ВИСШЕ ВОЕННОМОРСКО УЧИЛИЩЕ
„НИКОЛА Й. ВАПЦАРОВ“

9002 Варна, ул. „В. Друмев“ № 73

ДОКУМЕНТАЦИЯ КЪМ КУРСОВ ПРОЕКТ

ТЕМА: Bank Account Management System
ДИСЦИПЛИНА: Основи на програмирането и алгоритмизация

РАЗРАБОТЧИК:
Светлозар Деянов Драгнев

специалност: Информационни и комуникационни технологии (1. курс)
факултет „Инженерен“ | **катедра „Електроника“**
фак. номер: 1224-25120
период на разработка: ноември - декември 2025 г.

КОНТАКТИ:
имейл: 122425120@naval-acad.bg | svetlozardragnev@gmail.com
GitHub: www.github.com/svetdrag

Съдържание:

1. КОНЦЕПЦИЯ И ЦЕЛ НА ПРОЕКТА

1.1. Цел

Целта на проекта е създаването на конзолно приложение, симулиращо работата на банкова система за управление на потребителски акаунти и техните сметки. Програмата демонстрира основни принципи на процедурното програмиране, работа с външни файлове (бази данни), алгоритмично мислене и структуриране на кода чрез модулност.

1.2. Концепция и обхват на проекта

Чрез конзолното приложение напълно успешно може да се демонстрира реално банково приложение. Може както всеки потребител да се вписва в своя акаунт, така и да се създават нови. Всеки потребителски профил разполага с три имена на титуляр, номер на сметка и наличност в нея, както и ПИН код. Интегрирани са всички базови банкови операции, които може да се използват(депозиране и теглене на пари, изпращане на пари до друг банков потребител, смяна на ПИН, показване на наличен баланс по сметка, изготвяне на банково извлечение). Приложението може да бъде лесно достъпно и използвано от всички заинтересовани лица.

1.3. Използвани технологии и среда за разработка

Цялото приложение е написано на програмния език C, като за създаване на банковото извлечение се използва HTML с вътрешен CSS. За съхраняване, обновяване и използване на информация от бази данни се използват два текстови (.txt) файла. Използваната среда за разработка на цялата логика е Visual Studio Code. Проектът бива синхронизиран и качван в предварително създадено repository в GitHub посредством GitHub Desktop.

Използваните стандартни библиотеки на C:

A. <stdio.h> - използва се за входно-изходните операции като printf, scanf; основна роля при работа с файлове (*fopen, fclose, fprintf, fscanf)*;
B. <stdlib.h> - управление на паметта и конвертиране на типове данни; функция „atoi" за преобразуване на низове в числа, както и „system()";
C. <string.h> - обработка на низове; функция "strcmp" (*сравняване на ПИН)*, "strcpy" (*копиране)*, "strtok" (*разделяне на данни при четене)*;
D. <ctype.h> - валидация на данни; „isdigit()" се използва за проверка дали въведеният ПИН съдържа само цифри;
E. <time.h> - време и дати; записване на точна дата на транзакциите в лог файла и датата на издаване на банковото извлечение.

2. АНАЛИЗ НА ИЗИСКВАНИЯТА

2.1. Функционални изисквания

Функционалните изисквания описват поведението на системата и услугите, които тя предоставя на крайния потребител.

Управление на потребители (User Management) - системата предоставя механизми за създаване и автентификация на потребителски профили:

A. Регистрация на нов потребител:
   Системата позволява въвеждане на лични данни: Име, Презиме и Фамилия.
   Потребителят задава свой секретен ПИН код (4 до 6 цифри).
   Автоматично генериране на сметка: При успешна регистрация, алгоритъмът автоматично генерира уникален 4-цифрен номер на банкова сметка (Account ID), който служи за идентификация в системата. Този номер се съобщава на потребителя веднага след записа.
   Начален баланс: Всеки новосъздаден акаунт стартира с нулев баланс (0.00 BGN).

