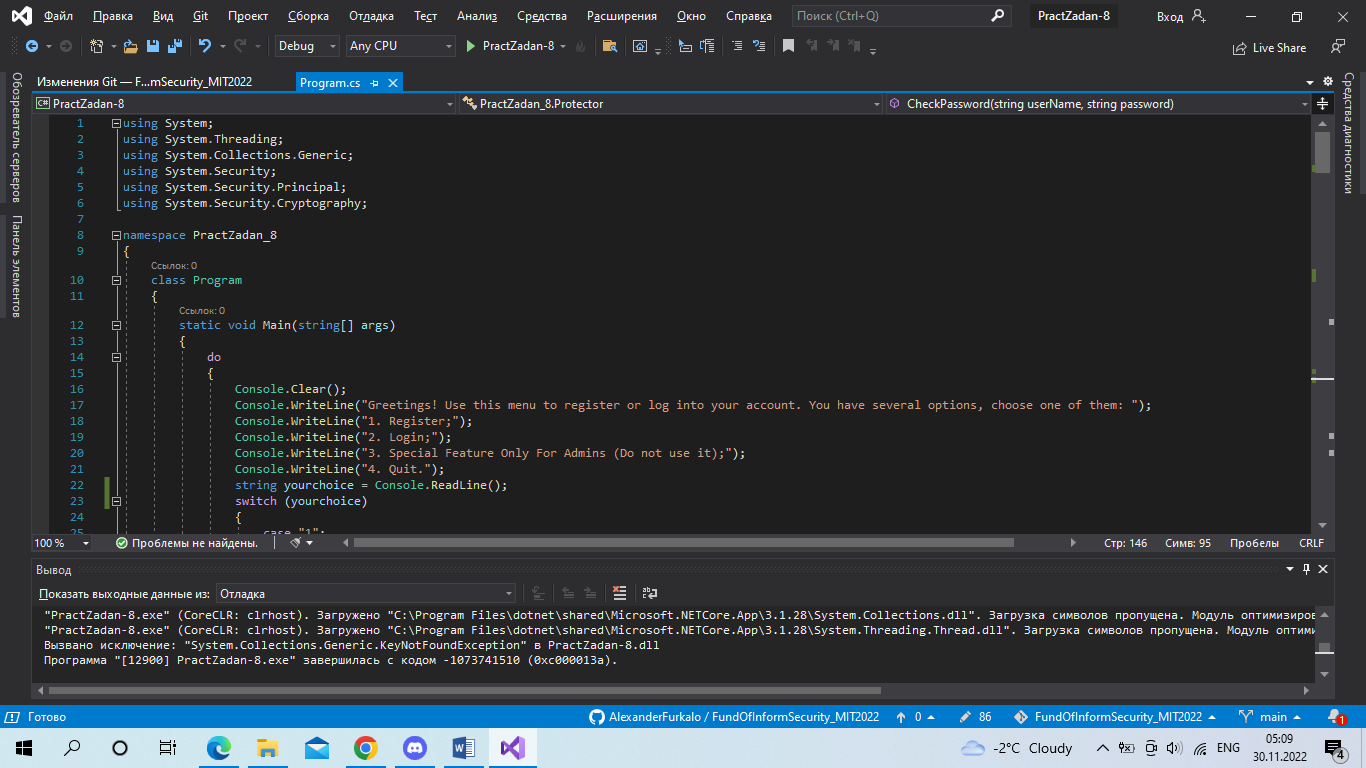
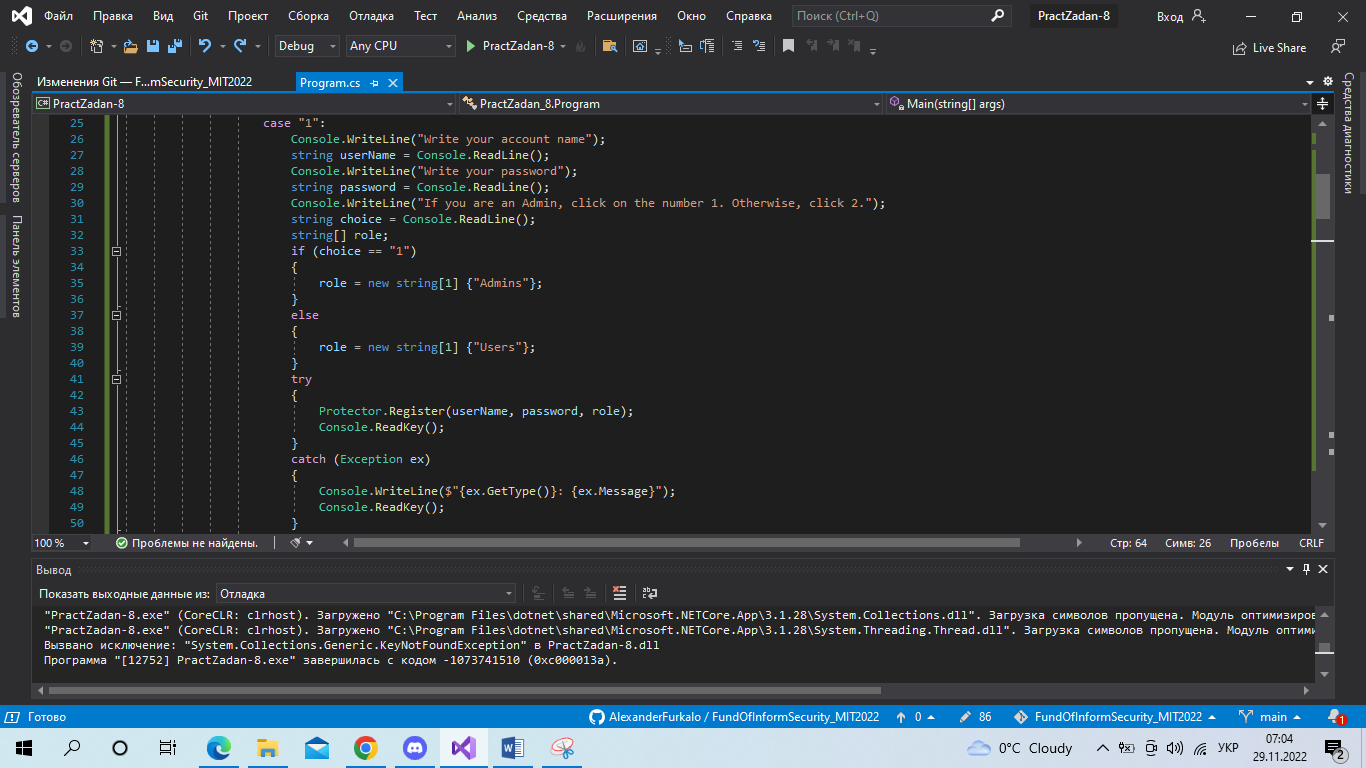
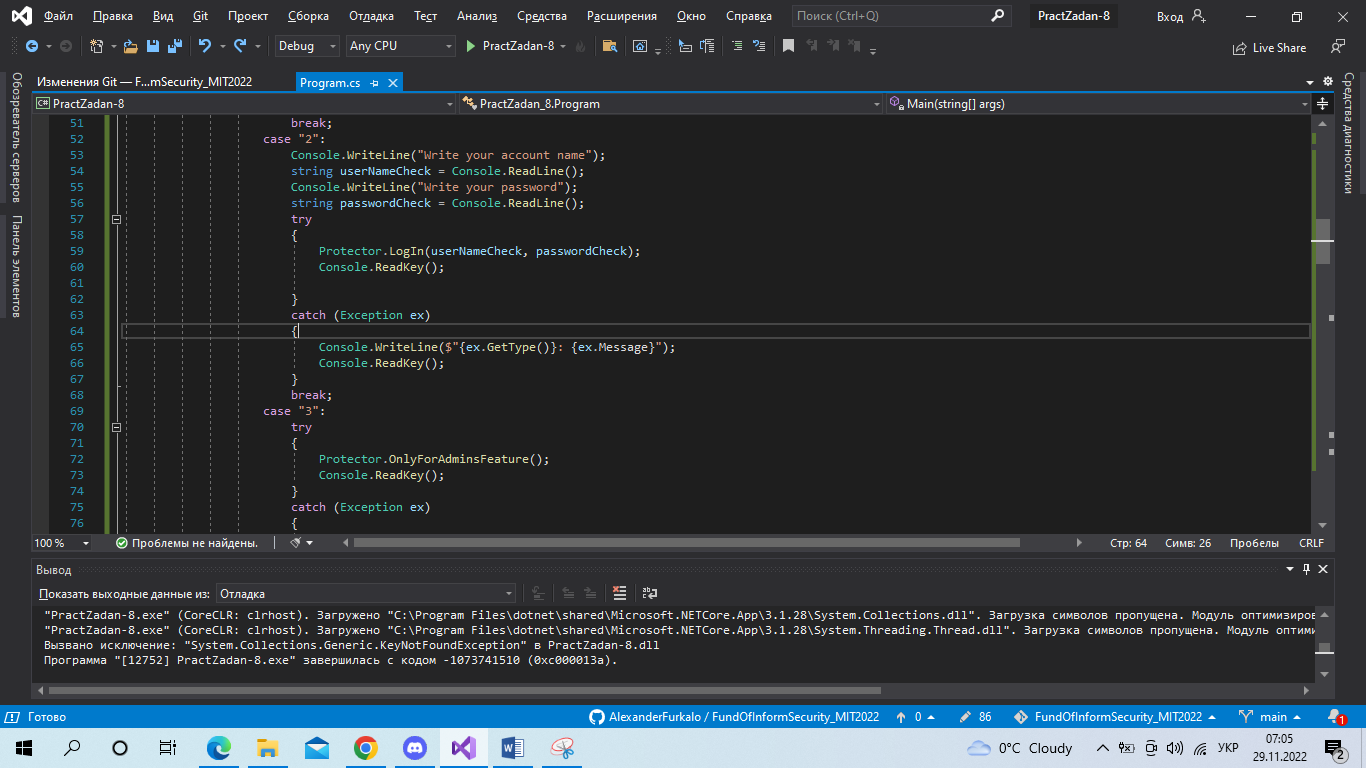
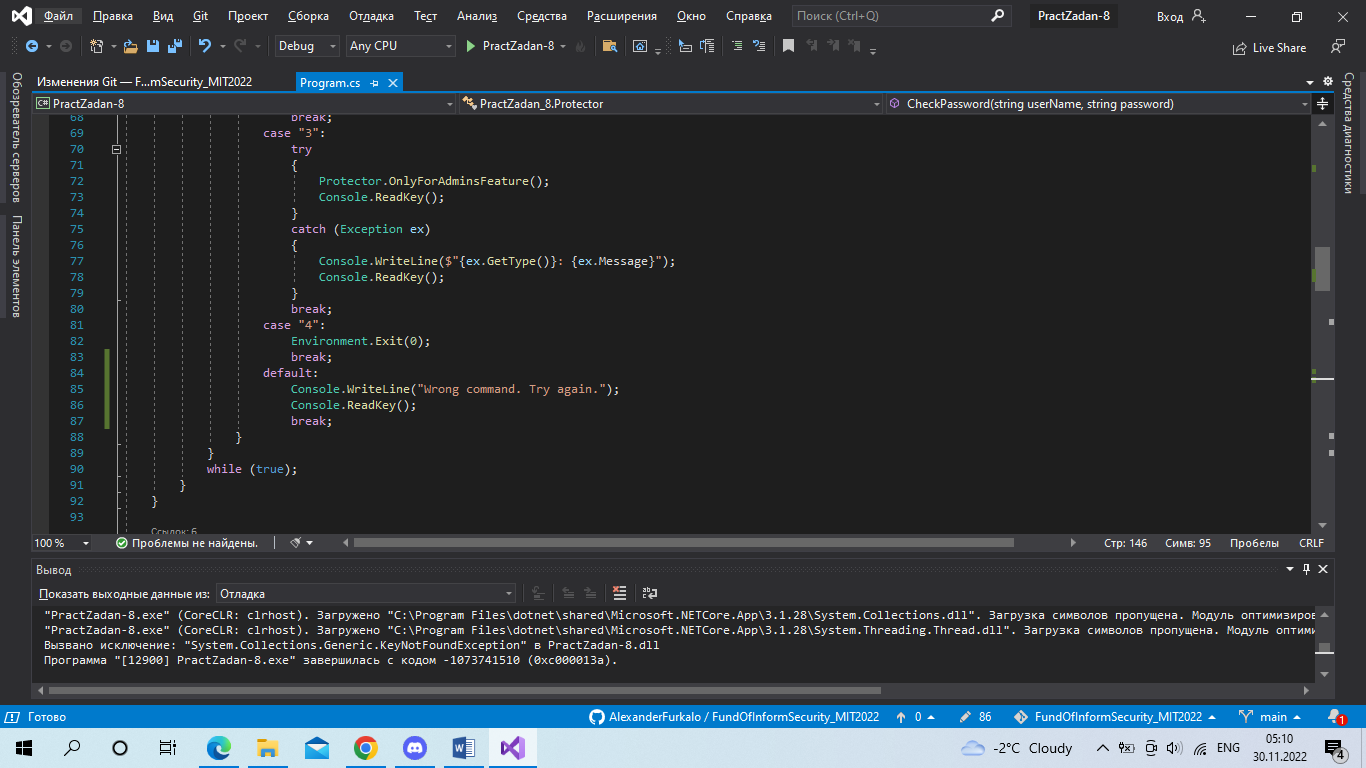
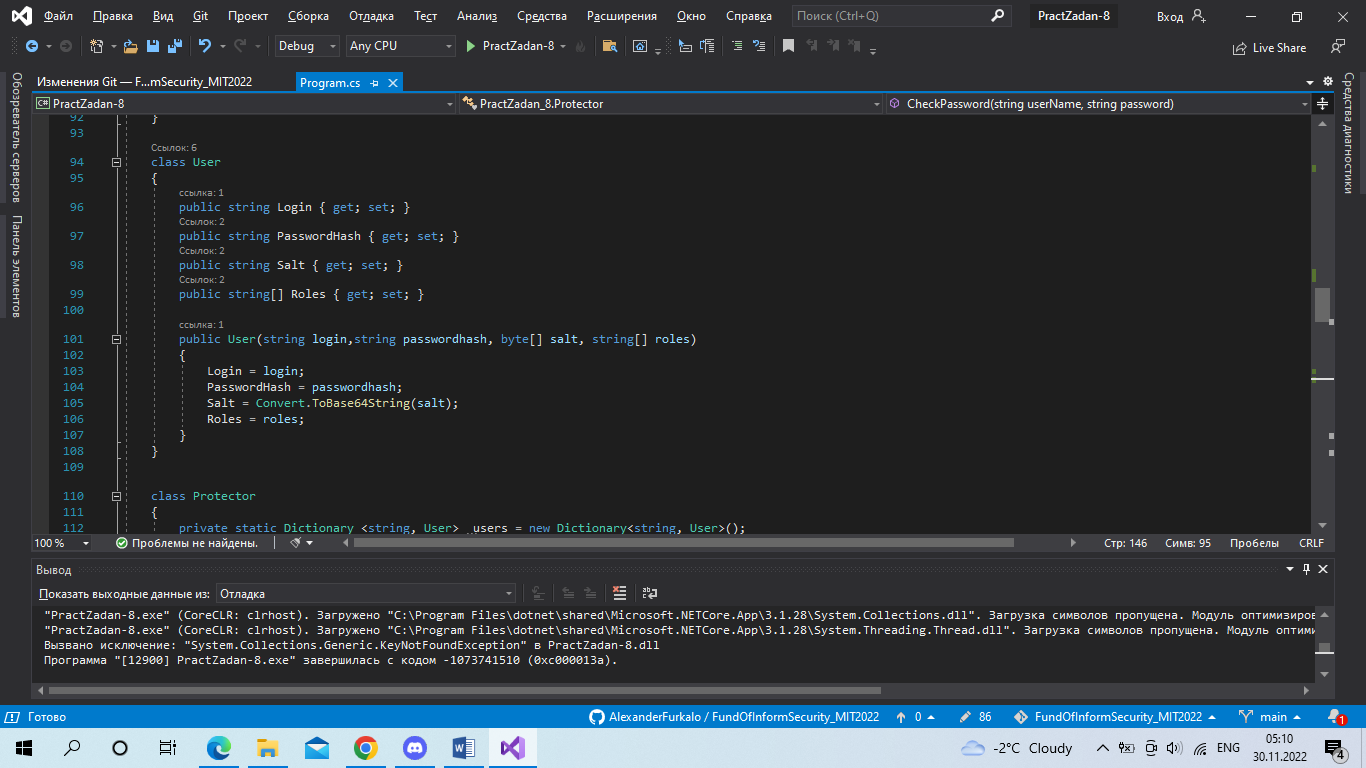
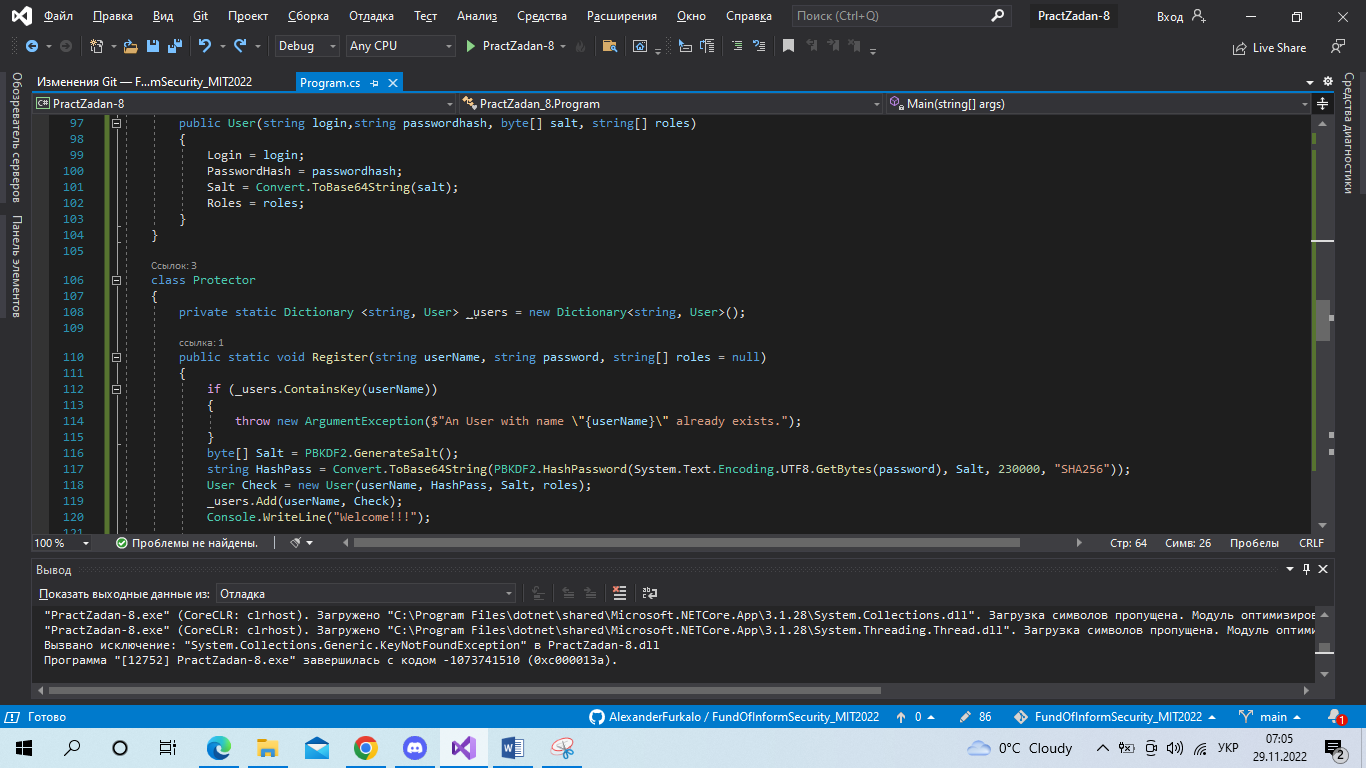
