Wydział Informatyki Politechniki Białostockiej Przedmiot: Komunikacja Człowiek-Komputer	Data oddania projektu: 13.11.2018 r.
Zespół: 1. Michał Kuc 2. Marek Ciborowski	Prowadzący: mgr inż. Łukasz Gadomer Ocena:

1. **Opis projektu.**

Projekt ma na celu stworzenie platformy do wypełniania ankiet, który tworzą użytkownicy. Każdy ma możliwość założenia konta, stworzenia ankiety (która zawiera pytania jednokrotnego, wielokrotnego wyboru, umożliwia tworzenie własnych odpowiedzi), przeglądania jej wyników czy obserwowania użytkowników (lista obserwowanych użytkowników może nam ułatwić znalezienie ankiet, które interesują nas najbardziej).

2. **Opis funkcjonalności.**

Survey:

a) **Configuration.cs:**
   i) **ConsoleInitialize()** - metoda do inicjalizowania konsoli;
   ii) **SetConsoleSize()** - metoda do ustawiania rozmiarów konsoli (tylko na starcie programu);
   iii) **ConsoleClearToArtAscii()** - czyszczenie ekranu do napisu "SURVEY" w art ascii;
   iv) **CurrentConsoleLineClear()** - czyszczenie linii w konsoli, w której aktualnie znajduje się kursor;
   v) **CurrentConsoleLineClear(int currentXposition)** - czyszczenie linii w konsoli, w której aktualnie znajduje się kursor i powrót do pozycji określonej w parametrze;
   vi) **MainMenu(List<MenuOptions> listOptions)** - wywoływanie menu, które wyświetla 3 opcje na ekranie maksymalnie, z możliwością przesuwania po kolejnych opcjach. Listę opcji otrzymuje w parametrze; Opcje, wywoływane przez nas znajdują się w pliku **MenuOptions.cs** (dwie metody, bo są dwa menu);

b) **Program.cs:**

Zawiera 2 metody, jedna wywoływana przy starcie systemu (metoda główna) i druga do powrotu do ekranu początkowego, który dodatkowo wyświetla komunikat przekazany z poprzedniego okna.

a) **ChangePassword.cs:** widok do zmiany hasła
b) **ChooseAccount.cs:** widok, który w parametrze otrzymuje listę użytkowników, którzy będą wyświetlani w zależności od potrzeb (wszystkie konta, obserwowani, obserwujący, głosujący, itp.);
c) **ChooseSurvey.cs:** widok analogiczny do **ChooseAccount.cs** tylko w odniesieniu do ankiet;
d) **CreateAccount.cs:** widok do tworzenia konta;
e) **CreateSurvey.cs:** widok do tworzenia ankiet;
f) **DeleteAccount.cs:** widok do usuwania konta;
g) **FillSurvey.cs:** widok do uzupełniania ankiet;
h) **PersonView.cs:** widok do wyświetlania danych o użytkowniku;
i) **ShowResult.cs:** widok do wyświetlania wyników ankiety;
j) **SignInView.cs:** widok do logowania się do systemu;

RepositoryLayer:

Każde repozytorium zawiera podstawowe metody do tworzenia poszczególnych modelów, dodawania do bazy, usuwania z bazy, wyszukiwania poszczególnych danych na podstawie głównie numeru ID (rzadko inne parametry są wykorzystywane do wyszukiwania np. po loginie i haśle). Repozytorium może zawierać metody wyspecjalizowane np. walidację loginu, hasła, emailu itp. Te zostaną opisane poniżej:

a) **AccountRepository.cs:** klasa zawierająca funkcjonalność użytkownika powiązaną z bazą danych oraz walidacją konta
   i) **DidFillSurvey(int? accountID, int? surveyID)** - metoda sprawdzająca, czy użytkownik o danym nr ID wypełniał ankietę o danym nr ID;
   ii) **IsEmailCorrect(string email)** - walidacja adresu email podanego przy rejestracji konta (sprawdzanie, czy format jest poprawny (wykorzystuje metodę prywatną **IsValidEmail(string email)** oraz czy nie jest zajęty przez innego użytkownika);
   iii) **IsNicknameCorrect(string nick)** - sprawdzenie, czy nick ma odpowiednią ilość znaków oraz czy nie jest zajęty przez innego użytkownika;
   iv) **IsFollowed(int followerId, int followedId)** - sprawdzanie, czy podany w pierwszym parametrze użytkownik nie obserwuje użytkownika podanego w drugim parametrze;
   v) **hashPassword(string password)** - metoda do szyfrowania hasła (cel: bezpieczeństwo);

b) **AnswerRepository.cs**
   i) **GetAccountsVoters(int? answerID)** - zwraca listę kont, która głosowała na daną odpowiedź;
   ii) **GetQuantityOfVotes(int? answerID)** - zwraca liczbę głosów na daną odpowiedź;

c) **SurveyRepository.cs**
   i) **IsAnonymous(int? id)** - zwraca wartość true, gdy ankieta o podanym nr ID jest anonimowa, w przeciwnym wypadku false;

d) **QuestionRepository.cs:**
   i) **GetQuestionCategory(bool canAddOwnAnswer, bool isSingleChoice)** - na podstawie wartości boolowskich zwraca odpowiednią kategorię pytania o podanych parametrach (istnieją tylko 4 kategorie);

e) **UserSecurityRepository.cs:**
   i) **IsLoginFree(string login)** - walidacja loginu przy rejestracji konta;