B. Вход в системата *(authentication)*:
   Процесът на вход изисква въвеждане на валидна комбинация от "Номер на сметка" и "ПИН код".
   Системата извършва проверка в базата данни *(accounts.txt)*.
   При неуспешен опит *(грешен номер и/или парола)*, достъпът се отказва и се извежда съответното съобщение за грешка.

Финансови операции - сърцевината на приложението е симулацията на банкови транзакции, които променят състоянието на баланса в реално време:

A. Депозиране на средства *(Deposit)*:
   Позволява на потребителя да внесе положителна сума по своята сметка.
   Системата валидира дали въведената стойност е по-голяма от 0 и съдържа само цифри.
   Балансът се актуализира незабавно в оперативната памет и се записва във файла.

B. Теглене на пари *(Withdraw)*:
   Позволява изтегляне на суми от сметката.
   Проверка за наличност: Преди извършване на транзакцията, системата проверява дали текущият баланс е достатъчен за покриване на исканата сума и входните данни са само цифри. Ако средствата са недостатъчни, операцията се блокира.

C. Банков превод *(Transfer)*:
   Позволява прехвърляне на средства от текущия потребител към друг регистриран потребител.
   Изисква въвеждане на: Номер на сметка на получателя, Име, Презиме и Фамилия на получателя *(за потвърждение)* и Сума.
   Системата извършва операцията: едновременно намалява баланса на изпращача и увеличава този на получателя, гарантирайки цялост на данните.

Отчетност и Справки *(Reporting)* - системата осигурява прозрачност на действията чрез визуализация на данните:

A. Справка на баланс: Потребителят може по всяко време да види текущата си наличност на екрана.
B. Генериране на Банково извлечение *(.html файл)*:
   Уникална функционалност, която създава външен файл *(напр. statement_1005.html)*. Файлът съдържа форматирана таблица с информация за титуляра, текущия баланс, дата и час на издаване. Документът е стилизиран с вграден CSS за професионален външен вид и се отваря автоматично в браузъра по подразбиране на операционната система.

2.2. Нефункционални изисквания

Нефункционалните изисквания описват атрибутите за качество на софтуера – как той функционира под натоварване и какви стандарти спазва.

Сигурност на данните *(Data Security & Encryption)* - за да се защити конфиденциалността на потребителите, системата не съхранява паролите в чист текст.

A. Алгоритъм на криптиране: Използва се модифициран Цезаров шифър *(Caesar Cipher)*.
B. Механизъм: При запис във файла accounts.txt, всяка цифра от ПИН кода се измества с фиксиран ключ *(напр. +3)*. При четене *(вход в системата)*, данните се декриптира *(изместване -3)*, за да бъдат използвани от програмата.

Ако файлът с базата данни бъде откраднат или отворен от неоторизирано лице, ПИН кодовете ще бъдат нечетими и неизползваеми без знанието за алгоритъма и ключа.

Устойчивост на данните *(Data Persistence)* - системата е проектирана да запазва състоянието си между отделните стартирания. Всички данни се съхраняват в структурирани текстови файлове *(accounts.txt и transactions.txt)*, разположени в директория data/.

Валидация на входа и обработка на грешки (*Input Validation & Error Handling)* - приложението е защитено срещу некоректни потребителски действия, които биха могли да доведат до срив *(crash)*.

A. Защита на входния буфер: Имплементирани са механизми за изчистване на стандартния вход *(stdin)* след всяко въвеждане. Това предпазва от проблеми при въвеждане на символи *(char)* там, където се очакват числа *(int/double)*.
B. Логически проверки: Не се допускат отрицателни суми при транзакции. ПИН кодът се проверява дали съдържа само цифри *(isdigit проверка)*. Не се допуска превод към несъществуваща сметка.

2.3. Потребителски роли

В настоящата версия на системата е дефинирана една основна роля, която обхваща всички взаимодействия с интерфейса.

