

**Wydział Nauk   
Informatyczno-Technologicznych**

**Projekt aplikacji**

Temat: Aplikacja Bankowa

Prowadzący: Przemysław Grabowski

Wykonujący projekt:

Adrian Kozłowski, Dawid Kucisz, Piotr Gałązka, Artur Lubiński  
Studia Stacjonarne I stopnia  
Kierunek: Informatyka  
Semestr: IV, grupa I  
Data: 04.05.2025

Spis treści

[Diagramy 3](#_Toc197269546)

[1. Diagram Czynności 3](#_Toc197269547)

[2. Diagram Przypadków użycia 3](#_Toc197269548)

[3. Diagram Klas 3](#_Toc197269549)

[4. Diagram Sekwencji 3](#_Toc197269550)

[Funkcje Programu 4](#_Toc197269551)

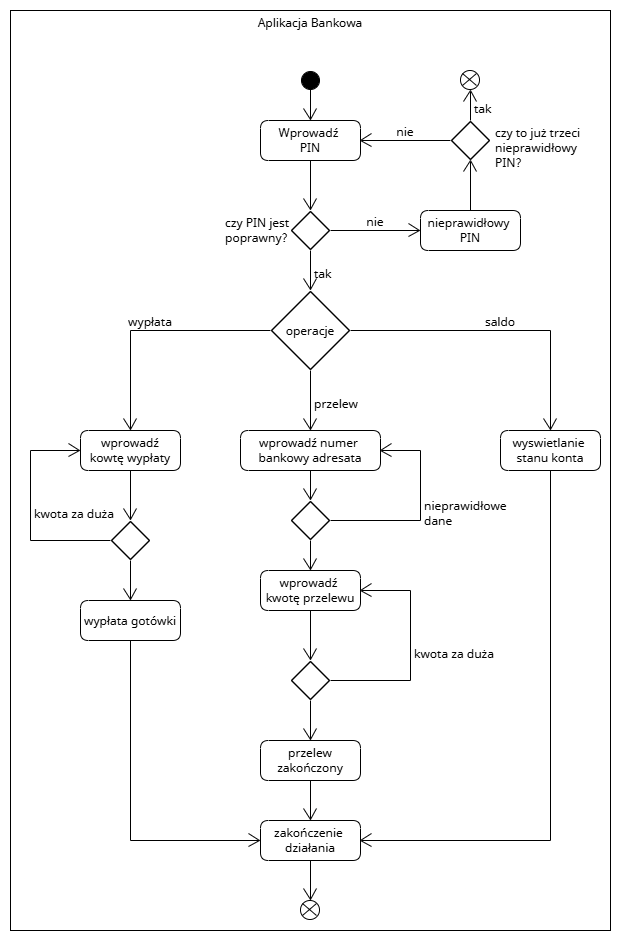
[Testy 5](#_Toc197269552)

[Wyniki testów 6](#_Toc197269553)

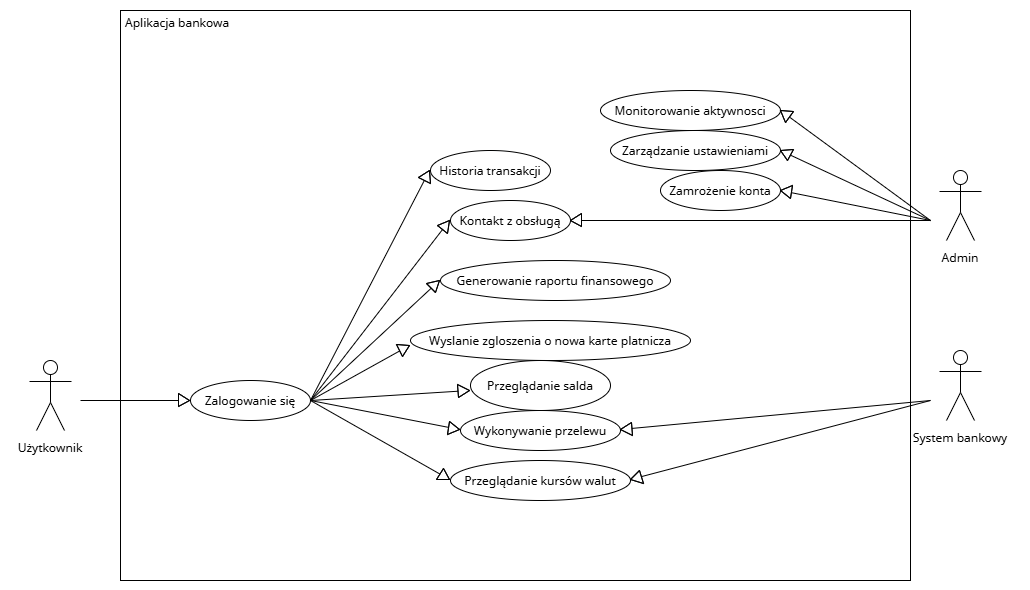
[Podział Obowiązków 6](#_Toc197269554)

# Diagramy

## Diagram Czynności



## Diagram Przypadków użycia

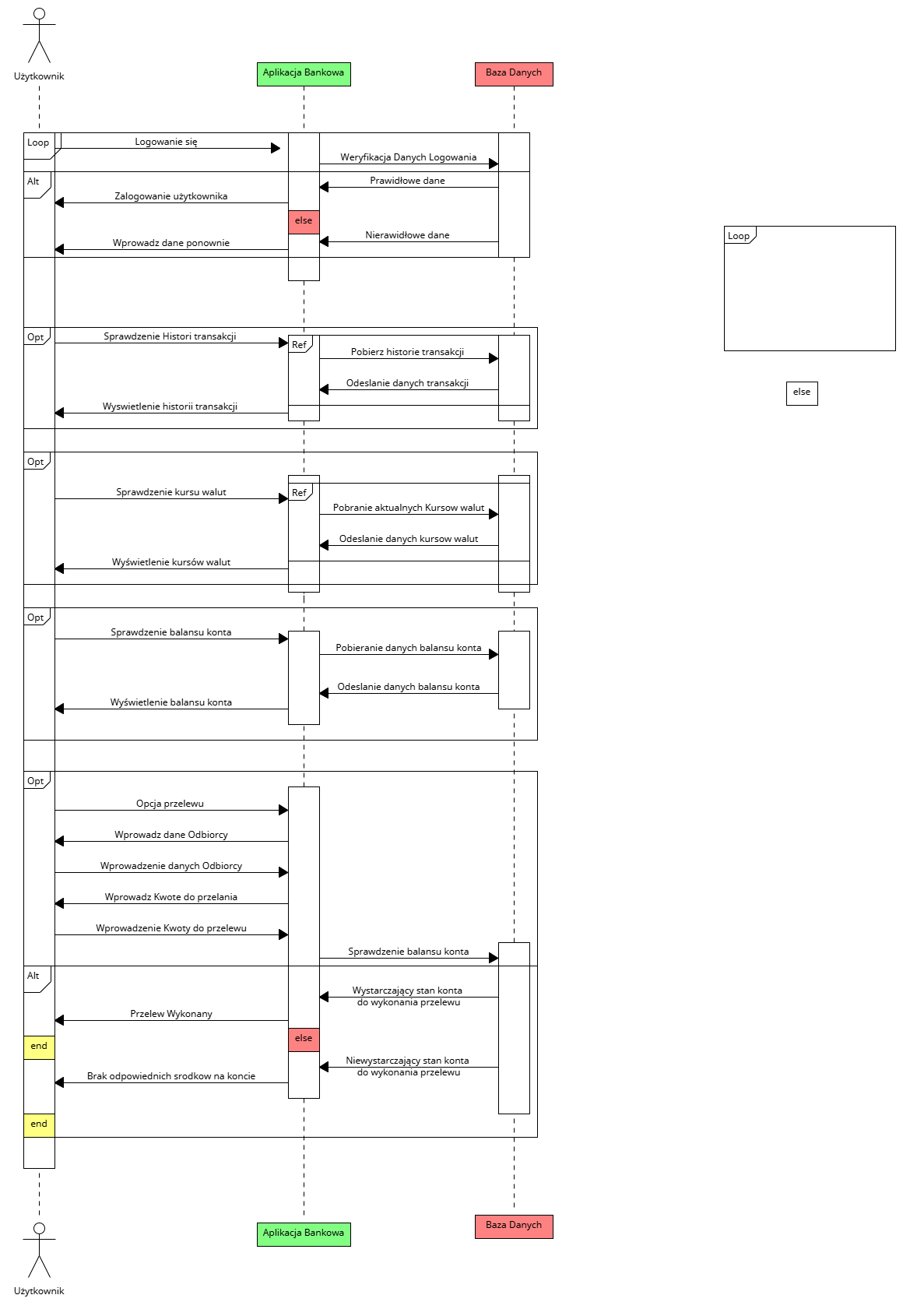


## Diagram Klas

A diagram of a computer program

AI-generated content may be incorrect.

## Diagram Sekwencji



# Funkcje Programu

*class Account:*

*def \_\_init\_\_(self, account\_id, balance=0.0):*

*self.account\_id = account\_id*

*self.balance = balance*

*self.transactions = []*

*def get\_balance(self):*

*return self.balance*

*def add\_funds\_via\_transfer(self, amount):*

*if amount <= 0:*

*raise ValueError("Amount must be positive")*

*self.balance += amount*

*self.transactions.append(f"Incoming transfer: {amount}")*

*return self.balance*

*def schedule\_payment(self, amount):*

*if amount <= 0:*

*raise ValueError("Amount must be positive")*

*if amount > self.balance:*

*raise ValueError("Insufficient funds")*

*self.balance -= amount*

*self.transactions.append(f"Scheduled payment: {amount}")*

*return self.balance*

*def transfer(self, target\_account, amount):*

*self.schedule\_payment(amount)*

*target\_account.add\_funds\_via\_transfer(amount)*

*self.transactions.append(f"Transfer to {target\_account.account\_id}: {amount}")*

*return self.balance*

*def get\_transaction\_history(self):*

*return self.transactions*

*class User:*

*def \_\_init\_\_(self, username, password):*

*self.username = username*

*self.password = password*

*self.logged\_in = False*

*def login(self, username, password):*

*self.logged\_in = self.username == username and self.password == password*

*return self.logged\_in*

# Testy

*import pytest*

*from bank\_app import Account, User*

*def test\_add\_funds\_via\_transfer():*

*acc = Account(1, 100)*

*assert acc.add\_funds\_via\_transfer(50) == 150*

*assert acc.add\_funds\_via\_transfer(1) == 151*

*with pytest.raises(ValueError):*

*acc.add\_funds\_via\_transfer(0)*

*with pytest.raises(ValueError):*

*acc.add\_funds\_via\_transfer(-10)*

*def test\_schedule\_payment():*

*acc = Account(1, 100)*

*assert acc.schedule\_payment(50) == 50*

*assert acc.schedule\_payment(10) == 40*

*with pytest.raises(ValueError):*

*acc.schedule\_payment(0)*

*with pytest.raises(ValueError):*

*acc.schedule\_payment(100)*

*def test\_transfer():*

*acc1 = Account(1, 200)*

*acc2 = Account(2, 50)*

*assert acc1.transfer(acc2, 50) == 150*

*assert acc2.get\_balance() == 100*

*with pytest.raises(ValueError):*

*acc1.transfer(acc2, 300)*

*with pytest.raises(ValueError):*

*acc1.transfer(acc2, -5)*

*def test\_get\_balance():*

*acc = Account(1, 123.45)*

*assert acc.get\_balance() == 123.45*

*acc.add\_funds\_via\_transfer(10)*

*assert acc.get\_balance() == 133.45*

*acc.schedule\_payment(3.45)*

*assert acc.get\_balance() == 130.0*

*def test\_get\_transaction\_history():*

*acc = Account(1)*

*acc.add\_funds\_via\_transfer(100)*

*acc.schedule\_payment(30)*

*history = acc.get\_transaction\_history()*

*assert "Incoming transfer: 100" in history*

*assert "Scheduled payment: 30" in history*

*assert len(history) == 2*

*def test\_login():*

*user = User("john", "1234")*

*assert user.login("john", "1234") is True*

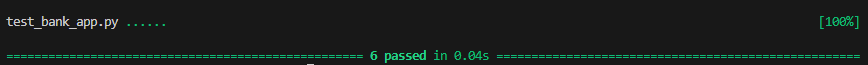
*assert user.logged\_in is True*

*assert user.login("john", "wrong") is False*

*assert user.login("wrong", "1234") is False*

*assert user.login("", "") is False*

# Wyniki testów



# Podział Obowiązków

Diagram czynności: Piotr Gałązka

Diagram przypadków użycia: Adrian Kozłowski

Diagram klas: Artur Lubiński

Diagram sekwencji: Dawid Kucisz

Funkcje i testy programu: Adrian Kozłowski, Dawid Kucisz, Piotr Gałązka, Artur Lubiński