НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра програмного забезпечення комп’ютерних систем

**КУРСОВА РОБОТА**

***з дисципліни "Основи програмування"***

Виконала: Балагуш Ольга Іванівна

Група: КП-01

Допущено до захисту

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2 семестр 2020/2021

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра програмного забезпечення комп’ютерних систем

|  |  |
| --- | --- |
| Узгоджено  Керівник роботи    \_\_\_\_\_\_\_\_\_\_/Гадиняк Р.А./ | ЗАХИЩЕНА "\_\_"\_\_\_\_\_\_\_\_\_2021р.  з оцінкою\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_/Гадиняк Р.А./ |

**Програмний додаток електронної соціальної мережі**

Виконавець роботи

Балагуш Ольга Іванівна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021р.

Зміст

[**Вступ**](#_cr2eoqpdh4mm) **4**

[**1. Аналіз предметної галузі**](#_b73pd3816f8p) **5**

[1.1. Аналіз вимог до функціональності програмних засобів](#_qz6vreit6wm5) 5

[**2. Аналіз мов програмування та технологій розроблення**](#_ewtylcuh8bvc) **7**

[2.1. Мова програмування С#](#_fh0j2unhluf2) 7

[2.2. Фреймворк .NET](#_ayd32pcv0jlc) 7

[2.3. База даних SQLite](#_hul0g052l08h) 7

[2.4. Бібліотека Terminal.Gui](#_6jqqk9eelxe) 7

[2.5. Бібліотека Microsoft.Data.Sqlite](#_28bxse9w7ext) 8

[2.6. Бібліотеки ScrottPlot і System.Drawing.Common](#_pt5i1swgedmp) 8

[**3. Опис розроблених програмних засобів**](#_kbi4mqs5pjyt) **8**

[3.1. Загальна структура системи](#_nlu4ckytvy2k) 8

[3.2. Структура бази даних](#_ny4p6qq0v8a9) 10

[3.3. Модулі репозитаріїв даних](#_p5yku72ilzdo) 10

[3.4. Модуль експорту й імпорту](#_bl0vwivvre05) 15

[3.5. Модуль авторизації](#_s1zshrw4ghmq) 16

[3.6. Модуль генерація звіту](#_44vniah0l8eo) 16

[3.7. Бібліотека AccessDataLib](#_fautz727jfd5) 16

[**4. Аналіз розроблених програмних засобів**](#_g0l8gqqu407h) **17**

[4.1. Особливості реалізації](#_5ajysoaxe2s5) 17

[4.2. Інтерфейс користувача](#_c8eyxdz8sp3u) 17

[**Висновки**](#_6dbqg2jsslr5) **24**

# Вступ

В процесі розвитку комунікацій часто виникає необхідність ділитись інформацією з великою масою людей.

Проблемами поширення інформації на велику аудиторію є відсутність платформ. Виникає необхідність створення соціальної мережі, яка б дала можливість публікувати користувачам цієї платформи, пости від свого імені та коментувати інші. Саме тому пропонується розробка програмного додатку електронної соціальної мережі.

Створення системи, що дає можливість публікувати пости та коментувати їх, є актуальною задачею. Оскільки соціальна мережа дає можливість за секунди надіслати важливу інформацію з життя користувача, та швидко поширювати інформацію на велику аудиторію.

Даний проект присвячено розробленню програмного додатку електронної соціальної мережі.

# 1. Аналіз предметної галузі

## 1.1. Аналіз вимог до функціональності програмних засобів

В процесі аналізу вимог до системи було виділено такі функціональні вимоги:

1. Наявний консольний інтерфейс користувача. Команди використані відповідно до їхнього основного призначення.

a. Наявно декілька станів інтерфейсу користувача.

b. Дотримується єдиний стиль для всіх команд

1. Наявні стани для створення і редагування даних.

a. Можна видаляти дані (з підтвердженням цієї дії).

b. Наявний стан редагування даних має вже заповнені поля із попередніми значеннями. Є можливість відмінити оновлення сутності.

1. Використання списків та таблиць для показу колекцій даних.

a. Дані в таблиці не є лише відображенням однієї таблиці БД, а містять склеєні чи змінені дані.

b. Дані у списках та таблицях пагінуються, доступний пошук та фільтрація їх записів.

1. Всі вхідні дані від користувача та з файлів перевіряються, помилки обробляються, користувачу відображається інформація про помилки.
2. Програма має можливість завантаження зображень з файлової системи, їх збереження у сховище даних та їх показ у інтерфейсі користувача.
3. Програма має можливість експорту-імпорту даних у форматі даних XML. Є можливість зберігання даних у файлах, та передача даних до інших процесів через протокол TCP.
4. Програма взаємодіє з реляційною базою даних. SQL запити до бази даних не конкатенуються, а їх виконання винесено у спеціальний окремий модуль (вид репозиторія).

a. База даних містить мінімум 3 таблиці. Таблиці в БД приведені до 1-ї та 2-ї нормальних форм.

b. Між таблицями бази даних є one-to-many та/або many-to-many зв'язки

1. Наявна реєстрація користувачів у системі. Паролі користувачів не зберігаються у відкритому виді, а хешуються обраним алгоритмом хешування рядків.

a. Наявний стан аутентифікації. Пароль користувача при введені прихований. Цей стан має сповіщення про некоректність введених користувачем даних.

b. Програма авторизує доступу до даних по користувачах. У базі є дані, що належать конкретному користувачу (наприклад, авторство сутностей).

1. Система дозволяє генерувати зображення та файлові звіти на основі даних, що зберігаються у БД.
2. Частина проекту винесена у власні бібліотеки класів.
3. Наявний проект з модульними тестами функціональності системи.

# 2. Аналіз мов програмування та технологій розроблення

## 2.1. Мова програмування С#

Під час створення програми були використані такі можливості C# як: об’єктно-орієнтованої розробки, обробки винятків, використання подій.

## 2.2. Фреймворк .NET

Можливості, платформи, що були застосовані: забезпечення об’єктно-орієнтованого середовища, для зберігання і виконання коду, середовище в якому гарантується безпечне виконання коду, забезпечення принципів розробки для різних типів додатків.

## 2.3. База даних SQLite

Базою даних SQLite були використані такі можливості, як можливості створення запитів до БД та роботи з реляційними базами даних.

## 2.4. Бібліотека Terminal.Gui

***Terminal.Gui*** - це бібліотека, що призначена для створення консольних програм на основі C#. За допомогою цієї бібліотеки створений подійно-орієнтованого графічного інтерфейсу за допомогою графічного редактора вікон, також використана можливість автоматичного створення подій, використання стандартних діалогових вікон (вікно повідомлення, вікно вибору файлу, директорії і тд) та використання стандартних елементів керування.