Роля: Банков Клиент (Client / Standard User)

A. **Права и достъп:** Има пълен достъп до собствените си средства и лични данни. Може да инициира транзакции само от свое име. Може да генерира документи *(извлечения)* само за своята сметка.
B. **Ограничения:** Няма достъп до списъка с други потребители. Не може да вижда баланса на чужди сметки. Не може да изтрива записи от лог-файла с транзакции.

3. АРХИТЕКТУРА НА СИСТЕМАТА

3.1. Общ преглед на файловата структура и организация на кода

Проектът се състои от няколко модула, за да бъде улеснена четимостта, поддръжката и бъдещето развитие.

Логическата структура е разделена на: bank.h – хедър файл, в който се съдържат всички дефиниции на структури, глобални константи и прототипи на функции; bank_functions.c – основен файл с изходен код, където са имплементирани всички функции *(цялата бизнес логика)*; main_bank.c – начална точка на програма, управлява се главното меню и потребителските входни данни; README.md – основно описани на проекта за GitHub repository-то; LICENSE - файл, в който се съдържа MIT лицензът за използване на проекта; директория data/ - в нея се съдържат двата текстови файла с функция на бази данни.

3.2. Описание на програмните модули

A. **bank.h (дефиници):**
   В този хедър файл е описана основната структура User, която моделира един банков потребител в паметта. Тя съдържа полета за номер на сметка *(accountNumber)*, ПИН код, три имена и текущ баланс. Тук са дефинирани и макроси *(като MAX_USERS и FILENAME)*, както и декларирани всички функции с ключовата дума extern за глобалните променливи.

B. **bank_functions.c (функционалност):**
   i. **void loadUsersFromFile()** - Тази функция се стартира автоматично при пускане на програмата. Нейната задача е да отвори файла data/accounts.txt и да прочете записаните потребители ред по ред. При четенето, тя извиква декриптиращия алгоритъм, за да преобразува шифрираните ПИН кодове в четим вид, преди да ги зареди в оперативната памет *(масива users)*.

   ii. **void saveUsersToFile()** - Функцията отговаря за устойчивостта на данните. Тя се вика след всяка промяна *(регистрация, транзакция)*. Тя отваря файла в режим на запис *(overwrite)*, обхожда масива с потребители и записва актуалното състояние. Преди запис, ПИН кодът на всеки потребител се криптира наново, за да не се съхранява в чист вид.

iii. **void caesarCipher(char *pin, int mode)** - Функция, реализираща алгоритъма за сигурност "Цезаров шифър". Приема като аргумент указател към ПИН кода и режим на работа *(1 за криптиране, -1 за декриптиране)*. Чрез модулна аритметика измества всяка цифра от паролата с фиксиран ключ, осигурявайки базова защита на личните данни.

iv. **int login()** - Управлява процеса на автентификация. Изисква от потребителя въвеждане на номер на сметка и ПИН. След това обхожда заредените в паметта потребители и сравнява въведените данни. Връща индекса на потребителя при успех или -1 при грешка.

v. **void registerUser()** - Създава нов профил в системата. Изисква въвеждане на три имена и желана парола. Функция автоматично генерира уникален 4-цифрен IBAN *(номер на сметка)* и задава начален баланс от 0.00 евро. Накрая извиква функцията за запис във файл.

vi. **void depositMoney()** - Позволява на текущо влезлия потребител да внесе средства. Функцията извършва валидация, за да гарантира, че сумата е положително число, добавя я към баланса и записва промяната, като същевременно създава запис в историята на транзакциите.

vii. **void withdrawMoney()** - Реализира тегленето на пари. Основната ѝ роля е да провери дали потребителят разполага с достатъчно наличност. Ако балансът е достатъчен, сумата се изважда; в противен случай се извежда съобщение за грешка.

viii. **void transferMoney()** - Управлява преводите между клиенти. Изисква номер на сметка и трите имена на получателя и сума. Функцията търси получателя в базата данни, проверява наличността на изпращача и извършва операцията – намалява парите на единия и ги увеличава на другия, записвайки промените незабавно.

