Міністерство освіти і науки України

Національний університет «Запорізька Політехніка»

Кафедра програмних засобів

ЗВІТ

з розрахуноково-графічного завдання №2

з дисципліни «Soft skills, групова динаміка та комунікації» на тему:

«КОМАНДНА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

Виконали:

студент групи КНТ-122 О. А. Онищенко

студент групи КНТ-132 Р. О. Маряхін

Прийняли:

доцент: В. М. Льовкін

2023

2 КОМАНДНА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Мета роботи

2.1.1 Навчитися розробляти програмне забезпечення, працюючи в команді.

2.1.2 Навчитися оформлювати програмну документацію.

Завдання до роботи

2.3.1 Ознайомитися з основними теоретичними відомостями за темою роботи, використовуючи дані методичні вказівки, а також рекомендовану літературу.

2.3.2 Сформувати команду з двох студентів та отримати у викладача індивідуальне завдання.

2.3.3 Завести акаунт на сервісі GitHub та створити команду для роботи в Slack.

2.3.4 Узгодити розподіл зобов'язань для виконання завдання, використовуючи Slack.

2.3.5 Створити репозиторій для роботи над проектом та дозволити доступ до проекту обом користувачам.

2.3.6 Завантажити у віддалений репозиторій існуючі файли проекту.

2.3.7 Створити дві гілки проекту: для презентації ревізій та для відлагодження проекту.

2.3.8 Налаштувати доступ до Git-репозиторію в інтегрованому середовищі розробки Eclipse.

2.3.9 Виконати реалізацію проекту в Eclipse та зазначити, яка частина коду ким була розроблена.

2.3.10 Визначити додаткову функцію та реалізувати її, працюючи над нею одночасно вдвох.

2.3.11 Завершити роботу над проектом.

2.3.12 Відповідно до діючих стандартів індивідуально оформити програмний документ, узгоджений з викладачем, на розроблене програмне забезпечення.

2.3.13 Оформити звіт з роботи.

2.3.14 Відповісти на контрольні запитання.

Короткі теоретичні відомості

Для роботи з Git в інтегрованих середовищах розробки існують спеціальні плагіни, наприклад, EGit для Eclipse.

Для того щоб інсталювати даний плагін, необхідно в Eclipse обрати пункт меню Help → Eclipse Marketplace та в рядку пошуку ввести EGit, після чого інсталювати відповідний плагін.

Далі необхідно створити репозиторій на основі нового проекту. Після того, як проект створено, необхідно обрати з контекстного меню проекту Team → Share Project та виконати конфігурацію репозиторію.

Текст програми

```
# main.py file are made by Maryakhin Roman

from colorama import Fore, Back, Style
from additional import get_neighbours, update_pole
import random

cycle = 0

def print_pole(pole, count):
    if count == 0:
        print("\n  ╔═ Початкове поле:")
```

```
    else:
        print(f"  ╔═ Крок {count}:")
    for row in pole:
        print("  ╟ ", end="")
        for cell in row:
            if cell == 1:
                print("🟢 ", end="")
            else:
                print("  ", end="")
        print()
    print("  ╚═", end="")


def inputValues(size):
    # Shared Function, made together

    pole = []
    if input("   1 - ручний\n   2 - автоматичний\nВиберіть метод
введення: ") == "1":
        print(
            f"  => Введіть початкову конфігурацію поля {size}x{size} (1 -
жива клітина, 0 - мертва клітина):")
        pole = []
        for i in range(size):
            print("  -> ", end="")
            ryad = list(map(int, input().split()))
            pole.append(ryad)

    # -------- Onyshchenko Oleh part
    else:

        pole = [[random.choice([1, 0])
                 for _ in range(size)] for _ in range(size)]

    return pole
    # --------

def start_game():
    global cycle
    count = 0

    print(" " * 23 + Fore.LIGHTYELLOW_EX + "\033[1m。☆✲★▬▬▬▬▬▬▬▬▬▬▬▬▬▪
Вітаємо! ▪▬▬▬▬▬▬▬▬▬▬▬▬★✲☆。\n" + Fore.RESET + Fore.RED +
          " " * 32 + "▓▓" + Fore.RESET + " Це гра про життя клітин " +
Fore.GREEN + "▓▓" + Fore.CYAN + """
    ╔═══════════════════════════════════ ¤ Правила ¤
═══════════════════════════════════╗
    ╟ 1. Клітина вмирає якщо вона має менше двох, або більше трьох живих
клітин сусідів; ╟
```

```
    ╟ 2. Жива клітина, яка має дві або три живі клітин сусідів, продовжує
жити;          ╟
    ╟ 3. Мертва клітина, у якій рівно три живі клітини серед сусідів,
стає живою.        ╟
    ╚═══════════════════════════════════ ¤¤¤¤¤¤¤¤¤¤¤
═══════════════════════════════════╝
    \033[0m""" + Fore.RESET)

    size = int(input("  =========> Введіть розмір поля: "))

    pole = inputValues(size)

    print_pole(pole, count)
    print(' Для наступного кроку натисніть Enter. Для виходу введіть
"0"')

    boards = []

    while input() != "0":
        count += 1
        boards.append(pole)
        pole = update_pole(pole)

        print_pole(pole, count)

        if all(cell == 0 for row in pole for cell in row):
            print(" Гру завершено! На полі немає живих клітин.")
            break
        elif pole in boards:
            print(" Гру завершено! На полі немає змін.")
            break

    print(Fore.CYAN+f"\n  Завершених життєвих циклів: " +
          Fore.GREEN+f"{cycle}"+Fore.RESET)


start_game()
```

```
# Both function are made by Onyshchenko Oleh

cycle = 0


def get_neighbours(pole, row, col):
    rows = len(pole)
    cols = len(pole[0])
    count = 0
```