## 2.5. Бібліотека Microsoft.Data.Sqlite

Бібліотека ***Microsoft.Data.Sqlite*** дає можливість реалізувати абстракції для підключень, команд, модулів читання даних і інших елементів бази даних SQLite.

## 2.6. Бібліотеки ScrottPlot і System.Drawing.Common

Бібліотеки застосовані для роботи з зображеннями. Так, за допомогою бібліотек у програмі застосовується генерація графіку та заміна зображення у файлі docx під час генерації звіту.

# 3. Опис розроблених програмних засобів

## 3.1. Загальна структура системи

Програмні засоби реалізовані у вигляді десктопного додатку з графічним інтерфейсом користувача. Структурна схема системи зображена на рис.



Рис. Структурна схема системи

Проект розбитий на такі модулі:

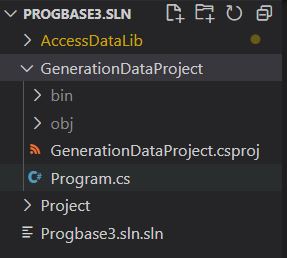
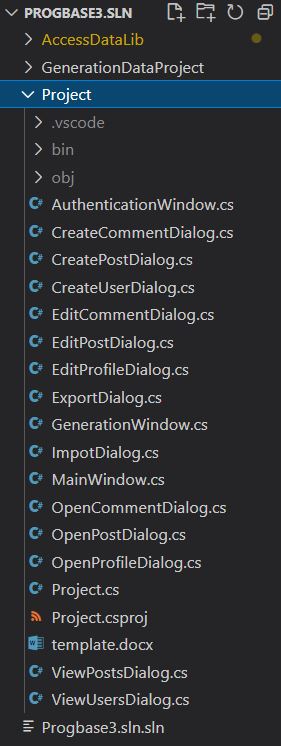
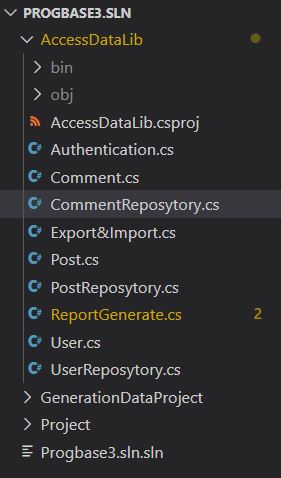


Рис. Модулі проекту

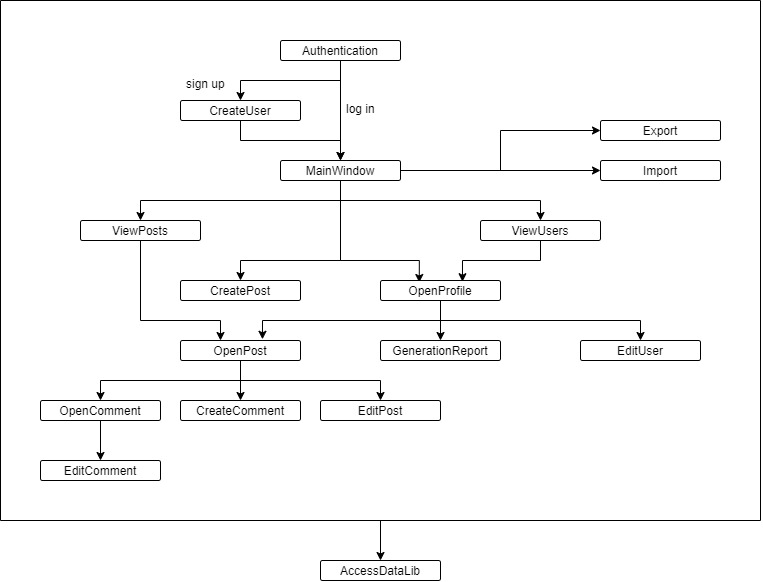


Рис. Діаграма залежностей модулів IDE

## 3.2. Структура бази даних

На рис. представлена ERD-діаграма структури бази даних системи.

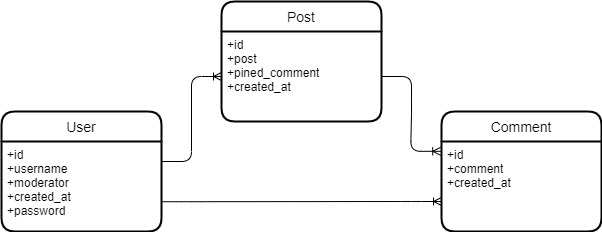


Рис. Структура бази даних

БД має 6 таблиць, 3 з яких таблиці сутностей: users, posts, comments, та ще 3 таблиці зв'язків ***one2many***: user\_comment, user\_post, post\_comment.



Рис. Таблиці БД

## 3.3. Модулі репозитаріїв даних

У модулях репозиторіїв (*UserRepository.cs, PostRepository.cs, CommentRepository.cs*) створенні методи виконання базових CRUD операцій над об’єктами відповідних сутностей, а також додаткові методи для знаходження зв'язків, створення пагінованого списку та генерації звіту.

Для прикладу *UserRepository*:

|  |
| --- |
| **UserRepository.сs** |
| using System;  using System.Collections.Generic;  using Microsoft.Data.Sqlite;  namespace AccessDataLib  {  public class UserRepository  {  private SqliteConnection connection;  private int numberOfElementsOnPage = 5;  public UserRepository(SqliteConnection connection)  {  this.connection = connection;  }  // методи для знаходження коментарів та постів юзера  public List<Comment> UserComments(long userID)  {  connection.Open();    SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT user\_comment.id\_user, comments.id, comments.pinned, comments.comment, comments.created\_at  FROM user\_comment, comments WHERE user\_comment.id\_user = $idUser  AND comments.id = user\_comment.id\_comment";  command.Parameters.AddWithValue("$idUser", userID);  SqliteDataReader reader = command.ExecuteReader();  List<Comment> comments = new List<Comment>();  while (reader.Read())  {  string commenttext = reader.GetString(2);  string createdAt = reader.GetString(4);  Comment newcomment = new Comment(commenttext, createdAt);  newcomment.id = int.Parse(reader.GetString(1));  newcomment.pinned = reader.GetString(3);  comments.Add(newcomment);  }  reader.Close();  connection.Close();  User user = GetByID(userID);  user.comments = comments;  return comments;  }  public List<Post> UserPosts(long userID)  {  connection.Open();    SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT user\_post.id\_user, posts.id, posts.post, posts.created\_at  FROM user\_post, posts WHERE user\_post.id\_user = $idUser  AND posts.id = user\_post.id\_post";  command.Parameters.AddWithValue("$idUser", userID);  SqliteDataReader reader = command.ExecuteReader();  List<Post> posts = new List<Post>();  while (reader.Read())  {  string postText = reader.GetString(2);  string createdAt = reader.GetString(3);  Post newpost = new Post(postText, createdAt);  newpost.id = int.Parse(reader.GetString(1));  posts.Add(newpost);  }  reader.Close();  connection.Close();  User user = GetByID(userID);  user.posts = posts;  return posts;  }  // методи для генерації звіту  public List<Post> NumberOfPostsFromTo(long userId, DateTime startDate, DateTime endDate)  {  List<Post> currentposts = this.UserPosts(userId);  List<Post> posts = new List<Post>();  foreach (Post item in currentposts)  {  if(item.createdAt >= startDate && item.createdAt < endDate)  {  posts.Add(item);  }  }  return posts;  }  public List<Comment> NumberOfCommentsFromTo(long userId, DateTime startDate, DateTime endDate)  {  List<Comment> currentcomments = this.UserComments(userId);  List<Comment> comments = new List<Comment>();  foreach (Comment item in currentcomments)  {  if(item.createdAt >= startDate && item.createdAt < endDate)  {  comments.Add(item);  }  }  return comments;  }  // методи для пагінованого списку  private long GetCount()  {  connection.Open();    SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT COUNT(\*) FROM users";    long count = (long)command.ExecuteScalar();  return count;  }  public int NumberOfPages(string searchValue, int numberOfElementsOnPage)  {  this.numberOfElementsOnPage = numberOfElementsOnPage;  int count = 0;  if(searchValue != "")  {  List<User> allusers = this.GetSearchValue(searchValue, numberOfElementsOnPage);  foreach (User item in allusers)  {  count++;  }  }  else  {  count = (int)GetCount();  }  if(count % numberOfElementsOnPage == 0)  {  return count / numberOfElementsOnPage;  }  else  {  return count / numberOfElementsOnPage + 1;  }  }  public List<User> UserssOnPage(int numberOfPage)  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT \* FROM users LIMIT $numberOfElementsOnPage OFFSET $fromElement";  command.Parameters.AddWithValue("$numberOfElementsOnPage", this.numberOfElementsOnPage);  command.Parameters.AddWithValue("$fromElement", (numberOfPage - 1) \* this.numberOfElementsOnPage);  SqliteDataReader reader = command.ExecuteReader();  List<User> users = new List<User>();  while(reader.Read())  {  string username = reader.GetString(1);  // int moderator = int.Parse(reader.GetString(2));  string password = reader.GetString(4);  DateTime createdAt = DateTime.Parse(reader.GetString(3));  User user = new User(username, password, createdAt.ToString());  user.id = int.Parse(reader.GetString(0));  if(int.Parse(reader.GetString(2)) == 0)  {  user.moderator = false;  }  else  {  user.moderator = true;  }  users.Add(user);  }  reader.Close();  connection.Close();  return users;  }  public List<User> GetSearchPage(string searchValue, int page, int numberOfElementsOnPage)  {  if(searchValue == "")  {  return this.UserssOnPage(page);  }  return GetSearchValue(searchValue, numberOfElementsOnPage);  }  private List<User> GetSearchValue(string searchValue, int numberOfElementsOnPage)  {  List<User> concerts = new List<User>();  for(int i = 1; i <= NumberOfPages("", numberOfElementsOnPage); i++)  {  List<User> currentConcerts = UserssOnPage(i);  foreach (User item in currentConcerts)  {  if(item.username.Contains(searchValue))  {  concerts.Add(item);  }  }  }  return concerts;  }  // знаходження юзера за ім'ям  public User GetByUsername(string username)  {  connection.Open();    SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT \* FROM users WHERE username = $username";  command.Parameters.AddWithValue("$username", username);    SqliteDataReader reader = command.ExecuteReader();  while (reader.Read())  {  string createdAt = reader.GetString(3);  string password = reader.GetString(4);  User user = new User(username, password, createdAt);  user.id = int.Parse(reader.GetString(0));  if(int.Parse(reader.GetString(2)) == 0)  {  user.moderator = false;  }  else  {  user.moderator = true;  }  reader.Close();    connection.Close();  return user;  }  reader.Close();    connection.Close();  return null;  }  //CRUD  public long Insert(User user)  {  connection.Open();    SqliteCommand command = connection.CreateCommand();  command.CommandText =  @"  INSERT INTO users (username, moderator, created\_at, password)  VALUES ($username, $moderator, $createdAt, $password);    SELECT last\_insert\_rowid();  ";  command.Parameters.AddWithValue("$username", user.username);  if(user.moderator == true)  {  command.Parameters.AddWithValue("$moderator", 1);  }  else  {  command.Parameters.AddWithValue("$moderator", 0);  }  command.Parameters.AddWithValue("$createdAt", user.createdAt);  command.Parameters.AddWithValue("$password", user.password);    long newId = (long)command.ExecuteScalar();  user.id = newId;    connection.Close();  return newId;  }  public User GetByID(long userID)  {  connection.Open();    SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT \* FROM users WHERE id = $id";  command.Parameters.AddWithValue("$id", userID);    SqliteDataReader reader = command.ExecuteReader();  while (reader.Read())  {  string username = reader.GetString(1);  int moderator = int.Parse(reader.GetString(2));  string createdAt = reader.GetString(3);  string password = reader.GetString(4);  User user = new User(username, password, createdAt);  user.id = int.Parse(reader.GetString(0));  if(int.Parse(reader.GetString(2)) == 0)  {  user.moderator = false;  }  else  {  user.moderator = true;  }  reader.Close();    connection.Close();  return user;  }  reader.Close();    connection.Close();  return null;  }  public bool Update(long userID, User newuser)  {  connection.Open();    SqliteCommand command = connection.CreateCommand();  command.CommandText = @"UPDATE users SET username = $username, moderator = $moderator WHERE id = $id";  command.Parameters.AddWithValue("$id", userID);  command.Parameters.AddWithValue("$username", newuser.username);  if(newuser.moderator == true)  {  command.Parameters.AddWithValue("$moderator", 1);  }  else  {  command.Parameters.AddWithValue("$moderator", 0);  }    int nChanged = command.ExecuteNonQuery();    if (nChanged == 0)  {  connection.Close();  return false;  }  connection.Close();  return true;  }  public bool Delete(long userID)  {  connection.Open();    SqliteCommand command = connection.CreateCommand();  command.CommandText = @"DELETE FROM users WHERE id = $id";  command.Parameters.AddWithValue("$id", userID);    int nChanged = command.ExecuteNonQuery();    if (nChanged == 0)  {  connection.Close();  return false;  }  connection.Close();  return true;  }  }  } |

## 3.4. Модуль експорту й імпорту

Модуль ***експорту*** дозволяє експортувати всі відфільтровані по співпадінню за заданим рядком пости з усіма їх коментарями у XML-файл. Користувач має можливість задати рядок та назву файлу, в який відбудеться експорт.