ix. **void exportStatementHTML()** - Генерира динамичен HTML файл *(напр. statement_1005.html)*. Функцията конструира уеб страница чрез C код, добавяйки CSS стилове за оформление и таблица с данните на потребителя. Накрая използва системно повикване *(system)*, за да отвори автоматично генерирания файл в подразбиращия се браузър.

x. **void logTransaction(int userIndex, const char *type, double amount)** - Помощна функция за съхранение на информация. Тя отваря файла data/transactions.txt в режим на добавяне *(append)* и записва ред, съдържащ: дата, име на потребителя, вид операция *(напр. "Deposit")* и сумата. Това създава история на действията.

xi. **int isValidPin(char *pin)** - Валидираща функция, която използва библиотеката <ctype.h>. Тя проверява дали всеки символ във въведения низ е цифра. Предпазва програмата от грешки при въвеждане на букви или символи в полето за ПИН.

xii. **int verifyPin()** - Спомагателна функция за сигурност, която се използва преди извършване на чувствителни операции *(като промяна на ПИН код)*. Тя изисква от потребителя да въведе текущия си ПИН повторно, за да потвърди самоличността си. Връща 1 *(true)*, ако въведеният код съвпада с този в паметта, и 0 *(false)* при несъответствие.

xiii. **void showBalance()** - Функция за справка, която визуализира текущото финансово състояние на потребителя. Тя чете стойността от полето balance на текущия потребител и я отпечатва на екрана, форматирана до втори знак след десетичната запетая *(например "100.50 EUR")*, без да извършва промени по данните.

xiv. **void changePin()** - Позволява на потребителя да смени своя код за достъп. Процесът включва три стъпки: 1) Верификация на стария ПИН *(чрез verifyPin)*; 2) Въвеждане и валидация на новия ПИН *(да съдържа само цифри)*; 3) Обновява структурата в паметта и веднага извиква saveUsersToFile(), за да запази промяната *(с новото криптиране)* във файла.

xv. **int openFile(char *filename)** - Системна функция, която осигурява връзката между C програмата и операционната система. Нейната задача е да конструира текстов низ с команда *(напр. "start statement1005.html")* и да я подаде на конзолата чрез функцията system(). Това води до автоматично отваряне на генерирания HTML отчет в браузъра по подразбиране. Функцията управлява динамична памет *(calloc)*, за да създаде безопасен буфер за командата.