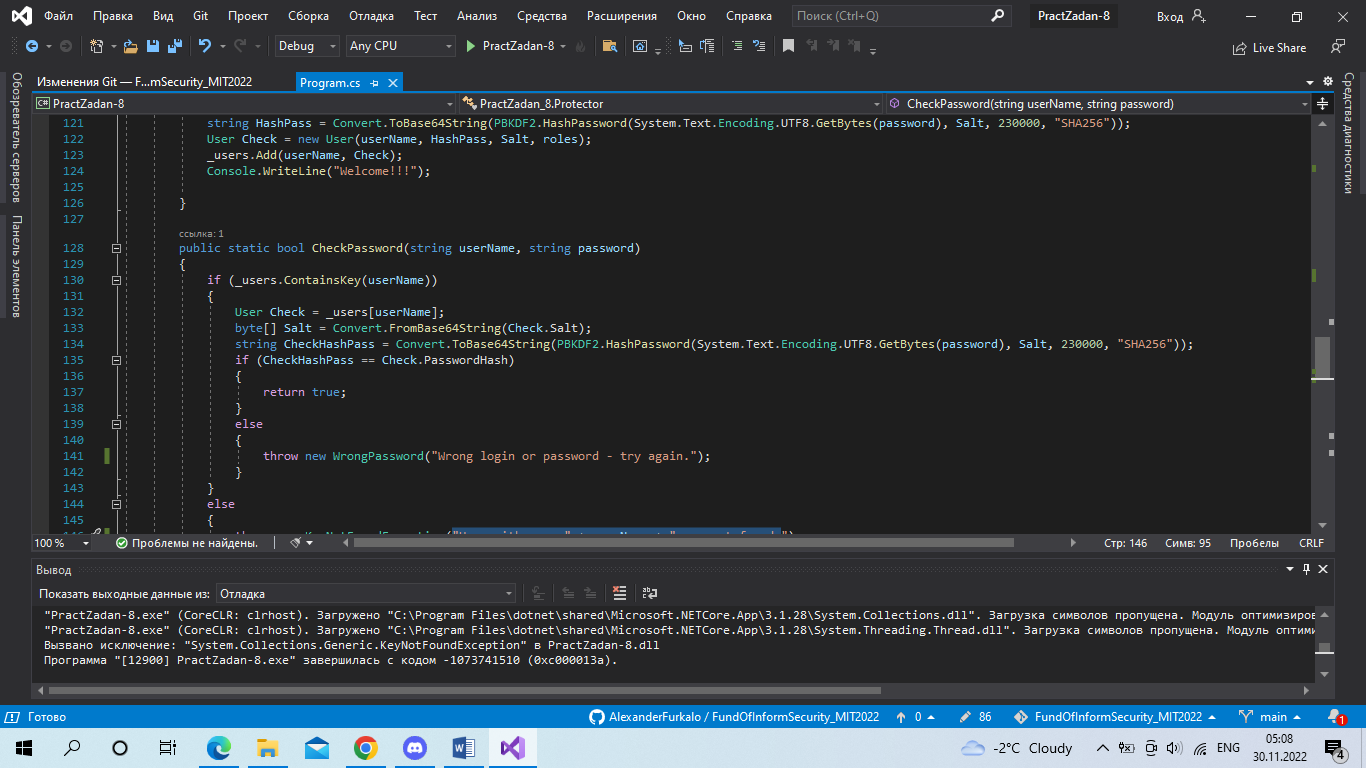
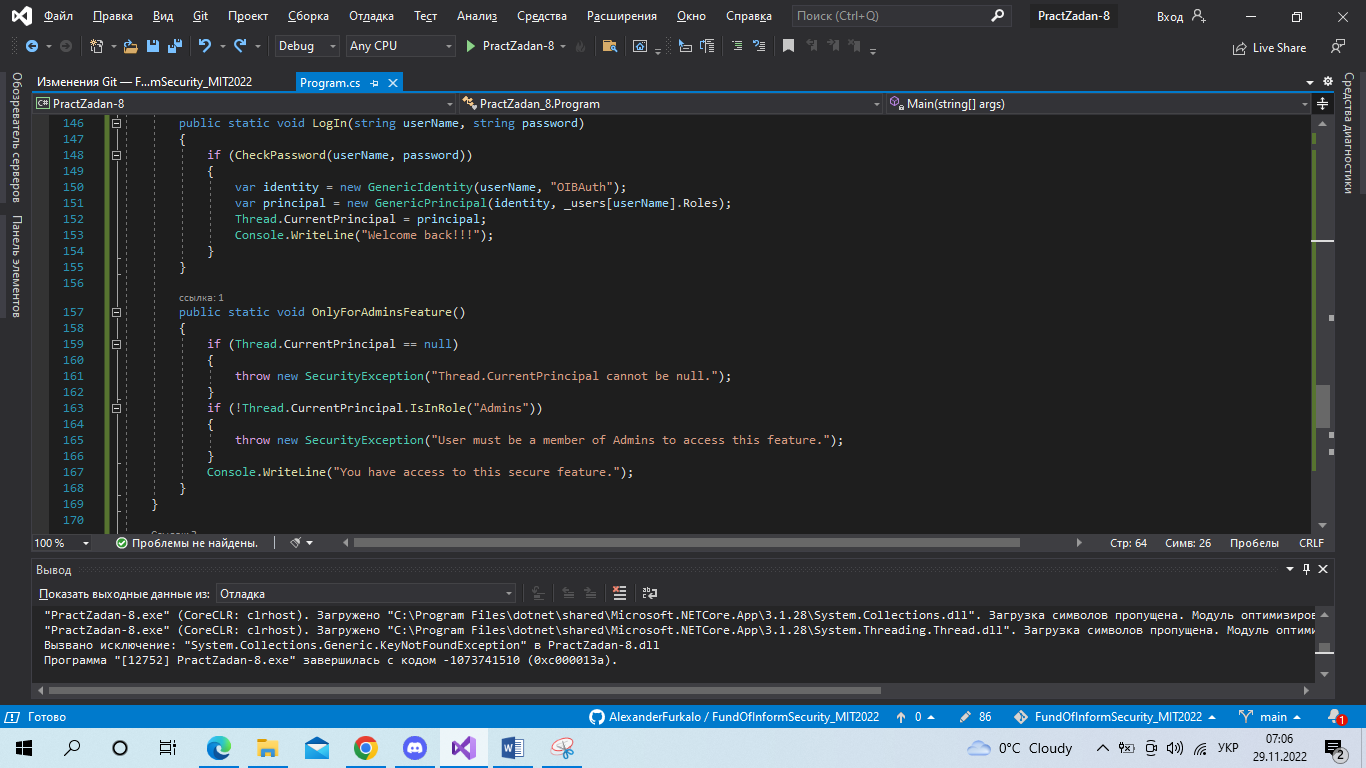
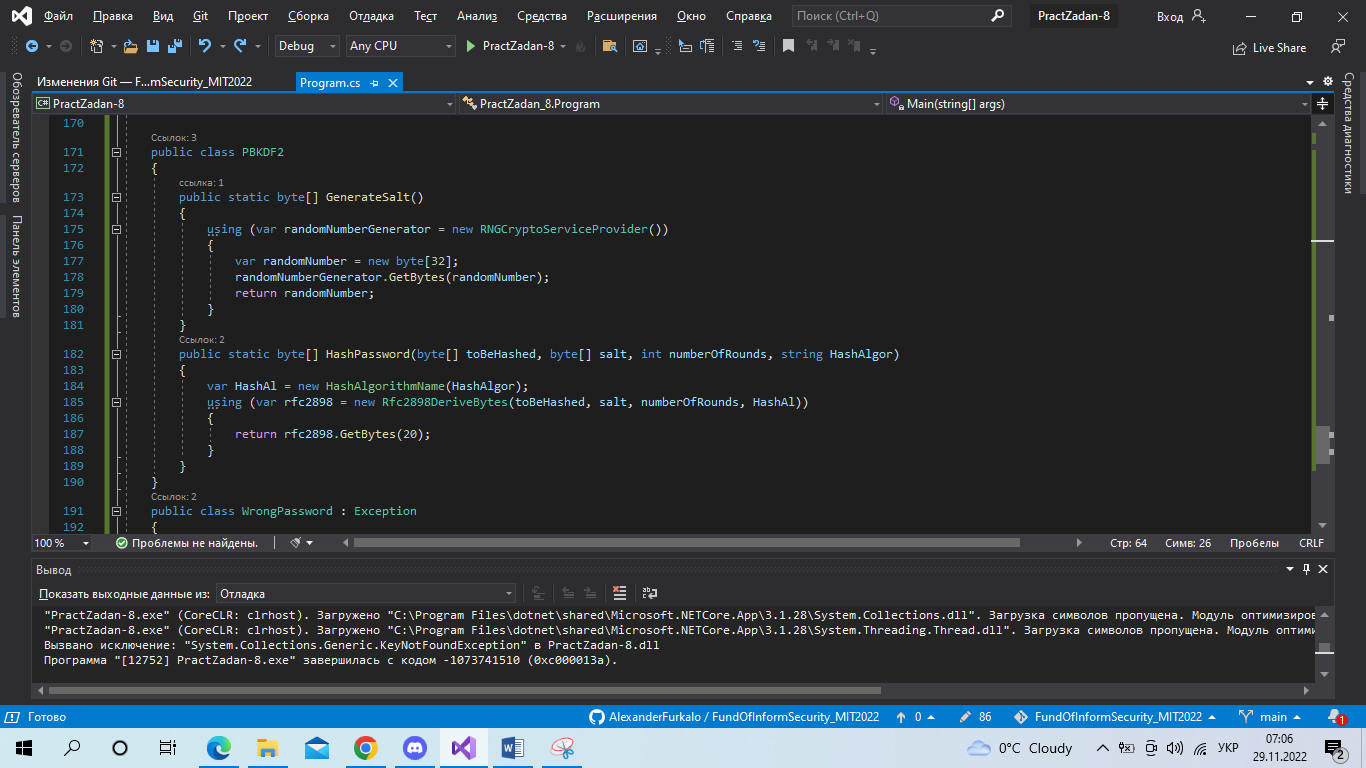
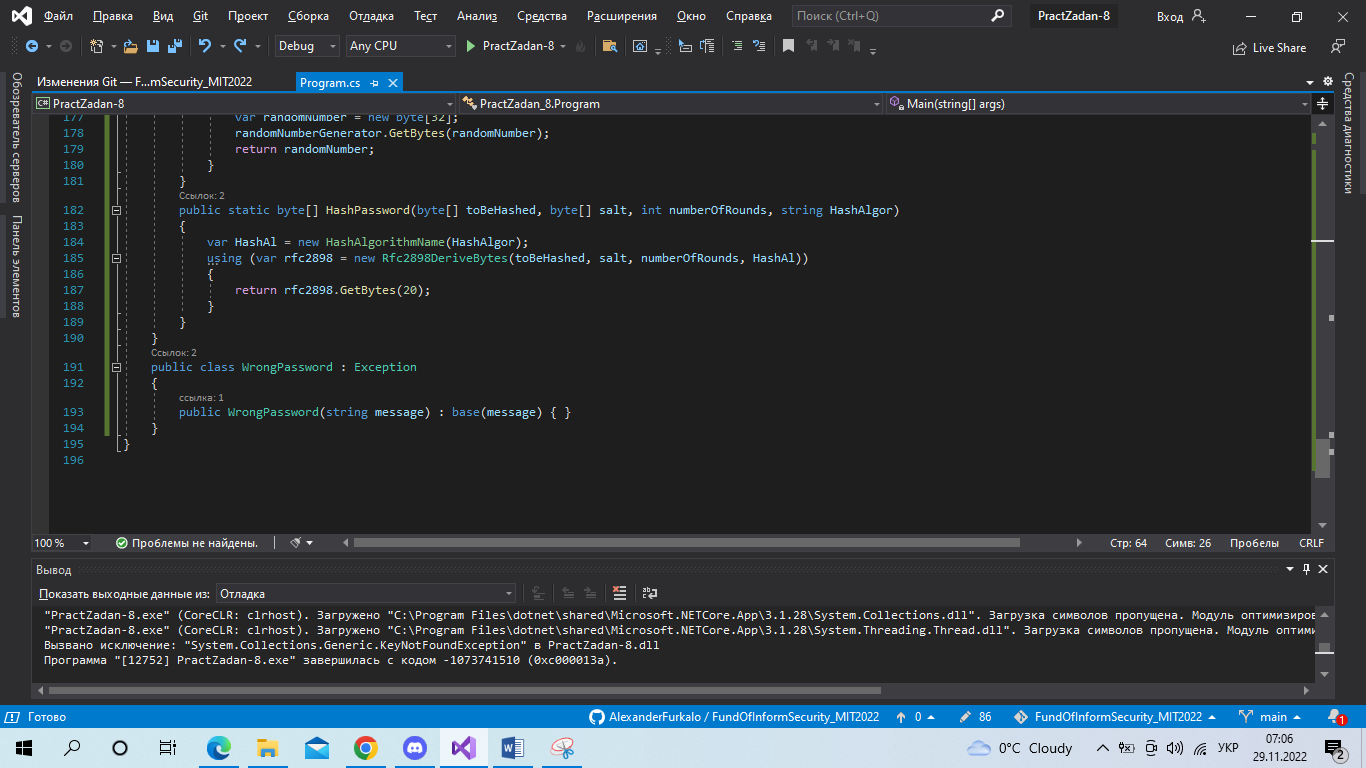
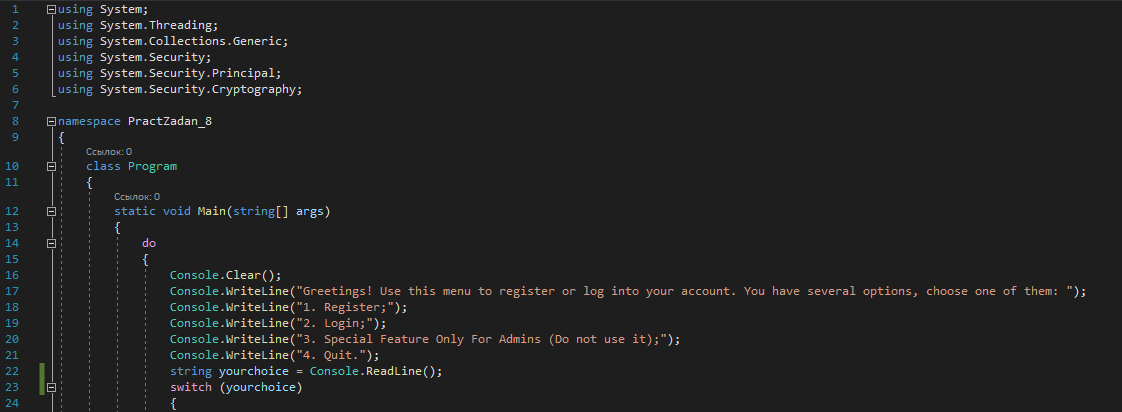
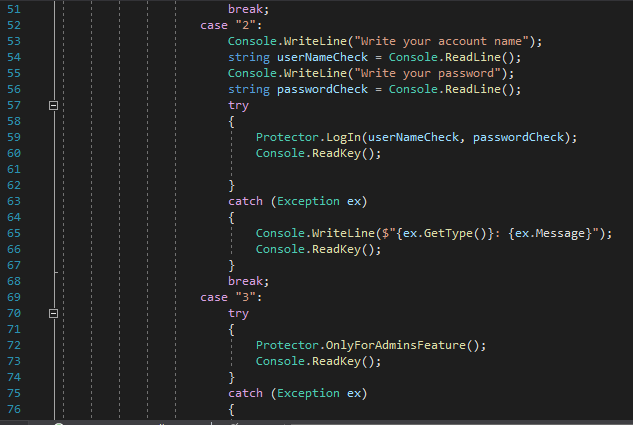
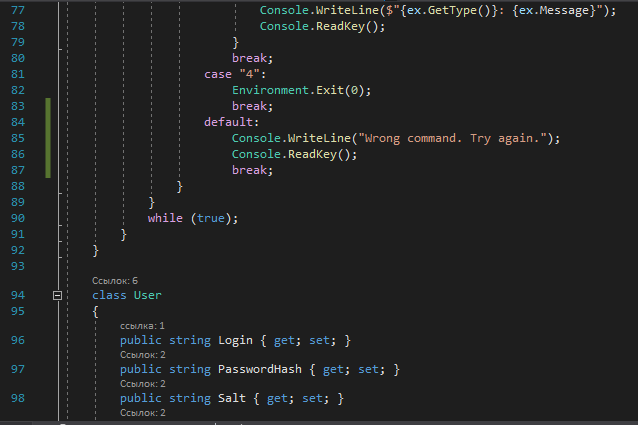
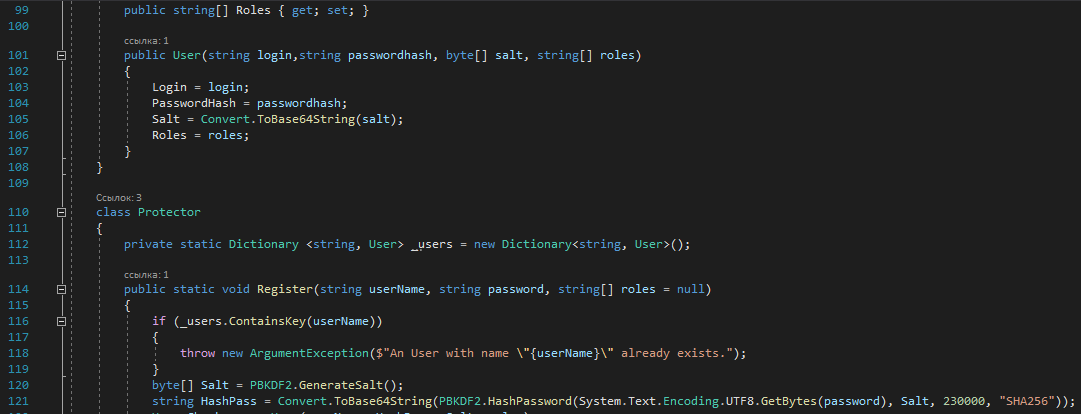
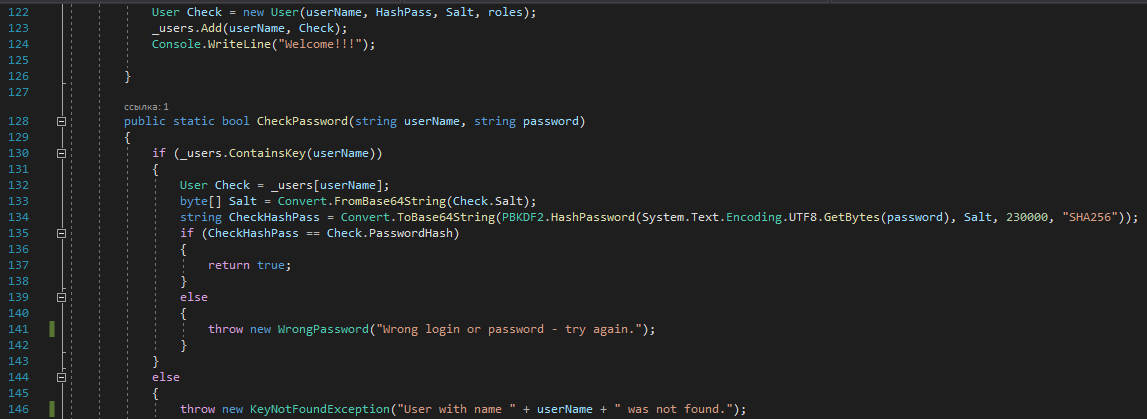
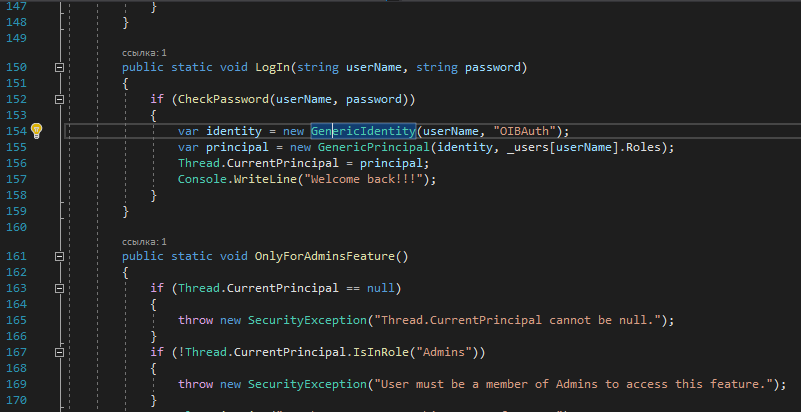
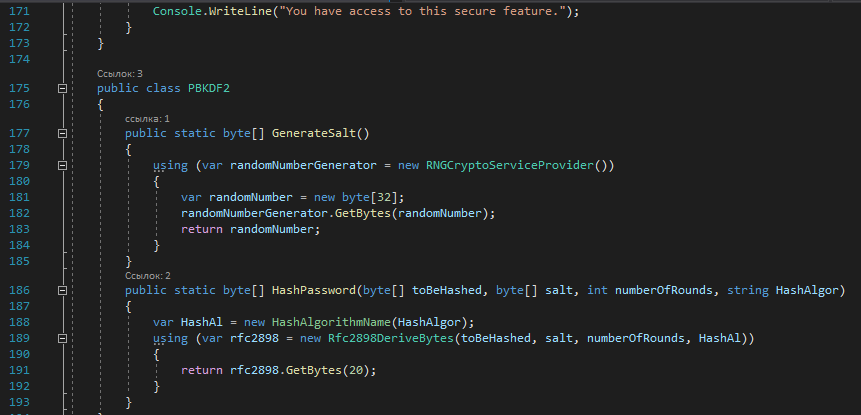
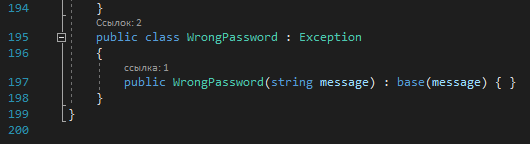
ПРАКТИЧНА РОБОТА №8  
З ОСНОВ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ  
СТУДЕНТА КИЇВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ІМЕНІ ТАРАСА ШЕВЧЕНКО  
ФАКУЛЬТЕТУ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ  
КАФЕДРИ МЕРЕЖЕВИХ ТА ІНТЕРНЕТ ТЕХНОЛОГІЙ  
ДРУГОГО КУРСУ, ДРУГОЇ ПІДГРУПИ  
ОЛЕКСАНДРА ОЛЕКСАНДРОВИЧА ФУРКАЛА  
ЗА ТЕМОЮ “АВТЕНТИФІКАЦІЯ ТА АВТОРИЗАЦІЯ КОРИСТУВАЧІВ”  
ЗВІТ  
30.11.2022

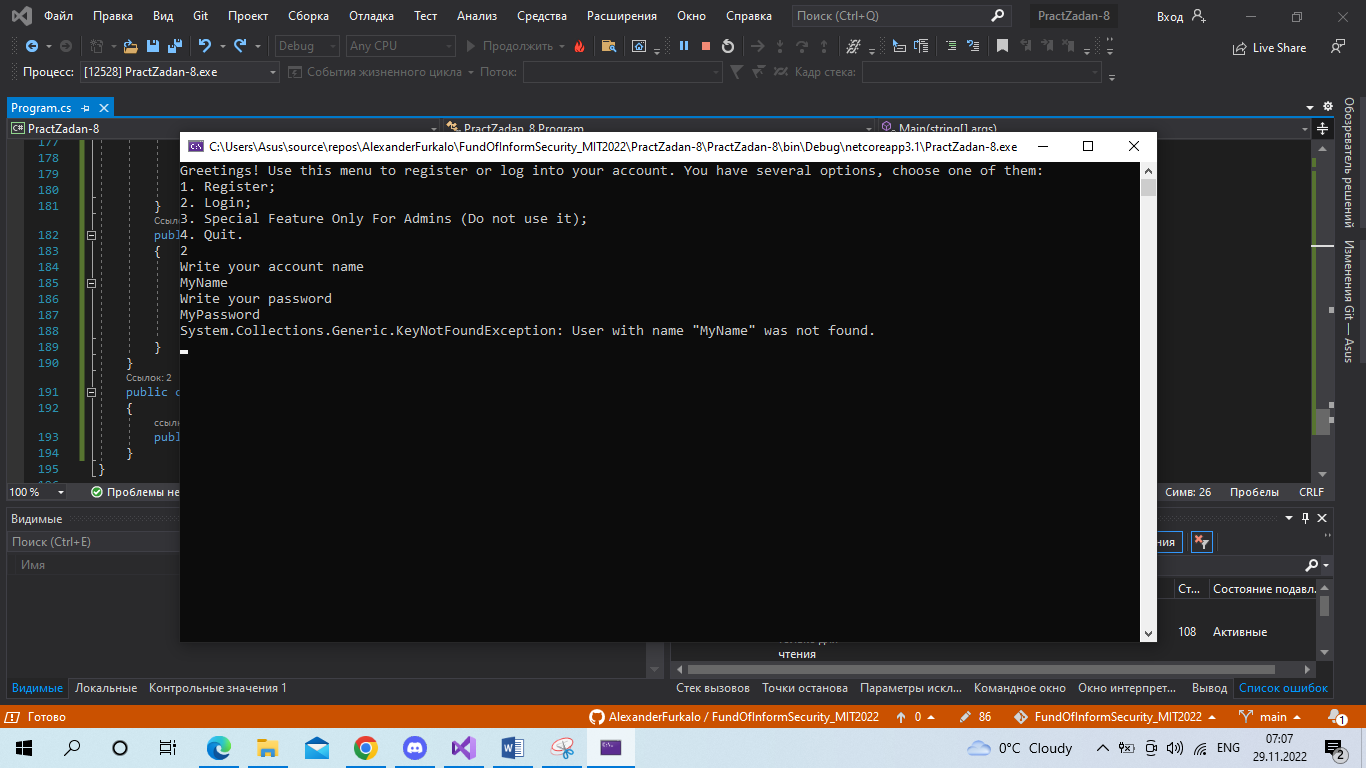
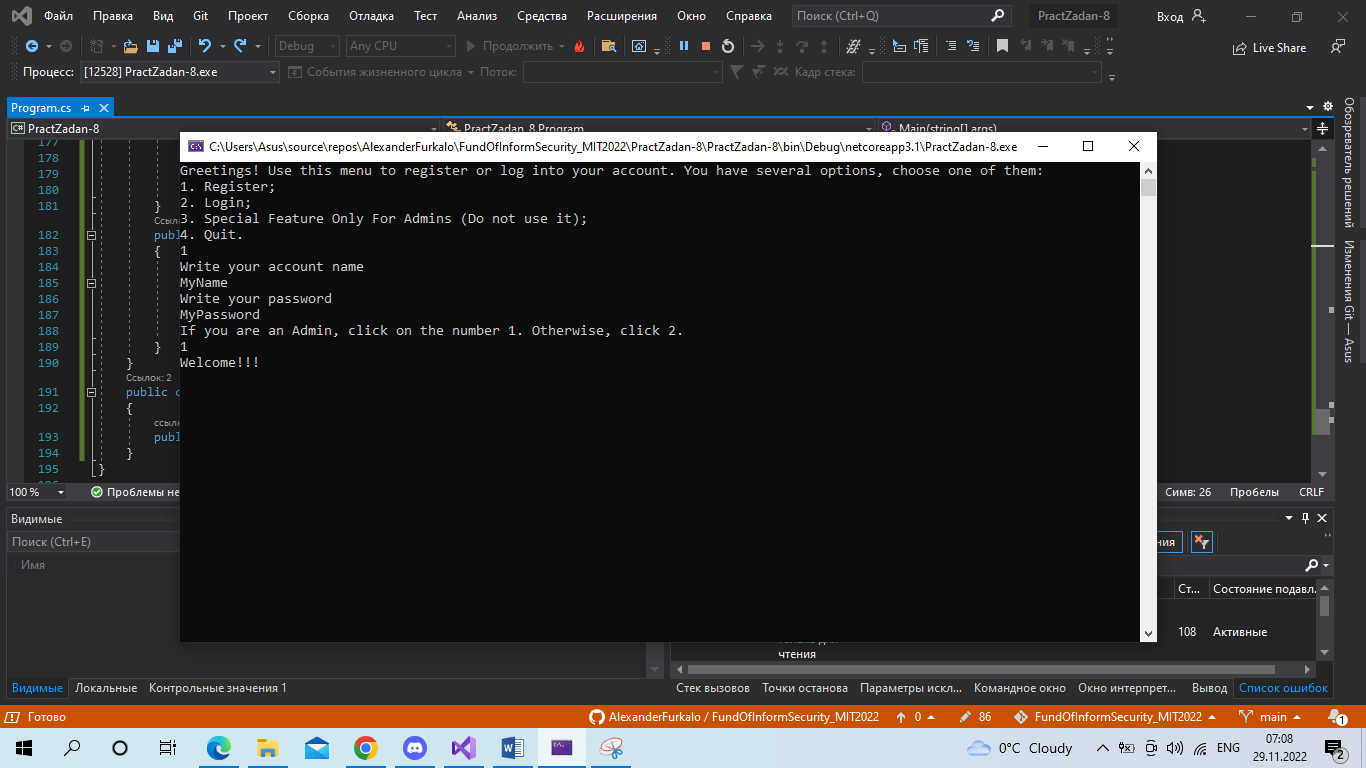
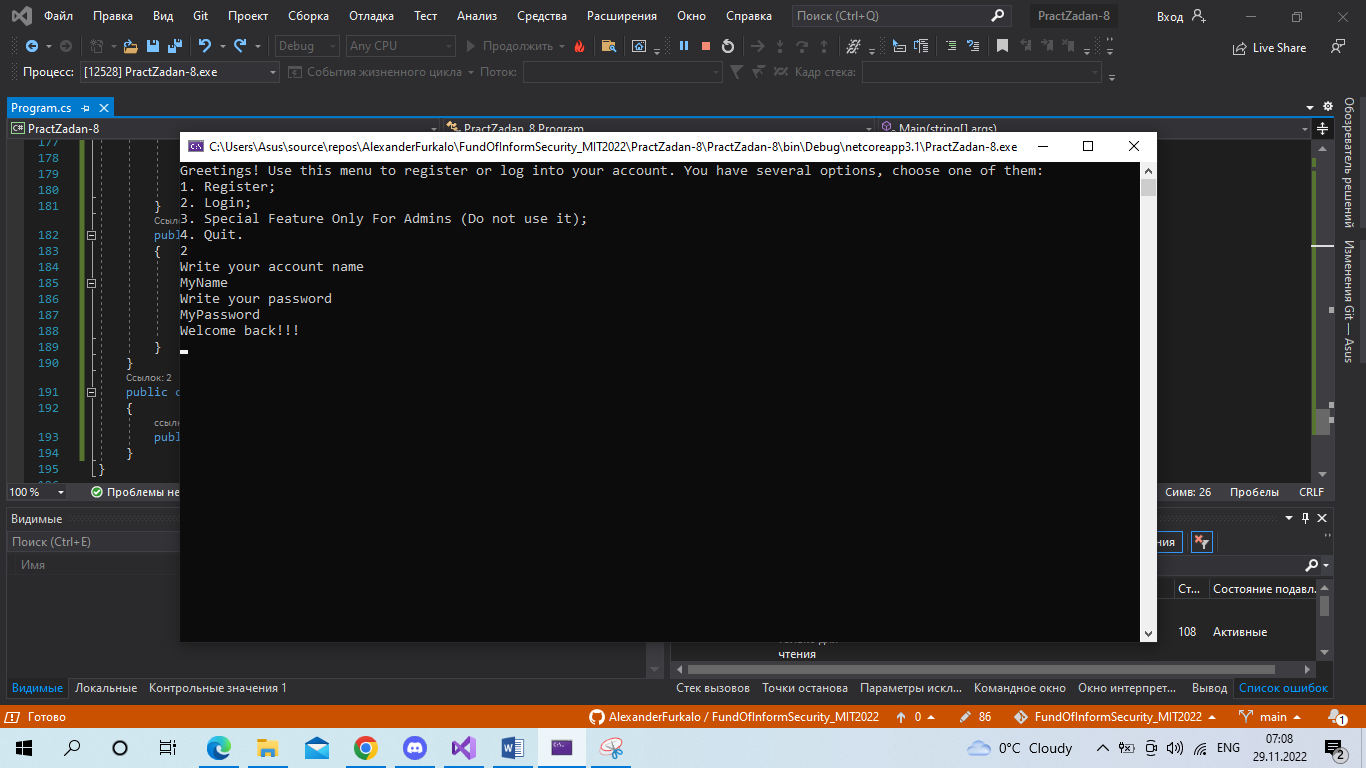
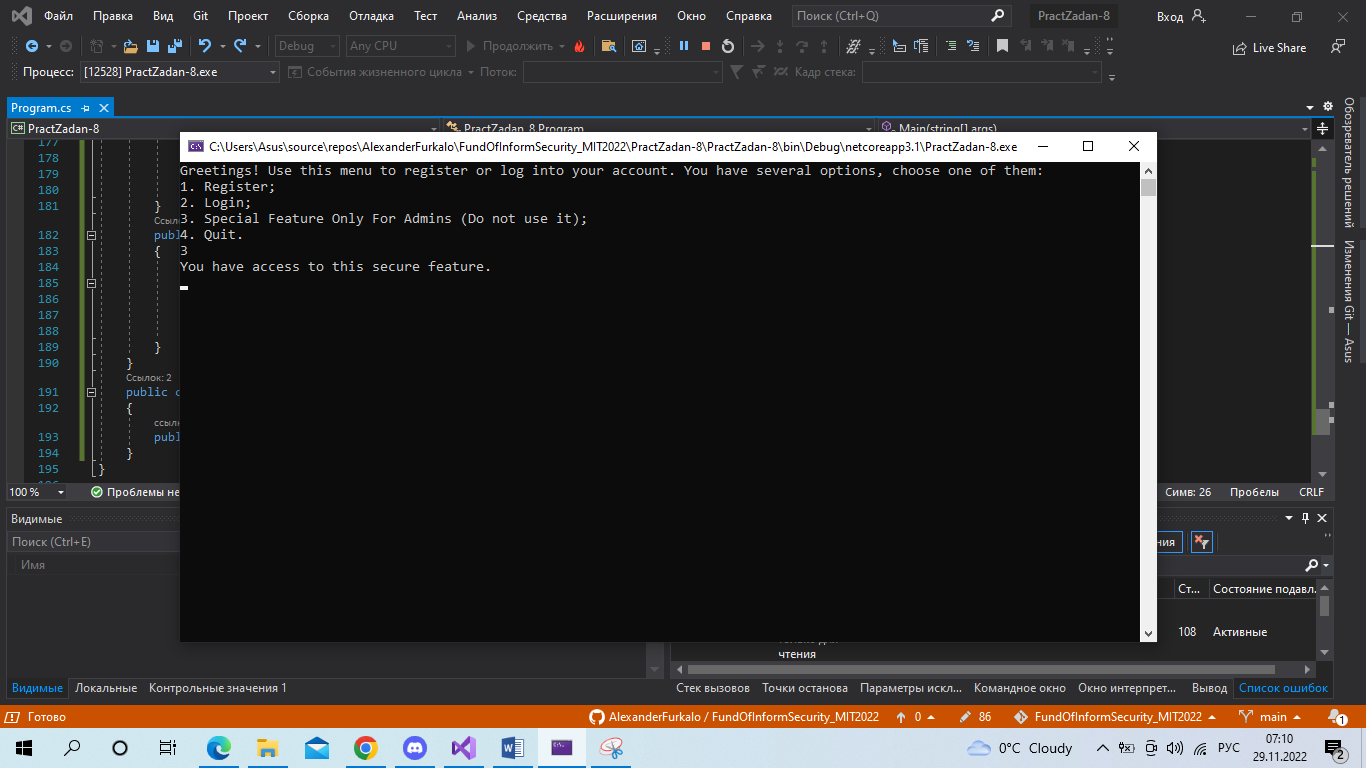
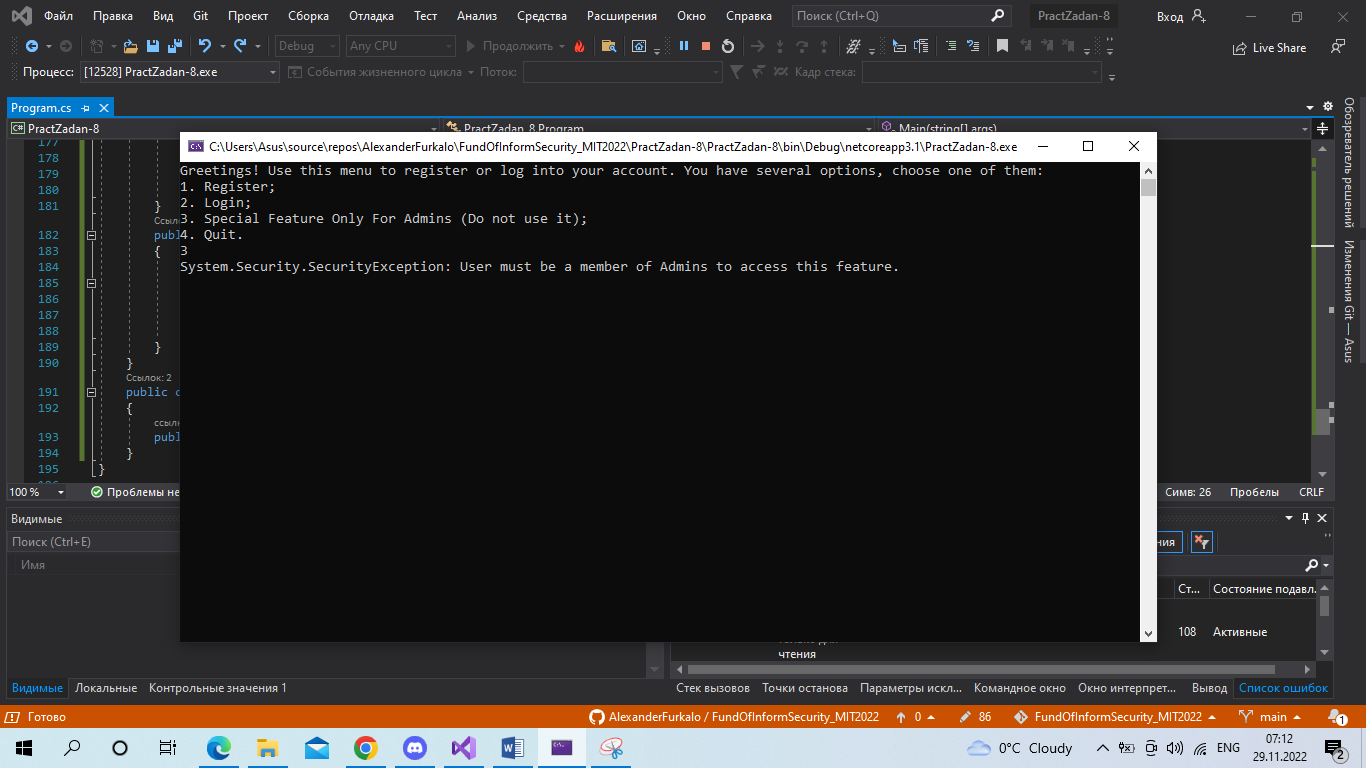
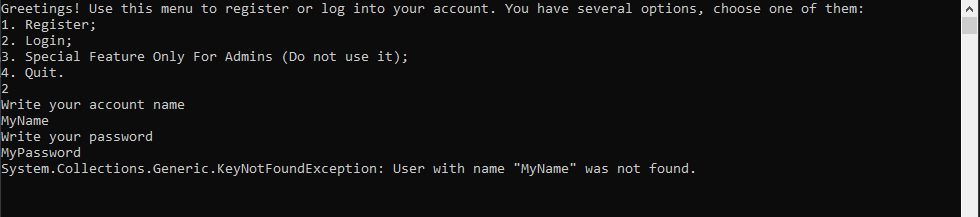
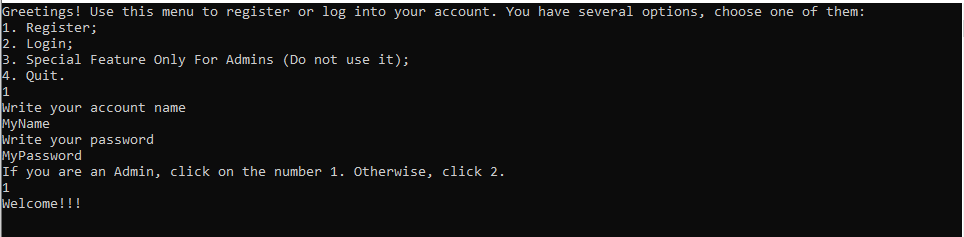
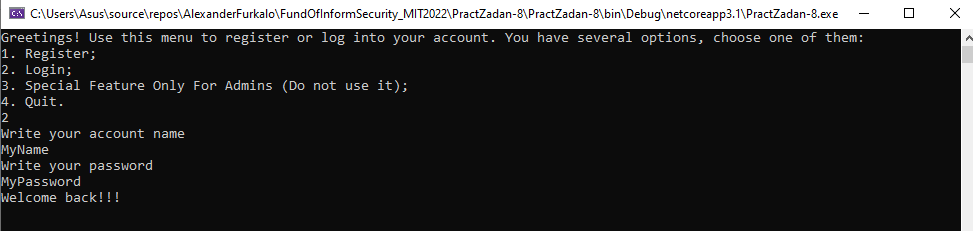
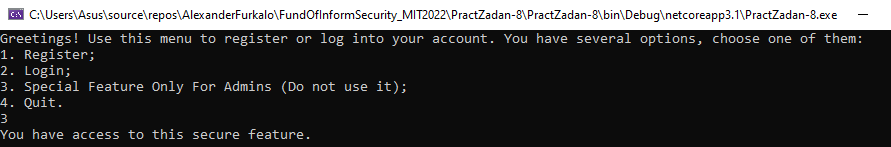
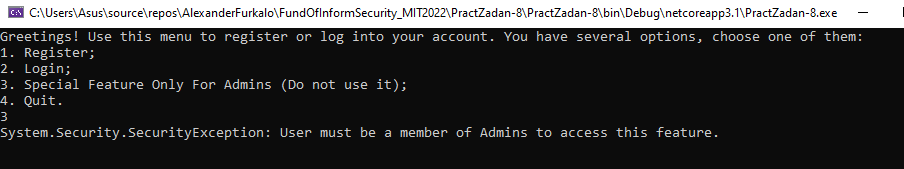
Завдання цієї практичної роботи були такі:  
1. Написати програму, яка виконує реєстрацію чотирьох користувачів, що мають різну приналежність ролей; здійснює автентифікацію користувача та надає або відхиляє доступ до певної частини коду (окремого методу).  
2. Оформити звіт.

Автентифікація – це процес верифікації особи користувача шляхом перевірки його облікових даних із даними, що зберігаються в надійному джерелі. Облікові дані можуть містити в собі поєднання імені користувача та пароля, біометричні дані. Після автентифікації користувачі можуть взаємодіяти з системою, наприклад, взнавати до яких груп або ролей вони належать та виконувати дозволені запити обробки інформації.

Авторизація – це процес верифікації приналежності до груп або ролей перед наданням доступу до ресурсів, наприклад, до функцій додатку або до даних. Авторизація може базуватися на верифікації особи користувача. Проте кращим методом забезпечення безпеки вважається авторизація, що базується на приналежності до групи або ролі: такий підхід дозволяє змінити приналежність у майбутньому, не вдаючись до перепризначення прав доступу

Перейдемо до першого завдання:

  
  
  
  
  
  
  
  
  
  
Трохи наблизивши:  
  
  
  
  
  
  
  
  


Результати роботи:  
  
  
  
  
  
  
  
  
  
Якщо здійснити Login з акаунта - юзера:   


Main, “головне меню”, зроблене через do та “while true”. Тому, поки не буде натиснуто команду номер 4, тобто вихід, програма не закінчує свою роботу після виконання інших команд.  
Оператор switch вибирає список операторів, який потрібно виконати, на основі відповідності шаблону виразу. Через string yourchoice = Console.ReadLine() реалізується те, як користувач вибиратиме команди - на вибір є всього 4 варіанти (case 1-4), будь-які інші призводять до того, що буде виведено "Wrong Command. Try again" через опцію default.  
Так як меню при виклику стирає всю попередню інформацію консолі через Console.Clear(), у коді використовується Console.ReadKey() - у цьому випадку користувач має час для того, щоб прочитати виведену інформацію, і очищення консолі (або наступна команда) відбудеться лише після натискання наступної будь-якої кнопки.   
Серед команд варто виділити третю, яка виводить повідомлення про допуск до спеціальної функції – вона має три можливі результати, а саме:  
- “Thread.CurrentPrincipal cannot be null." - якщо не було успішно виконано операцію login, і зараз немає активного користувача  
- "User must be a member of Admins to access this feature." – якщо активний користувач не має ролі Admin  
- "You have access to this secure feature." – якщо активний користувач має роль Admin.  
Перевірка виконується через public static void OnlyForAdminsFeature(): Thread.CurrentPrincipal повертає або задає поточного учасника потоку (на основі ролей), та Thread.CurrentPrincipal.IsInRole визначає, чи належить поточний учасник до потрібної ролі.  
Клас User представляє користувача у системі за властивостями імені, пароля (вже хеширован), “солі” (Convert.ToBase64String, для того щоб було в вигляді string) та ролі. Процес реєстрації досить простий – спочатку перевіряється, чи є вже користувач з цим ім’ям (за вимогами роботи, цей крок знаходиться в Register), та якщо ні – генерує сіль (код для цього був запозичений з минулої практичної роботи і працює за таким же вивченим принципом - можна тільки помітити те, що на відміну від використаної тоді версії, в завданні не потрібно використовувати декілька алгоритмів хешування, тому в коді використовується тільки один, SHA256, та також вказуємо параметр кількості ітерацій – як у минулій роботі, це початкове значення для варіанту 23, тобто 230.000), хешує пароль та додає нового користувача через \_users.Add.  
LogIn – автентифікація користувача, спочатку потрібно ввести ім’я (userName) та пароль (password). Після цього, виконується CheckPassword: спочатку у словнику шукається ім'я користувача, і якщо такого немає, виводиться повідомлення ""User with name " + userName + " was not found.", інакше через словник виводиться сіль, яка потім використовується для хешування введеного пароля, і порівнювання результату з даними зі словника. Якщо збігу немає, отримуємо повідомлення "Wrong login or password - try again.". Якщо є, повертаємось до LogIn – створюється екземпляр автентифікованого користувача (new Generic Identity), прив’язує його до обраних ролей (var principal = new GenericPrincipal(identity, \_users[userName].Roles)), та в кінці – присвоєння його потоку, в якому виконується програма.

Висновки

У цій практичній роботі, ми дізналися більше про автентифікацію та авторизацію, навчилися розрізняти ці поняття. Також ми написали програму, яка виконує реєстрацію чотирьох користувачів, що мають різну приналежність ролей; здійснює автентифікацію користувача та надає або відхиляє доступ до певної частини коду.