3. **Szczególnie interesujące zagadnienia projektowe:**
   a) sposób poruszania się w konsoli przy użyciu klawiatury (enter do zatwierdzania, escape do powrotu do poszczególnych widoków oraz strzałki do poruszania się po menu). Wykorzystana do tego odpowiednie switch case'y:

Configuration.cs (poruszanie się w poziomie)

```
while (true)
{
      key = ConsoleKey.B;
      if (Console.KeyAvailable)
            key = Console.ReadKey(true).Key;


      switch (key)
      {
```

```
            case ConsoleKey.LeftArrow:
            Console.SetCursorPosition(positionX, Console.WindowHeight / 2);
            CurrentConsoleLineClear(positionX);

            for (int j = 2; j >= 0; j--)
            {
                if (j == 1)
                    Console.ForegroundColor = Color.Red;
                    else
                        Console.ForegroundColor = Color.White;

                    int z = (i - j) % quantityOfOptions;
                    if (z == (-1))
                Console.Write(listOptions[quantityOfOptions -
1].GetName());
                     else if (z == (-2))
                    {
                       Console.Write(listOptions[quantityOfOptions -
2].GetName());
                        i = quantityOfOptions;
                    }
                    else
                        Console.Write(listOptions[z].GetName());
            }
            i = (i - 1) % quantityOfOptions;

        Console.SetCursorPosition(positionX, Console.WindowHeight / 2 + 5);
            CurrentConsoleLineClear(positionX);

            break;

            case ConsoleKey.RightArrow:
            Console.SetCursorPosition(positionX, Console.WindowHeight / 2);
            CurrentConsoleLineClear(positionX);
            for (int j = 0; j < 3; j++)
            {
                 if (j == 1)
                    Console.ForegroundColor = Color.Red;
                else
                    Console.ForegroundColor = Color.White;
                 Console.Write(listOptions[(i + j) %
quantityOfOptions].GetName());
            }
            i = (i + 1) % quantityOfOptions;
        Console.SetCursorPosition(positionX, Console.WindowHeight / 2 + 5);
             CurrentConsoleLineClear(positionX);
             break;
```

```
            case ConsoleKey.Enter:
                ConsoleClearToArtAscii();
                listOptions[i].OptionFunction();
                break;

            case ConsoleKey.Escape:
                ConsoleClearToArtAscii();
                MainMenu(Options.GetMainOptions());
              break;
        }
    }
}
```

ChooseAccount.cs (przechodzenie pionowe)

```
while (true)
{
      foreach (DisplayedAccount displayedAccount in displayedAccounts)
      {
           Console.SetCursorPosition(positionX,
displayedAccount.accountPositionY);
           if (currentlySelectedAccount == displayedAccount.accountNumber)
                Console.ForegroundColor = Color.Red;
                else
                     Console.ForegroundColor = Color.White;
                int customSpaceBreakLength = 24 -
           displayedAccount.account.nickname.Length;
                Console.WriteLine(displayedAccount.account.nickname + new
string(' ', customSpaceBreakLength) + displayedAccount.numberOfFollowers +
                        bigSpaceBreak + spaceBreak +
displayedAccount.numberOfSurveysCreated);
      }
      key = ConsoleKey.B;
      if (Console.KeyAvailable)
            key = Console.ReadKey(true).Key;
      switch (key)
      {
            case ConsoleKey.UpArrow:
                currentlySelectedAccount--;
                if (currentlySelectedAccount == 0)
                      currentlySelectedAccount = lastAccountNumber;
                break;

             case ConsoleKey.DownArrow:
                  currentlySelectedAccount++;
                  if (currentlySelectedAccount == lastAccountNumber+1)
                        currentlySelectedAccount = 1;
                       break;
```

```
                case ConsoleKey.Enter:
                    Configuration.ConsoleClearToArtAscii();
                    PersonView.Show(account, displayedAccounts.Find(t =>
t.accountNumber == currentlySelectedAccount).account);
                        break;

                case ConsoleKey.Escape:
                    Configuration.ConsoleClearToArtAscii();
                    AfterSignIn.ComeBack(account, "Returned to main menu.");
                        break;
        }
}
```

b) przekazywanie konta jako parametru do widoków, aby mieć system, który pamięta, jaki użytkownik jest zalogowany i wyświetlanie rzeczy odpowiednio dla zalogowanego użytkownika (jeżeli jest autorem ankiet, to taki komunikat dostaje, jeżeli wybierze swoje konto, to otrzyma informacje, że to jest jego konto, itp);

4. **Instrukcja instalacji:**

Przygotowany został plik Setup.exe, który po naciśnięciu włącza typowy instalator aplikacji. Wybieramy lokalizację, gdzie należy zainstalować program i instalujemy. Ikona pojawi się na pulpicie o nazwie “Survey App”.

5. **Instrukcja użytkownika:**

Nowy użytkownik ma możliwość założenia własnego konta po podaniu swoich danych. Następnie ma do wyboru wiele opcji, m.in.: dodawanie własnych ankiet, przeglądanie ich wyników, obserwowanie innych użytkowników, wypełnianie ankiet, zmianę hasła jak również wylogowanie się. Istnieje również możliwość usunięcia konta.

6. **Wnioski:**

Mieliśmy możliwość poznać sposoby tworzenia aplikacji konsolowych w języku c# co było niezwykle zajmującym doświadczeniem. Ponadto poszerzyliśmy naszą wiedzę z zakresu Entity Framework, a także tworzenia instalatorów.

7. **Samoocena: 4.5**