C. **main_bank.c (главен модул)**

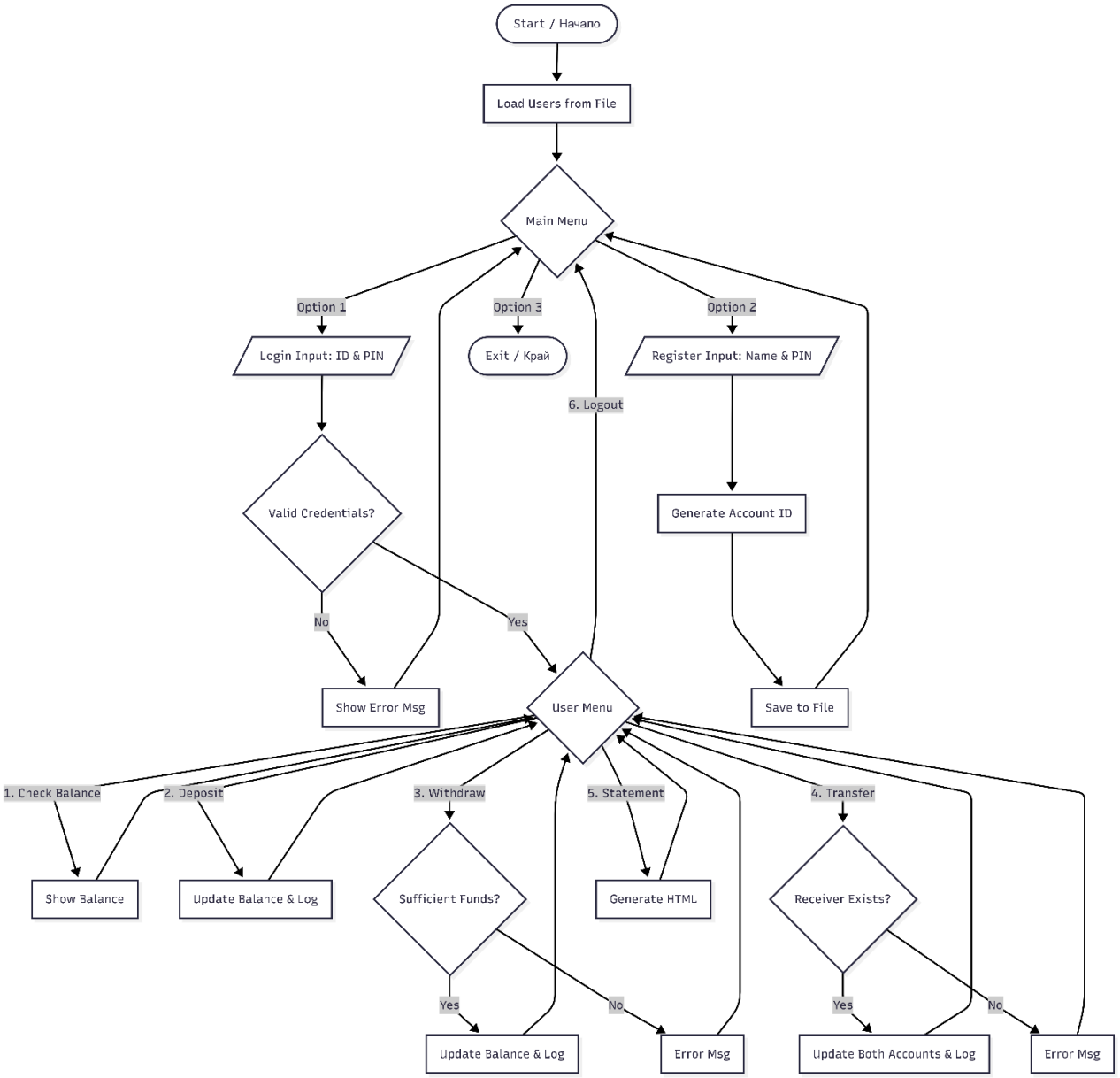
Съдържа функцията main() – инициализира се програмата чрез loadUserFormFile() и се влиза в while цикъл, който визуализира главното меню. Чрез конструкцията switch се насочва потребителя към съответната функционалност, която е избрал *(Login, Register или Exit)*.

D. **Бази данни**

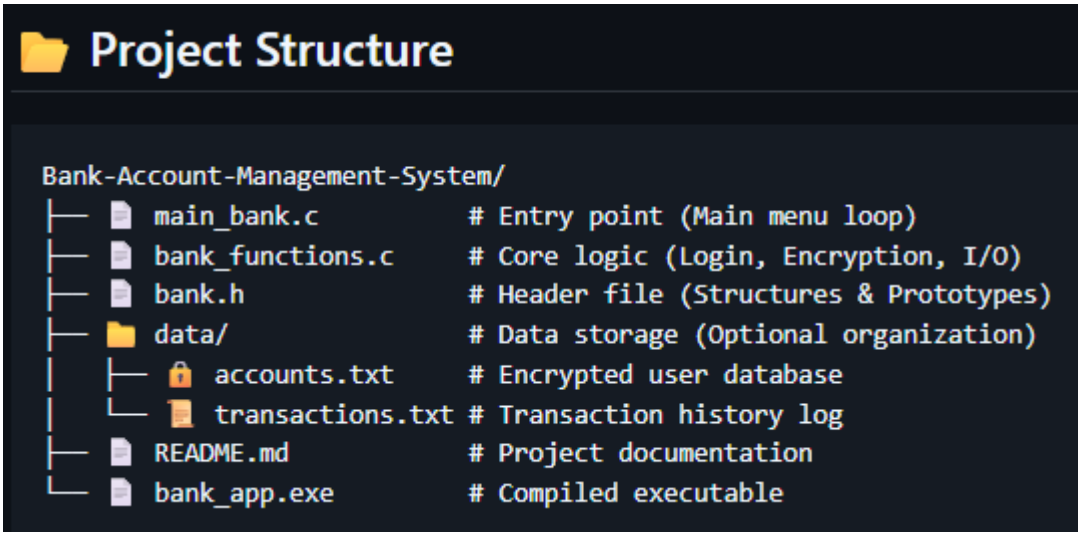
i. **data/accounts.txt** – всеки ред съдържа данните за даден потребител. С оглед на сигурност, при записване на ПИН кодът на всеки потребител, той бива криптиран предварително.

