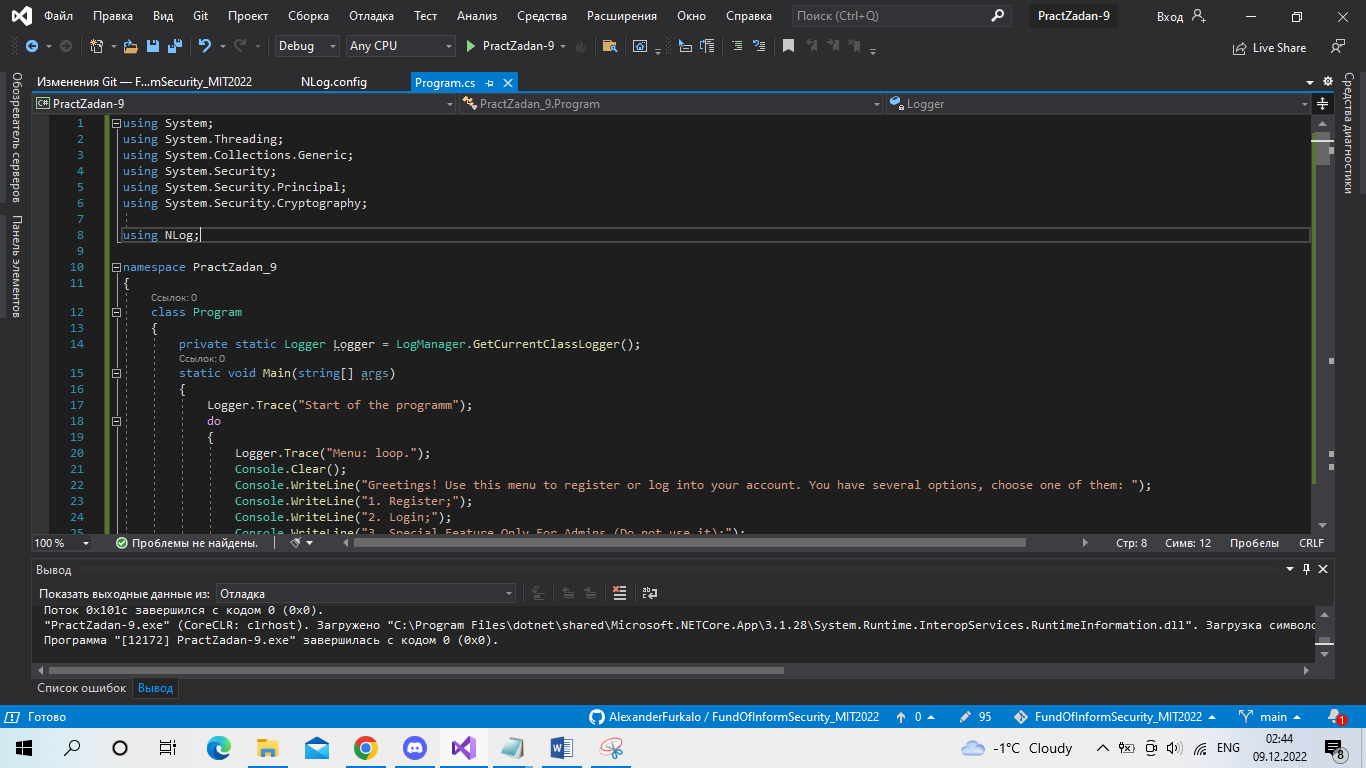
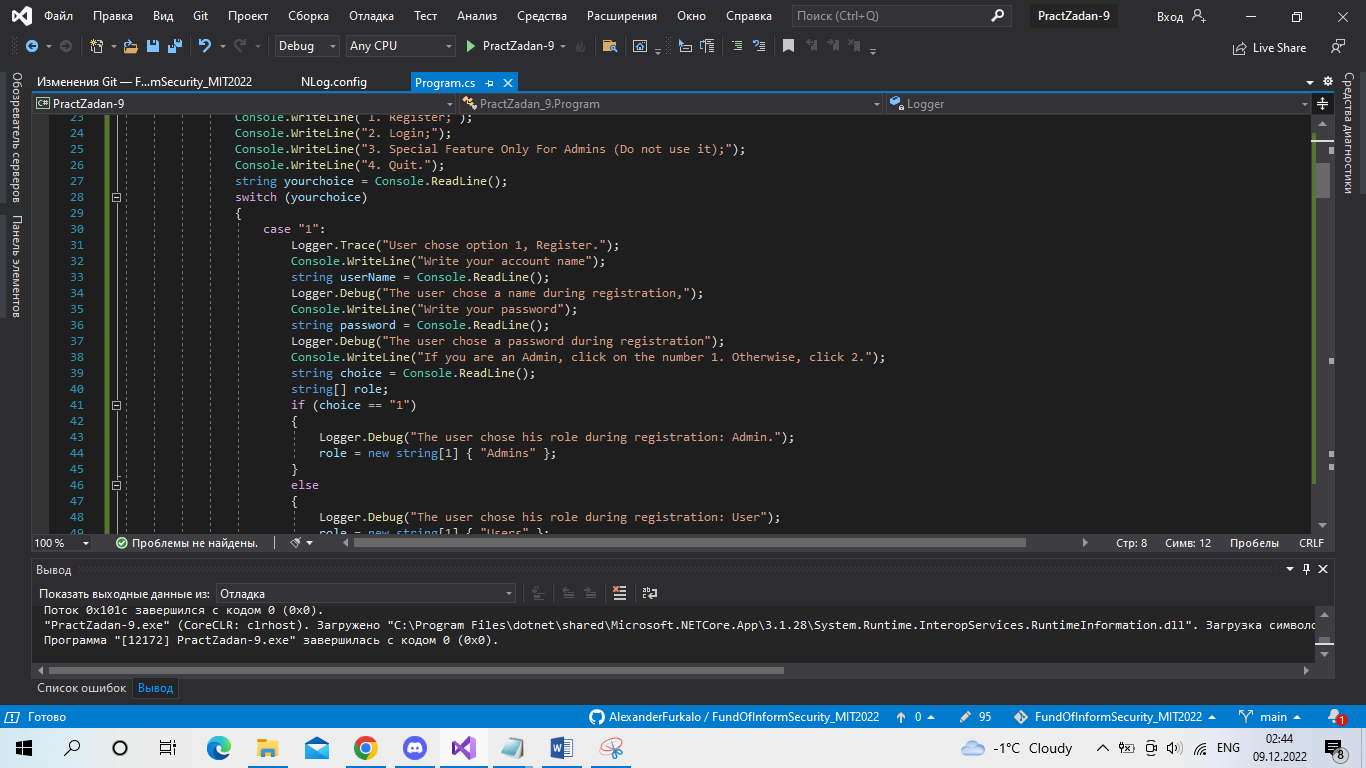
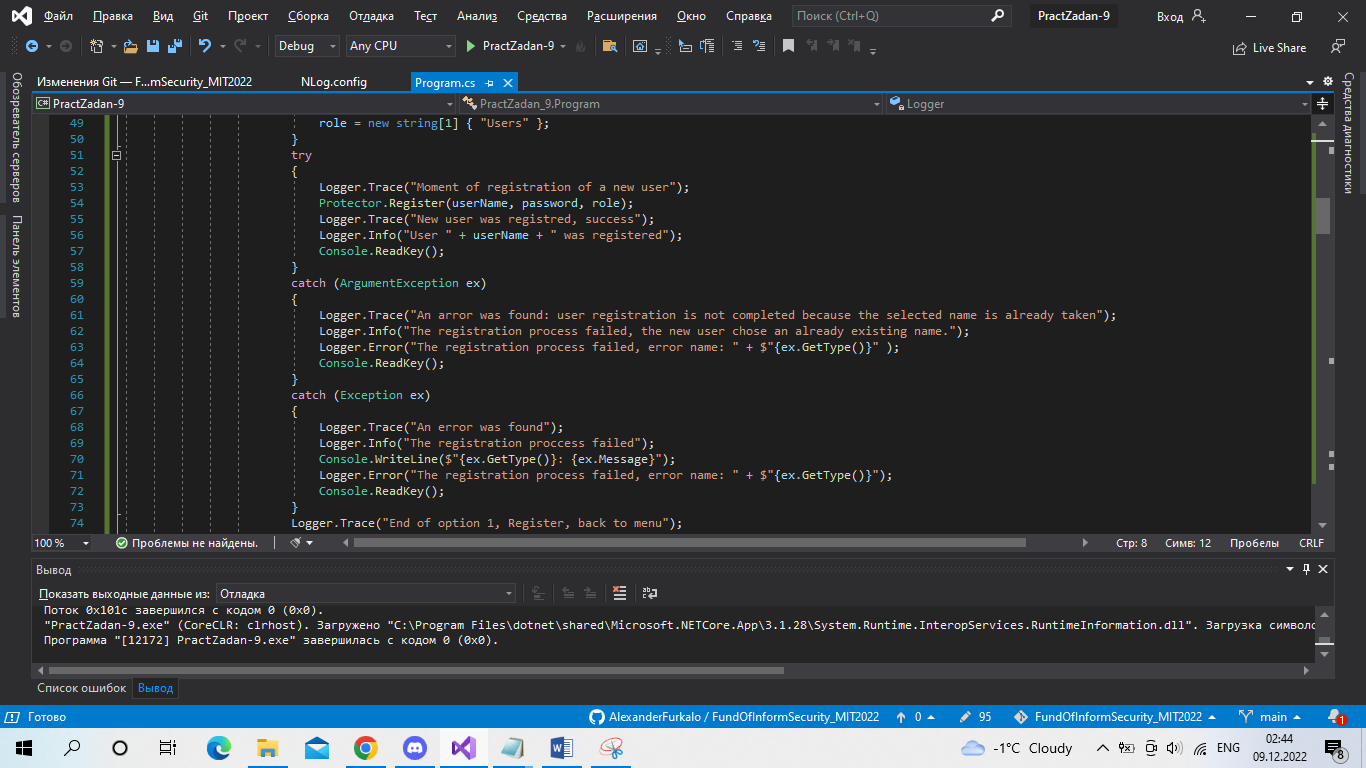
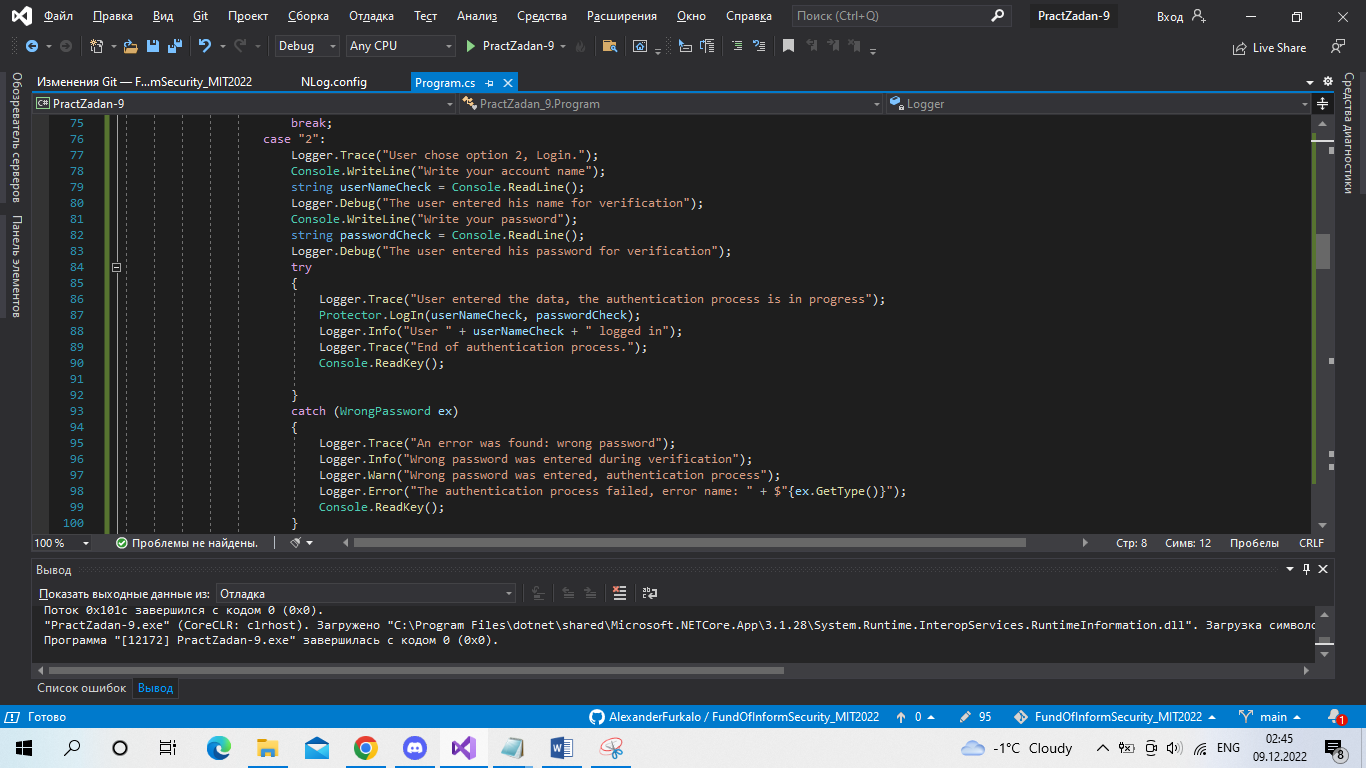
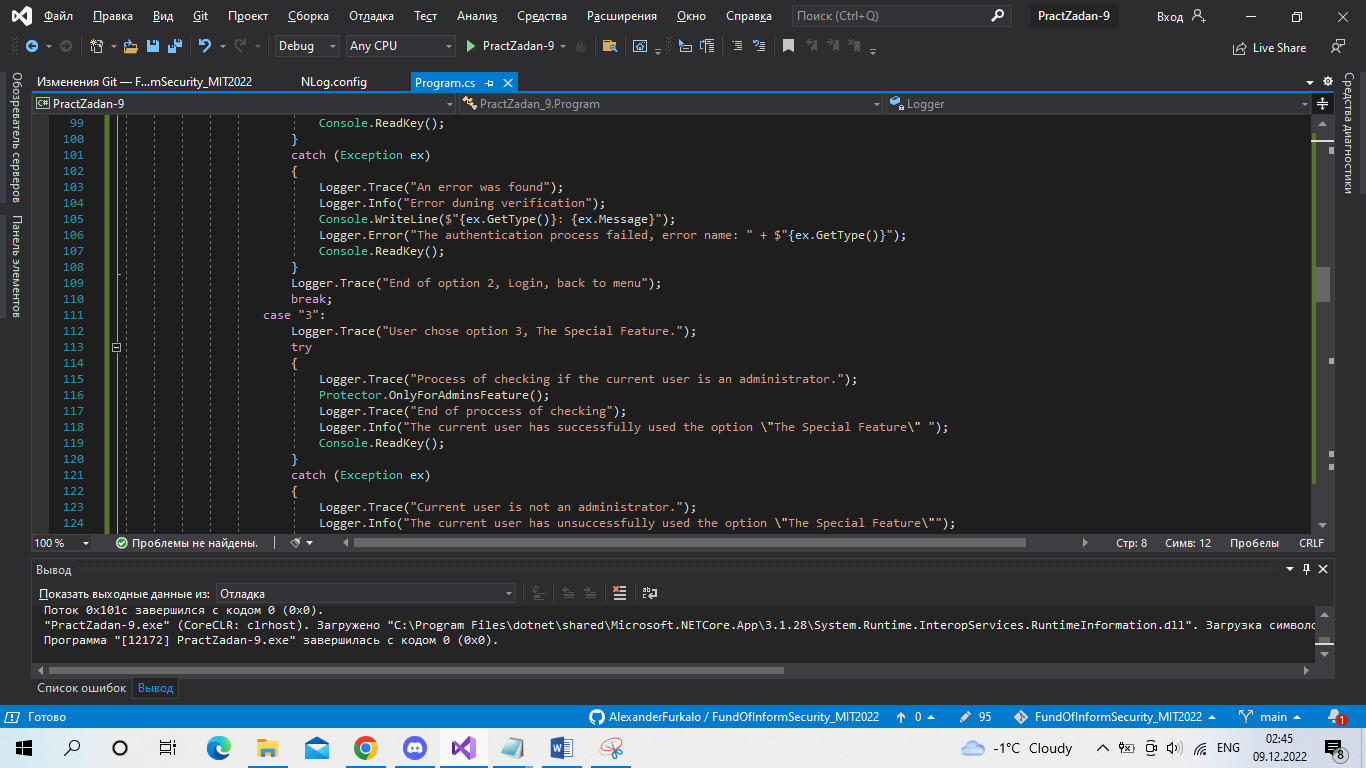
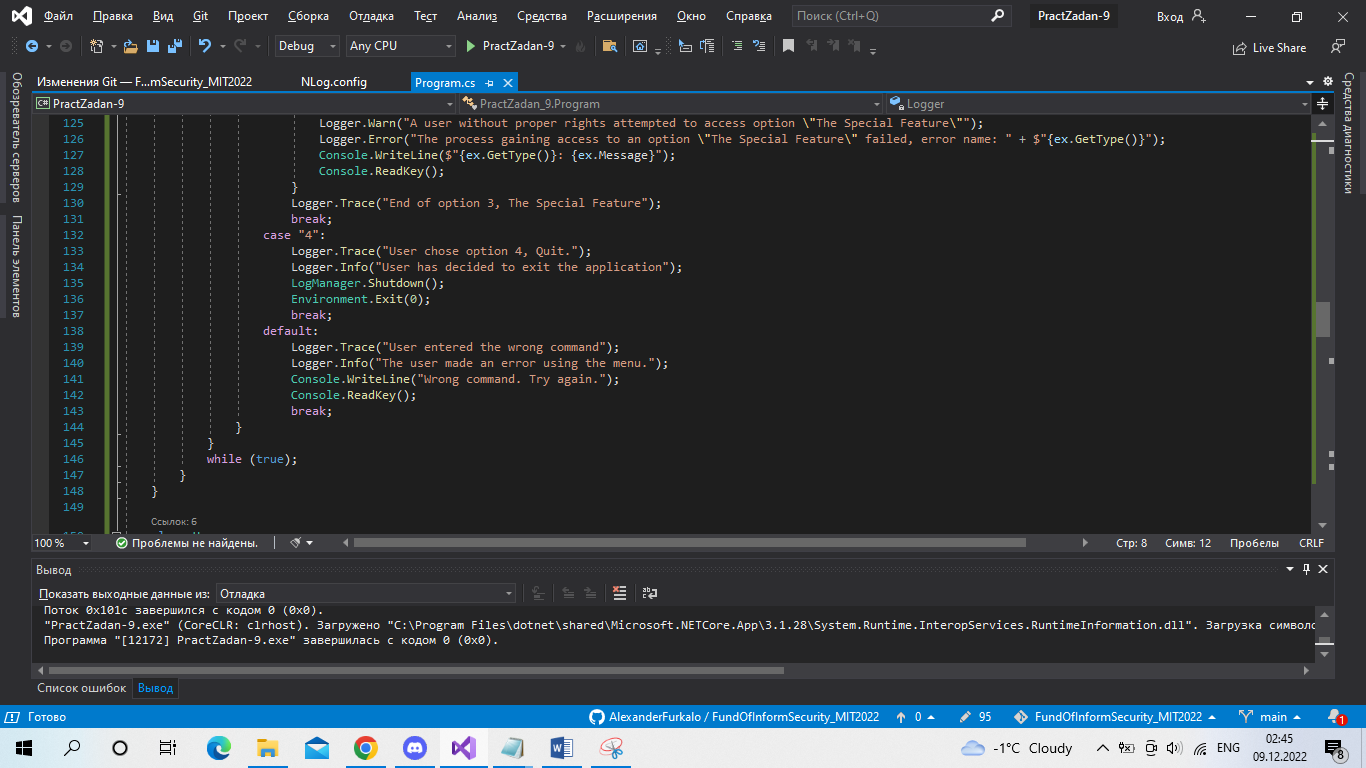