Модуль ***імпорту*** дозволяє імпортувати експортовані пости і коментарі з даними.

## 3.5. Модуль авторизації

Модуль ***авторизації*** дозволяє зареєструватись чи увійти користувачу, також виконує при реєстрації хешує пароль та перевіряє на унікальність введене ім’я (якщо ім'я вже існує відхиляє запит на створення нового користувача), при вході виконує перевірку ім’я користувача та відповідного захешованого пароля.

## 3.6. Модуль генерація звіту

Модуль ***генерація звіту*** дозволяє користувачу задати рік, за який генерувати статистику, та файл, у який її генерувати. Також у модулі генерується графік кількості постів та коментарів користувача по місяцях, з інформацією про пости і коментарі протягом часового проміжку (дата початку, дата закінчення, кількість постів протягом проміжку, кількість коментарів протягом проміжку, пост з найбільшою кількістю коментарів в цей період, зображення з графіком).

Звіт генерується за допомогою розархівування шаблону docx та заміни даних.

## 3.7. Бібліотека AccessDataLib

Бібліотека ***AccessDataLib*** містить у собі модулі сутностей, репозиторії цих сутностей, модуль авторизації, генерації звіту та модуль експорту та імпорту. Бібліотека дозволяє працювати з даними у БД, експортувати сутності з бази даних та імпортувати їх, генерувати статистику постів і коментарів користувача та графік, а також перевіряти користувача під час авторизації та хешувати паролі.

# 4. Аналіз розроблених програмних засобів

## 4.1. Особливості реалізації

***Короткий опис***:

***особливості авторизації користувачів*** – користувач отримує доступ до програми лише після успішної авторизації.

***ролі користувачів*** – користувач може бути “модератор” або “користувач”. При реєстрації автоматично встановлюється роль “користувач”. Змінювати ролі інших користувачів може лише “модератор”. Також лише модератори мають право змінювати, видаляти та додавати сутності до бази.

***спосіб зберігання паролів*** – паролі зберігаються в базі у вигляді хешованого рядка за допомогою алгоритму хешування *SHA256*.

***формати даних, серіалізація*** - імпорт відбувається з файлу з форматом даних *XML*. При експорті інформація з бази серіалізується у *XML*-файл.

***інші спеціальні можливості*** – користувач має можливість згенерувати звіт на основі інформації користувача про пости, створені користувачем, та коментарі, під постом.

## 4.2. Інтерфейс користувача

Для показу даних інтерфейс користувача складається з:

1. Вікно авторизації
2. Головне вікно
3. Діалогове вікно створення користувача
4. Діалогове вікно створення поста
5. Діалогове вікно створення коментаря
6. Діалогове вікно перегляду профіля користувача
7. Діалогове вікно перегляду поста
8. Діалогове вікно перегляду коментаря
9. Діалогове вікно редагування профіля користувача
10. Діалогове вікно редагування поста
11. Діалогове вікно редагування коментаря
12. Діалогове вікно перегляду пагінованого списку усіх користувачів
13. Діалогове вікно перегляду пагінованого списку усіх постів
14. Діалогове вікно експорту
15. Діалогове вікно імпорту
16. Діалогове вікно генерації звіту

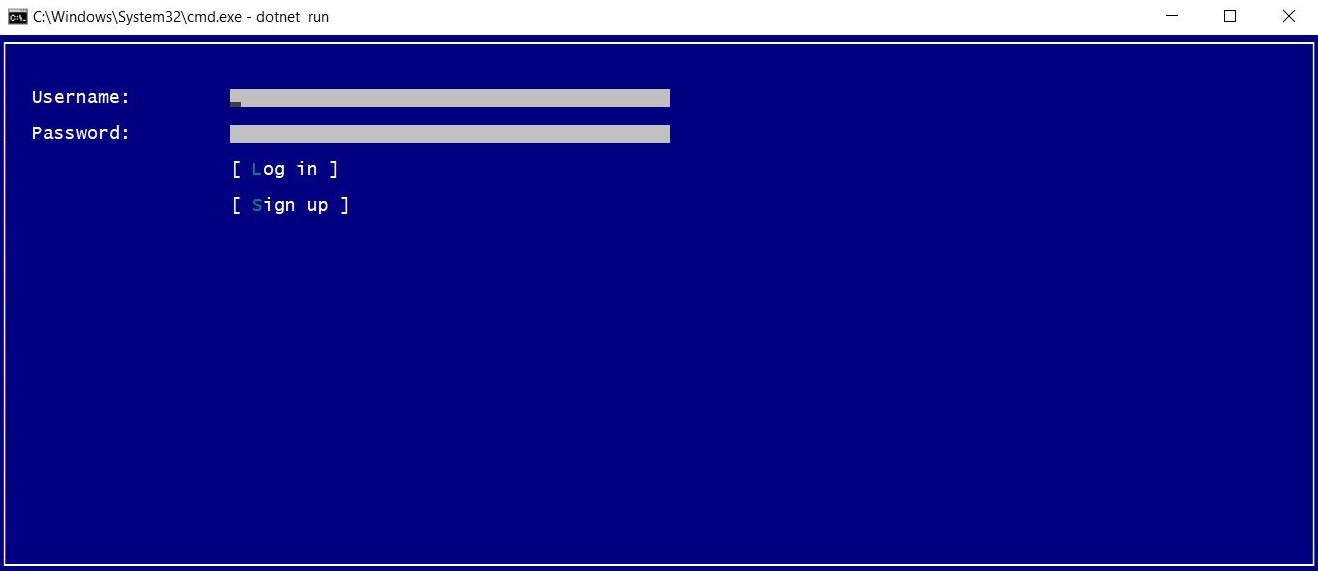


Рис. Вікно авторизації



Рис. Головне вікно



Рис. Діалогове вікно створення користувача

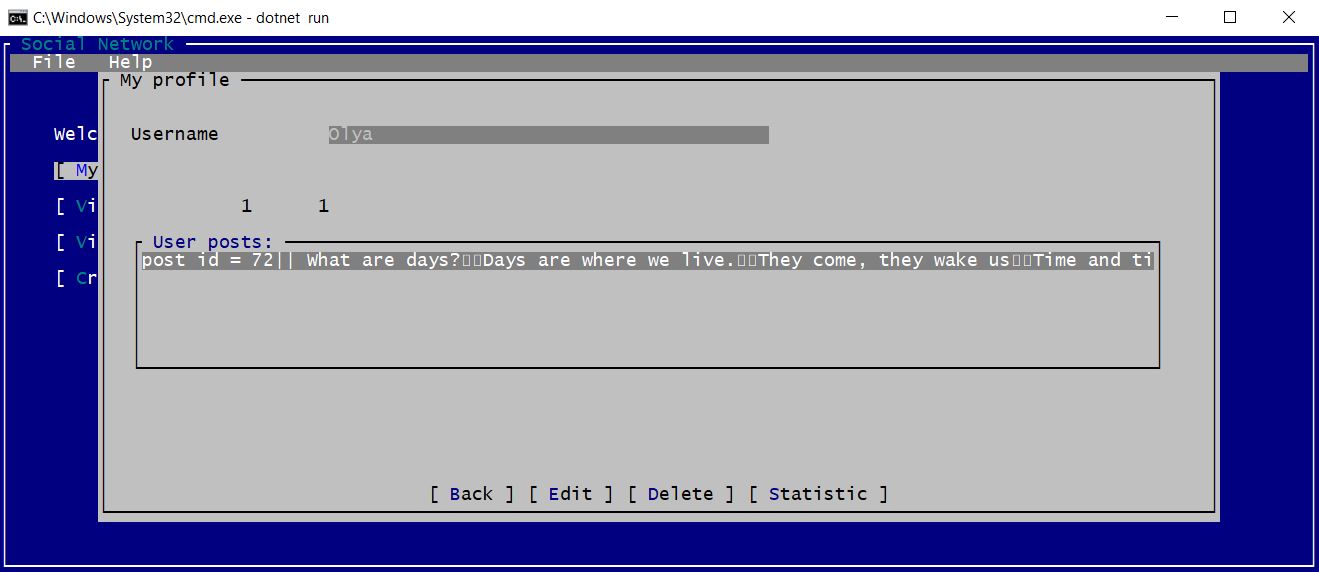


Рис. Діалогове вікно перегляду профілю користувача



Рис. Діалогове вікно редагування профілю користувача

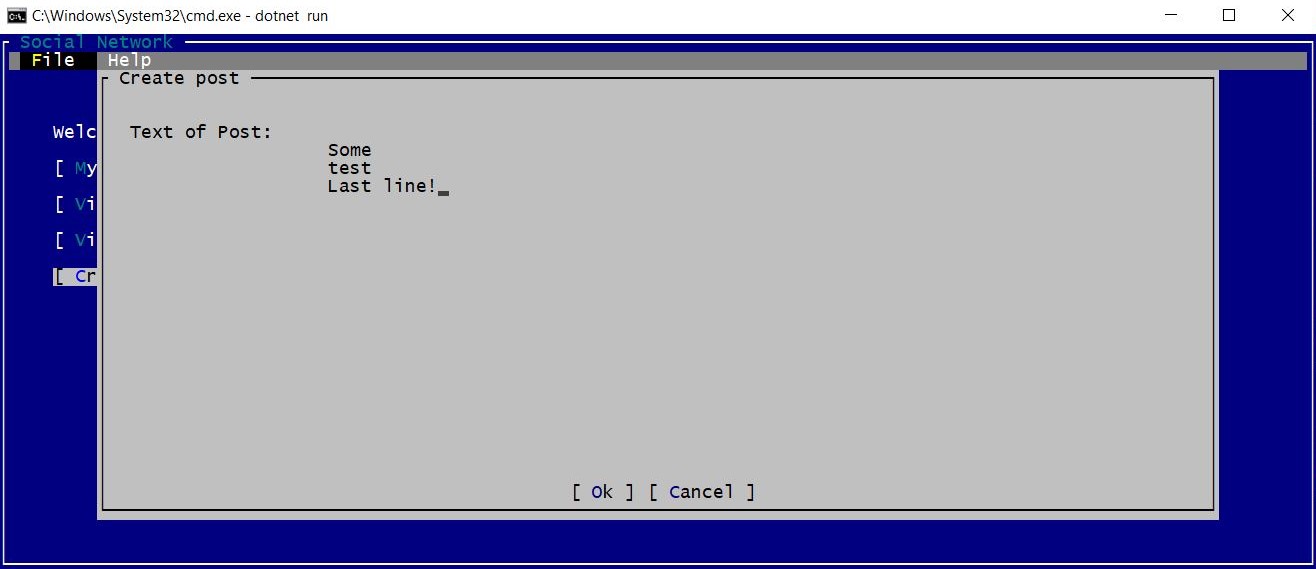


Рис. Діалогове вікно створення поста

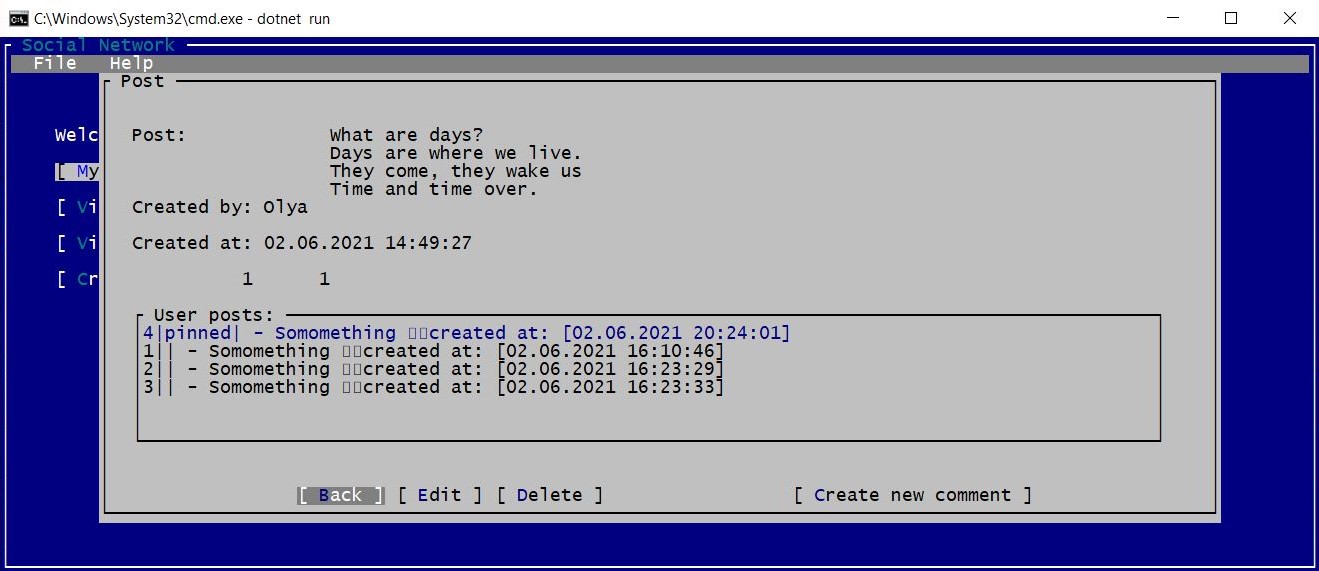


Рис. Діалогове вікно перегляду поста

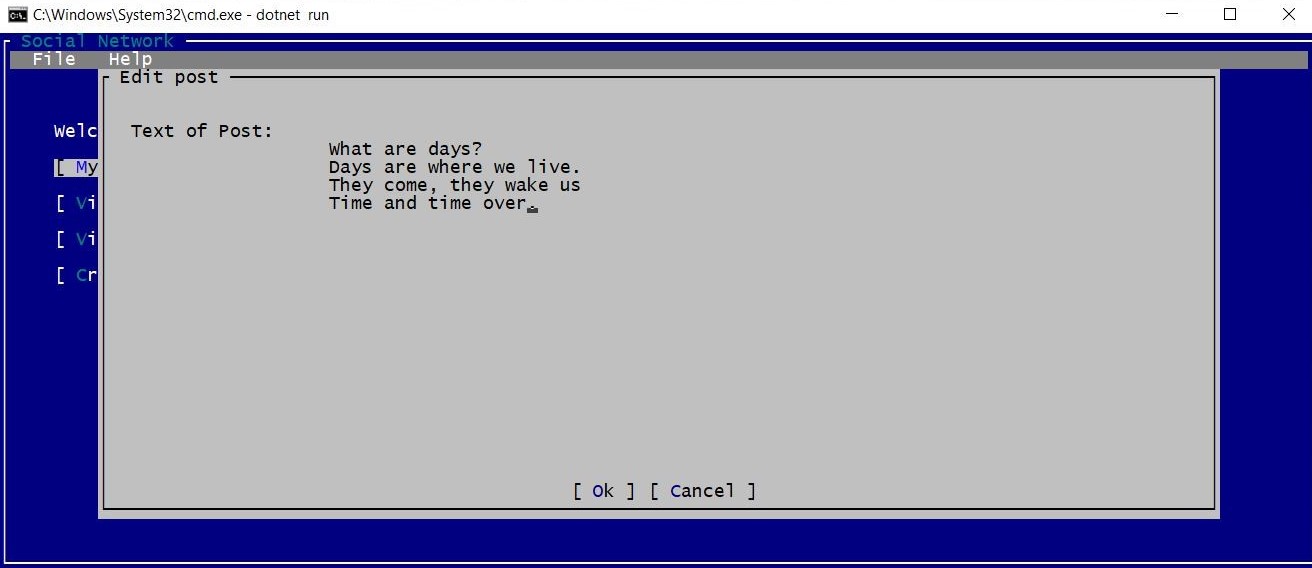


Рис. Діалогове вікно редагування поста



Рис. Діалогове вікно створення коментаря

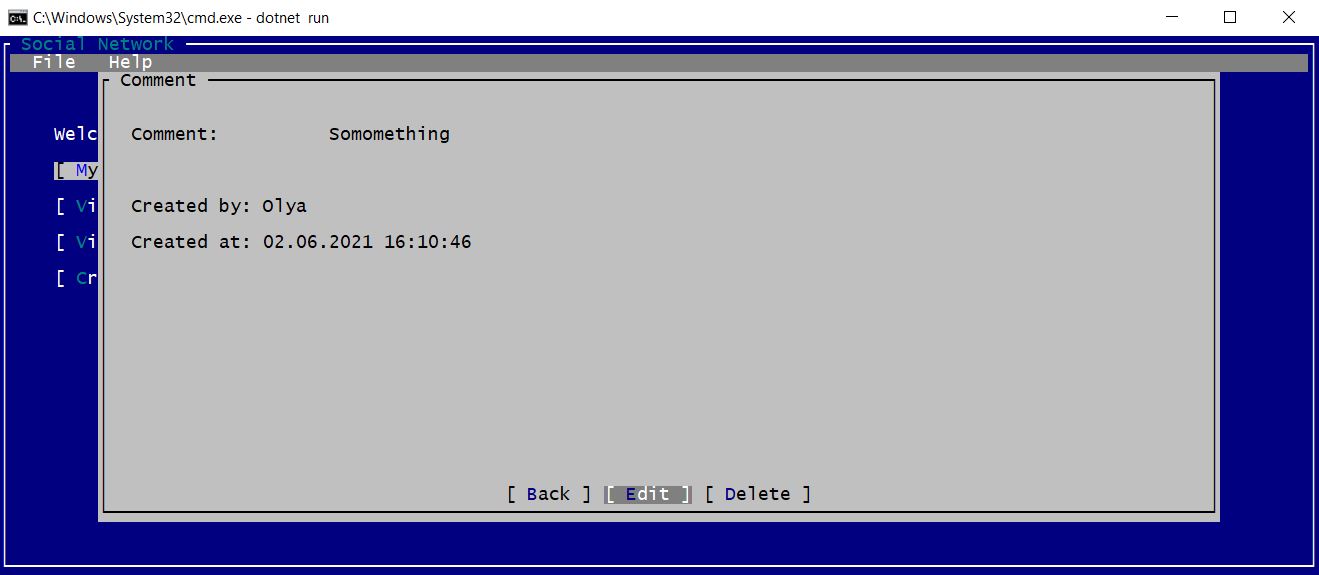


Рис. Діалогове вікно перегляду коментаря

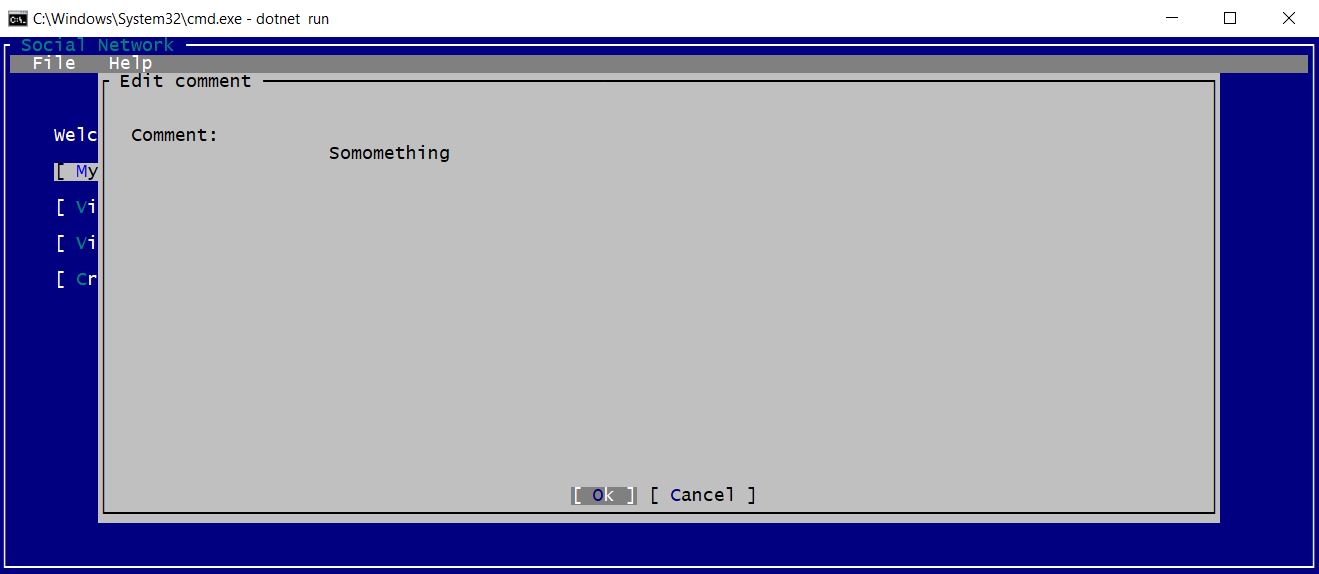


Рис. Діалогове вікно редагування коментаря

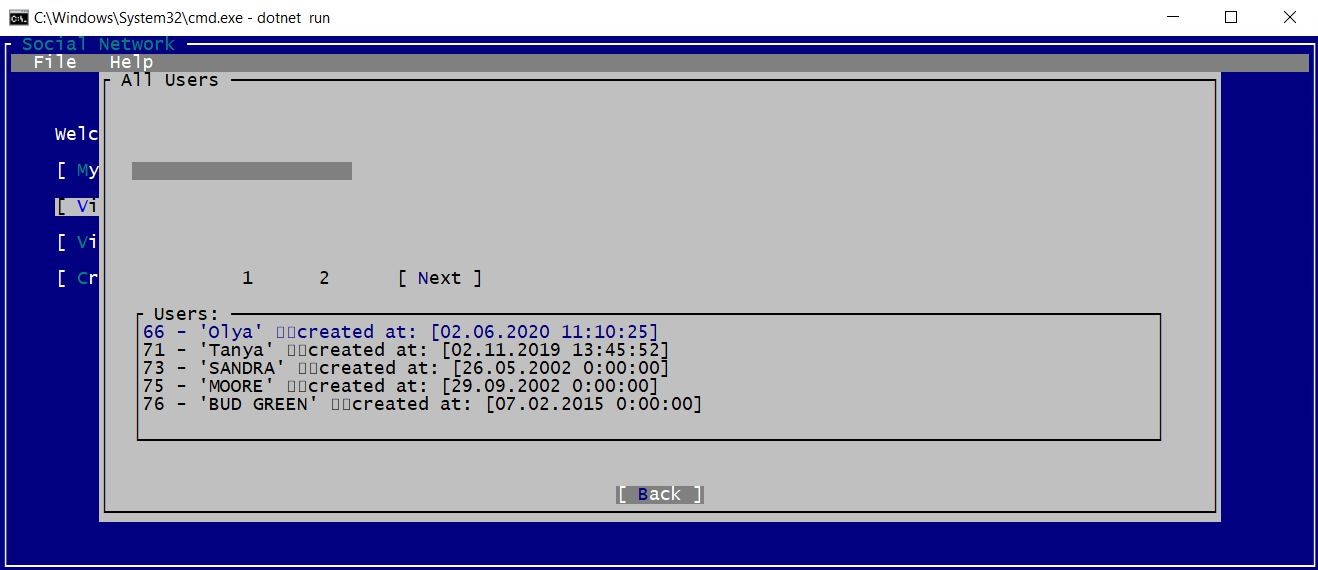


Рис. Діалогове вікно списку усіх користувачів

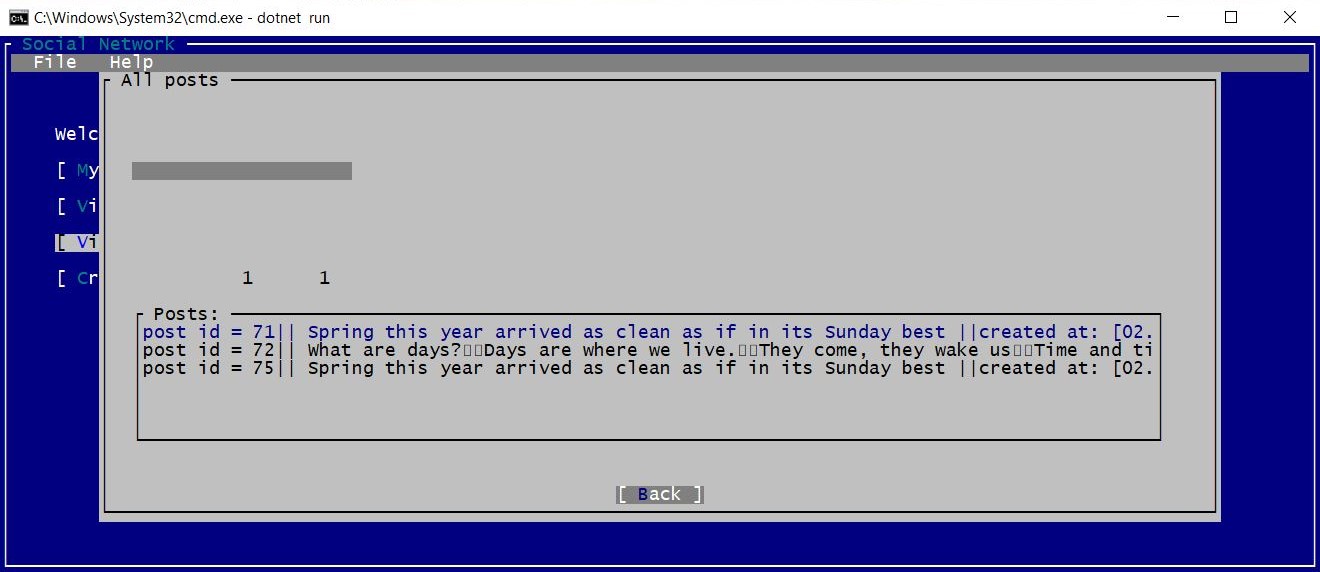


Рис. Діалогове вікно списку усіх постів



Рис. Діалогове вікно експорту

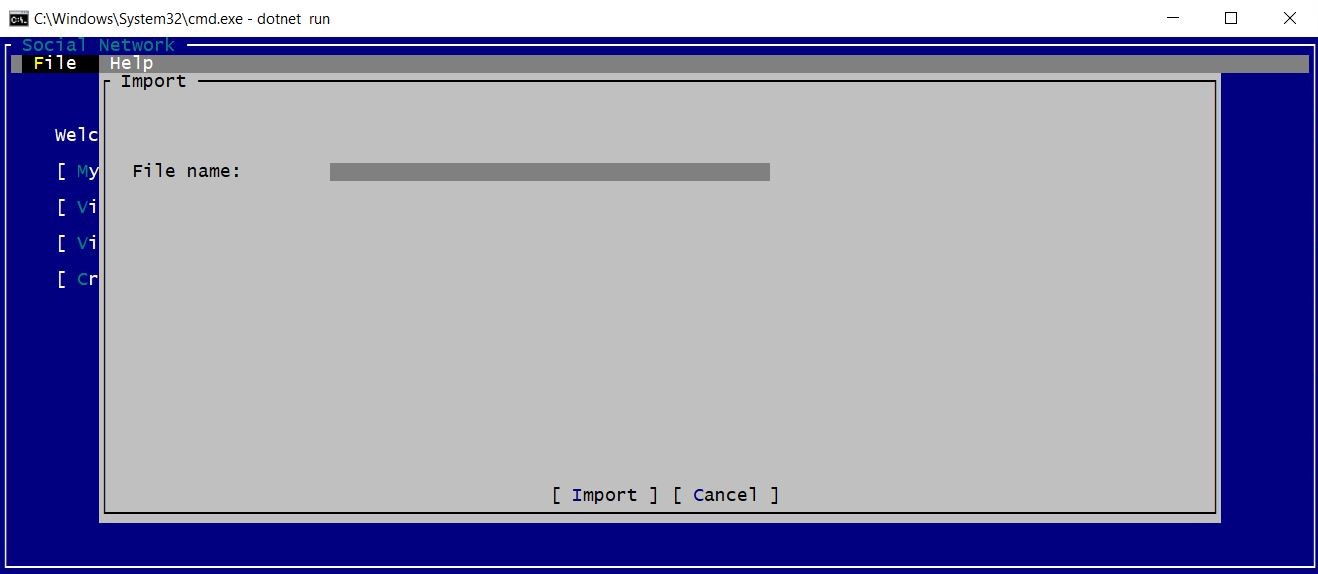


Рис. Діалогове вікно імпорту

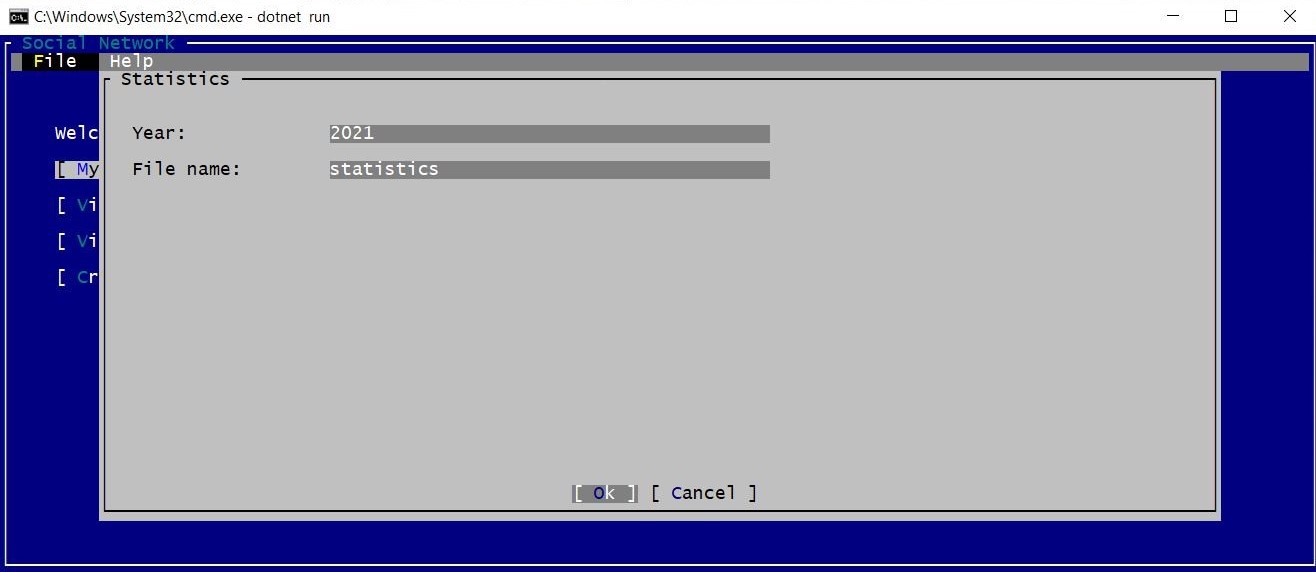


Рис. Діалогове вікно генерації звіту

# Висновок

Метою даного курсового проекту було розроблення десктопного додатку з подійно-орієнтованим інтерфейсом користувача - програмного додатку електронної соціальної мережі.

Розроблена система:

* забезпечує авторизований доступ до соціальної мережі;
* дозволяє користувачеві створювати сутності постів та коментарів;
* дає можливість редагувати та робити інші модифікації сутностей;

Особливу увагу під час розроблення даного програмного продукту було приділено розробленню зручності у користуванні для користувача.

Розробка виконана у повному обсязі, всі вимоги враховані, продукт протестовано.

Використання розробленого додатку дозволить користувачам вести активне соціальне життя та ділитись інформацією з іншими користувачами платформи, а також використовувати інші доступні функції додатку.