ii. **data/transactions.txt** – лог файл, пази хронологично историята на всички банкови операции от всички потребители. Използва се при създаване на банково извлечение.

3.3. Диаграма на процесите (Алгоритъм)



3.4. Организация на файловете



4. ИМПЛЕМЕНТАЦИЯ НА КЛЮЧОВИ АЛГОРИТМИ

4.1. Алгоритъм за сигурност (Caesar Cipher)

Вдъхновено от курса CS50 на Harvard, приложението използва алгоритъм **"Цезаров шифър"** за защита на ПИН кодовете във файла `accounts.txt`. При запис всяка цифра се измества с определен ключ, а при четене се декриптира. Това гарантира, че данните не са четими в чист текст. Всяка цифра от ПИН кода се третира като число и се измества с фиксирана стойност *(ключ)*, дефинирана чрез макрос CAESAR_KEY.

A. **Шифроване *(енкрипция)*:** C = (x + k) % 10. Където x е оригиналната цифра, k е ключът, а модулното деление (% 10) гарантира, че резултатът винаги ще е едноцифрено число (0-9).
B. **Разшифроване (декрипция):** обратната операция се извършва при четене от файла, за да може програмата да работи с оригиналната парола в паметта.

Този метод предотвратява възможността ПИН кодовете да бъдат прочетени директно, ако някой отвори текстовия файл с текстов редактор.

4.2. Алгоритъм за генериране на банково извлечение (HTML файл)

Една от отличителните черти на системата е възможността за експорт на данни в уеб формат. Алгоритъмът работи на следните стъпки:

A. **Конструиране на името:** Програмата динамично създава име на файл, уникално за потребителя *(напр. statement_1005.html)*, използвайки sprintf.
B. **Генериране на съдържание:** Чрез серия от fprintf команди, програмата записва стандартен HTML5 код.
    - Включва CSS стилове в <style> секцията за оформление *(цветове, шрифтове, граници на таблици, лого)*.
    - Попълва <table> с данните на текущия потребител *(Имена, Баланс)*.
    - Добавя Timestamp *(дата на генериране)*.
C. **Стартиране:** Използва се системна команда *(system("start ...") за Windows)*, която инструктира операционната система да отвори новосъздадения файл с браузъра по подразбиране.

4.3. Механизъм за валидация на входни данни

За да се осигури стабилност на програмата, е разработен механизъм за проверка на потребителския вход.

- **Проблем:** Стандартната функция scanf може да остави символи в буфера, което води до прескачане на следващи команди или безкраен цикъл при въвеждане на букви вместо цифри.
- **Решение:**
    1. **Изчистване на буфера:** След всяко въвеждане се прилага логика за изчистване на stdin *(стандартния вход)*, за да се премахнат остатъчни символи за нов ред *(\n)*.
    2. **Проверка за цифри (isdigit):** При регистрация или промяна на ПИН, функцията isValidPin обхожда въведения низ символ по символ. Ако открие нецифров символ, операцията се отхвърля.
    3. **Логически проверки:** системата не позволява въвеждане на отрицателни суми при депозит или суми, надвишаващи баланса при теглене.