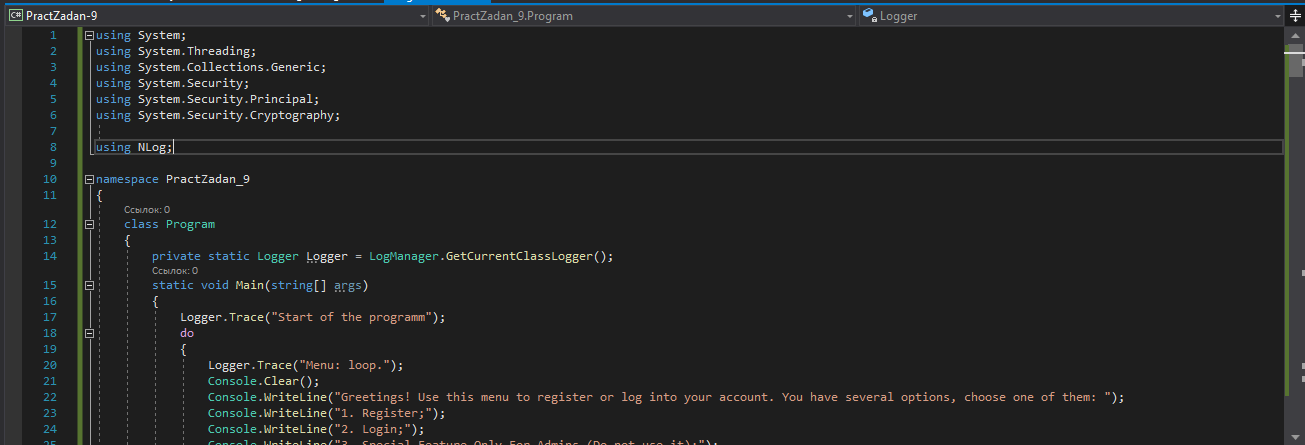
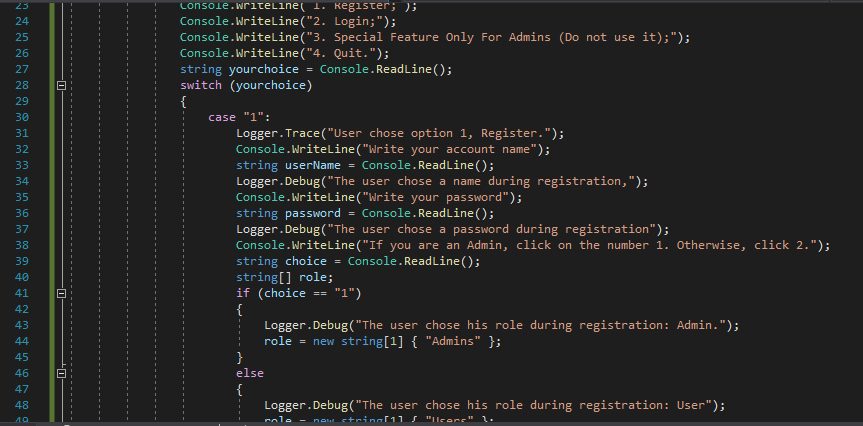
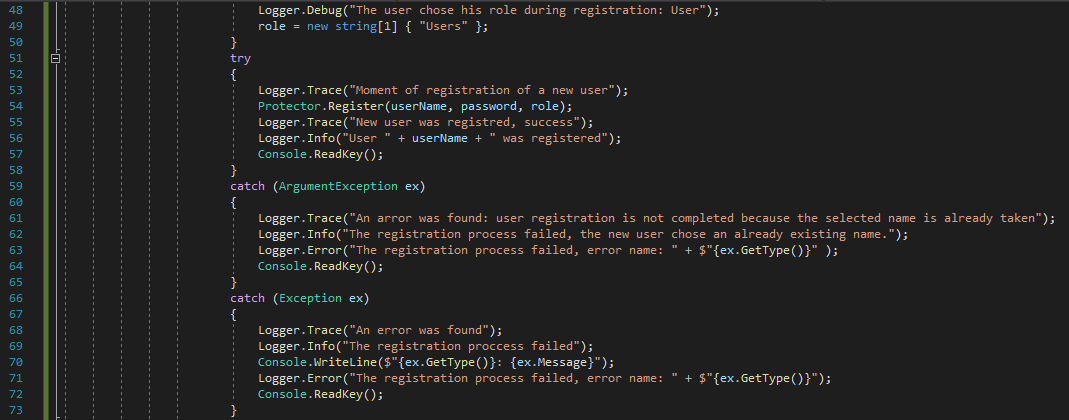
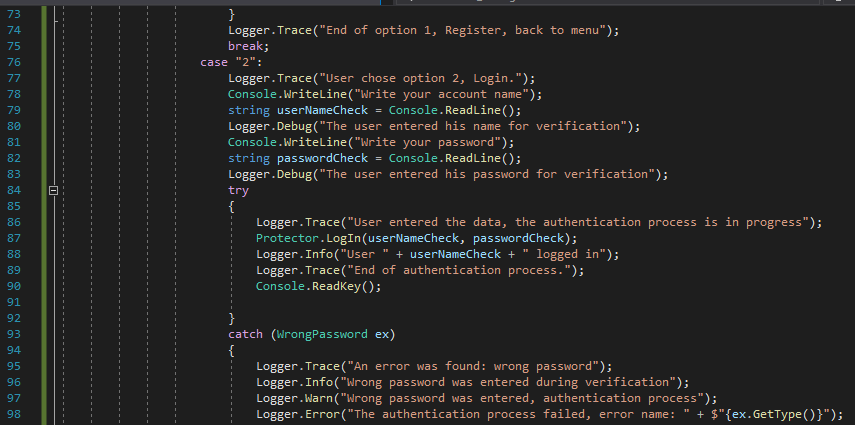
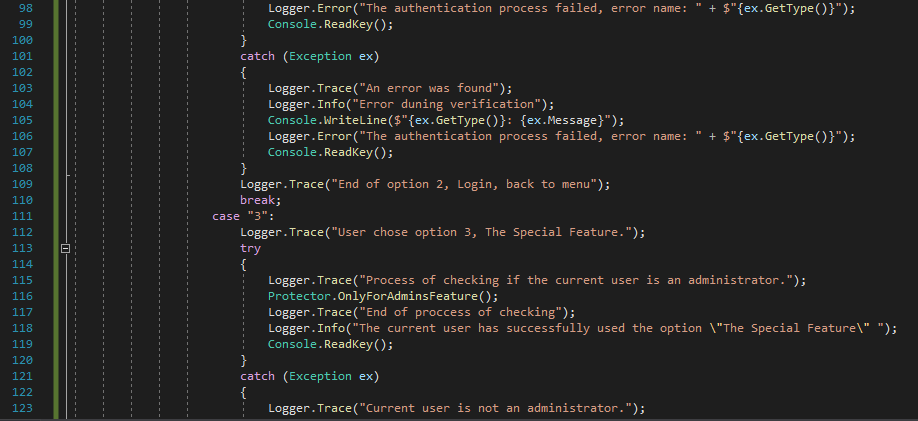
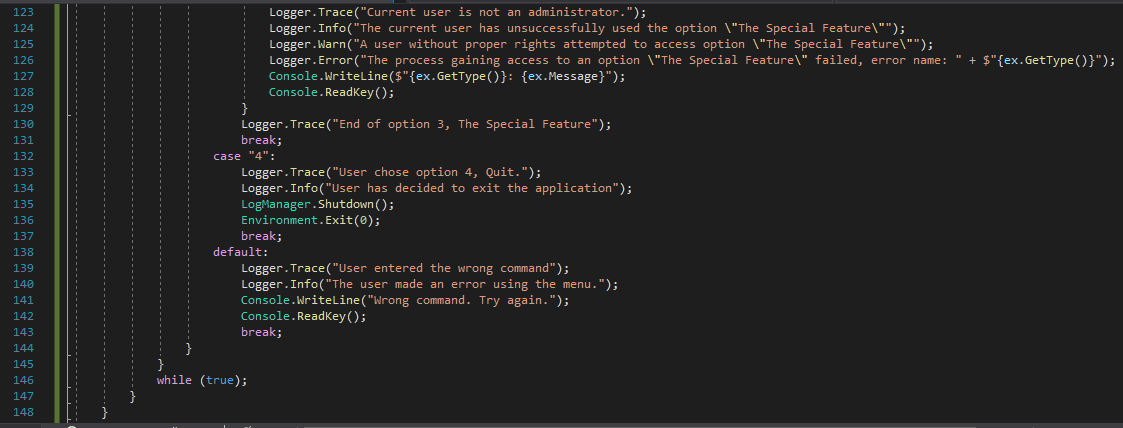
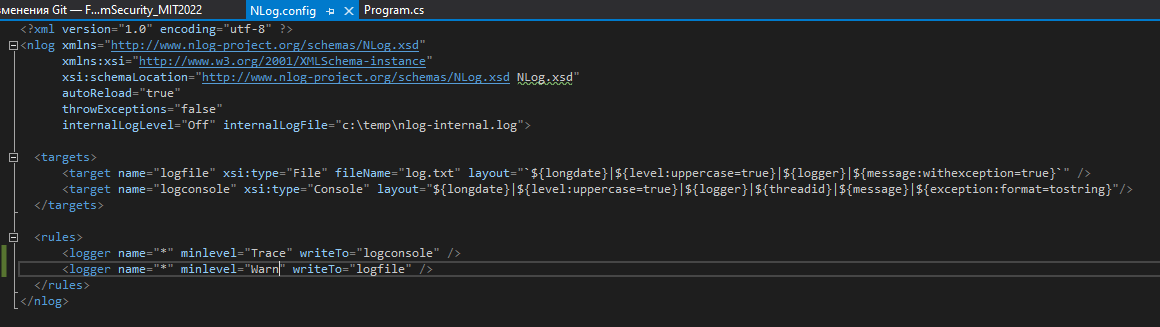
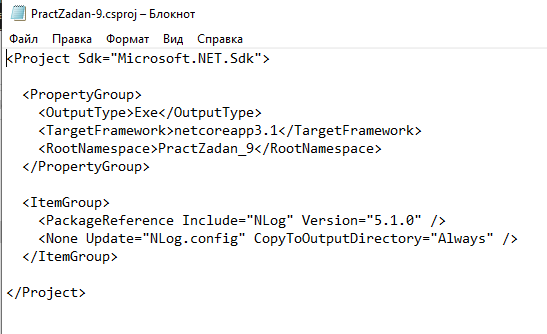
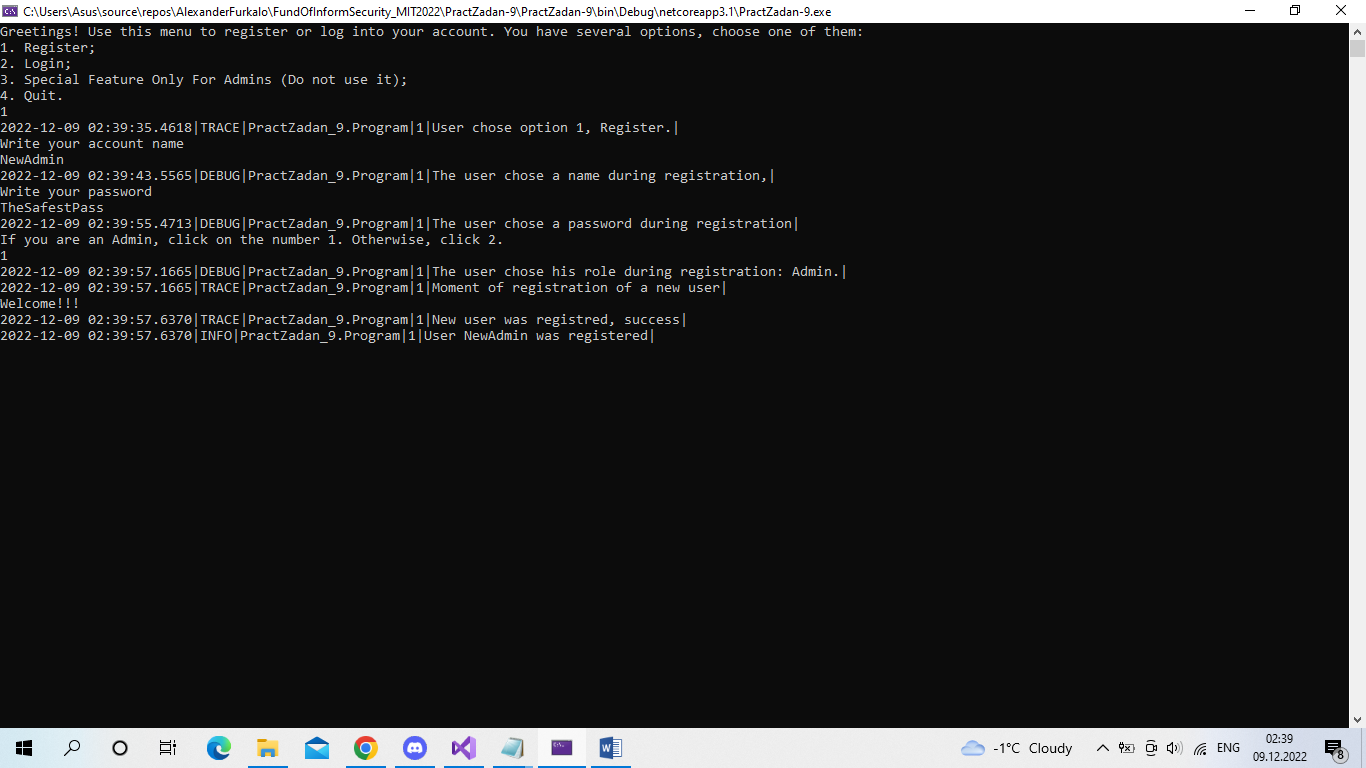
ПРАКТИЧНА РОБОТА №9  
З ОСНОВ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ  
СТУДЕНТА КИЇВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ІМЕНІ ТАРАСА ШЕВЧЕНКО  
ФАКУЛЬТЕТУ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ  
КАФЕДРИ МЕРЕЖЕВИХ ТА ІНТЕРНЕТ ТЕХНОЛОГІЙ  
ДРУГОГО КУРСУ, ДРУГОЇ ПІДГРУПИ  
ОЛЕКСАНДРА ОЛЕКСАНДРОВИЧА ФУРКАЛА  
ЗА ТЕМОЮ “ЛОГУВАННЯ”  
ЗВІТ  
09.12.2022

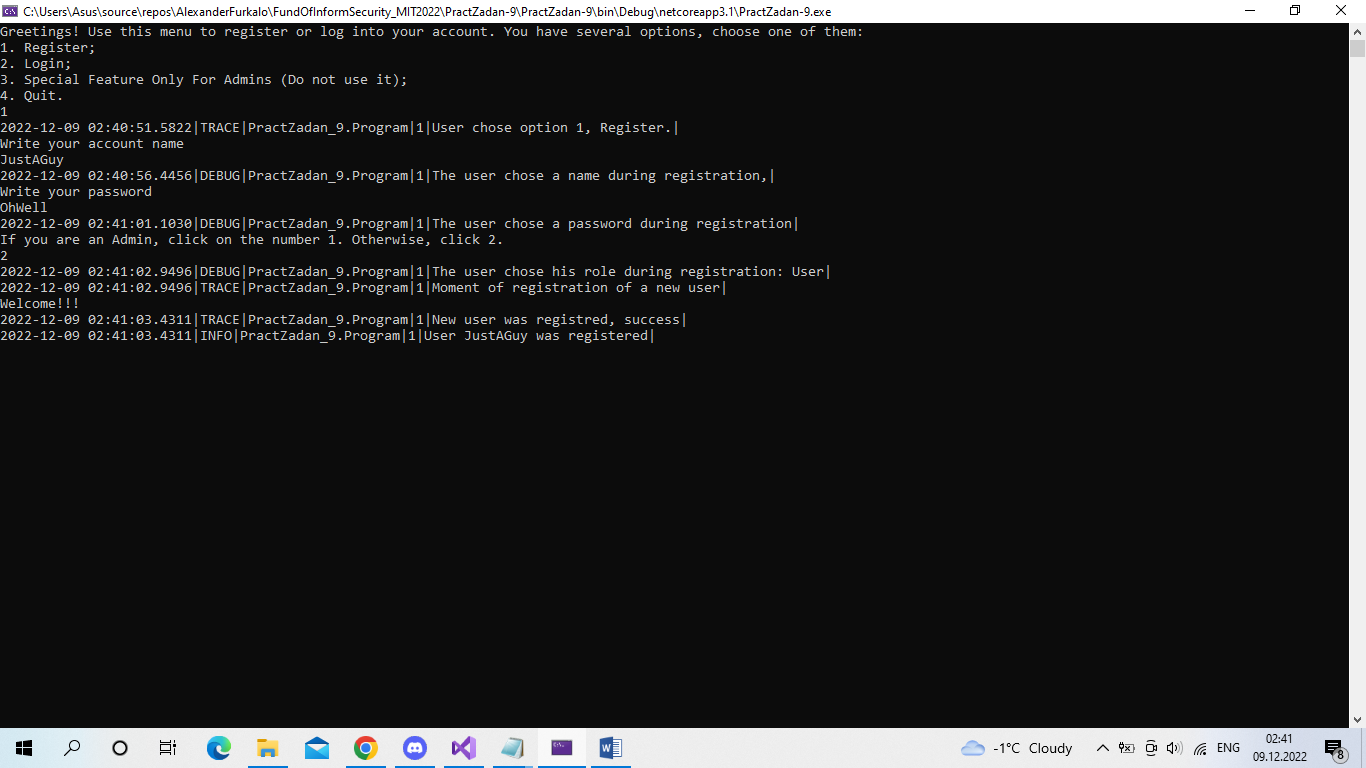
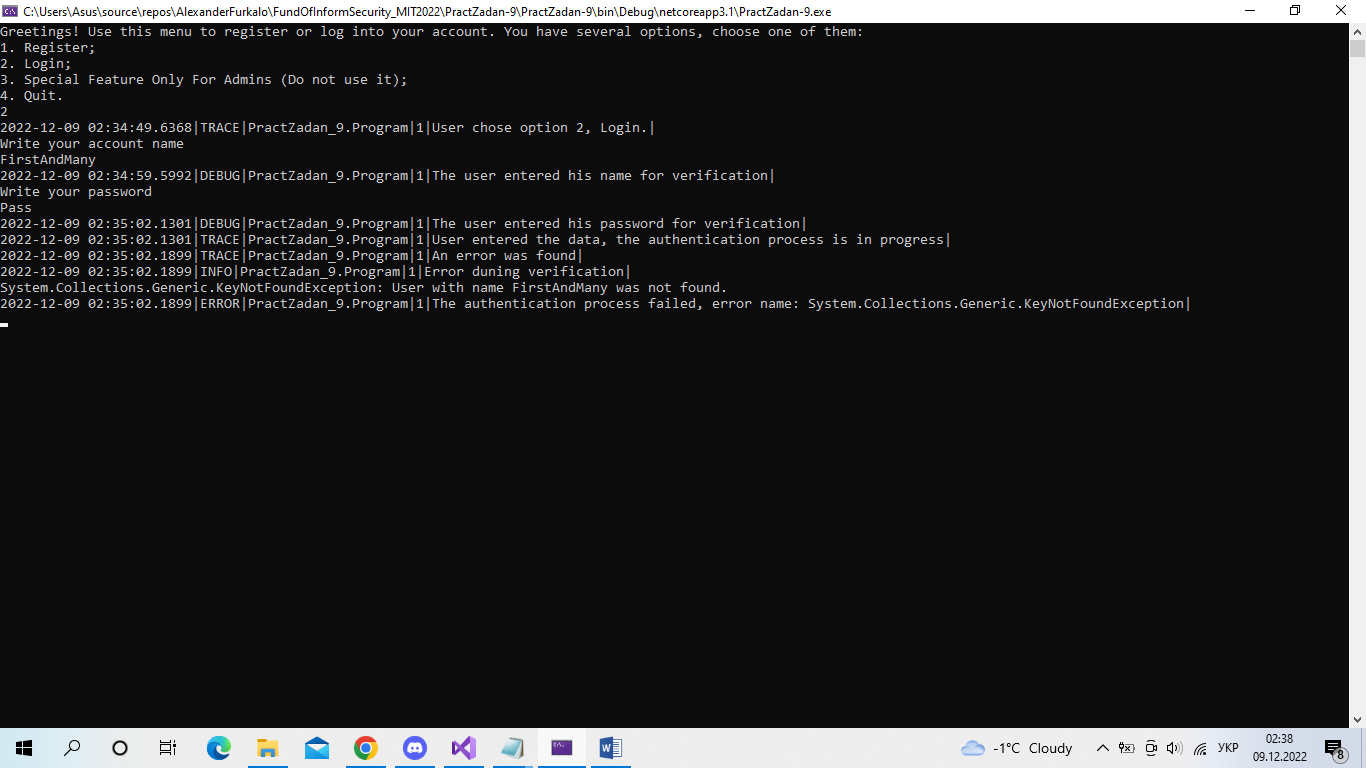
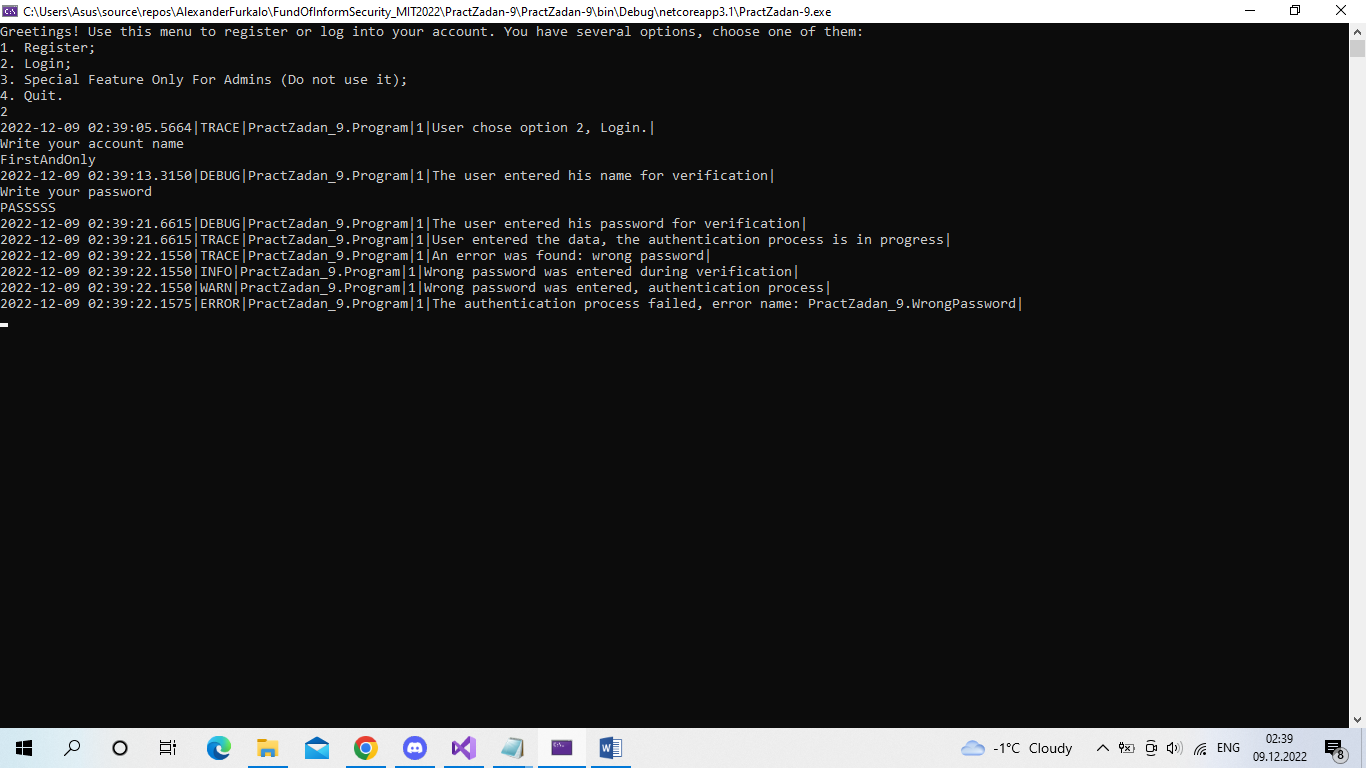
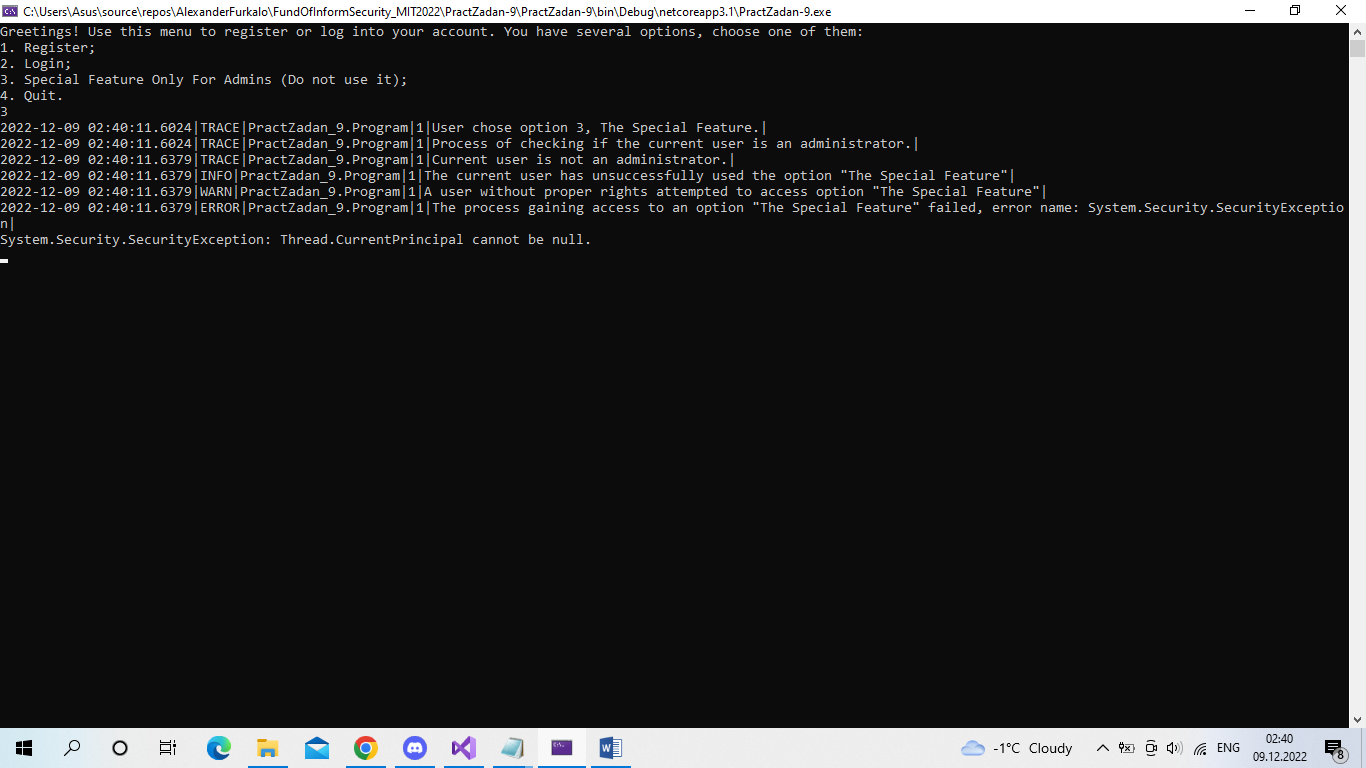
Завдання цієї практичної роботи були такі:  
1. Взяти за основу проект попереднього завдання та скопіювати його. В існуючий проект додати функціонал логування. Для логування потрібно застосувати всі рівні. Рівні Trace, Debug, Info - виводити логи на консоль, включати додаткову інформацію (дата/час, Проект/клас/метод, попереднє значення змінної / нове значення змінної). Рівні Warn, Error, Fatal - на консоль та у файл, включати додаткову інформацію (дата/час, Проект/клас/метод, попереднє значення змінної / нове значення змінної, опис підозрілої дії / екземпляр типу Exception);  
2. Оформити звіт.

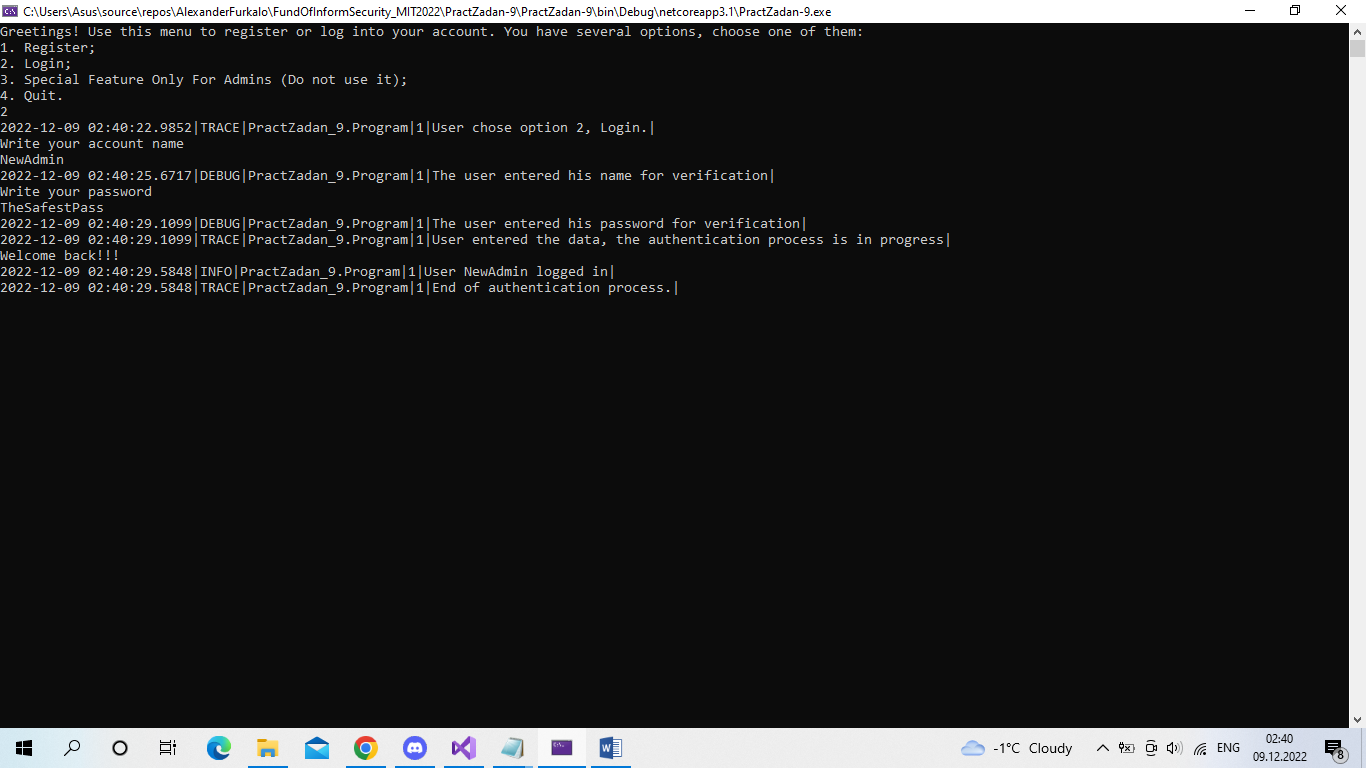
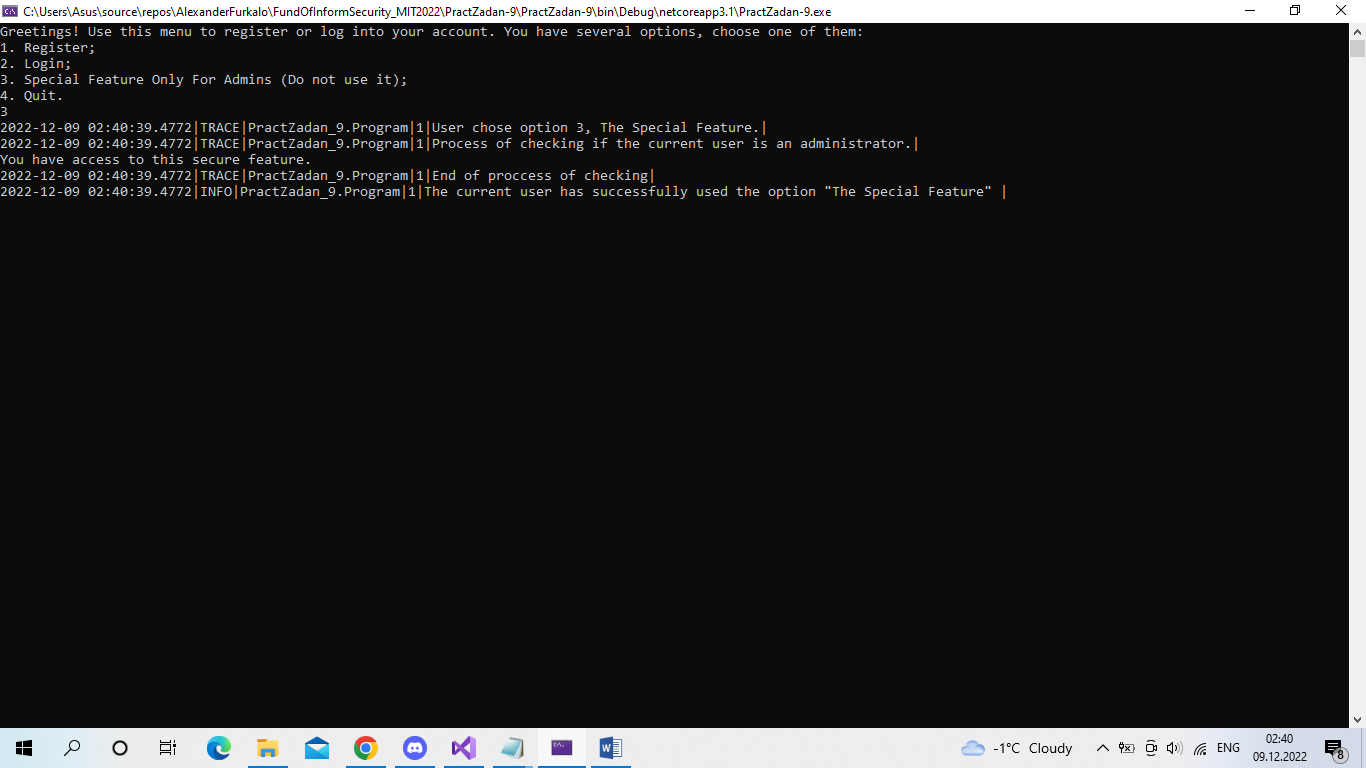
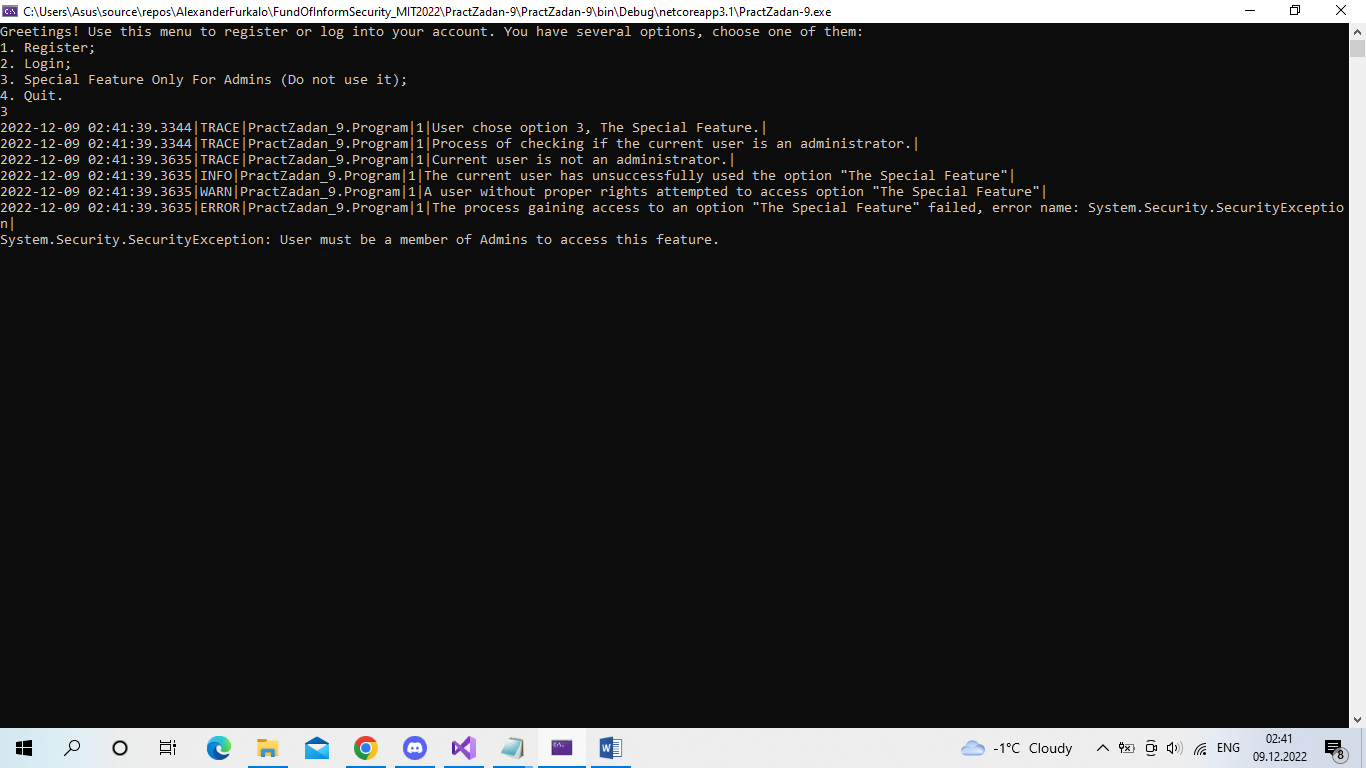
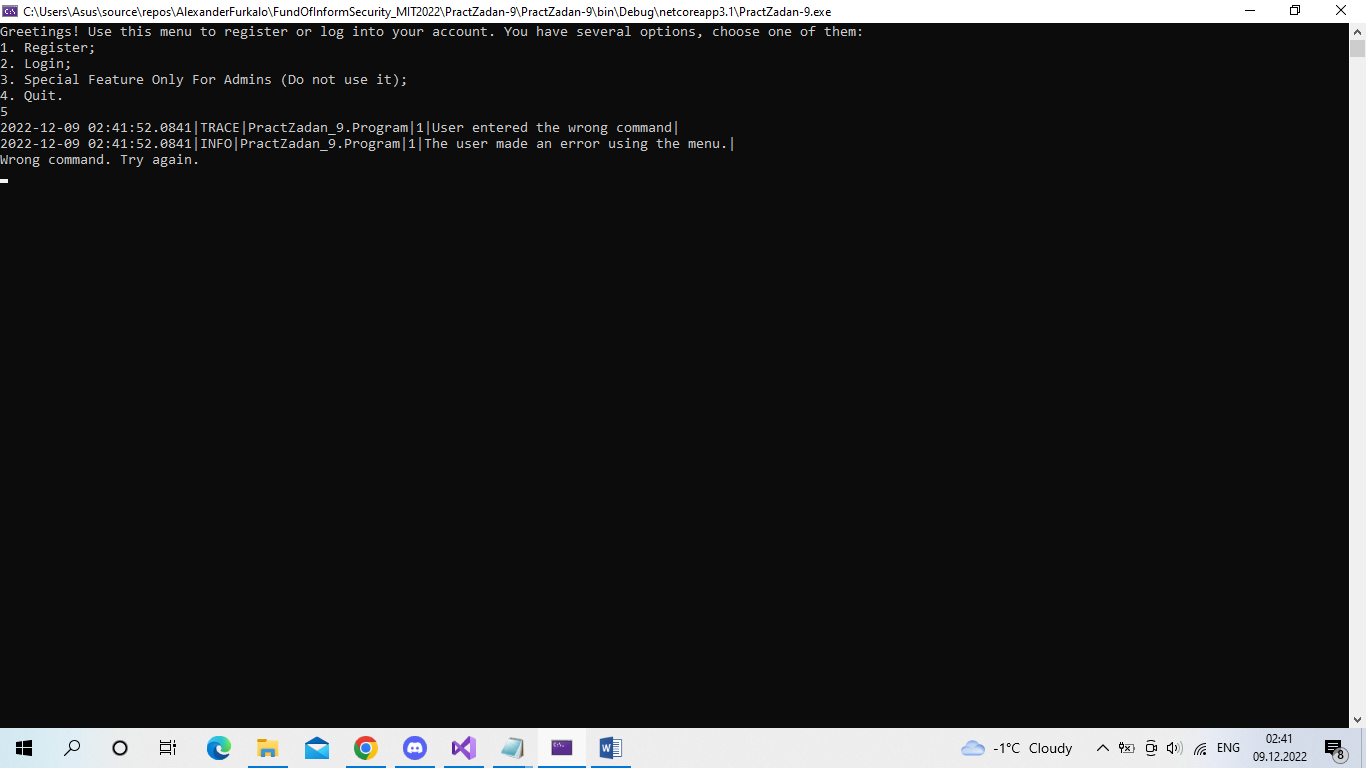
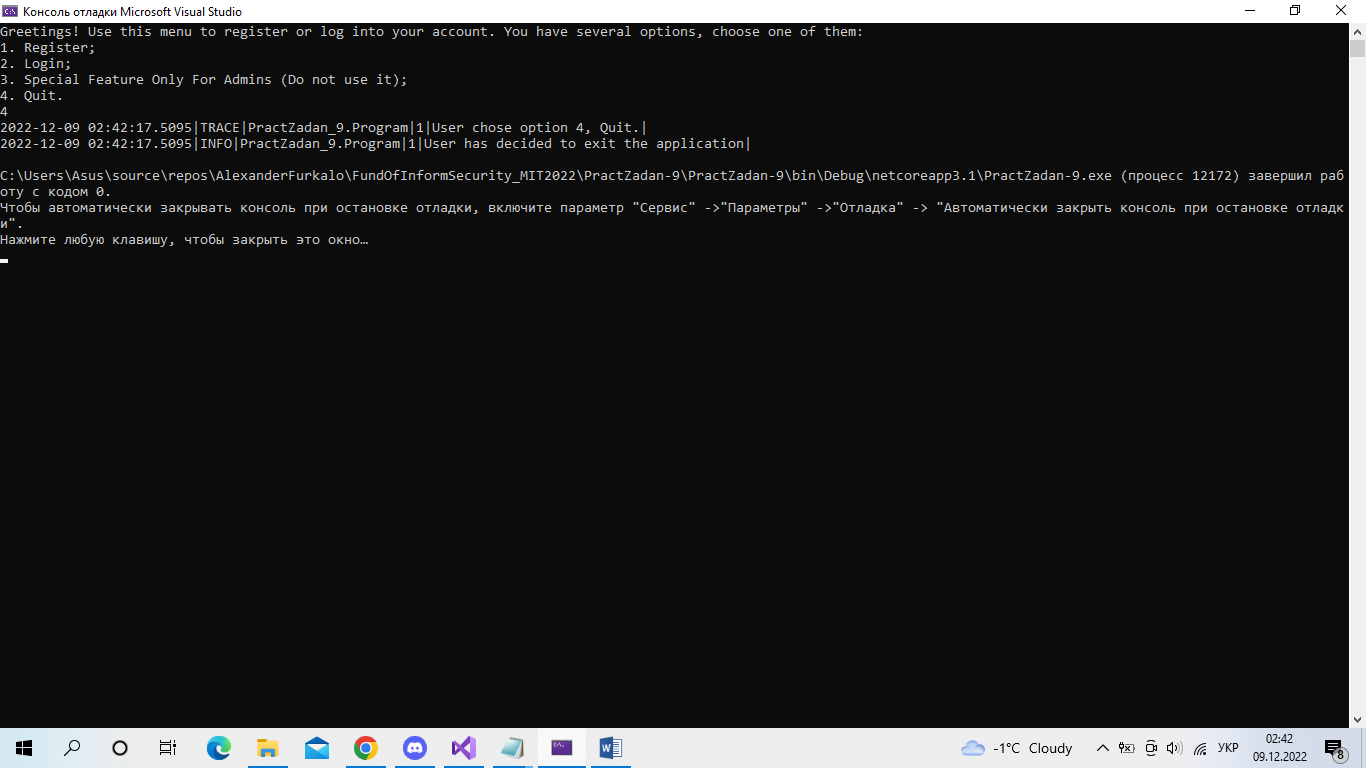
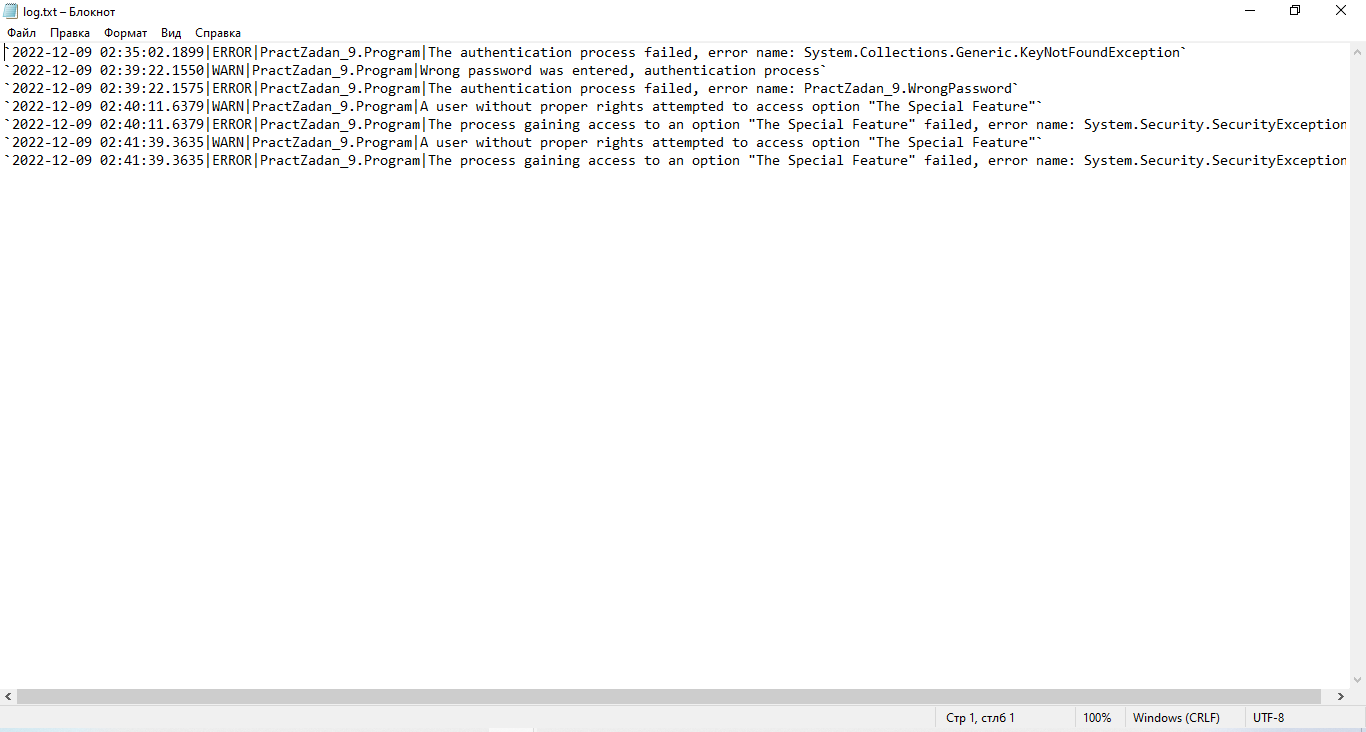
Логування – хронологічний запис подій, параметрів, станів системи, значень змінних, помилок та оброблених-необроблених ситуацій. запис логів дозволяє відповісти на питання, що відбувалося, коли та за яких обставин. Логування потрібне для постійного моніторингу програми, на відстеження проведених дій та виникнення будь-яких помилок у роботі. Завдяки логам, можна швидко зрозуміти, які дії призвели до якихось помилок, завдяки чому їх можна буде виправити швидше - застосовуватися вони можуть, наприклад, якраз для пошуку помилок.  
Логування має шість рівнів:   
- Trace – найбільш багатослівний рівень, який використовується в основному для розробки, записуються всі кроки виконання програми  
- Debug - налагодження поведінки програми з позиції внутрішніх подій, записуються зміни значення змінних та переходу у розгалуженнях або циклах.  
- Info – інформація, яка висвітлює прогрес програми, записується інформація стосовно предметної області (тобто, реєстрація та вхід/вихід користувача, у нашому випадку).   
- Warn – попередження про якісь проблеми перевірки або інші збії, записується “підозріла” активність (невірний логін або пароль)  
- Error – помилки, через які або функціонал програми вийшов з ладу, або були виявлені якісь винятки Exceptions, записуються оброблені виняткові ситуації  
- Fatal – найсерйозніші помилки та винятки, через які програма збирається припинити роботу, записуються необроблені виняткові ситуації

Перейдемо до завдання:  
  
  
  
  
  
  
Крім цього, код повністю ідентичний тому, що був у практичній роботі 8.  
Трохи наблизивши:  
  
  
  
  
  


NLog.config  
  
.csproj  


Результати роботи:  


  
  
Якщо здійснити Login з акаунта - юзера:   
  
Якщо обрати щось інше, крім цифр 1-4:  
  
Вихід  
  
Файл з повідомленнями з Warn-Error-Fatal  


Логування реалізовано завдяки NLog – бібліотека для .NET-додатків. Щоб використати її, потрібно створити файл NLog.config. Цей файл має такі теги <targets> та <rules> - перше визначає цілі та вихідні дані журналу, друге визначає правила маршрутизації журналу.   
NLog.config знаходиться у папці з практичною, log.txt записується в bin->Debug->netcoreapp3.1  
Targets має такі параметри, як xsi:type (цільовий тип, в нас вказан Файлом) та fileName (назва файлу, в який буде йти запис). У категорії layout йдуть подальші налаштування, а саме:  
- longdate: Дата й час у довгому форматі, який можна сортувати  
- level: рівень журналу або рівня  
- uppercase: Перетворює результат виведення layout на верхній регістр.  
- logger: Назва логгеру (викликається у програмі для ведення логу)  
- message: Повідомлення (відформатованого) журналу-логу  
- exception: Інформація про винятки, надана через виклик одного з методів Logger  
- threadid: Ідентифікатор поточного потоку  
Через Rules, ми визначаємо, які рівні будуть виводитися у консолі (усі рівні), та які будуть записуватися в файл (тільки Warn, Error та Fatal).  
У коді, спочатку потрібно вказати logger логер як static – його створення разом з LogManager.GetCurrentClassLogger має невеликі витрати. Після цього, достатньо вказувати у потрібних місцях Logger.ПотрібнійРівень (“Повідомлення”) – вказівки, за якими створювалися повідомлення, були написані на початку звіту (Trace – майже будь-яка дія, Debug – нові змінні та їх змінення, Info – крок реєстрації або верифікації, Warn – “підозрілі” дії, Error – помилки та винятки)

Висновки

У цій практичній роботі, ми дізналися що таке логування та змогли реалізувати його функціонал у проекті.