Практическая работа 1

Задание 1

Тест-кейсы

Id	Приоритет	Модуль	Подмодуль	Название	Шаги	Результат
1	Высокий	Hello Printer	Generate Message	Проверка количества строк в сообщении	1. Вызвать метод GenerateMessage. 2. Разделить результат на строки по символу новой строки. 3. Проверить, что количество строк = 3.	Сообщение должно содержать 3 строки.
2	Высокий	Hello Printer	Generate Message	Проверка первой строки сообщения	1. Вызвать метод GenerateMessage. 2. Разделить результат на строки по символу новой строки. 3. Проверить, что первая строка = "Hello, world!".	Первая строка должна быть "Hello, world!".
3	Средний	Hello Printer	Generate Message	Проверка второй строки сообщения	1. Вызвать метод GenerateMessage. 2. Разделить результат на строки по символу новой строки. 3. Проверить, что вторая строка = "And hi again!".	Вторая строка должна быть "And hi again!".
4	Средний	Hello Printer	Generate Message	Проверка количества восклицательных знаков в последней строке	1. Создать объект Random с фиксированным значением (например, 0). 2. Вызвать метод GenerateMessage с этим объектом. 3. Проверить, что количество восклицательных	Количество восклицательных знаков должно быть от 5 до 50 (включительно).

					знаков в последней строке от 5 до 50.	

Исходный код тестируемого модуля

```
internal class HelloPrinter
{
	public static void PrintMessage()
	{
		string message = GenerateMessage();
		Console.WriteLine(message);
	}
	public static string GenerateMessage(Random random = null)
	{
		random ??= new Random();
		int numberOfExclamations = random.Next(5, 51);
		return $"Hello, world!\nAnd hi again!\n{new string('!',
numberOfExclamations)}";
	}
}
```

Задание 2

Тест кейсы

Id	Приоритет	Модуль	Подмодуль	Название	Шаги	Результат
1	Высокий	HumanResourceAnalyzer	GetNumberOfPeople	Проверка обработки валидного количества человек	1. Ввести число 5. 2. Метод возвращает введенное значение.	Метод возвращает 5.
2	Высокий	HumanResourceAnalyzer	GetNumberOfPeople	Проверка обработки невалидного количества человек	1. Ввести число 30 (не в диапазоне). 2. Ввести число 5.	Метод возвращает 5.
3	Средний	HumanResourceAnalyzer	GetPeopleData	Проверка валидных обработки валидных введенных данных о человеке	1. Ввести имя "John". 2. Ввести фамилию "Doe". 3. Ввести возраст 25.	Метод возвращает список с одним объектом Person (John Doe, 25).
4	Средний	HumanResourc	GetValidName	Проверка валидации корректно	1. Ввести имя "John".	Метод возвращает "John".

		eAnalyzer		введенного имени		
5	Средний	HumanResourceAnalyzer	GetValidName	Проверка валидации некорректно введенного имени	1. Ввести "123". 2. Ввести пустую строку. 3. Ввести "John".	Метод возвращает "John".
6	Средний	HumanResourceAnalyzer	GetValidAge	Проверка валидации корректно введенного возраста	1. Ввести возраст 25	Метод возвращает 25.
7	Средний	HumanResourceAnalyzer	GetValidAge	Проверка валидации некорректно введенного возраста	1. Ввести "abc". 2. Ввести -1. 3. Ввести 125. 4. Ввести 25.	Метод возвращает 25.
8	Низкий	HumanResourceAnalyzer	DisplayStatistics	Проверка вычисления статистики	1. Передать список Person с данными: John Doe (25), Jane Smith (30), Max Brown (22).	Консоль выводит: "Youngest age: 22, Oldest age: 30, Average age: 25.67".

Исходный код тестируемого модуля

```
internal class HumanResourceAnalyzer
{
 public void Run()
 {
       int numberOfPeople = GetNumberOfPeople();
       List<Person> people = GetPeopleData(numberOfPeople);

       DisplayPeople(people);
       DisplayStatistics(people);
 }

 public int GetNumberOfPeople()
 {
       int number;
       while (true)
       {
             try
             {
                   Console.WriteLine("Enter the number of people (1-20):");
                   number = Convert.ToInt32(Console.ReadLine());
                   if (number >= 1 && number <= 20)
                   {
                         break;
                   }
                   Console.WriteLine("Enter a number in the correct range (1-20).");
```

```
            }
            catch
            {
                  Console.WriteLine("Enter a valid number!");
            }
      }
      return number;
 }

 public List<Person> GetPeopleData(int number)
 {
      var people = new List<Person>();
      for (int i = 0; i < number; i++)
      {
            string name = GetValidName($"Enter name of the {i + 1} person");
            string surname = GetValidSurname($"Enter surname of the {i + 1}
person");
            int age = GetValidAge($"Enter age of the {i + 1} person");

            people.Add(new Person { Name = name, Surname = surname, Age = age });
      }
      return people;
 }

 public string GetValidName(string prompt)
 {
      while (true)
      {
            Console.WriteLine(prompt);
            string? name = Console.ReadLine();
            if (!string.IsNullOrWhiteSpace(name) && name.All(char.IsLetter) &&
char.IsUpper(name[0]))
            {
                  return name;
            }
            Console.WriteLine("Enter a valid name (starting with an uppercase
letter and only letters)!");
      }
 }

 public string GetValidSurname(string prompt)
 {
      while (true)
      {
            Console.WriteLine(prompt);
            string? surname = Console.ReadLine();
            if (!string.IsNullOrWhiteSpace(surname) && surname.All(char.IsLetter)
&& char.IsUpper(surname[0]))
            {
                  return surname;
            }
            Console.WriteLine("Enter a valid surname (starting with an uppercase
letter and only letters)!");
      }
 }

 public int GetValidAge(string prompt)
 {
      while (true)
      {
            try
```

```
            {
                  Console.WriteLine(prompt);
                  int age = Convert.ToInt32(Console.ReadLine());
                  if (age >= 1 && age <= 120)
                  {
                        return age;
                  }
                  Console.WriteLine("Enter an age in the correct range (1-120).");
            }
            catch
            {
                  Console.WriteLine("Enter a valid number for age!");
            }
      }
 }

 public void DisplayPeople(List<Person> people)
 {
      foreach (var person in people)
      {
            Console.WriteLine($"{person.Surname} {person.Name} {person.Age}");
      }
 }

 public void DisplayStatistics(List<Person> people)
 {
      int youngestAge = people.Min(p => p.Age);
      int oldestAge = people.Max(p => p.Age);
      double averageAge = people.Average(p => p.Age);

      Console.WriteLine($"Youngest age: {youngestAge}, Oldest age: {oldestAge},
Average age: {averageAge:F2}");
 }
}
```

Задание 3

Тест кейсы

Id	Приоритет	Модуль	Подмодуль	Название	Шаги	Результат
1	Высокий	RectangularService	Run	Ввод корректных значений для длины и ширины	1. Установить ввод длины = 5 и ширины = 10. 2. Вызвать Run. 3. Проверить вывод в консоль.	Вывод: "The area of the rectangle is: 50".
2	Средний	RectangularService	Run	Ввод некорректного значения для длины	1. Установить ввод "abc" для длины, затем дважды ввести 10 для длины и ширины. 2. Вызвать Run.	Сообщение об ошибке: "Invalid input. Please enter a positive number".

					3. Проверить вывод в консоль.	
3	Средний	RectangularService	Run	Ввод некорректного значения для ширины	1. Установить ввод 10 для длины, затем "xyz" для ширины, затем дважды ввести 10 для длины и ширины. 2. Вызвать Run. 3. Проверить вывод в консоль.	Сообщение об ошибке: "Invalid input. Please enter a positive number".
4	Средний	RectangularService	Run	Ввод нулевого значения для длины	1. Установить ввод 0 для длины, затем дважды ввести 10 для длины и ширины. 2. Вызвать Run. 3. Проверить вывод в консоль.	Сообщение об ошибке: "Invalid input. Please enter a positive number".
5	Средний	RectangularService	Run	Ввод нулевого значения для ширины	1. Установить ввод 10 для длины, затем 0 для ширины, затем дважды ввести 10 для длины и ширины. 2. Вызвать Run. 3. Проверить вывод в консоль.	Сообщение об ошибке: "Invalid input. Please enter a positive number".
6	Средний	RectangularService	Run	Ввод отрицательного значения для длины	1. Установить ввод -5 для длины, затем дважды ввести 10 для длины и ширины. 2. Вызвать Run. 3. Проверить вывод в консоль.	Сообщение об ошибке: "Invalid input. Please enter a positive number".
7	Средний	RectangularService	Run	Ввод отрицательного значения для ширины	1. Установить ввод 10 для длины, затем -5 для ширины, затем дважды ввести 10 для длины и ширины. 2. Вызвать Run.	Сообщение об ошибке: "Invalid input. Please enter a positive number".

					3. Проверить вывод в консоль.	
8	Средний	RectangularService	Run	Ввод значений на границе допустимого диапазона	1. Установить ввод 1e100 для длины и ширины. 2. Вызвать Run. 3. Проверить вывод в консоль.	Вывод: "The area of the rectangle is: ...".
9	Средний	RectangularService	Run	Ввод значения, превышающего максимальное допустимое	1. Установить ввод 1e101 для длины, затем дважды ввести 10 для длины и ширины. 2. Вызвать Run. 3. Проверить вывод в консоль.	Сообщение об ошибке: "Invalid input. Please enter a positive number".
10	Высокий	RectangularService	CalculateArea	Проверка исключения при отрицательной длине	1. Создать экземпляр RectangularService. 2. Вызвать CalculateArea с длиной -1 и шириной 10.	Выброс исключения ArgumentOutOfRangeException с сообщением об ошибке.
11	Высокий	RectangularService	CalculateArea	Проверка исключения при отрицательной ширине	1. Создать экземпляр RectangularService. 2. Вызвать CalculateArea с длиной 10 и шириной -1.	Выброс исключения ArgumentOutOfRangeException с сообщением об ошибке.
12	Высокий	RectangularService	CalculateArea	Проверка исключения при длине, превышающей максимальное значение	1. Создать экземпляр RectangularService. 2. Вызвать CalculateArea с длиной 1e101 и шириной 10.	Выброс исключения ArgumentOutOfRangeException с сообщением об ошиб
13	Высокий	RectangularService	CalculateArea	Проверка исключения при ширине, превышающей максимальное значение	1. Создать экземпляр RectangularService. 2. Вызвать CalculateArea с длиной 10 и шириной 1e101.	Выброс исключения ArgumentOutOfRangeException с сообщением об ошибке.

Исходный код тестируемого модуля

```
 internal class RectangularService
 {
        public double MaxValue { get; set; } = 1e100;
       public void Run()
       {
             double length = GetDimension("Enter the length of the rectangle
(positive number less than 1e100):");
             double width = GetDimension("Enter the width of the rectangle (positive
number less than 1e100):");

             double area = CalculateArea(length, width);
             Console.WriteLine($"The area of the rectangle is: {area}");
       }

       private double GetDimension(string prompt)
       {
             double dimension;
             while (true)
             {
                   Console.WriteLine(prompt);

                   string input = Console.ReadLine();
                   bool isValid = double.TryParse(input, out dimension);

                   if (isValid && dimension > 0 && dimension <= MaxValue)
                   {
                         break;
                   }
                   else
                   {
                         Console.WriteLine("Invalid input. Please enter a positive
number greater than 0 and less than or equal to " + MaxValue);
                   }
             }

             return dimension;
       }

       public double CalculateArea(double length, double width)
       {
             if (width <= 0 || width > MaxValue )
             {
                   throw new ArgumentOutOfRangeException("width", "Width must be
positive value less than or equal to " + MaxValue);
             }
             if (length <= 0 || length > MaxValue)
             {
                   throw new ArgumentOutOfRangeException("lenght", "Length must be
positive value less than or equal to " + MaxValue);
             }

             return length * width;
       }
 }
```

Задание 4

Тест кейсы

Id	Приоритет	Модуль	Подмодуль	Название	Шаги	Результат
1	Высокий	GradientPrinter	GenerateTableRows	Проверка генерации строк таблицы	1. Вызвать метод GenerateTableRows. 2. Подсчитать количество строк в возвращенном HTML.	HTML с 256 строками (<tr> элементов).
2	Средний	GradientPrinter	GenerateTableRows	Проверка цвета первой строки	1. Вызвать метод GenerateTableRows. 2. Проверить наличие строки с цветом rgb(255, 255, 255).	Строка html таблицы с цветом rgb(255, 255, 255).
3	Средний	GradientPrinter	GenerateTableRows	Проверка цвета последней строки	1. Вызвать метод GenerateTableRows. 2. Проверить наличие строки с цветом rgb(0, 0, 0).	Строка html таблицы с цветом rgb(0, 0, 0).
4	Высокий	GradientPrinter	SaveHtmlToFile	Проверка выброса исключения при null содержимом	1. Вызвать метод SaveHtmlToFile с null аргументом.	Выброс исключения ArgumentNullException с параметром "content".
5	Высокий	GradientPrinter	SaveHtmlToFile	Проверка выброса исключения при отсутствии директории	1. Установить FilePath в недействительный путь. 2. Вызвать метод SaveHtmlToFile с корректным содержимым.	Выброс исключения InvalidOperationException с сообщением "The specified directory was not found".
6	Высокий	GradientPrinter	SaveHtmlToFile	Проверка выброса исключения при отсутствии прав на запись	1. Установить FilePath в защищенный путь. 2. Вызвать метод SaveHtmlToFile с корректным содержимым.	Выброс исключения InvalidOperationException с сообщением "You do not have permission to write".

7	Высокий	GradientPrinter	SaveHtmlToFile	Проверка создания файла	1. Установить FilePath в доступный путь. 2. Вызвать метод SaveHtmlToFile с корректным содержимым. 3. Проверить файл.	Создан файл по указанному пути.

Исходный код тестируемого модуля

```
    internal class GradientPrinter
    {
        public string FilePath { get; set; } =
"D:\\uni\\VPO\\TLWD1\\TLWD1.Task4\\table_with_gradient.html";

        public string Print()
        {
            string htmlContent = GenerateHtmlContent();
            SaveHtmlToFile(htmlContent);
            return $"HTML-file created. Path: {FilePath}";
        }

        public string GenerateHtmlContent()
        {
            string tableRows = GenerateTableRows();
            return $"""
                    <!DOCTYPE html>
                    <html>
                    <head>
                        <meta name=\"viewport\" content=\"width=device-width,
initial-scale=1.0\" />
                        <title>Index</title>
                    </head>
                    <body style=\"margin: 0px; padding: 0px 0px 0px 0px\">
                        <table style=\"width: 100%; border-collapse:
collapse;\">{tableRows}</table>
                    </body>
                    </html>
                    """;
        }

        public string GenerateTableRows()
        {
            var rows = string.Empty;

            for (int i = 255; i > -1; i--)
            {
                string color = $"rgb({i}, {i}, {i})";
                rows += $"<tr style=\"background-color: {color}; height:
3px\"><td></td></tr>";
            }

            return rows;
        }

       public void SaveHtmlToFile(string content)
```

```
        {
                if (content == null)
                {
                        throw new ArgumentNullException(nameof(content), "Content cannot
be null");
                }

                try
                {
                        using (StreamWriter streamWriter = new StreamWriter(FilePath))
                        {
                                streamWriter.WriteLine(content);
                        }
                }
                catch (UnauthorizedAccessException ex)
                {
                        throw new InvalidOperationException("You do not have permission
to write to this file path.", ex);
                }
                catch (DirectoryNotFoundException ex)
                {
                        throw new InvalidOperationException("The specified directory was
not found.", ex);
                }
                catch (IOException ex)
                {
                        throw new InvalidOperationException("An error occurred while
writing to the file.", ex);
                }
                catch (Exception ex)
                {
                        throw new InvalidOperationException("An unexpected error
occurred.", ex);
                }
        }

  }
```

Задание 5

Тест кейсы

Id	Приоритет	Модуль	Подмодуль	Название	Шаги	Результат
1	Высокий	FileSearcher	Конструктор	Проверка выброса исключения при null значении для directoryWrapper	1. Создать экземпляр FileSearcher с null аргументом. 2. Проверить выброс исключения.	Выбрасывается ArgumentNullException.

2	Средний	FileSearcher	AskForDirectoryAndExtension	Проверка установки DirectoryPath и Extension при валидном вводе	1. Смоделировать ввод в формате C:\myfolder .txt. 2. Вызвать метод.	Устанавливаются DirectoryPath в C:\myfolder и Extension в .txt.
3	Средний	FileSearcher	AskForDirectoryAndExtension	Проверка, что значения не устанавливаются при некорректном вводе	1. Смоделировать ввод C:\myfolder. 2. Вызвать метод.	Печатается сообщение об ошибке, DirectoryPath и Extension остаются null.
4	Высокий	FileSearcher	SearchFilesInDirectory	Проверка вывода ошибки при отсутствии директории	1. Установить DirectoryPath в C:\myfolder. 2. Вызвать метод.	Печатается сообщение "The directory 'C:\myfolder' does not exist."
5	Высокий	FileSearcher	SearchFilesInDirectory	Проверка вызова метода SearchFiles при существующей директории	1. Установить существующий DirectoryPath и Extension. 2. Вызвать метод.	Печатается список файлов и подпапок из директории.
6	Средний	FileSearcher	SearchFiles	Проверка вывода ошибки при выбросе IOExceptio	1. Смоделировать выброс IOException в методе GetFiles. 2. Вызвать SearchFiles.	Печатается сообщение "Error fetching files from directory: ..."
7	Высокий	FileSearcher	GetFilesWithExtension	Проверка возвращения файлов при наличии подходящих файлов	1. Смоделировать возврат файлов. 2. Вызвать GetFilesWithExtension.	Возвращается массив файлов, ["file1.txt", "file2.txt"].
8	Средний	FileSearcher	GetFilesWithExtension	Проверка возврата пустого массива при исключении	1. Смоделировать выброс исключения в методе GetFiles. 2. Вызвать	Возвращается пустой массив.

					GetFilesWithExtension.	
9	Высокий	FileSearcher	GetSubDirectories	Проверка возврата подпапок при наличии таковых	1. Смоделировать возврат подпапок. 2. Вызвать GetSubDirectories.	Возвращается массив подпапок, ["subdir1", "subdir2"].
10	Средний	FileSearcher	GetSubDirectories	Проверка возврата пустого массива при исключении	1. Смоделировать выброс исключения в методе GetDirectories. 2. Вызвать GetSubDirectories.	Возвращается пустой массив.

Исходный код тестируемого модуля

```
 public class FileSearcher
 {
       public readonly IDirectoryWrapper _directoryWrapper;
       public string Extension { get; set; }
       public string DirectoryPath { get; set; }

       public FileSearcher(IDirectoryWrapper directoryWrapper)
       {
             _directoryWrapper = directoryWrapper ?? throw new
ArgumentNullException(nameof(directoryWrapper));
       }

       public void AskForDirectoryAndExtension()
       {
             Console.WriteLine("Enter <path> <extension> (e.g. C:\\myfolder .txt)");

             string input = Console.ReadLine();

             if (string.IsNullOrWhiteSpace(input))
             {
                   Console.WriteLine("Invalid input.");
                   return;
             }

             var inputParts = input.Split(' ');

             if (inputParts.Length != 2)
             {
                   Console.WriteLine("Invalid format. Please provide both directory
and extension.");
                   return;
             }

             DirectoryPath = inputParts[0];
             Extension = inputParts[1].StartsWith(".") ? inputParts[1] :
$".{inputParts[1]}";
```

```
            SearchFilesInDirectory();
        }

        public void SearchFilesInDirectory()
        {
            if (string.IsNullOrWhiteSpace(DirectoryPath) ||
string.IsNullOrWhiteSpace(Extension))
            {
                Console.WriteLine("Directory path or extension cannot be
empty.");
                return;
            }

            if (!_directoryWrapper.Exists(DirectoryPath))
            {
                Console.WriteLine($"The directory '{DirectoryPath}' does not
exist.");
                return;
            }

            try
            {
                SearchFiles(DirectoryPath);
            }
            catch (UnauthorizedAccessException ex)
            {
                Console.WriteLine($"Access denied to directory: {DirectoryPath}.
{ex.Message}");
            }
            catch (IOException ex)
            {
                Console.WriteLine($"I/O error while accessing the directory:
{DirectoryPath}. {ex.Message}");
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Unexpected error: {ex.Message}");
            }
        }

        public void SearchFiles(string directoryPath)
        {
            try
            {
                string[] files = GetFilesWithExtension(directoryPath);
                foreach (string file in files)
                {
                    Console.WriteLine(file);
                }

                string[] directories = GetSubDirectories(directoryPath);
                foreach (string directory in directories)
                {
                    Console.WriteLine(directory);
                    SearchFiles(directory);
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error during file search in directory:
{directoryPath}. {ex.Message}");
```

```
            }
        }

        public string[] GetFilesWithExtension(string directoryPath)
        {
            try
            {
                return _directoryWrapper.GetFiles(directoryPath, "*" +
Extension);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error fetching files from directory:
{directoryPath}. {ex.Message}");
                return Array.Empty<string>();
            }
        }

        public string[] GetSubDirectories(string directoryPath)
        {
            try
            {
                return _directoryWrapper.GetDirectories(directoryPath);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error fetching subdirectories from directory:
{directoryPath}. {ex.Message}");
                return Array.Empty<string>();
            }
        }
    }

    public interface IDirectoryWrapper
    {
        string[] GetFiles(string path, string searchPattern);
        string[] GetDirectories(string path);
        bool Exists(string path);
    }
```

Задание 6

Тест кейсы

Id	Приоритет	Модуль	Подмодуль	Название	Шаги	Результат
1	Высокий	DocumentDownloader	AskForURLAndSaveDir	Проверка установки URL и директории при корректном вводе	1. Моделировать ввод: http://example.com/document.pdf C:\Downloads. 2. Проверить значения URL и SaveDir.	Поля URL и SaveDir содержат http://example.com/document.pdf и C:\Downloads.

2	Средний	DocumentDownloader	AskForURLAndSaveDir	Проверка обработки некорректного URL	1. Ввести: invalid-url C:\Downloads. 2. Проверить вывод сообщения об ошибке.	Консоль выводит сообщение "Invalid URL format. Please try again."
3	Средний	DocumentDownloader	AskForURLAndSaveDir	Проверка создания директории при её отсутствии	1. Моделировать ввод: http://example.com/document.pdf C:\NewDir. 2. Проверить вызов метода CreateDirectory.	Метод CreateDirectory вызывается, и выводится сообщение "Directory does not exist. Creating directory...".
4	Высокий	DocumentDownloader	DownloadAndSave	Проверка загрузки и сохранения документа	1. Моделировать наличие директории. 2. Задать корректный URL и директорию. 3. Проверить, что данные загружаются и сохраняются.	Сообщение "Document from http://example.com/document.pdf resource was downloaded successfully to C:\Downloads" отображается, файл успешно сохранен.
5	Высокий	DocumentDownloader	DownloadAndSave	Проверка обработки некорректного URL при скачивании	1. Вызвать метод DownloadAndSaveDocumentAsync с некорректным URL.	Выброс исключения ArgumentException с сообщением "Invalid URL.".
6	Средний	DocumentDownloader	SaveFile	Проверка обработки ошибки доступа	1. Попытаться сохранить файл в защищенной директории. 2. Проверить вывод сообщения.	Консоль выводит сообщение "Access to the path is denied."
7	Средний	DocumentDownloader	SaveFile	Проверка обработки IOException	1. Задать условия для выбрасывания IOException. 2. Проверить вывод сообщения.	Консоль выводит сообщение "I/O error:

						[описание ошибки]".
8	Средний	DocumentDownloader	SaveFile	Проверка обработки непредвиденного исключения	1. Задать условия для выбрасывания общего исключения. 2. Проверить вывод сообщения.	Консоль выводит сообщение "Unexpected error when saving file: [описание ошибки]".

Исходный код тестируемого модуля

```
public class DocumentDownloader
{
	private readonly IHttpClient _httpClient;
	private readonly IFileService _fileService;

	public string URL { get; private set; }
	public string SaveDir { get; private set; }

	public DocumentDownloader(IHttpClient httpClient, IFileService fileService)
	{
		_httpClient = httpClient;
		_fileService = fileService;
	}

	public void AskForURLAndSaveDirectory()
	{
		while (true)
		{
			Console.WriteLine("Enter <url> <directory> (e.g. 
http://example.com/document.pdf C:\\Downloads)");

			string input = Console.ReadLine();
			if (string.IsNullOrWhiteSpace(input))
			{
				Console.WriteLine("Invalid input. Please try again.");
				continue;
			}

			var inputParts = input.Split(' ', 2);
			if (inputParts.Length != 2)
			{
				Console.WriteLine("Invalid format. Please provide both URL 
and directory.");
				continue;
			}

			URL = inputParts[0];
			SaveDir = inputParts[1];

			if (!IsValidUrl(URL))
			{
				Console.WriteLine("Invalid URL format. Please try again.");
				URL = null;
				continue;
```

```
                }

                if (!_fileService.Exists(SaveDir))
                {
                        Console.WriteLine("Directory does not exist. Creating
directory...");
                        try
                        {
                                _fileService.CreateDirectory(SaveDir);
                        }
                        catch (Exception ex)
                        {
                                Console.WriteLine($"Failed to create directory:
{ex.Message}");
                                continue;
                        }
                }
                break;
        }
    }

    public bool IsValidUrl(string url)
    {
        var pattern = @"^(http|https)://";
        return Uri.TryCreate(url, UriKind.Absolute, out _) &&
Regex.IsMatch(url, pattern);
    }

    public string GetFileNameFromUrl(string url)
    {
        try
        {
                string fileName = Path.GetFileName(new Uri(url).LocalPath);
                return string.IsNullOrEmpty(fileName) ? "downloaded_document" :
fileName;
        }
        catch (UriFormatException)
        {
                return "downloaded_document";
        }
    }

    public async Task<byte[]> DownloadFileAsync(string url)
    {
        HttpResponseMessage response = await _httpClient.GetAsync(url);
        response.EnsureSuccessStatusCode();
        return await response.Content.ReadAsByteArrayAsync();
    }

    public void SaveFile(string filePath, byte[] content)
    {
        try
        {
                _fileService.WriteAllBytes(filePath, content);
        }
        catch (UnauthorizedAccessException)
        {
                Console.WriteLine("Access to the path is denied.");
        }
        catch (IOException ex)
        {
```

```
                  Console.WriteLine($"I/O error: {ex.Message}");
            }
            catch (Exception ex)
            {
                  Console.WriteLine($"Unexpected error when saving file:
{ex.Message}");
            }
      }

      public async Task DownloadAndSaveDocumentAsync(string url, string
saveDirectory)
      {
            if (string.IsNullOrWhiteSpace(saveDirectory) ||
string.IsNullOrWhiteSpace(url))
            {
                  Console.WriteLine("Directory path or URL cannot be empty.");
                  return;
            }

            if (!IsValidUrl(url))
                  throw new ArgumentException("Invalid URL.");

            string fileName = GetFileNameFromUrl(url);
            string filePath = Path.Combine(saveDirectory, fileName);

            try
            {
                  byte[] fileContents = await DownloadFileAsync(url);
                  SaveFile(filePath, fileContents);
                  Console.WriteLine($"Document saved successfully to: {filePath}");
            }
            catch (HttpRequestException ex)
            {
                  Console.WriteLine($"Error downloading document: {ex.Message}");
            }
            catch (IOException ex)
            {
                  Console.WriteLine($"Error saving document: {ex.Message}");
            }
            catch (Exception ex)
            {
                  Console.WriteLine($"An unexpected error occurred: {ex.Message}");
            }

            Console.WriteLine($"Document from {url} resource was downloaded
successfully to {saveDirectory}");
      }
 }

 public interface IHttpClient
 {
      Task<HttpResponseMessage> GetAsync(string requestUri);
 }

 public interface IFileService
 {
      void WriteAllBytes(string path, byte[] bytes);
      bool Exists(string? path);
      DirectoryInfo CreateDirectory(string? path);

 }
```