5. РЪКОВОДСТВО ЗА ПОТРЕБИТЕЛЯ

5.1. Стартиране на програмата

Приложението е конзолно *(изпълнява се в терминала)*. За да бъде стартирано:

A. Отворете терминал *(Command Prompt / PowerShell)* в папката на проекта.
B. Уверете се, че разполагате с изпълнимия файл bank_app.exe. *(ако ли не – създайте го чрез въвеждане на командата* ***gcc bank_functions.c main_bank.c -o bank_app.exe*** *в терминала)*
C. Напишете командата **./bank_app** и натиснете Enter.
D. Ако папката data/ липсва, създайте я ръчно преди първото стартиране.

5.2. Първи стъпки

Ако за пръв път стартиране програмата и нямате регистриран банков профил:

A. От главното меню изберете опция **2. Register**.
B. Въведете Вашите: Име, Презиме и Фамилия *(на латиница)*.
C. Създайте 4-6 цифрен ПИН код.
D. **ВАЖНО:** Запишете си генерирания **Account Number** *(напр. 1005)*, който системата ще ви покаже. Той ви е необходим за вход.

5.3. Изпълнение на операции

След успешен вход *(Login)* с вашия номер и ПИН, получавате достъп до User Menu-то:

A. **Check Balance:** Показва текущата сума в евро *(EUR)*.
B. **Change Pin:** Възможност потребителят да смени своя PIN код.
C. **Deposit Money:** Изберете опцията и въведете сума за внасяне.
D. **Withdraw Money:** Позволява теглене, ако имате наличност.
E. **Transfer:** Изисква да знаете трите имена и номера на сметката на получателя.
F. **Export Statement:** Генерира HTML справка и автоматично я отваря в браузъра.

6. БЪДЕЩО РАЗВИТИЕ И УСТОЙЧИВОСТ

За да се превърне в пълноценен банков софтуер, проектът подлежи на следните надграждания:

1. **Бази данни:** Миграция от текстови файлове към SQL база данни (MySQL/PostgreSQL) за по-голяма бързина и сигурност.
2. **Графичен интерфейс (GUI):** Разработка на десктоп приложение с прозорци и бутони. Красива и иновативна визия.
3. **Мултивалутен портфейл:** Възможност за съхранение на EUR, USD, BGN и автоматично превалутиране. Имплементиране на функция за валутен калкулатор.
4. **Двуфакторна автентификация (2FA):** Изпращане на код по имейл или SMS при превод на средства и за създаване на потребителски акаунт.

7. ИЗПОЛЗВАНА ЛИТЕРАТУРА И ИЗТОЧНИЦИ

- Лекции по „Основи на програмирането и алгоритмизация“
- Курс CS50 – Harvard Edu
- „Програмният език C – второ издание“ – Б. В. Керниган, Д. М. Ричи
- AI асистент – Gemini
- YouTube клипове
- https://www.tutorialspoint.com/c_standard_library/c_function_localtime.htm
- https://www.w3schools.com/c/index.php
- https://stackoverflow.com/questions/1538420/difference-between-malloc-and-calloc
- https://www.mermaidchart.com/app/projects/50172307-5bc2-4606-9498-8e8b207a0254/diagrams/45fc1664-91c9-41a9-88cc-f74d2ff8a852/share/invite/eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkb2N1bWVudElEIjoiNDVmYzE2NjQtOTFjOS00MWE5LTg4Y2MtZjc0ZDJmZjhhODUyIiwiYWNjZXNzIjoiVmlldyIsImlhdCI6MTc2NjUxMjk1NX0.3aGVDlg0-lnt3jheP-6UvRzX-pQHjmK36d4YKEjYFcg

8. GitHub repository

Целият проект е качен в следното GitHub repository:

- https://github.com/SvetDrag/Bank-Account-Management-System

MIT лиценз: