

Dokumentacja techniczna do obsługi aplikacji bankowej

Autorzy:
Magdalena Denicka
Andrii Zapukhlyi

30 stycznia 2024

Spis treści

1	Wprowadzenie	2
2	Funkcje aplikacji i interfejs	2
3	Dodawanie konta bankowego	3
3.1	Kod źródłowy	4
4	Depozyt środków na konto	6
4.1	Kod źródłowy	7
5	Wycofanie środków z konta	8
5.1	Kod źródłowy	9
6	Sprawdzenie salda konta	10
6.1	Kod źródłowy	11
7	Zlecenie przelewu między kontami	12
7.1	Kod źródłowy	12
8	Wyświetlenie listy kont	14
8.1	Kod źródłowy	15
9	Modyfikacja danych	15
9.1	Kod źródłowy	16
10	Usuwanie konta bankowego	17
10.1	Kod źródłowy	18
11	Podział pracy	19

1 Wprowadzenie

Witamy w dokumentacji technicznej aplikacji do obsługi bankowości, stworzonej z myślą o zapewnieniu pracownikom intuicyjnego i prostego operowania. Dokument ten stanowi źródło informacji dla osób zainteresowanych zrozumieniem architektury i funkcji aplikacji.

Głównym celem aplikacji jest umożliwienie pracownikom banku łatwych, efektywnych i bezpiecznych operacji niezbędnych do bezproblemowego działania banku.

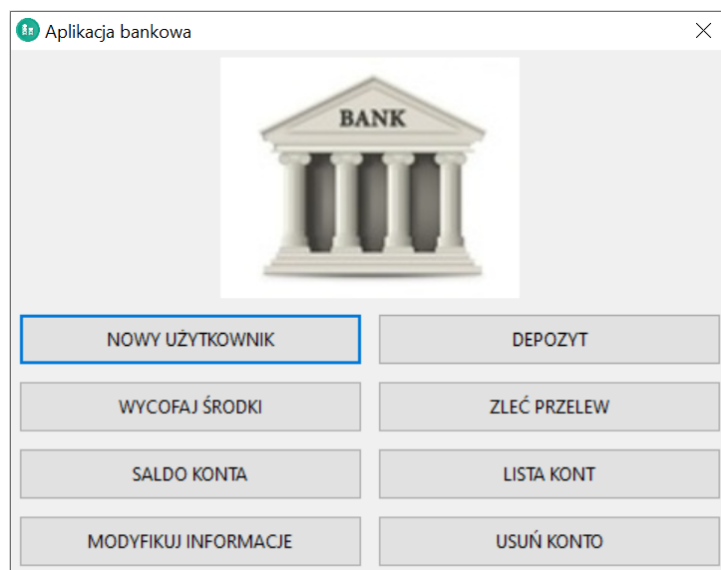
Dokumentacja została podzielona na oddzielne rozdziały, aby ułatwić zrozumienie różnych elementów aplikacji.

2 Funkcje aplikacji i interfejs

Rozdział dotyczy najważniejszych funkcjonalności aplikacji, niezbędnych do poprawnego działania banku. Poniżej lista funkcji aplikacji:

- Tworzenie i usuwanie konta bankowego
- Operacje na środkach – depozyty oraz wycofywanie
- Wgląd na saldo konkretnego konta
- Zlecanie przelewów między kontami bankowymi
- Wgląd na listę wszystkich użytkowników bankowości
- Modyfikacja danych konta
- Zapis kont do zewnętrznego pliku

Interfejs został zaprojektowany tak, aby był zrozumiały dla każdej osoby używającej aplikacji. Oparty jest o osiem funkcjonalnych przycisków przedstawionych na rysunku 1.

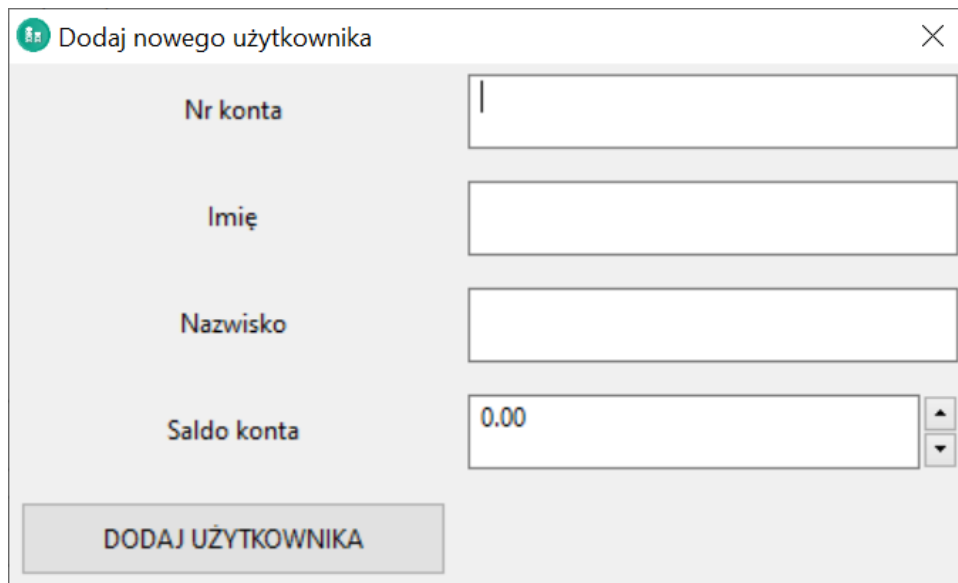


Rysunek 1: Wygląd aplikacji bankowej

3 Dodawanie konta bankowego

Aby dodać nowe konto bankowe w aplikacji, należy:

1. Otworzyć aplikację bankowości
2. Kliknąć przycisk **‘NOWY UŻYTKOWNIK’**
3. W polu „Nr konta” wpisać numer konta dodawanego użytkownika
4. W polu „Imię” wpisać pełne imię użytkownika
5. W polu „Nazwisko” wpisać pełne nazwisko użytkownika
6. W polu „Saldo konta” wpisać początkowy stan konta użytkownika w formie liczbowej
7. Kliknąć przycisk **‘DODAJ UŻYTKOWNIKA’** po wprowadzeniu i weryfikacji danych

The image shows a software dialog box titled "Dodaj nowego użytkownika" (Add new user) with a close button (X) in the top right corner. The dialog contains four input fields arranged vertically. The first field is labeled "Nr konta" (Account number) and is empty. The second field is labeled "Imię" (First name) and is empty. The third field is labeled "Nazwisko" (Last name) and is empty. The fourth field is labeled "Saldo konta" (Account balance) and contains the value "0.00"; it has small up and down arrow buttons to its right. At the bottom of the dialog is a button labeled "DODAJ UŻYTKOWNIKA" (Add user).

Rysunek 2: Dodawanie nowego użytkownika

3.1 Kod źródłowy

Sprawdzamy, czy podany numer konta istnieje w pliku db.txt. Jeśli plik jest otwarty, funkcja przechodzi do pętli while. Wewnątrz pętli, funkcja czyta każdą linię pliku za pomocą funkcji getline. Następnie linia ta jest przekazywana do strumienia wejściowego istream.

Następnie pobierany jest numer konta i zapisywany do zmiennej currentAccountNumber. Sprawdzamy czy currentAccountNumber jest równy accountNumber. Jeśli funkcja zwraca true to oznacza, że numer konta został znaleziony i zamyka plik, w przeciwnym razie zamyka plik i zwraca false.

```
1 bool isAccountNumberExists(int accountNumber)
2     { ifstream file("db.txt");
3         if (file.is_open()) {
4             string line;
5             while (getline(file, line)) {
6                 istream iss(line);
7                 int currentAccountNumber;
8                 iss >> currentAccountNumber;
9                 if (currentAccountNumber == accountNumber
10                    ) {
11                     file.close();
12                     return true; }
13             }
14             file.close();
15         }
16         return false;
17     }
```

Mamy teraz funkcję związaną z tworzeniem nowego użytkownika. Sprawdzamy, czy pola tekstowe nie są puste. Jeśli nie są wypełnione to wyświetlany jest komunikat informujący użytkownika o konieczności uzupełnienia wszystkich pól. Pole z numerem konta może zawierać tylko liczby. Numer konta jest konwertowany na liczbę całkowitą, a nazwisko i imię są konwertowane na ciągi znaków. Wartość salda pobierana jest z SpinCtrlDouble1. Funkcja wywołuje funkcję isAccountNumberExists (wcześniej omówioną), aby sprawdzić, czy nowo wprowadzony numer konta już istnieje w bazie danych. Jeśli numer konta jest unikalny, dane (numer konta, imię, nazwisko i saldo) są zapisywane do pliku db.txt.

Po zapisaniu danych wyświetlany jest komunikat informujący użytkownika, że konto zostało pomyślnie utworzone. Numer nowo utworzonego konta jest dodawany na początek listy elementów comboboxa.

```

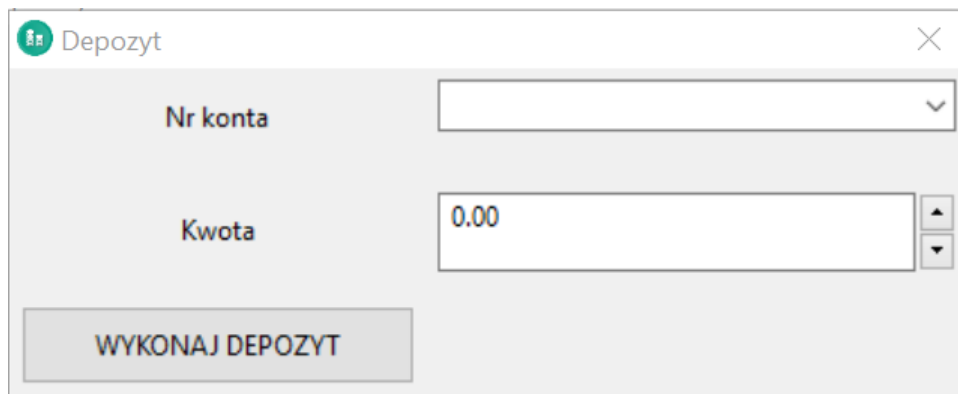
1 void wxNewUser::create_account(wxCommandEvent& event)
2     { if (TextCtrl1->IsEmpty() || TextCtrl2->IsEmpty() ||
3       TextCtrl3->IsEmpty()) {
4         wxMessageBox("Pole jest puste.", "Blad",
5           wxICON_ERROR);
6         return;
7     } else if (!TextCtrl1->GetValue().IsNumber()) {
8         wxMessageBox("Pole numer moze zawierac tylko
9         liczby.", "Blad", wxICON_ERROR);
10        return;
11    }
12    int number = stoi(w2s(TextCtrl1->GetValue()));
13    string name = w2s(TextCtrl2->GetValue());
14    string surname = w2s(TextCtrl3->GetValue());
15    double balance = SpinCtrlDouble1->GetValue();
16    if (isAccountNumberExists(number)) {
17        wxMessageBox(_("Numer konta juz istnieje!"),
18          "Blad", wxICON_ERROR);
19        return;
20    }
21    ofstream MyFile;
22    MyFile.open("db.txt", ios_base::app);
23    MyFile << number << "\t" << name << "\t" <<
24      surname << "\t" << balance << "\n";
25    MyFile.close();
26    wxMessageBox(_("Konto utworzone pomyslnie!"),
27      wxMessageBoxCaptionStr, wxICON_INFORMATION);
28    globalComboBoxItems.insert(globalComboBoxItems.
29      begin(), wxString::Format("%d", number));
30    Close(true);
31  }

```

4 Depozyt środków na konto

W celu dodania środków (depozytu) na określone konto bankowe trzeba:

1. Otworzyć aplikację bankowości
2. Kliknąć przycisk '**DEPOZYT**'
3. W polu „Nr konta” wpisać numer konta użytkownika
4. W polu „Kwota” wpisać kwotę, o ile powinien zwiększyć się stan konta użytkownika
5. Kliknąć przycisk '**WYKONAJ DEPOZYT**' po wprowadzeniu i weryfikacji danych



The screenshot shows a dialog box titled "Depozyt" with a close button (X) in the top right corner. Inside the dialog, there are two input fields. The first is labeled "Nr konta" and is a dropdown menu. The second is labeled "Kwota" and contains the value "0.00", with up and down arrow buttons to its right. At the bottom of the dialog is a button labeled "WYKONAJ DEPOZYT".

Rysunek 3: Depozyt

4.1 Kod źródłowy

Funkcja sprawdza poprawność wybranego numeru konta, aktualizuje saldo konta o podaną kwotę depozytu, a następnie informuje użytkownika o wyniku operacji.

```
1 void wxDepozyt::deposit(wxCommandEvent& event)
2     { if (ComboBox1->wxItemContainerImmutable::IsEmpty()) {
3         wxMessageBox("Pole jest puste.", "Blad",
4             wxICON_ERROR);
5         return;
6     } else if (!ComboBox1->GetValue().IsNumber()) {
7         wxMessageBox("Pole numer moze zawierac tylko
8             liczby.", "Blad", wxICON_ERROR);
9         return; }
10    int option = 1;
11    int number = stoi(w2s(ComboBox1->GetValue()));
12    double amount = SpinCtrlDouble1->GetValue();
13
14    fstream file(filePath);
15    ofstream tempFile(tempFilePath);
16    string line;
17    bool found = false;
18    while (getline(file, line)) {
19        istringstream iss(line);
20        int currentAccountNumber;
21        string name, surname;
22        double balance;
23        iss >> currentAccountNumber >> name >> surname >>
24            balance;
25        if (currentAccountNumber == number) {
26            if (option == 1) {
27                balance += amount;
28                found = true;
29            } else {
30                if (amount <= balance) {
31                    balance -= amount;
32                    found = true;
33                } else {
34                    break;
35                }
36            }
37        }
38        tempFile << currentAccountNumber << '\t' << name
39            << '\t' << surname << '\t' << balance << '\n';
40    }
41    file.close();
42    tempFile.close();
43    if (found && wxRenameFile(tempFilePath, filePath,
44        true)) {
```



```

40         wxMessageBox(_("Saldo zostalo pomyslnie
           zaktualizowane."), wxMessageBoxCaptionStr,
           wxICON_INFORMATION);
41     globalComboBoxItems.erase(std::find(
           globalComboBoxItems.begin(),
           globalComboBoxItems.end(), wxString::Format("%d", number)));
42     globalComboBoxItems.insert(globalComboBoxItems.
           begin(), wxString::Format("%d", number));
43     Close(true);
44 } else {
45     wxMessageBox(_("Nie znaleziono konta lub wystapil
           blad podczas aktualizacji salda."), "Error",
           wxICON_ERROR);
46     wxRemoveFile(tempFilePath);
47 }
48 }

```

5 Wycofanie środków z konta

Wycofanie środków z konta (odebraniu określonej kwoty) jest możliwe do zrobienia postępując wobec podanych niżej czynności:

1. Otworzyć aplikację bankowości
2. Kliknąć przycisk **‘WYCOFAJ ŚRODKI’**
3. W polu „Nr konta” wpisać numer konta użytkownika
4. W polu „Kwota” wpisać kwotę, o ile powinien zmniejszyć się stan konta użytkownika
5. Kliknąć przycisk **‘WYCOFAJ ŚRODKI’** po wprowadzeniu i weryfikacji danych

Rysunek 4: Wycofanie środków

5.1 Kod źródłowy

Funkcja jest odpowiedzialna za wypłatę środków z konta w aplikacji bankowej. Sprawdza poprawność numeru konta i kwoty wypłaty, aktualizuje saldo konta o odpowiednią kwotę, a następnie informuje użytkownika o wyniku operacji.

```
1 void wxWycofaj::withdraw(wxCommandEvent& event)
2     { if (ComboBox1->wxItemContainerImmutable::IsEmpty())
3         {wxMessageBox("Pole jest puste.", "Blad",
4           wxICON_ERROR);
5           return;
6         } else if (!ComboBox1->GetValue().IsNumber()) {
7             wxMessageBox("Pole numer moze zawierac tylko
8               liczby.", "Blad", wxICON_ERROR);
9             return; }
10    int option = 0;
11    int number = stoi(w2s(ComboBox1->GetValue()));
12    double amount = SpinCtrlDouble1->GetValue();
13    fstream file(filePath);
14    ofstream tempFile(tempFilePath);
15    string line;
16    bool found = false;
17    while (getline(file, line)) {
18        istringstream iss(line);
19        int currentAccountNumber;
20        string name, surname;
21        double balance;
22        iss >> currentAccountNumber >> name >>
23          surname >> balance;
24        if (currentAccountNumber == number) {
25            if (option == 1) {
26                balance += amount;
27                found = true;
28            } else {
29                if (amount <= balance) {
30                    balance -= amount;
31                    found = true;
32                } else {
33                    break;}}}
34        tempFile << currentAccountNumber << '\t' <<
35          name << '\t' << surname << '\t' << balance
36          << '\n';}
37    file.close();
38    tempFile.close();
39    if (found && wxRenameFile(tempFilePath, filePath,
40      true)) {
41        wxMessageBox(_("Saldo zostalo pomyslnie
42          zaktualizowane."), wxMessageBoxCaptionStr,
43          wxICON_INFORMATION);
```

```

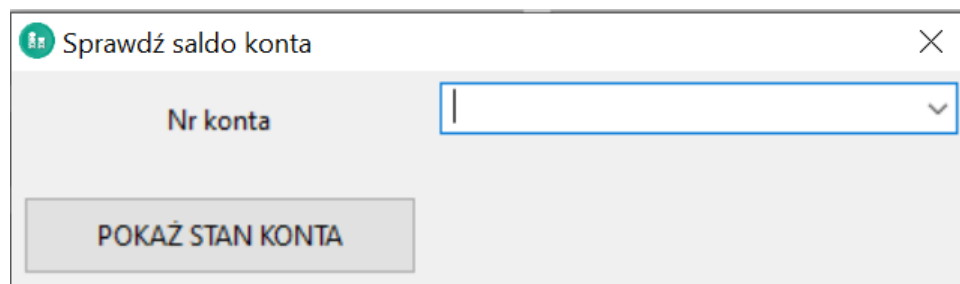
34         globalComboBoxItems.erase(std::find(
            globalComboBoxItems.begin(),
            globalComboBoxItems.end(), wxString::
            Format("%d", number)));
35         globalComboBoxItems.insert(
            globalComboBoxItems.begin(), wxString::
            Format("%d", number));
36         Close(true);
37     } else {
38         wxMessageBox(_("Nie znaleziono konta lub
            kwota jest wieksza od salda."), "Blad",
            wxICON_ERROR);
39         wxRemoveFile(tempFilePath);}}

```

6 Sprawdzenie salda konta

Chcąc sprawdzić saldo konkretnego konta użytkownika należy:

1. Otworzyć aplikację bankowości
2. Kliknąć przycisk '**SALDO KONTA**'
3. W polu „Nr konta” wpisać numer konta użytkownika
4. Kliknąć przycisk '**POKAŻ STAN KONTA**' po wprowadzeniu i weryfikacji danych



Rysunek 5: Sprawdzenie salda konta

6.1 Kod źródłowy

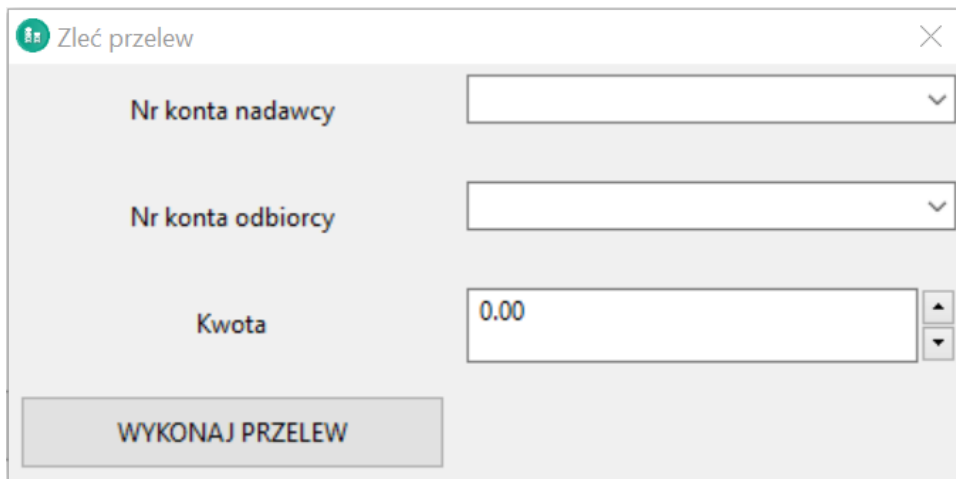
Funkcja sprawdza saldo wybranego konta w aplikacji bankowej. Jeśli numer konta jest poprawny i istnieje w pliku danych, wyświetla aktualne saldo. W przeciwnym razie informuje użytkownika o braku konta lub nieprawidłowym numerze.

```
1 void wxSaldo::OnButton1Click(wxCommandEvent& event)
2     {if (ComboBox1->wxItemContainerImmutable::IsEmpty())
3         {wxMessageBox("Pole jest puste.", "Blad",
4             wxICON_ERROR);
5             return;
6         } else if (!ComboBox1->GetValue().IsNumber()) {
7             wxMessageBox("Pole numer moze zawierac tylko
8                 liczby.", "Blad", wxICON_ERROR);
9             return; }
10    int num = stoi(w2s(ComboBox1->GetValue()));
11    ifstream file(filePath);
12    string line;
13    bool found = false;
14    while (getline(file, line)) {istringstream iss(line);
15        int currentAccountNumber;
16        string name, surname;
17        double balance;
18        iss >> currentAccountNumber >> name >> surname >>
19            balance;
20        if (currentAccountNumber == num) {
21            wxString balanceString = wxString::Format(_("%.1f"),
22                balance);
23            wxMessageBox(_("Saldo: ") + balanceString,
24                wxMessageBoxCaptionStr, wxICON_INFORMATION);
25            found = true;
26            globalComboBoxItems.erase(std::find(
27                globalComboBoxItems.begin(),
28                globalComboBoxItems.end(), wxString::Format("%d", num)));
29            globalComboBoxItems.insert(globalComboBoxItems.
30                begin(), wxString::Format("%d", num));
31            Close(true);
32            break;}}
33    file.close();
34    if (!found) {
35        wxMessageBox(_("Konto nie znaleziono."), "
36            Error", wxICON_ERROR);}}
```

7 Zlecenie przelewu między kontami

Przelew jest najważniejszą operacją związaną z bankowością. Aby zlecić przelew między kontami potrzeba:

1. Otworzyć aplikację bankowości
2. Kliknąć przycisk **'ZLEĆ PRZELEW'**
3. W polu „Nr konta nadawcy” wpisać numer konta użytkownika nadającego przelew
4. W polu „Nr konta odbiorcy” wpisać numer konta użytkownika odbierającego przelew
5. W polu „Kwota” wpisać kwotę przelewu
6. Kliknąć przycisk **'WYKONAJ PRZELEW'** po wprowadzeniu i weryfikacji danych

The image shows a graphical user interface window titled "Zleć przelew" (Transfer). It has a close button (X) in the top right corner. Inside the window, there are three input fields. The first is labeled "Nr konta nadawcy" (Sender account number) and is a dropdown menu. The second is labeled "Nr konta odbiorcy" (Recipient account number) and is also a dropdown menu. The third is labeled "Kwota" (Amount) and contains the text "0.00", with up and down arrow buttons to its right. At the bottom of the window is a button labeled "WYKONAJ PRZELEW" (Execute Transfer).

Rysunek 6: Zlecenie przelewu

7.1 Kod źródłowy

Funkcja zajmuje się operacją przekazywania środków między kontami w aplikacji bankowej. Sprawdza poprawność danych wejściowych, w tym numerów kont oraz dostępnych środków na koncie źródłowym. Następnie aktualizuje saldo kont zaangażowanych w transakcję i informuje użytkownika o wyniku operacji.

```
1 void wxPrzelew::transaction(wxCommandEvent& event)
2     {if (ComboBox1->wxItemContainerImmutable::IsEmpty() ||
        ComboBox2->wxItemContainerImmutable::IsEmpty()) {
        wxMessageBox("Pole jest puste.", "Bład", wxICON_ERROR)
        ;
        return;
3     } else if (!ComboBox1->GetValue().IsNumber() || !
4     ComboBox2->GetValue().IsNumber()) {wxMessageBox("
        Pole numer może zawierać tylko liczby.", "Bład",
        wxICON_ERROR);
```

```

5         return;}
6     int num = stoi(w2s(ComboBox1->GetValue()));
7     int recipient = stoi(w2s(ComboBox2->GetValue()));
8     double amount = SpinCtrlDouble1->GetValue();
9     fstream file(filePath);
10    ofstream tempFile(tempFilePath);
11    string line;
12    bool sourceAccountFound = false;
13    bool destinationAccountFound = false;
14    double sourceBalance;
15    while (getline(file, line)) {
16        istringstream iss(line);
17        int currentAccountNumber;
18        string name, surname;
19        double balance;
20        iss >> currentAccountNumber >> name >> surname >>
            balance;
21        if (currentAccountNumber == num) {
22            sourceAccountFound = true;
23            sourceBalance = balance;
24            if (amount <= sourceBalance) {
25                balance -= amount;
26                tempFile << currentAccountNumber << '\t'
                    << name << '\t' << surname << '\t' <<
                    balance << '\n';
27            } else {
28                wxMessageBox(_("Konto zrodlowe nie ma
                    wystarczajacego salda dla transakcji."
                    ), "Blad", wxICON_ERROR);
29                tempFile << line << '\n';
30                file.close();
31                tempFile.close();
32                wxRemoveFile(tempFilePath);
33                return;}
34        } else if (currentAccountNumber == recipient) {
35            destinationAccountFound = true;
36            balance += amount;
37            tempFile << currentAccountNumber << '\t' <<
                name << '\t' << surname << '\t' << balance
                << '\n';
38        } else {
39            tempFile << currentAccountNumber << '\t' <<
                name << '\t' << surname << '\t' << balance
                << '\n'; }}
40    file.close();
41    tempFile.close();
42    if (sourceAccountFound && destinationAccountFound &&
        wxRenameFile(tempFilePath, filePath, true)) {

```

```

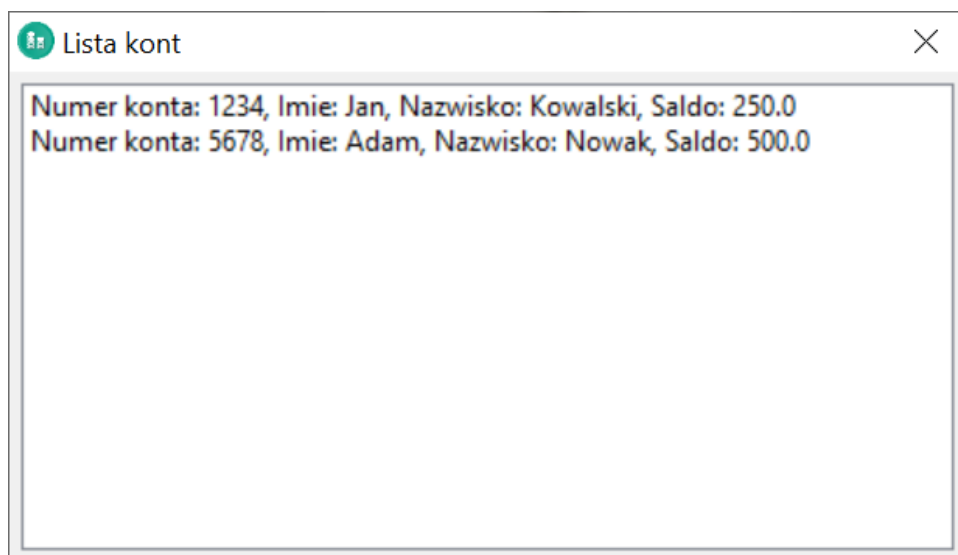
43     wxMessageBox(_("Transakcja zakończona pomyślnie.")
44     , wxMessageBoxCaptionStr, wxICON_INFORMATION);
45     globalComboBoxItems.erase(std::find(
46         globalComboBoxItems.begin(),
47         globalComboBoxItems.end(), wxString::Format("%d", num)));
48     globalComboBoxItems.insert(globalComboBoxItems.
49         begin(), wxString::Format("%d", num));
50     globalComboBoxItems.erase(std::find(
        globalComboBoxItems.begin(),
        globalComboBoxItems.end(), wxString::Format("%d", recipient)));
        globalComboBoxItems.insert(globalComboBoxItems.
            begin(), wxString::Format("%d", recipient));
        Close(true);
    } else {
        wxMessageBox(_("Błąd podczas wykonywania
        transakcji."), "Błąd", wxICON_ERROR);
        wxRemoveFile(tempFilePath);}}

```

8 Wyświetlenie listy kont

Wyświetlenie listy wszystkich kont jest osiągalne poprzez następujące czynności:

1. Otworzyć aplikację bankowości
2. Kliknąć przycisk **LISTA KONT**



Rysunek 7: Lista z przykładowymi kontami

8.1 Kod źródłowy

Odpowiada za tworzenie listy kont w aplikacji bankowej. Najpierw inicjuje okno dialogowe o tytule "Lista kont", w którym znajduje się przewijalne okno (ScrolledWindow1) oraz lista (ListBox1). Następnie, wczytuje dane z pliku "db.txt" zawierające informacje o numerze konta, imieniu, nazwisku i saldzie. Te informacje są dodawane do listy kont. Jeśli lista jest pusta, użytkownik otrzymuje komunikat informujący o braku danych.

```
1 wxLista::wxLista(wxWindow* parent,wxWindowID id,const wxPoint
   & pos,const wxSize& size)
2     {wxString filename = "db.txt";
3       ifstream file(filename.ToStdString());
4       ListBox1->Clear();
5       string line;
6       wxArrayString entries;
7       while (getline(file, line)) {istringstream iss(line);
8         int number;
9         string name, surname;
10        double balance;
11        if (iss >> number >> name >> surname >> balance) {
12          wxString entry = wxString::Format("Numer konta: %d
13            , Imie: %s, Nazwisko: %s, Saldo: %.1f",
14              number, name, surname, balance);
15            entries.Add(entry);}}
16        file.close();
17        if (!entries.IsEmpty()) {
18          ListBox1->InsertItems(entries, 0);
19        } else {
20          ListBox1->Append("Brak wpisow w bazie danych."); }}
```

9 Modyfikacja danych

W celu zaktualizowania danych takich jak imię, nazwisko lub saldo konta, należy:

1. Otworzyć aplikację bankowości
2. Kliknąć przycisk **'MODYFIKUJ INFORMACJE'**
3. W polu „Nr konta” wpisać numer konta użytkownika, którego dane będą zmieniane
4. W polu „Imię” wpisać niezmienione lub zaktualizowane imię użytkownika
5. W polu „Nazwisko” wpisać niezmienione lub zaktualizowane nazwisko użytkownika
6. W polu „Saldo konta” wpisać niezmienione lub zaktualizowane saldo konta użytkownika
7. Kliknąć przycisk **'MODYFIKUJ'** po wprowadzeniu i weryfikacji danych

Rysunek 8: Modyfikacja danych konta

9.1 Kod źródłowy

Funkcja modyfikuje dane konta w interfejsie aplikacji bankowej. Sprawdza poprawność danych wejściowych, a następnie aktualizuje plik. Po udanej operacji informuje użytkownika o pomyślnej modyfikacji konta, a w przypadku błędu wyświetla stosowny komunikat.

```

1 void wxModyfikuj::modify_account(wxCommandEvent& event)
2     {if (ComboBox1->wxItemContainerImmutable::IsEmpty()
3         || TextCtrl2->IsEmpty() || TextCtrl3->IsEmpty()) {
4         wxMessageBox("Pole jest puste.", "Bład",
5             wxICON_ERROR);
6         return;
7     } else if (!ComboBox1->GetValue().IsNumber()) {
8         wxMessageBox("Pole numer moze zawierac tylko
9             liczby.", "Bład", wxICON_ERROR);
10        return; }
11    int num = stoi(w2s(ComboBox1->GetValue()));
12    int new_balance = SpinCtrlDouble1->GetValue();
13    string new_name = w2s(TextCtrl2->GetValue());
14    string new_surname = w2s(TextCtrl3->GetValue());
15    fstream file(filePath);
16    ofstream tempFile(tempFilePath);
17    string line;
18    bool found = false;
19    while (getline(file, line)) {
20        istream iss(line);
21        int currentAccountNumber;
22        string name, surname;
23        double balance;
24        iss >> currentAccountNumber >> name >>

