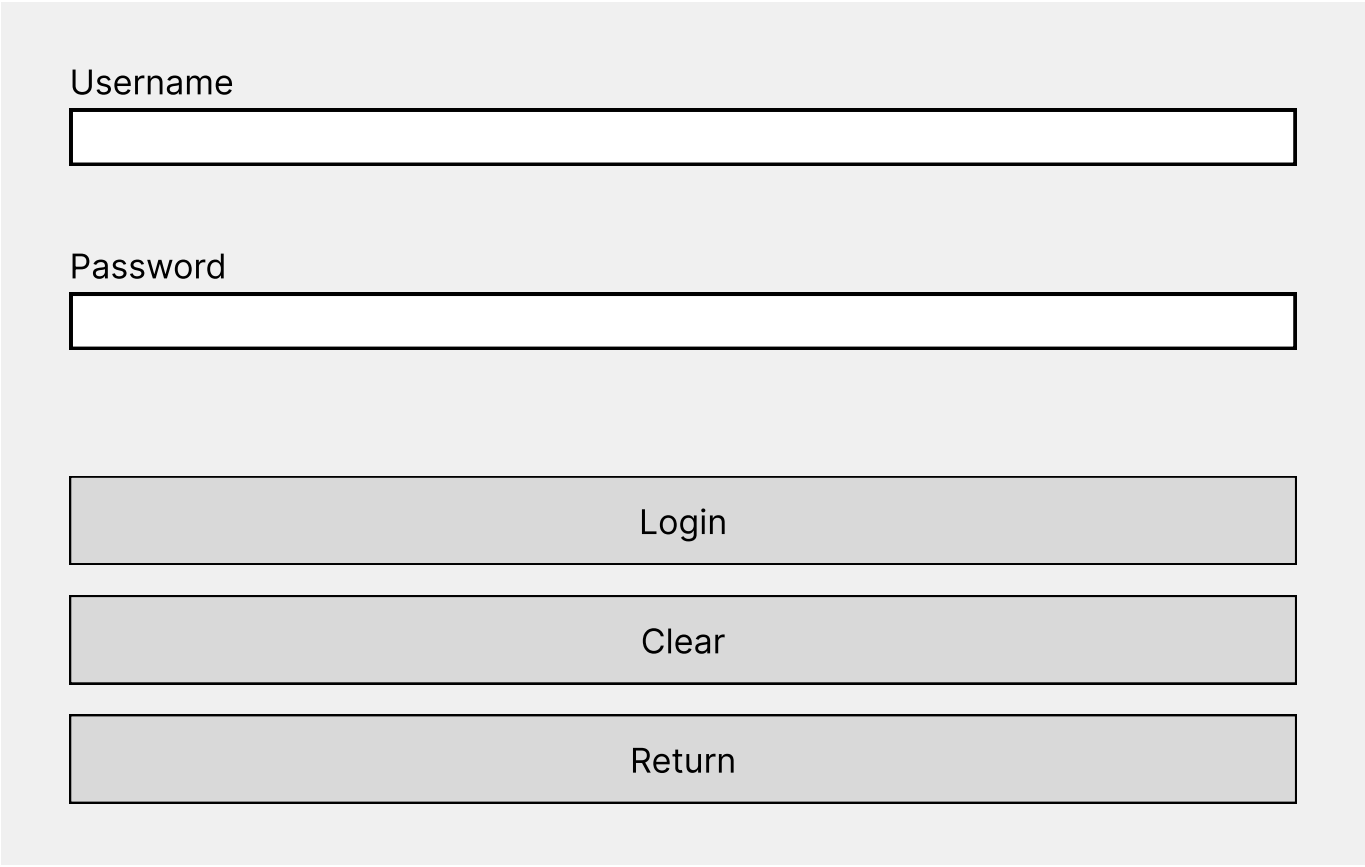
Spis treści

Spis tabel

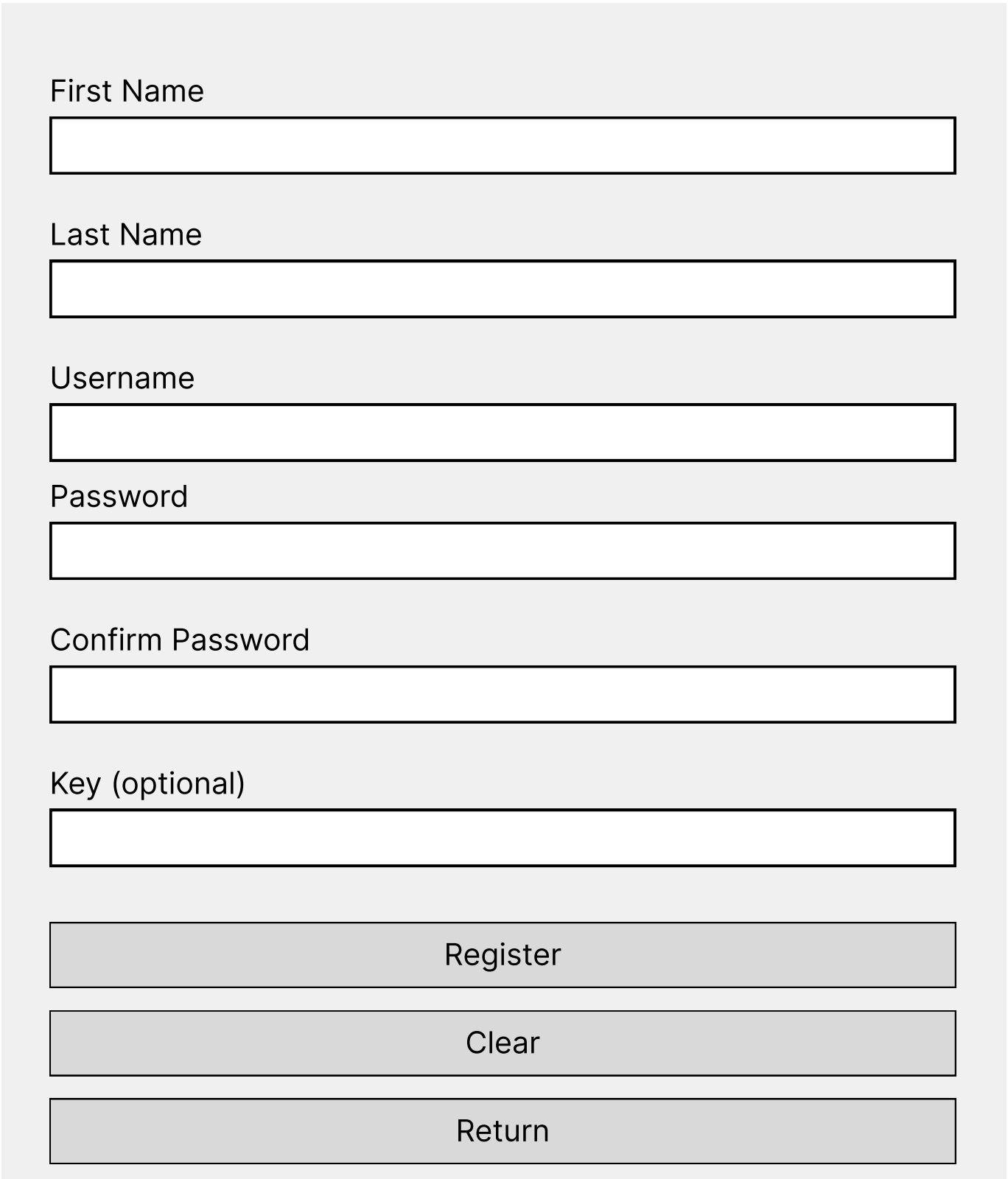
Spis rysunków

1 Makieta interfejsu graficznego

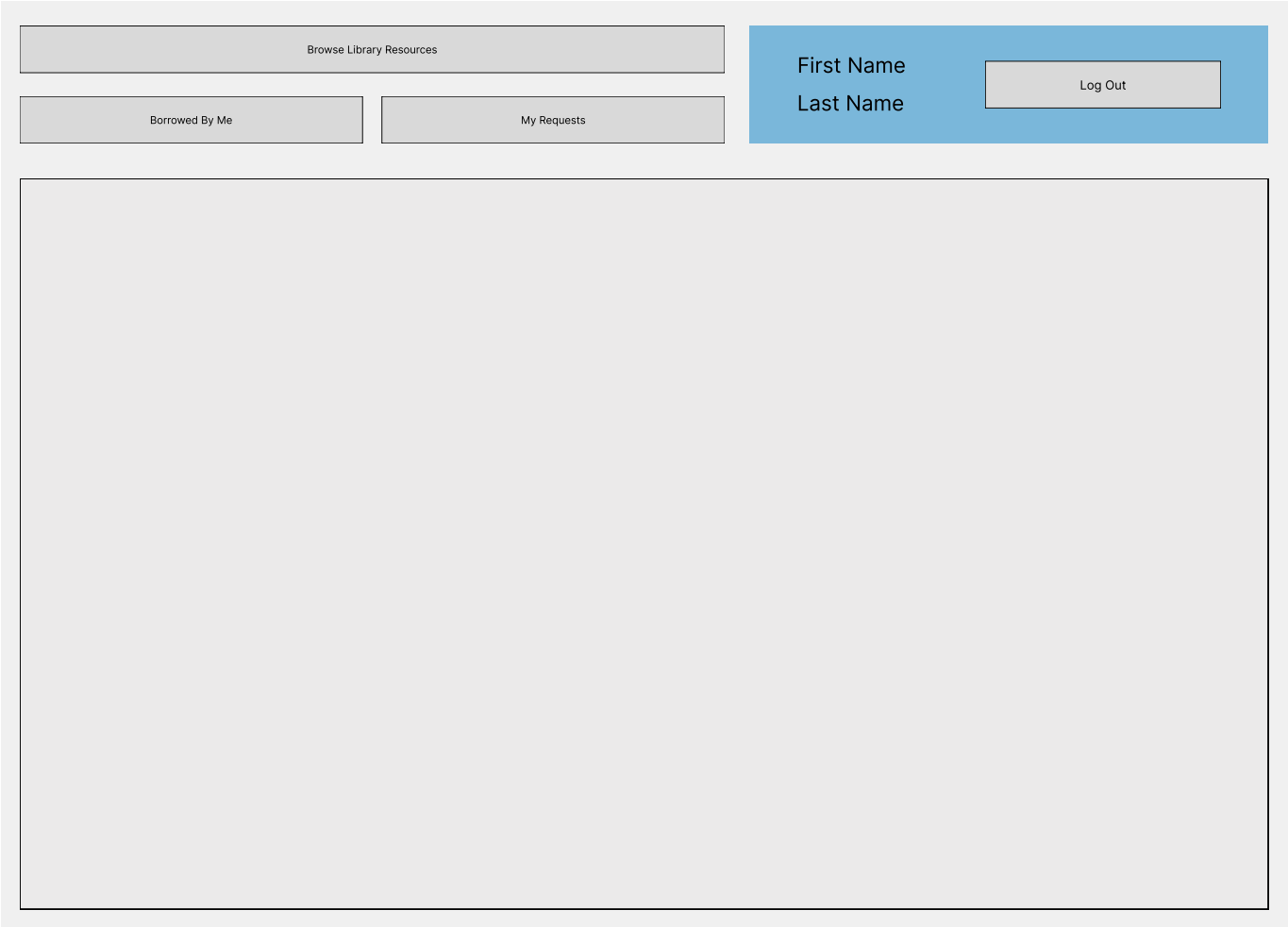
Rysunek 1: Makieta widoku powitalnego.



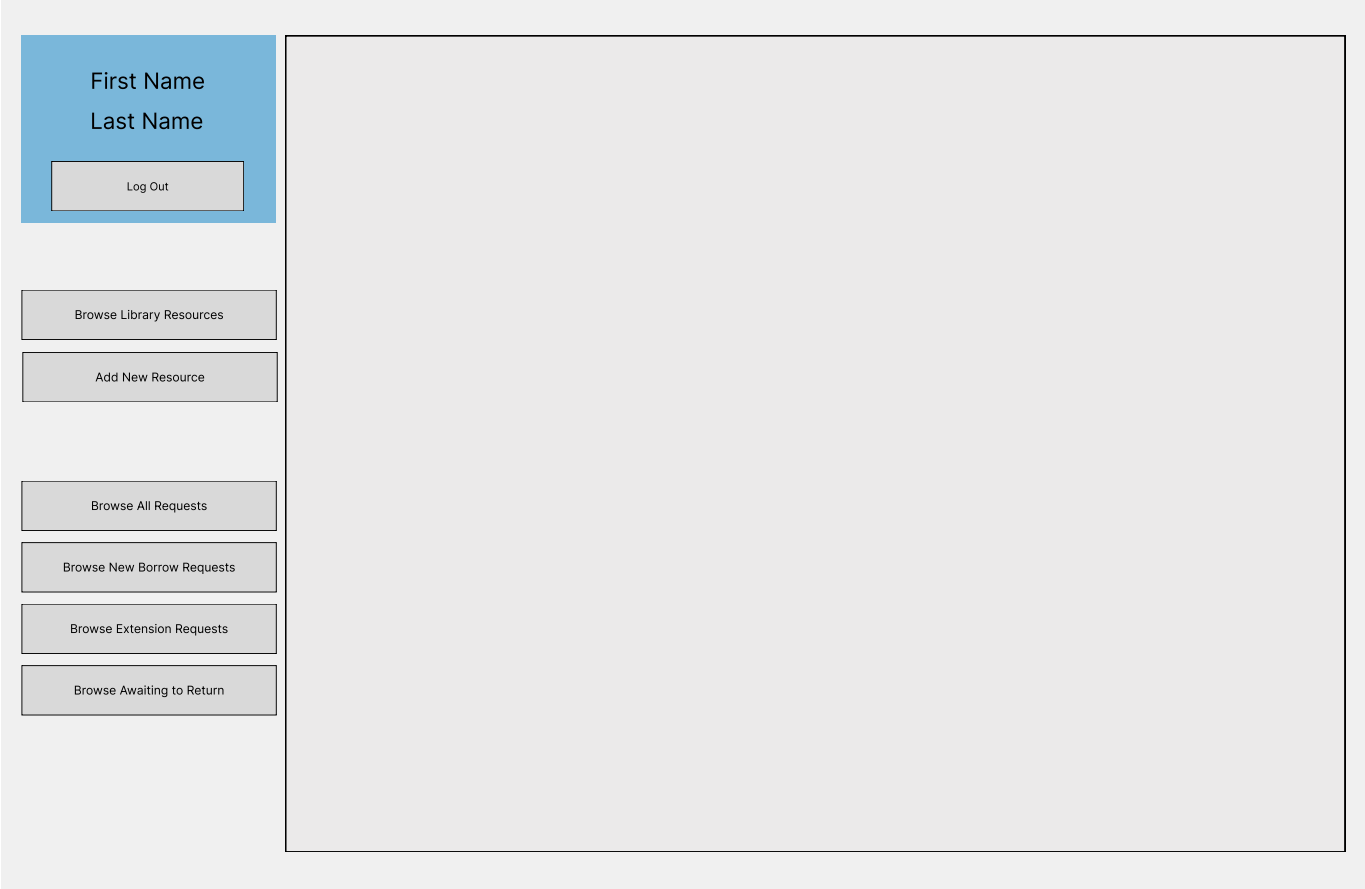
Rysunek 2: Makieta widoku logowania.



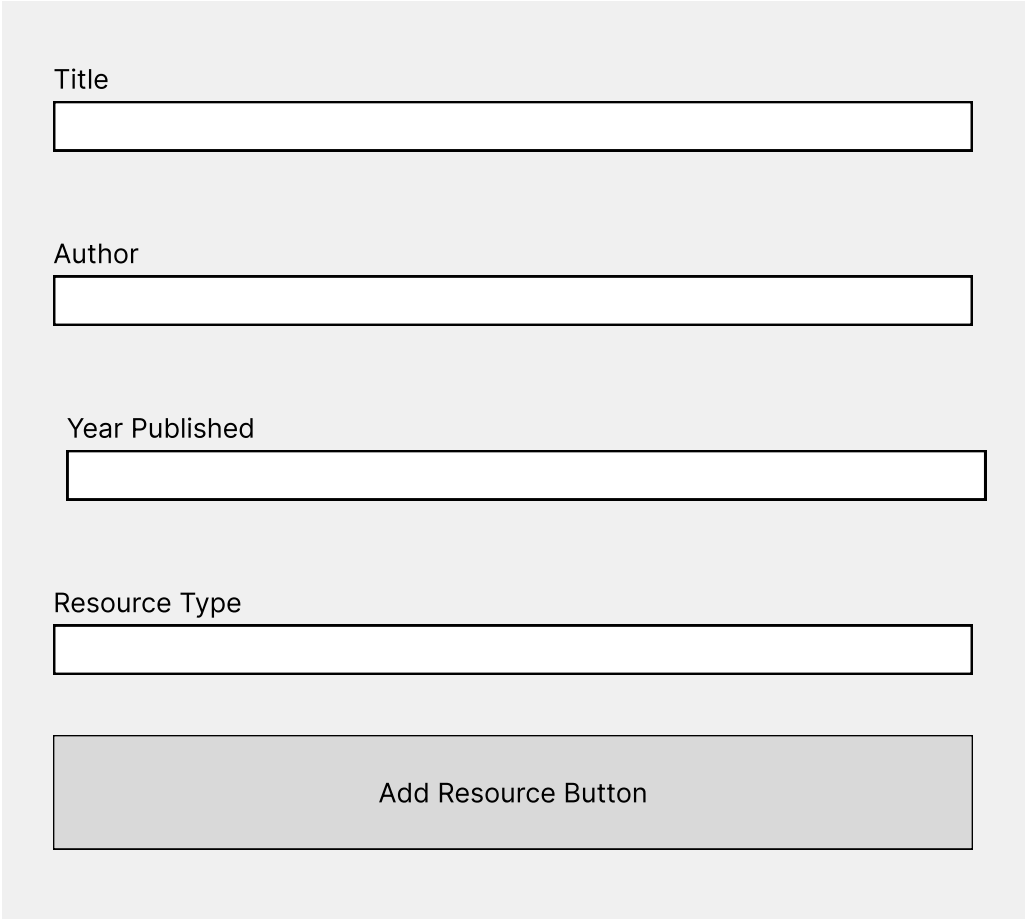
Rysunek 3: Makieta widoku rejestracji.



Rysunek 4: Makieta widoku klienta.

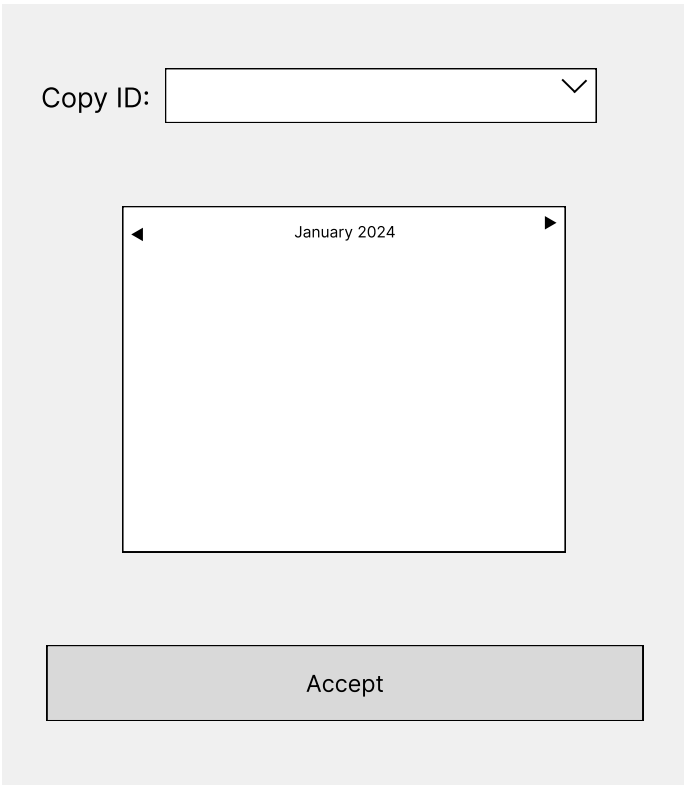


Rysunek 5: Makieta widoku pracownika.

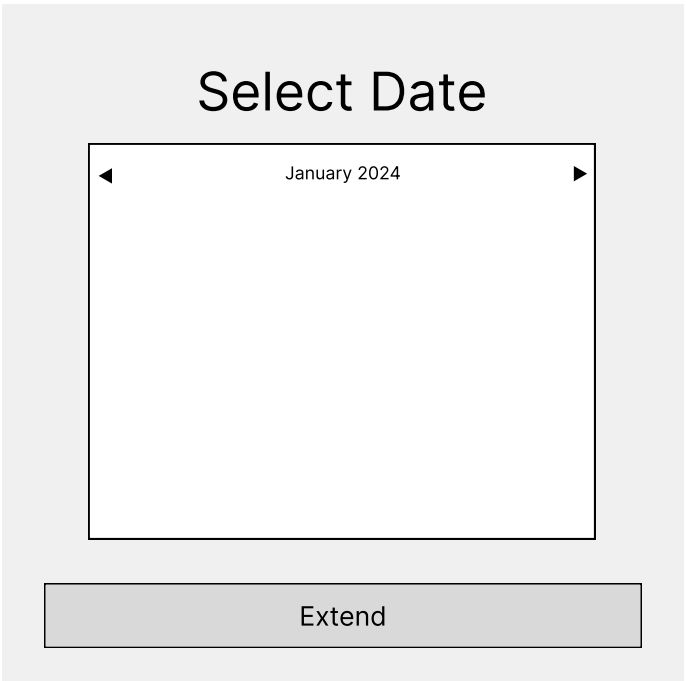


Rysunek 6: Makieta widoku dodawania zasobu.

Rysunek 7: Makieta widoku zarządzania kopiami zasobu.

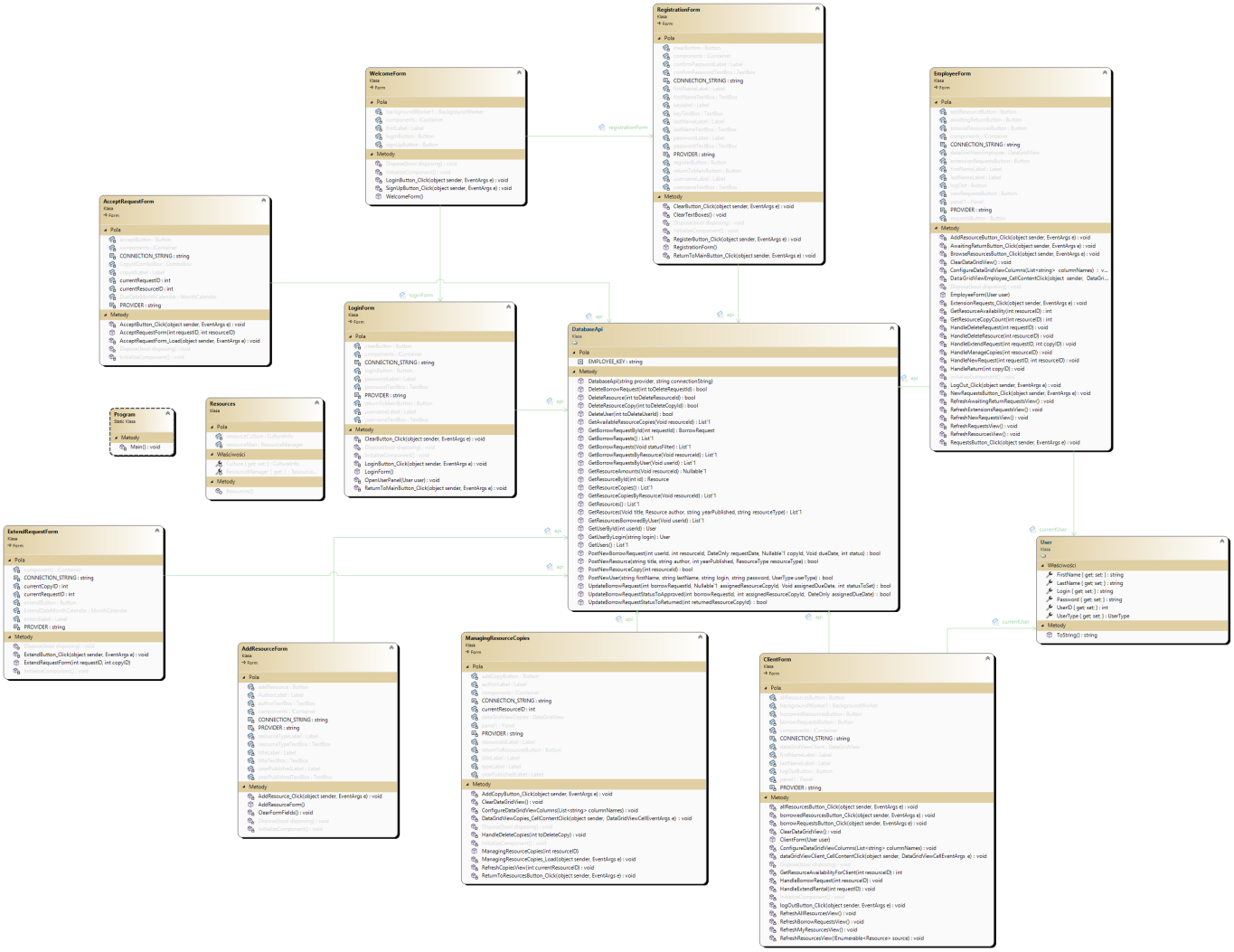


Rysunek 8: Makieta widoku akceptacji prośby o wypożyczenie.

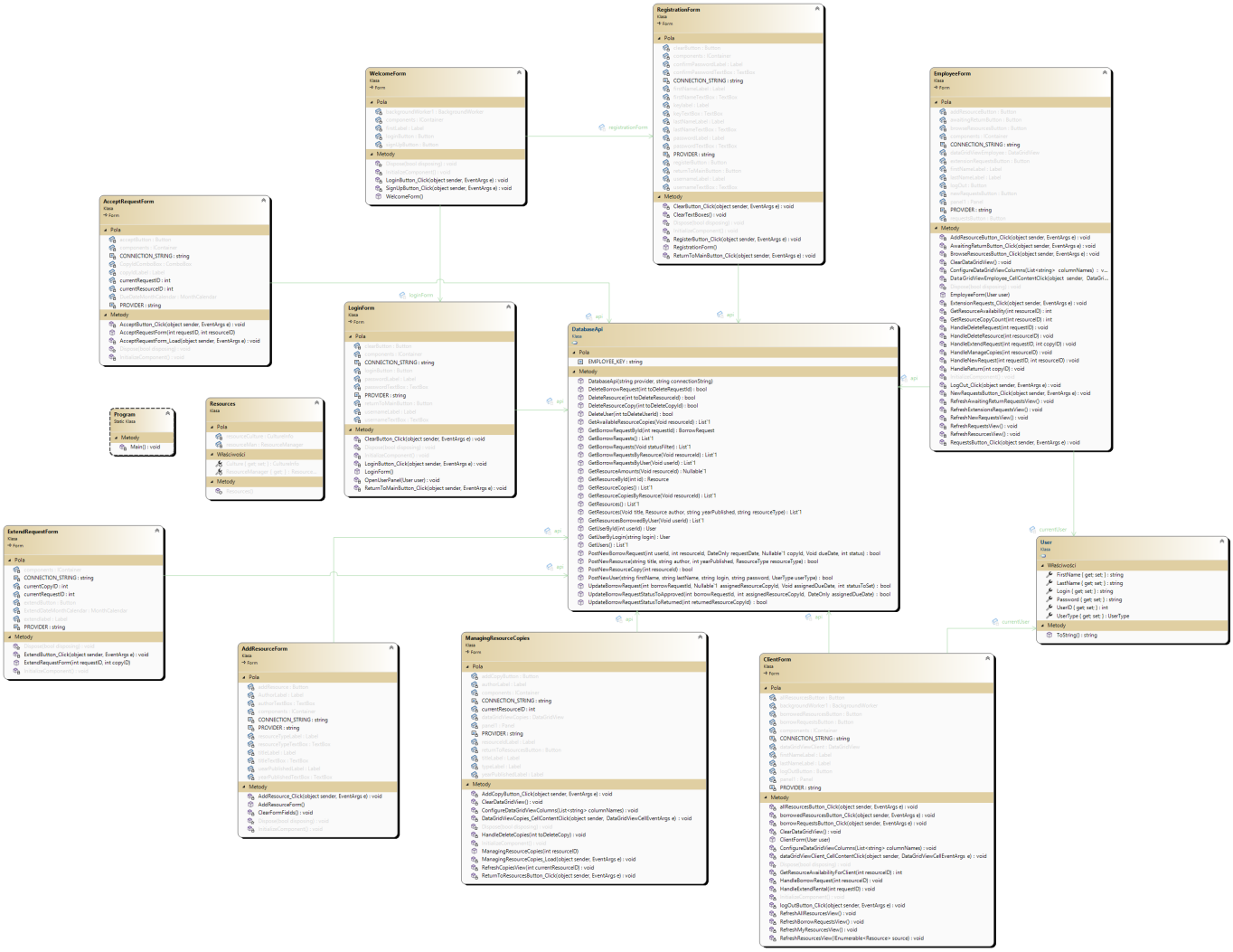


Rysunek 9: MAkieta widoku przedłużenia prośby o wypożyczenie.

2 Diagram klas



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



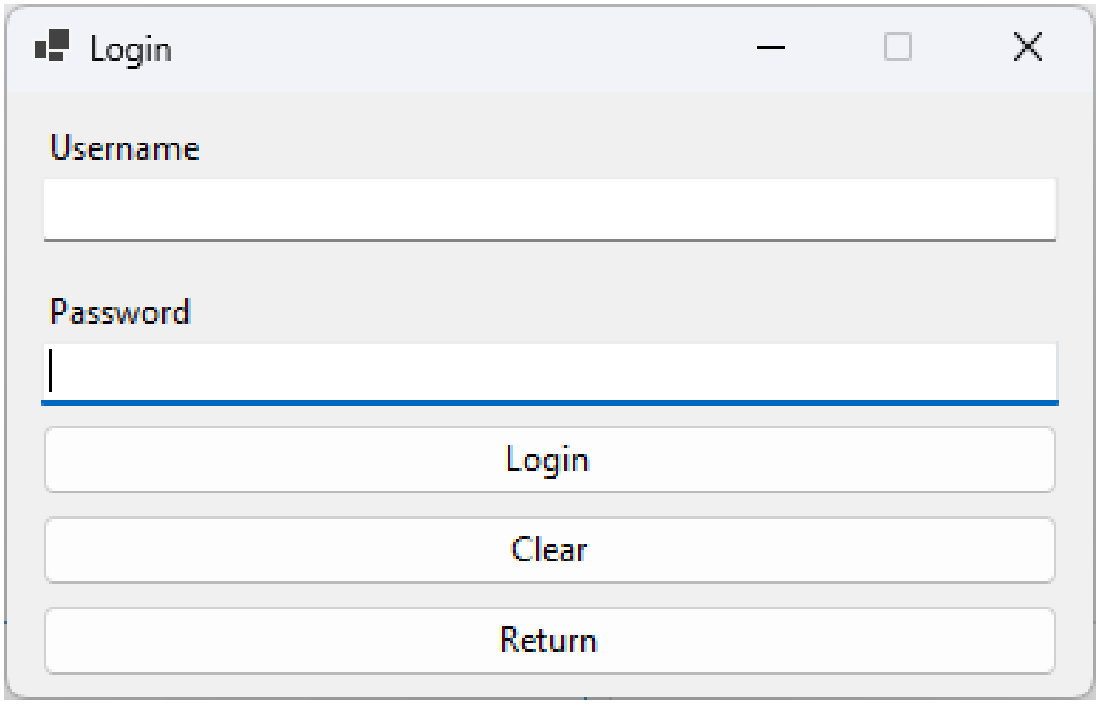
Rysunek 11: Diagram klas dla Back-End'u.

3 Wdrożenie aplikacji

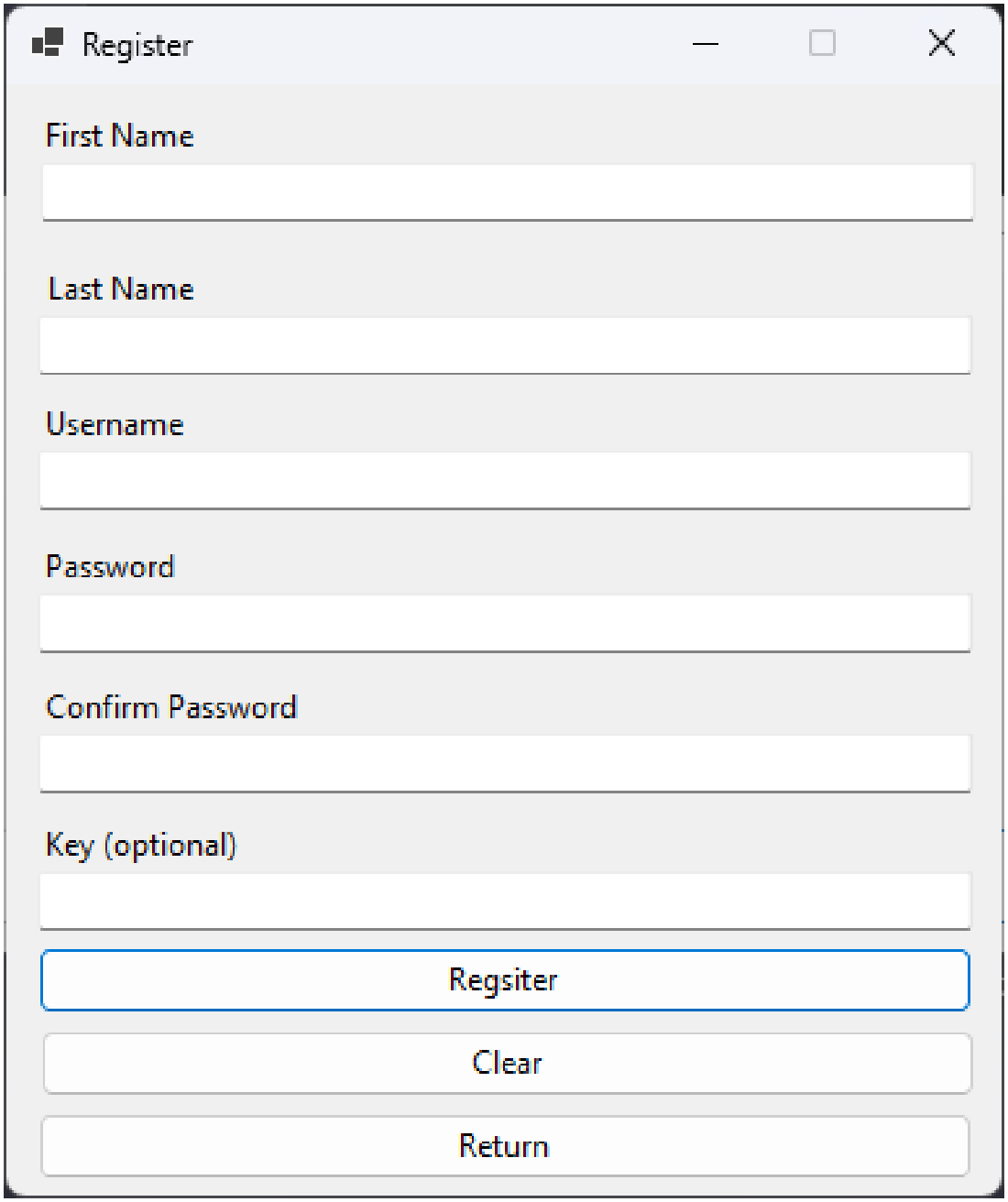
//screeny

3.1 Okna startowe

Rysunek 12: Okno powitalne.

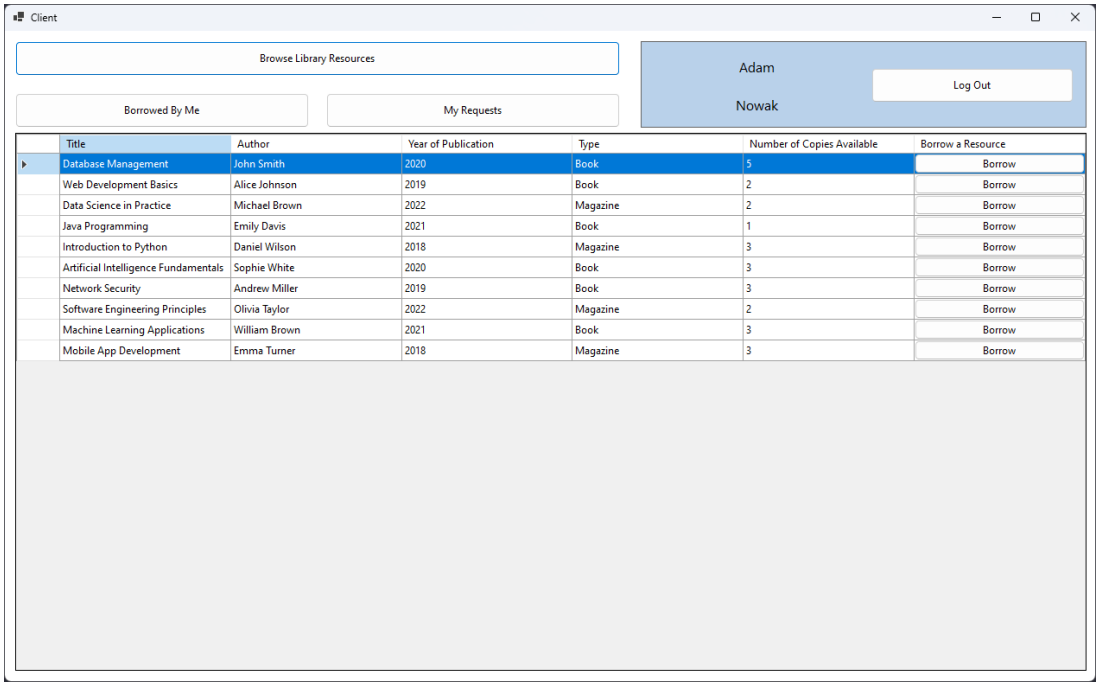


Rysunek 13: Okno logowania.

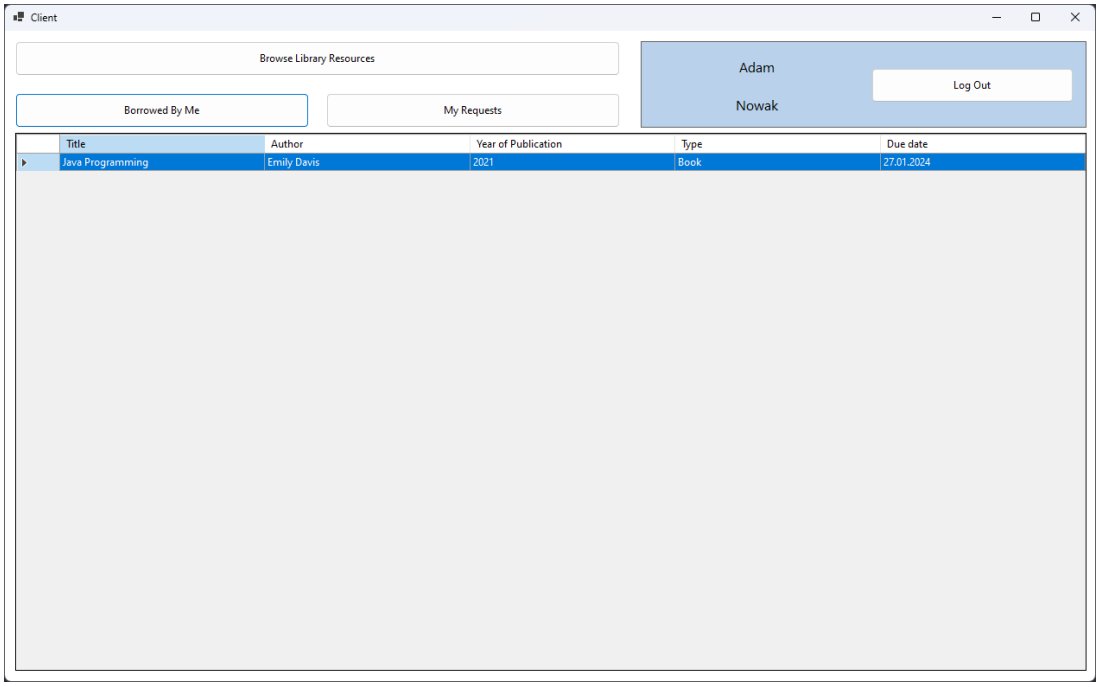


Rysunek 14: Okno rejestracji.

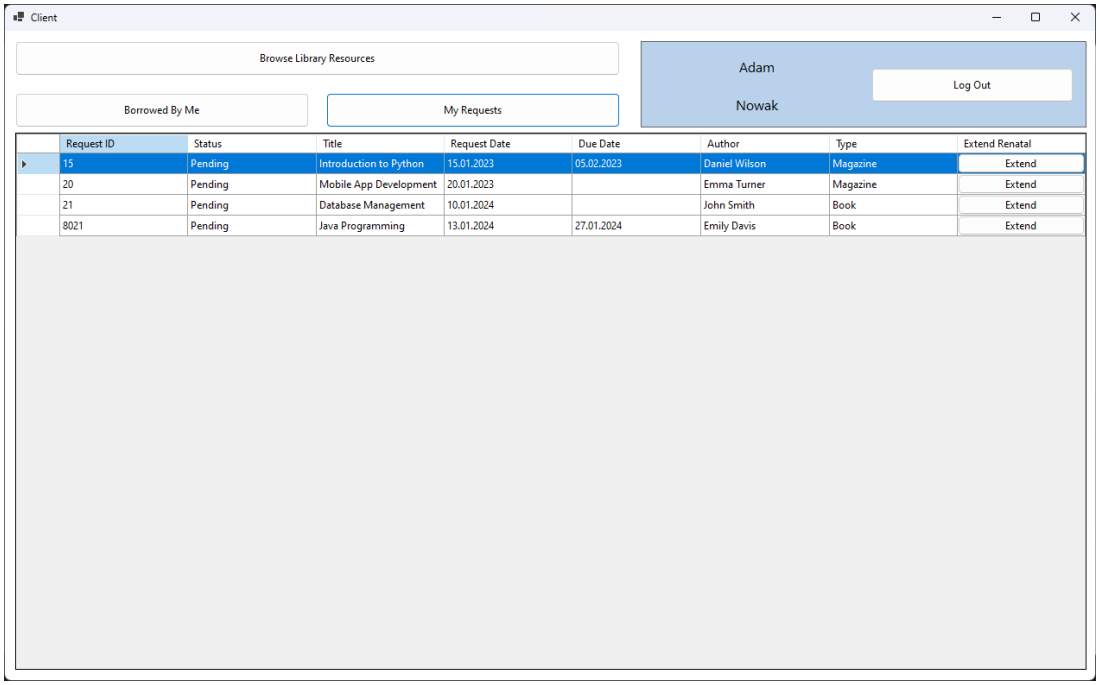
3.2 Okna klienta



Rysunek 15: Okno przeglądania zbioru biblioteki.

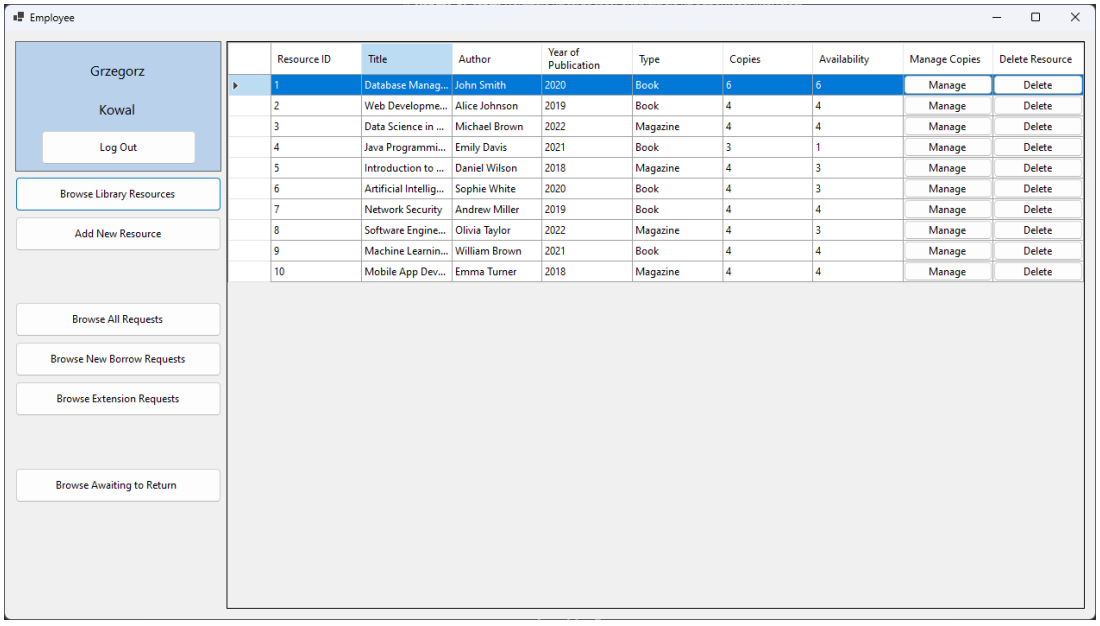


Rysunek 16: Okno przeglądania pożyczonych pozycji.

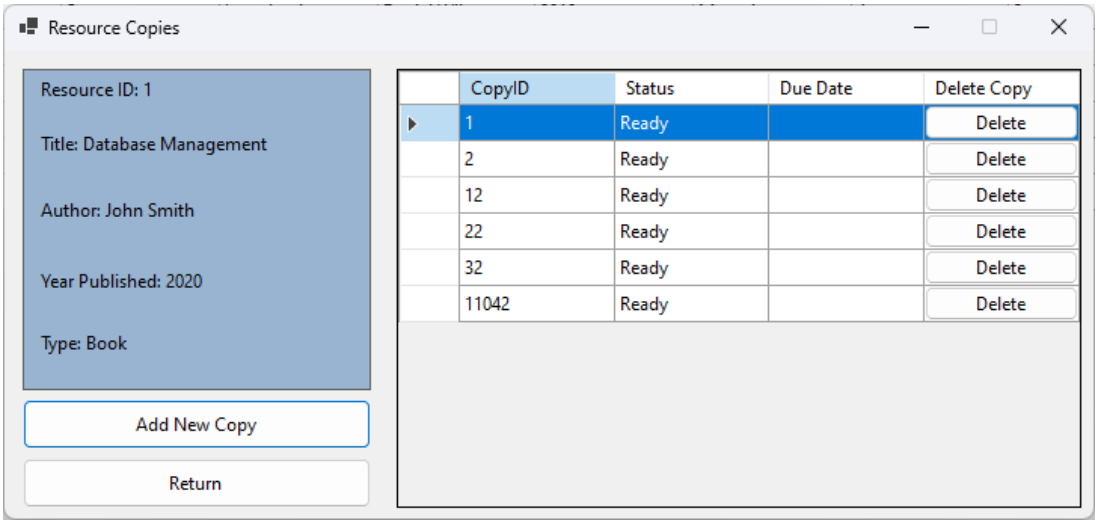


Rysunek 17: Okno przeglądania próśb o wypożyczenie.

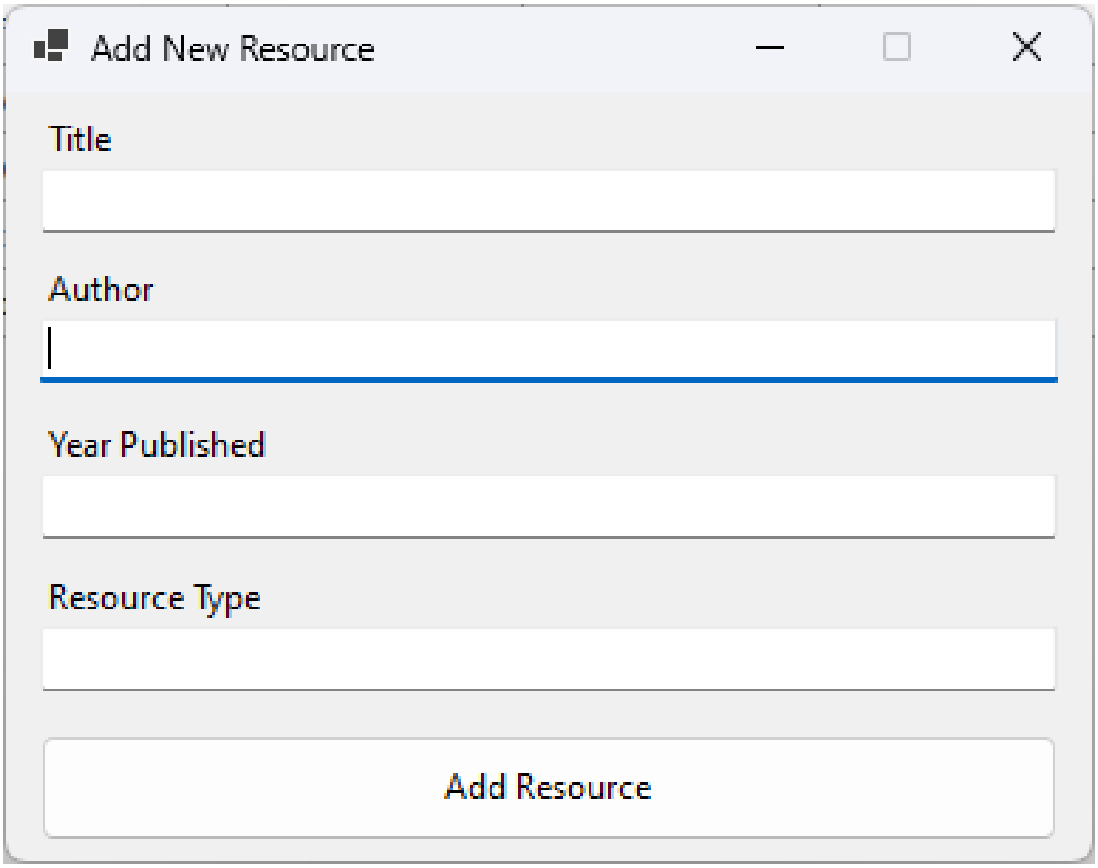
3.3 Okna pracownika



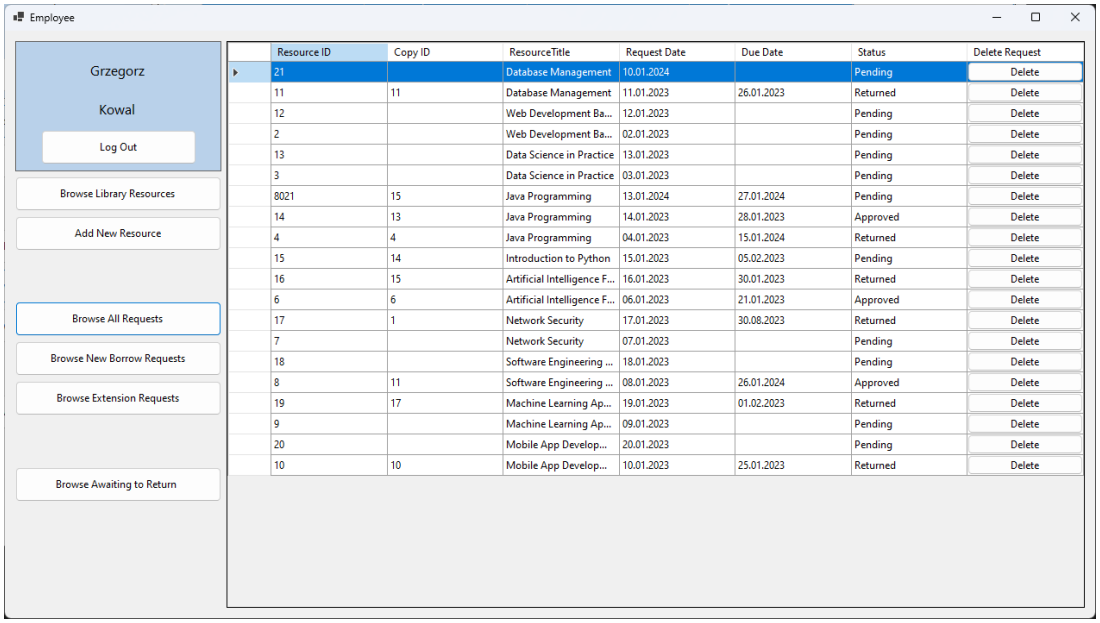
Rysunek 18: Okno przeglądania zasobów biblioteki.



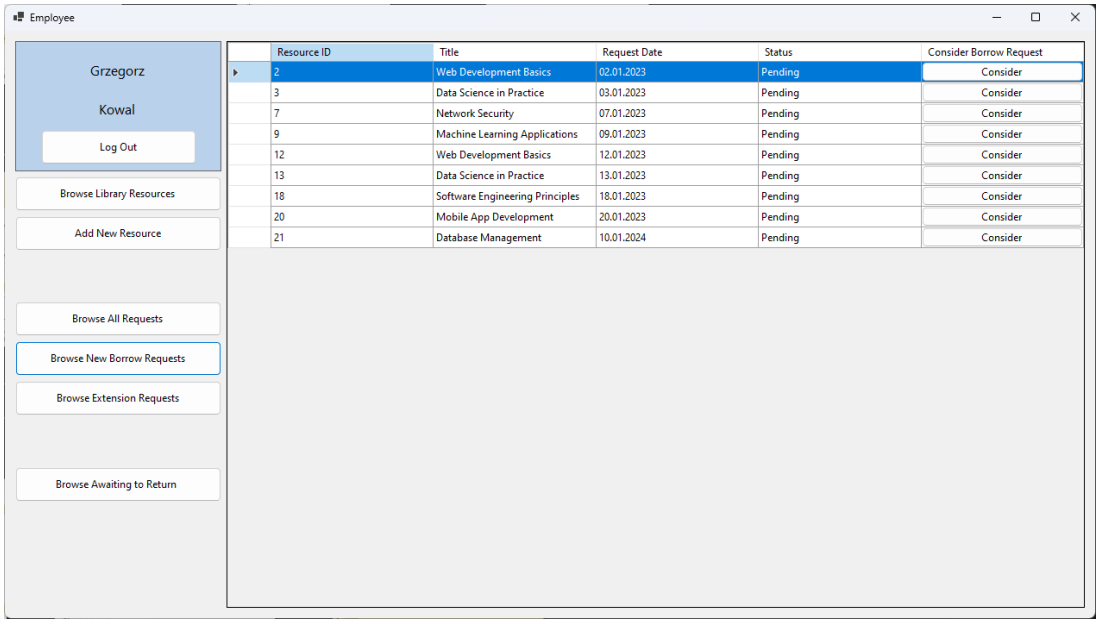
Rysunek 19: Okno zarządzania egzemplarzami zasobu.



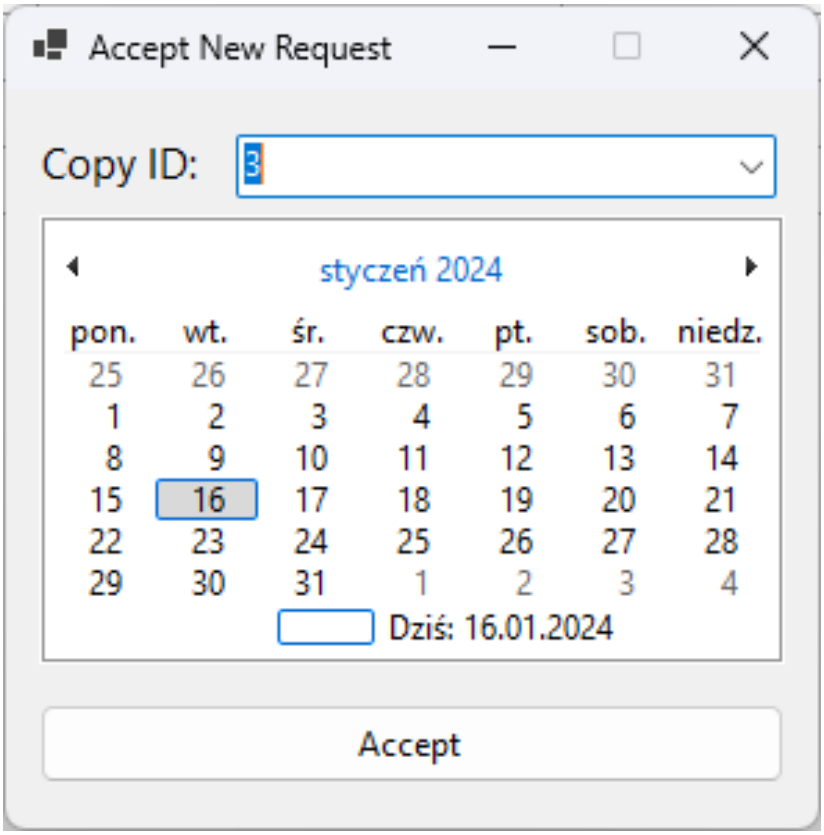
Rysunek 20: Okno dodawania nowego zasobu.



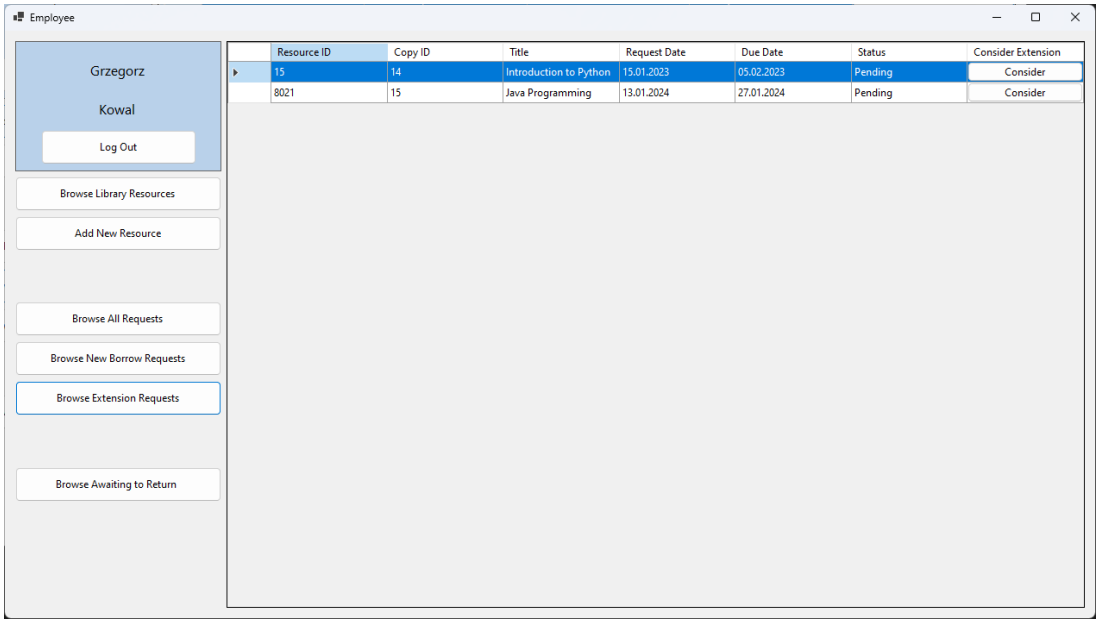
Rysunek 21: Okno przeglądania wszystkich próśb o wypożyczenie.



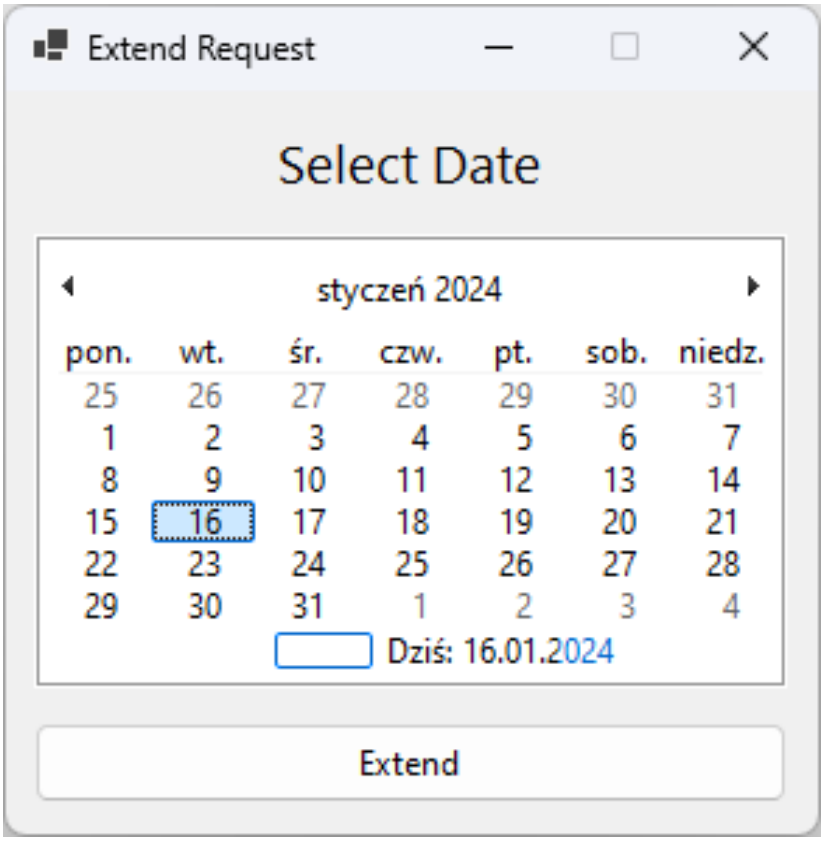
Rysunek 22: Okno przeglądania nowych próśb o wypożyczenie.



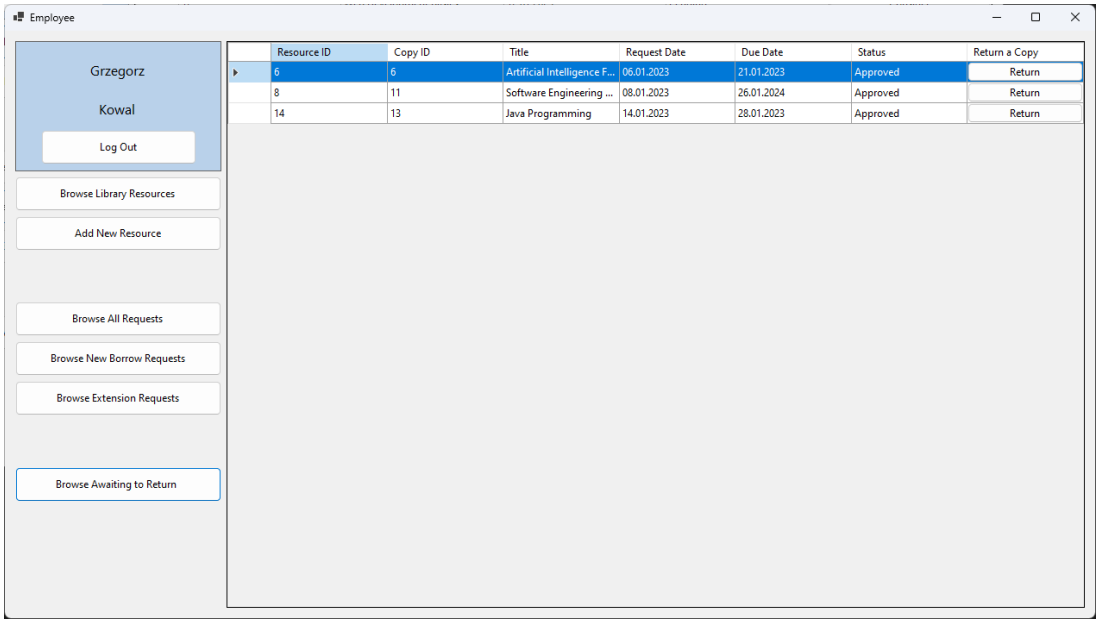
Rysunek 23: Okno akceptacji prośby o wypożyczenie.



Rysunek 24: Okno przeglądania próśb o przedłużenie wypożyczenia.



Rysunek 25: Okno akceptacji prośby o przedłużenie wypożyczenia.



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));
    }
```

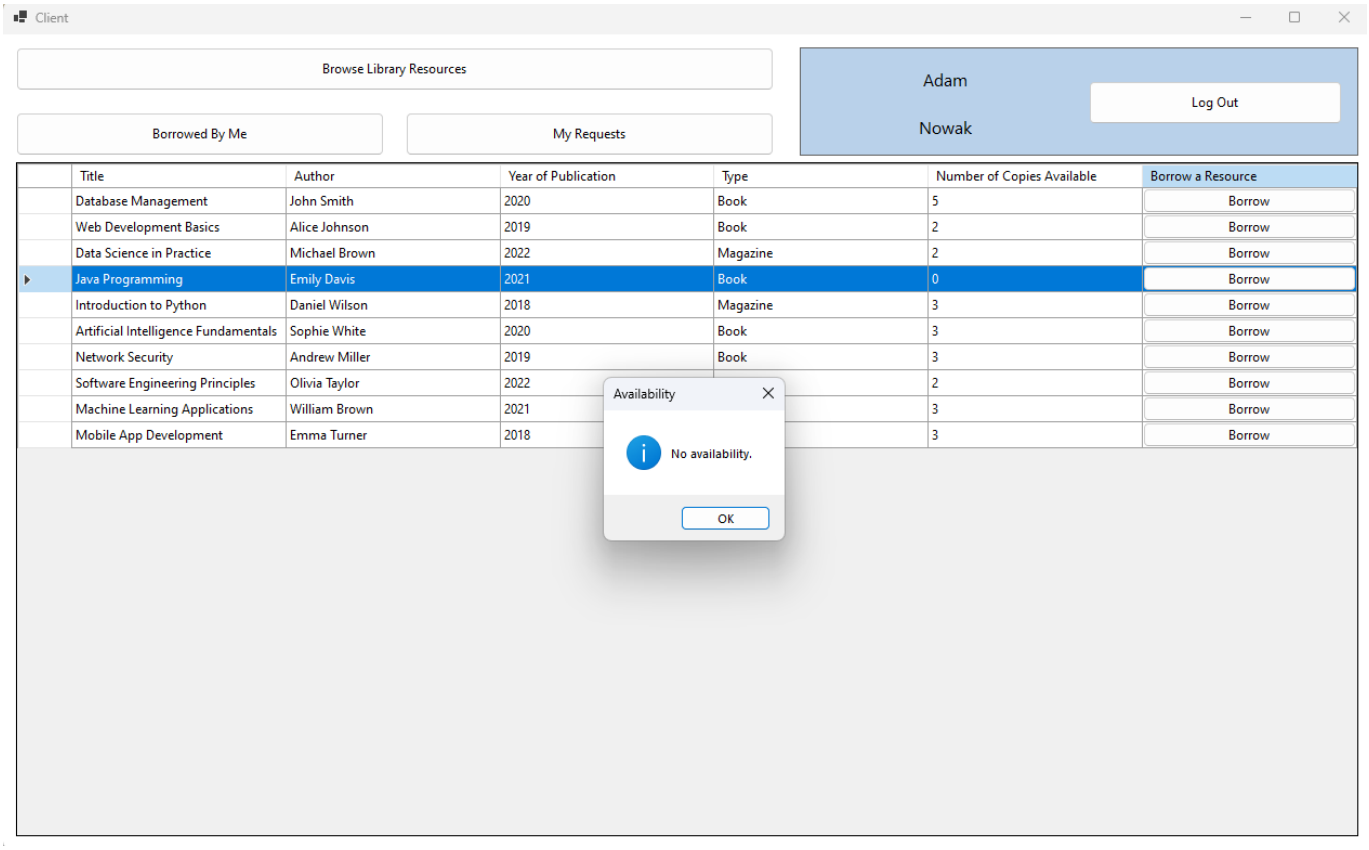
```
    var updatedBorrowedReq = api.GetBorrowRequests();
    Assert.NotNull(updatedBorrowedReq);
    Assert.True(updatedBorrowedReq.TrueForAll(
        ubr => ubr.Status == Status.Returned || ubr.Status == Status.Pending));

    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 &&
```

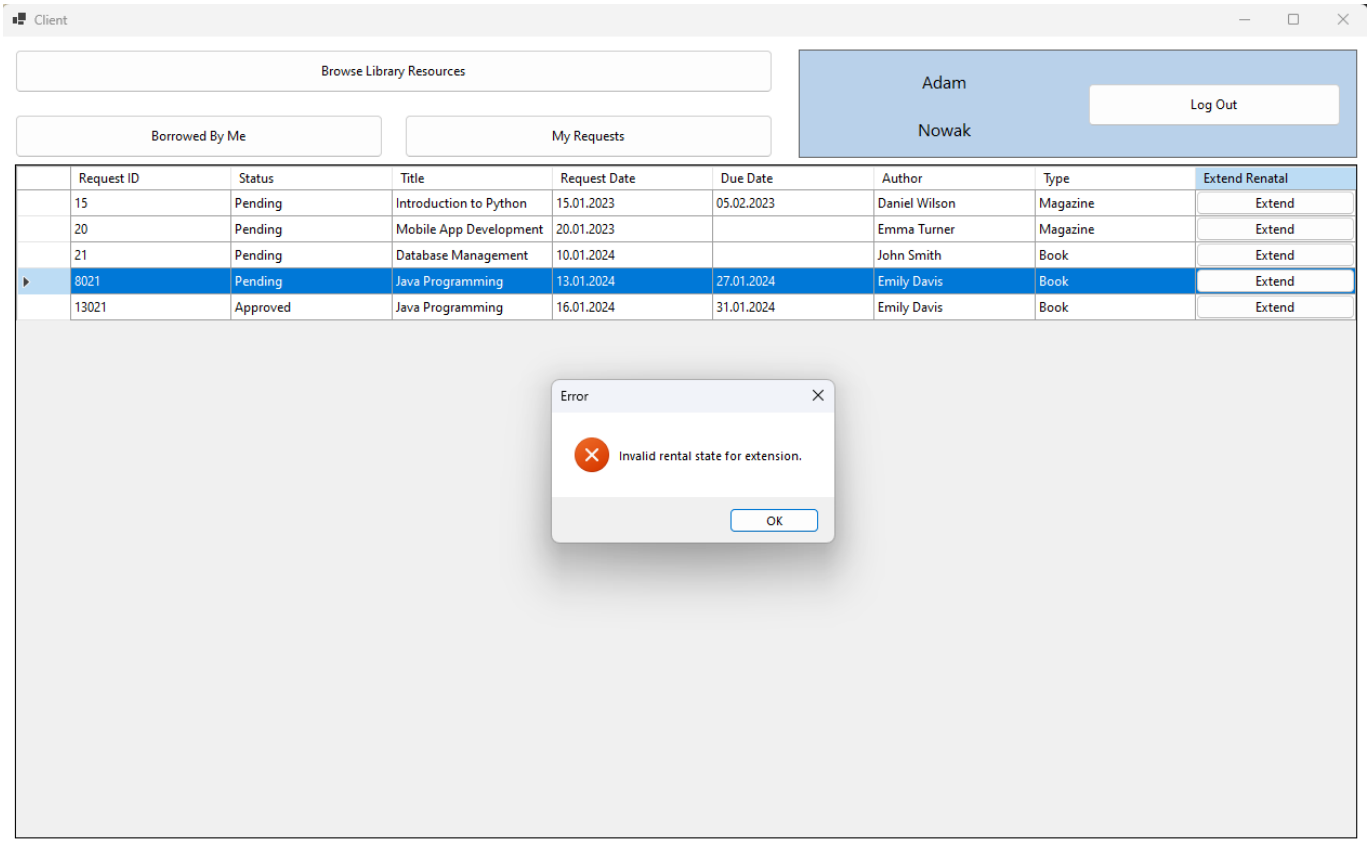
```
    res[0].ResourceID == e.ResourceID &&
    e.RequestDate == new DateOnly(1900, 1, 1) &&
    e.DueDate == null &&
    e.CopyID == null &&
    e.Status == Status.Pending).Single();

    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());
    }
}
```

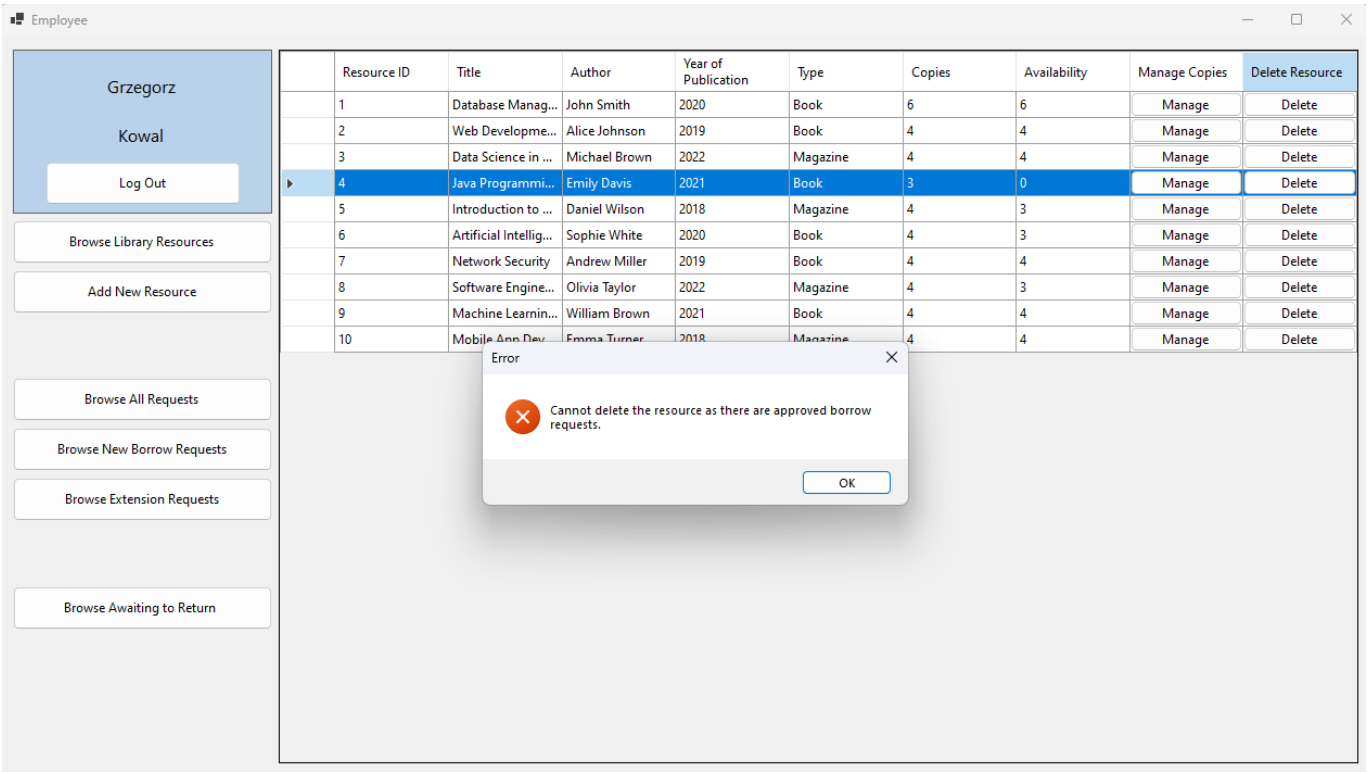
4.2 Testy GUI



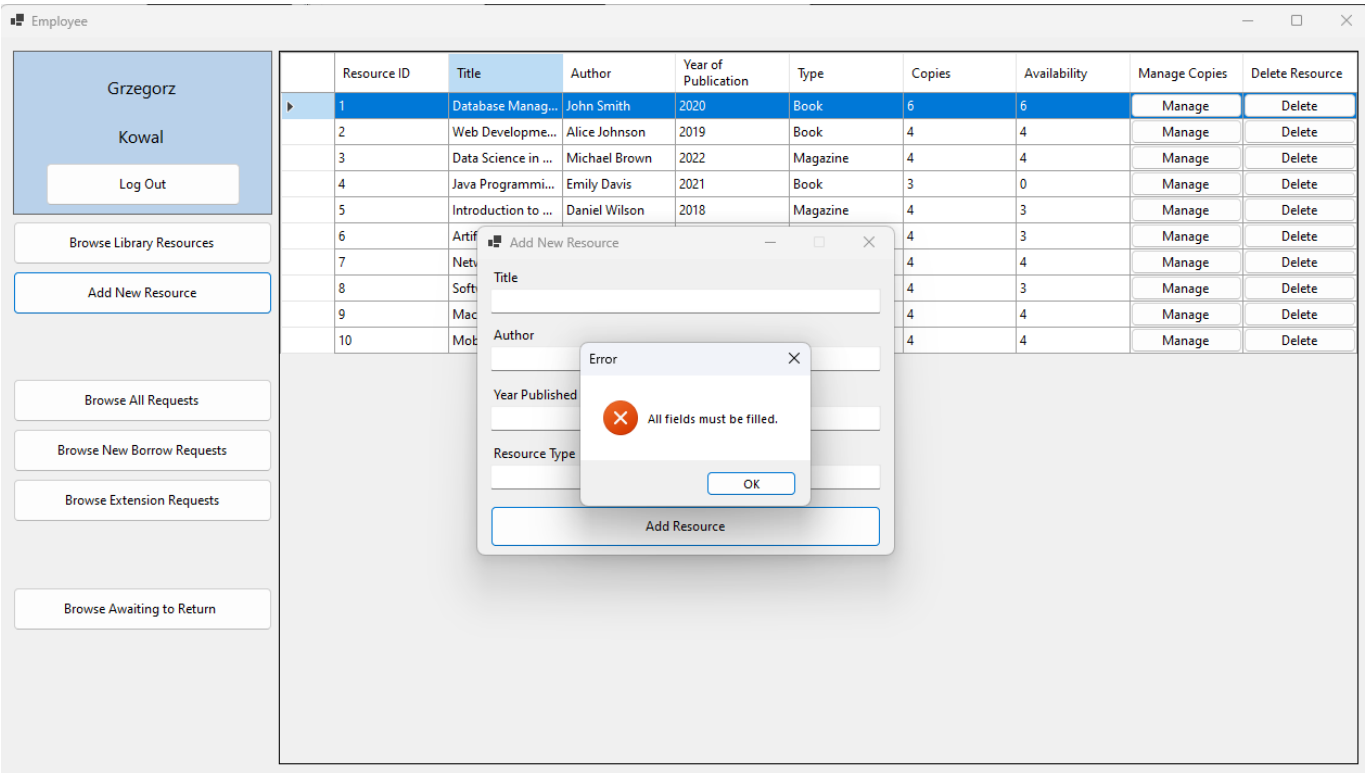
Rysunek 27: Test 1: Próba wypożyczenia zasobu, dla którego nie ma dostępnych kopii



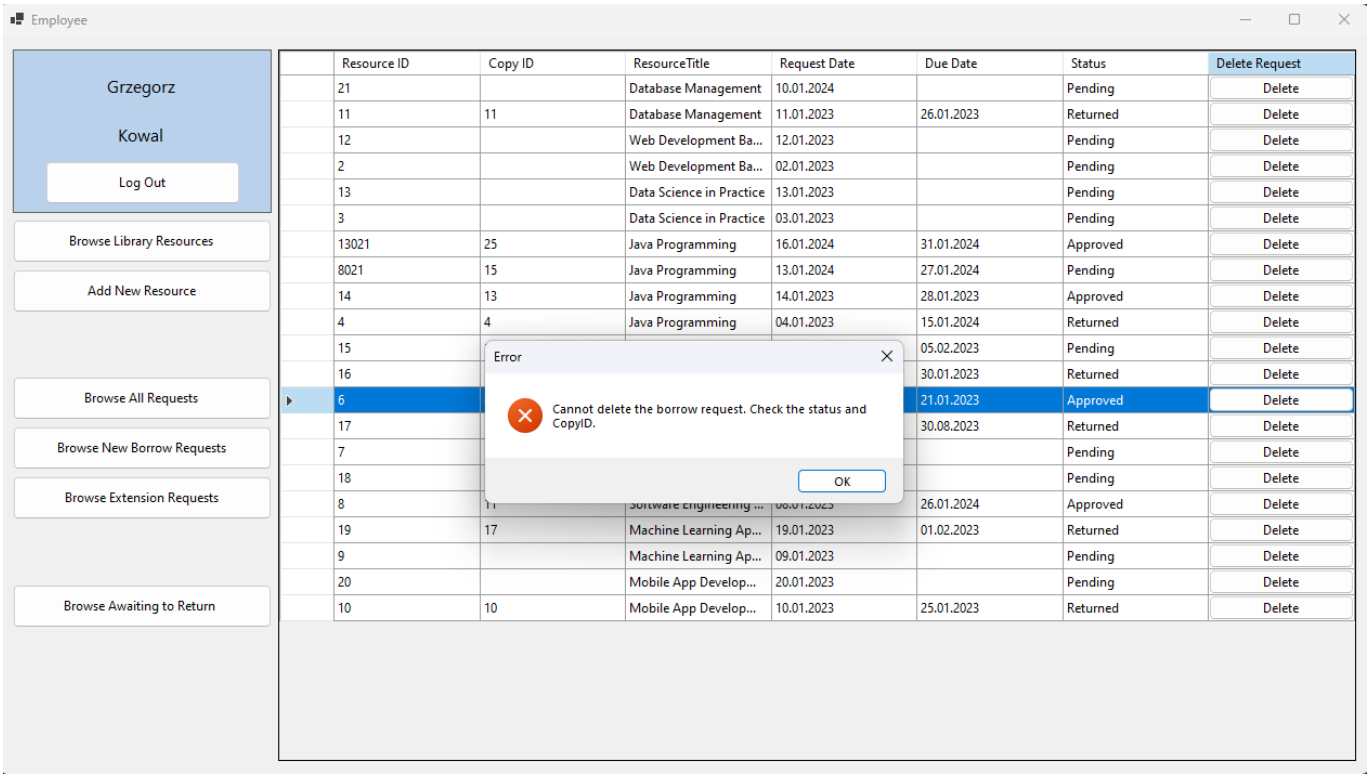
Rysunek 28: Test 2: Próba przedłużenia wypożyczenia egzemplarza, który nie jest wypożyczony przez użytkownika



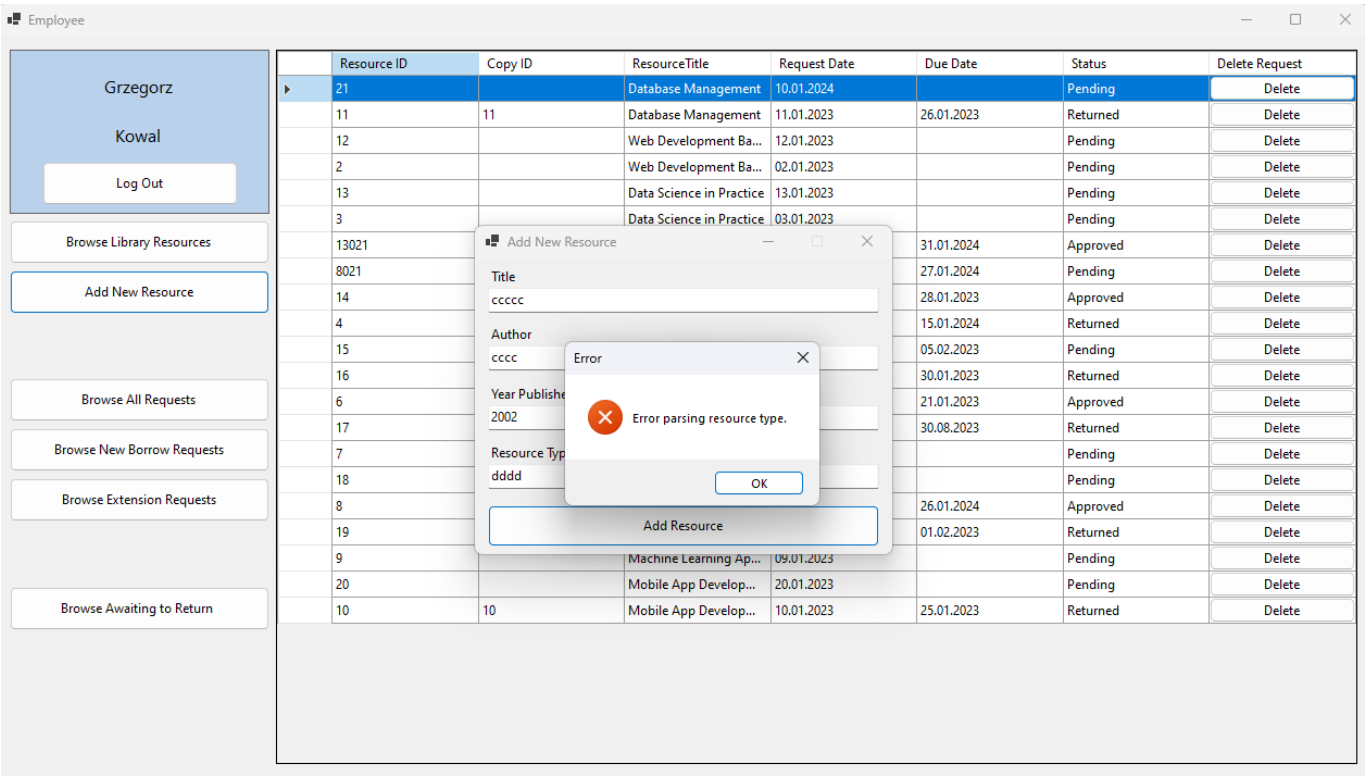
Rysunek 29: Test 3: Próba usunięcia zasobu, którego egzemplarz jest aktualnie wypożyczony klientowi



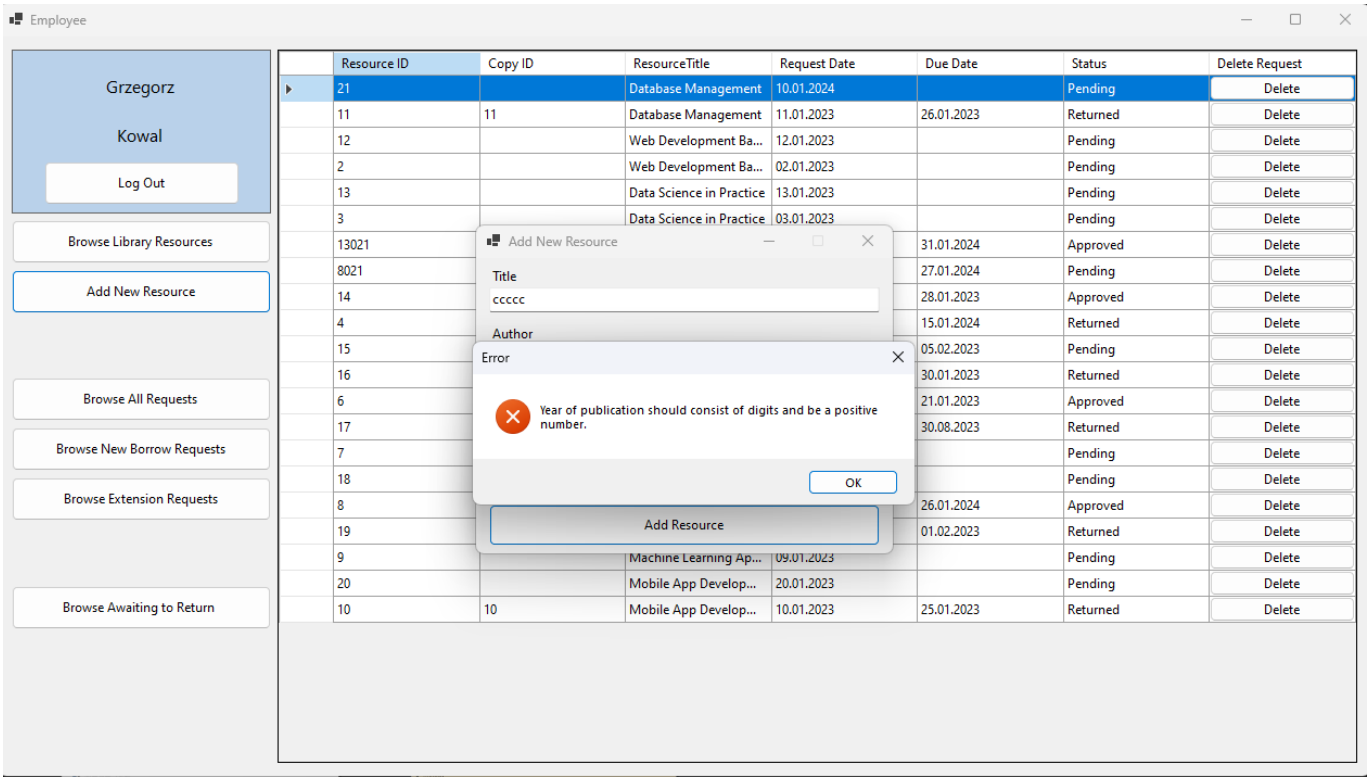
Rysunek 30: Test 4: Próba dodania zasobu bez uzupełnienia pól formularza



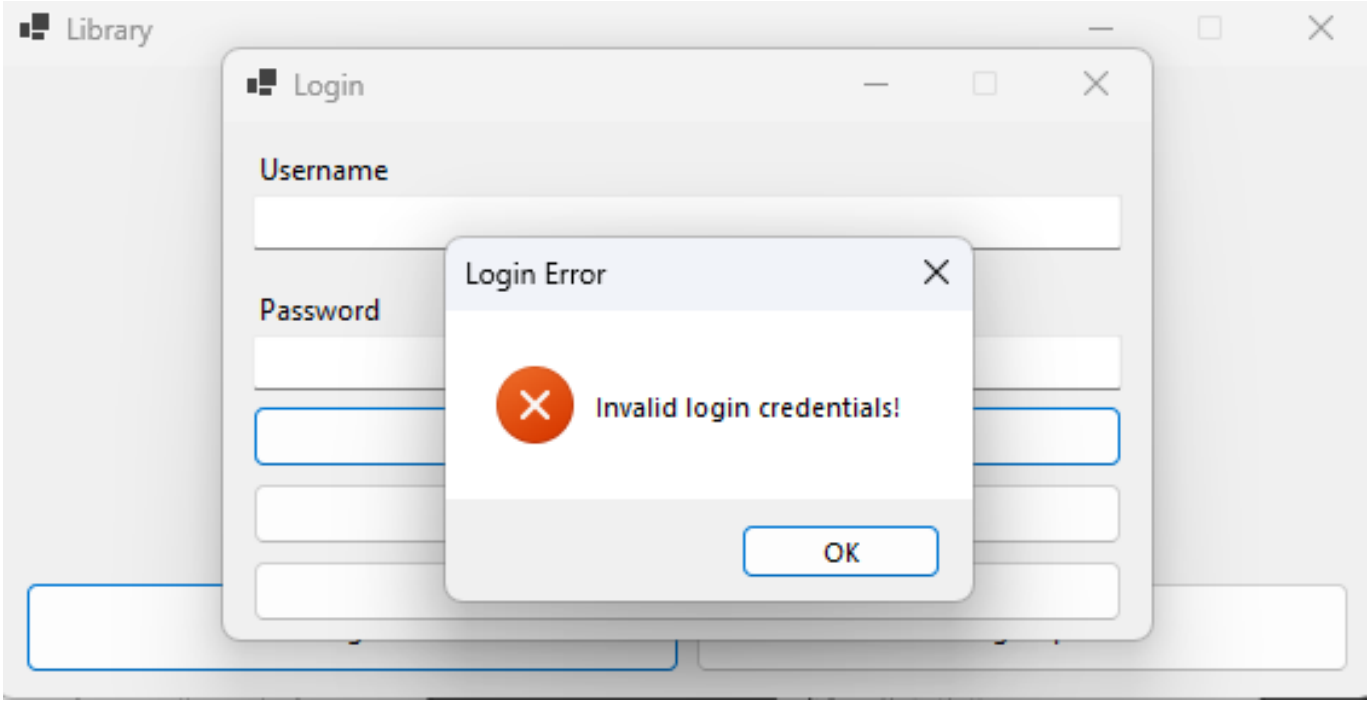
Rysunek 31: Test 5: Próba usunięcie prośby o wypożyczenie dla niezwróconej kopii.



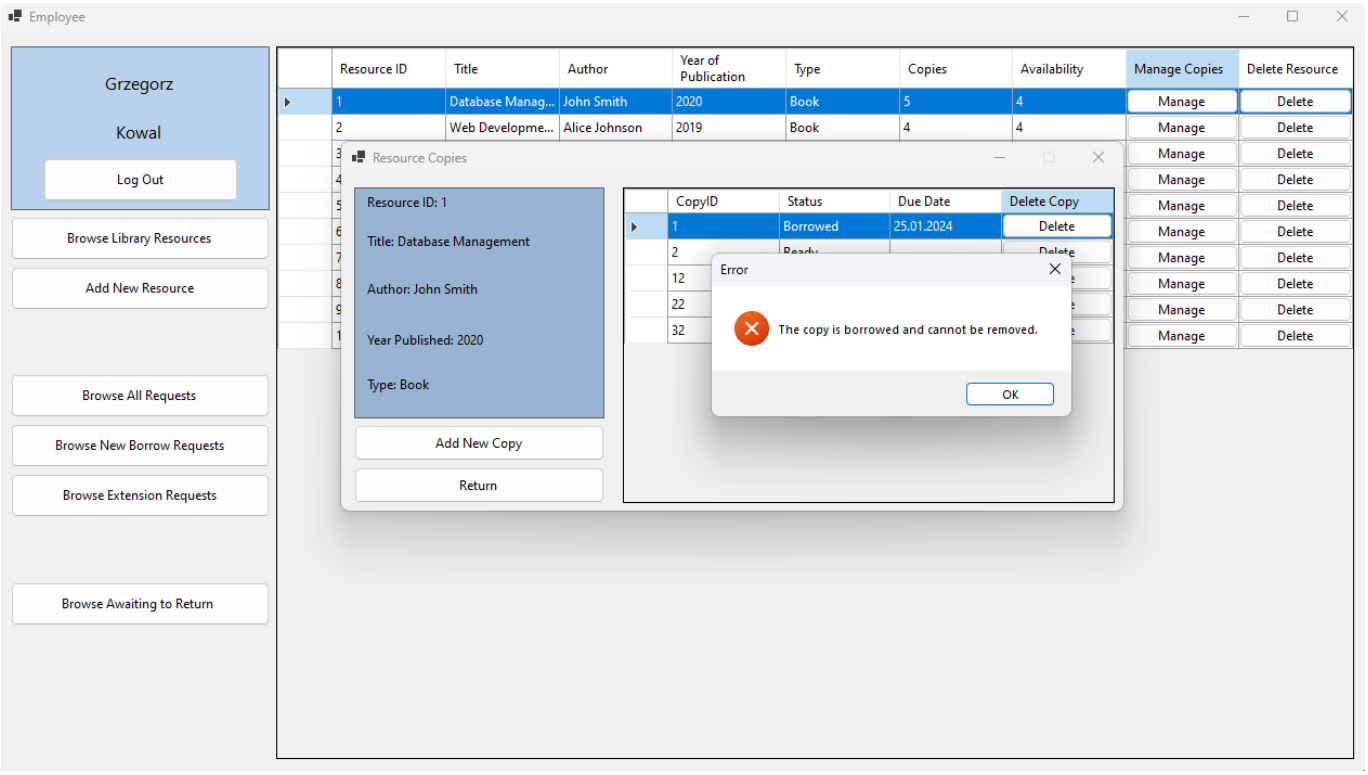
Rysunek 32: Test 6: Próba dodanie zasobu o nieznanym typie.



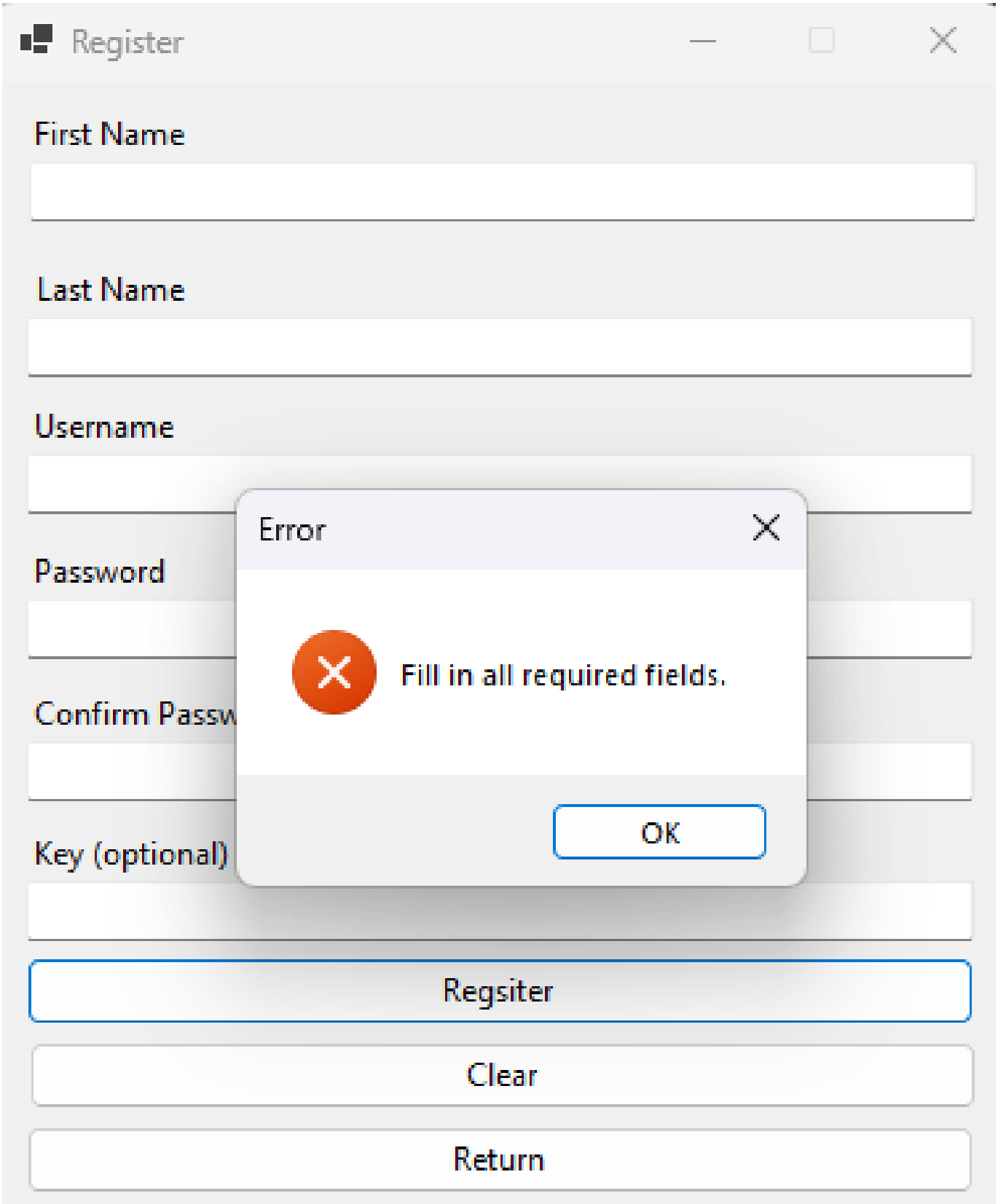
Rysunek 33: Test 7: Próba dodanie zasobu o niepoprawnej dacie wydania.



Rysunek 34: Test 8: Próba zalogowania bez podania danych



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



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