```

```

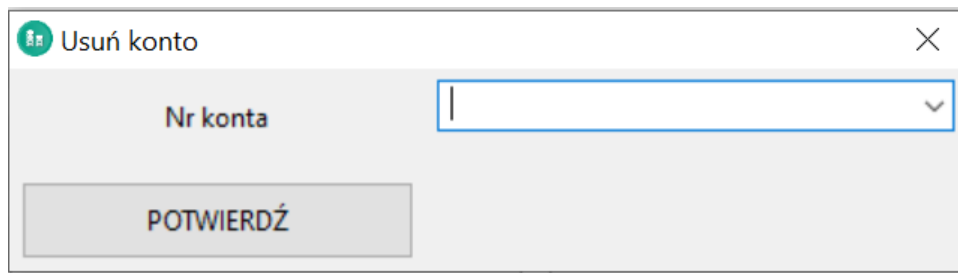
22         surname >> balance;
23     if (currentAccountNumber == num) {
24         found = true;
25         tempFile << currentAccountNumber << '\t'
26             << new_name << '\t' << new_surname <<
27             '\t' << new_balance << '\n';
28     } else {
29         tempFile << currentAccountNumber << '\t'
30             << name << '\t' << surname << '\t' <<
31             balance << '\n';}}
32 file.close();
33 tempFile.close();
34 if (found && wxRenameFile(tempFilePath, filePath,
35     true)) {
36     wxMessageBox(_("Konto zostalo pomyslnie
        zmodyfikowane."), wxMessageBoxCaptionStr,
        wxICON_INFORMATION);
    globalComboBoxItems.erase(std::find(
        globalComboBoxItems.begin(),
        globalComboBoxItems.end(), wxString::
        Format("%d", num)));
    globalComboBoxItems.insert(
        globalComboBoxItems.begin(), wxString::
        Format("%d", num));
    Close(true);
} else {
    wxMessageBox(_("Nie znaleziono konta lub
        wystapil blad podczas modyfikowania konta.
        "), "Error", wxICON_ERROR);
    wxRemoveFile(tempFilePath);}}

```

10 Usuwanie konta bankowego

Aby usunąć konto bankowe użytkownika trzeba postępować według poniższych kroków:

1. Otworzyć aplikację bankowości
2. Kliknąć przycisk **‘USUŃ KONTO’**
3. W polu „Nr konta” wpisać numer konta użytkownika, którego konto zostanie usunięte
4. Kliknąć przycisk **‘POTWIERDŹ’** po wprowadzeniu i weryfikacji danych



Rysunek 9: Usuwanie konta

10.1 Kod źródłowy

Funkcja usuwa konto z aplikacji bankowej. Sprawdza poprawność danych wejściowych, a następnie aktualizuje plik. Po udanej operacji informuje użytkownika o pomyślnym usunięciu konta, a w przypadku błędu wyświetla stosowny komunikat.

```

1 void wxUsun::delete_account(wxCommandEvent& event)
2     {if (ComboBox1->wxItemContainerImmutable::IsEmpty()){
3         wxMessageBox("Pole jest puste.", "Bład", wxICON_ERROR)
4         ;
5         return;
6     } else if (!ComboBox1->GetValue().IsNumber()) {
7         wxMessageBox("Pole numer moze zawierac tylko
8         liczby.", "Bład", wxICON_ERROR);
9         return; }
10    int num = stoi(w2s(ComboBox1->GetValue()));
11    fstream file(filePath);
12    ofstream tempFile(tempFilePath);
13    string line;
14    bool found = false;
15    while (getline(file, line)) {
16        istringstream iss(line);
17        int currentAccountNumber;
18        string name, surname;
19        double balance;
20        iss >> currentAccountNumber >> name >> surname >>
21        balance;
22        if (currentAccountNumber == num) {
23            found = true;
24        } else {
25            tempFile << currentAccountNumber << '\t' <<
26            name << '\t' << surname << '\t' << balance
27            << '\n'; }}
28    file.close();
29    tempFile.close();
30    if (found && wxRenameFile(tempFilePath, filePath,
31        true)) {
32        wxMessageBox(_("Konto usuniete pomyslnie."),
33            wxMessageBoxCaptionStr, wxICON_INFORMATION);

```

```

26         globalComboBoxItems.erase(std::find(
            globalComboBoxItems.begin(),
            globalComboBoxItems.end(), wxString::Format("%
27         Close(true);
28     } else {
29         wxMessageBox(_("Nie znaleziono konta lub wystapil
            blad podczas usuwania konta."), "Blad",
            wxICON_ERROR);
30         wxRemoveFile(tempFilePath); }}

```

11 Podział pracy

- GUI - Magdalena Denicka 100 %
- Plan projektu - Magdalena Denicka 50%, Andrii Zapukhlyi 50%
- Funkcjonalność i praca z bazą danych - Andrii Zapukhlyi 100%
- Dokumentacja - Magdalena Denicka 100%

Spis rysunków

1	Wygląd aplikacji bankowej	2
2	Dodawanie nowego użytkownika	3
3	Depozyt	6
4	Wycofanie środków	8
5	Sprawdzenie salda konta	10
6	Zlecenie przelewu	12
7	Lista z przykładowymi kontami	14
8	Modyfikacja danych konta	16
9	Usuwanie konta	18