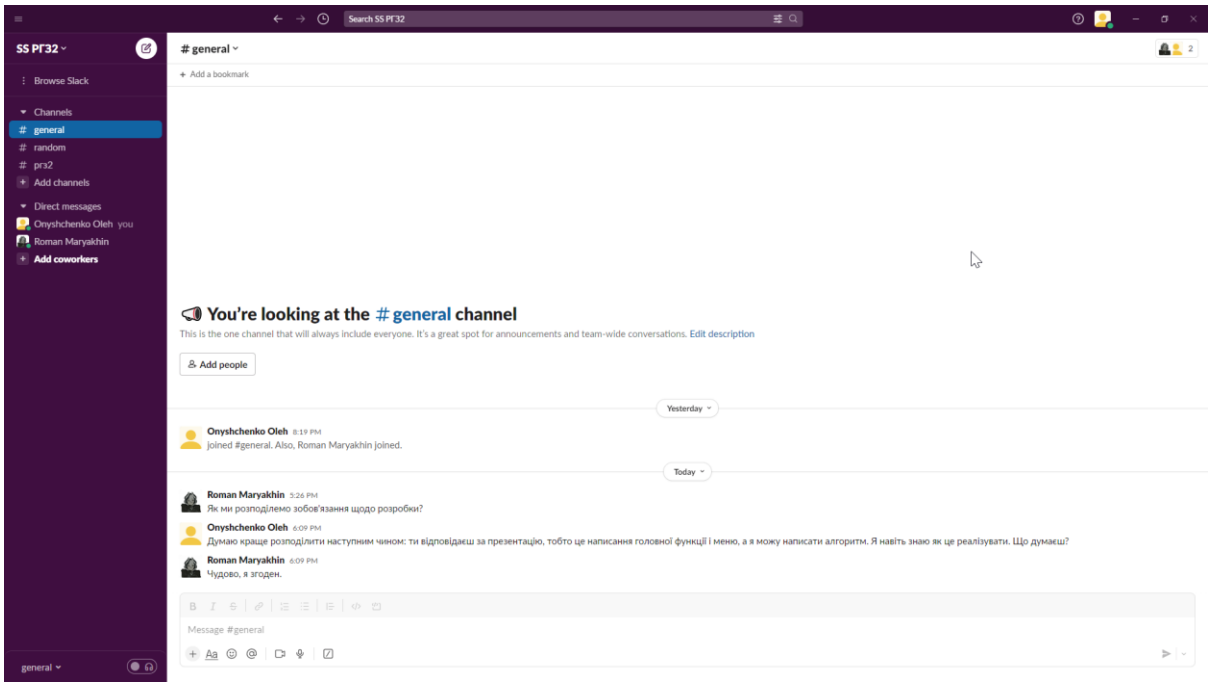
```
    offsets = [(-1, 0), (1, 0), (0, -1), (0, 1),
               (-1, -1), (-1, 1), (1, -1), (1, 1)]
    for offset in offsets:
        offset_row = row + offset[0]
        offset_col = col + offset[1]
        if 0 <= offset_row < rows and 0 <= offset_col < cols and
pole[offset_row][offset_col] == 1:
            count += 1
    return count


def update_pole(pole):
    global cycle
    new_pole = []
    for row in range(len(pole)):
        new_row = []
        for col in range(len(pole[row])):
            cell = pole[row][col]

            live_neighbours = get_neighbours(pole, row, col)

            if cell == 1:
                if live_neighbours in [2, 3]:
                    new_row.append(1)
                else:
                    cycle += 1
                    new_row.append(0)
            else:
                if live_neighbours == 3:
                    new_row.append(1)
                else:
                    new_row.append(0)

        new_pole.append(new_row)
    return new_pole
```

Опис розподілу відповідальності в команді

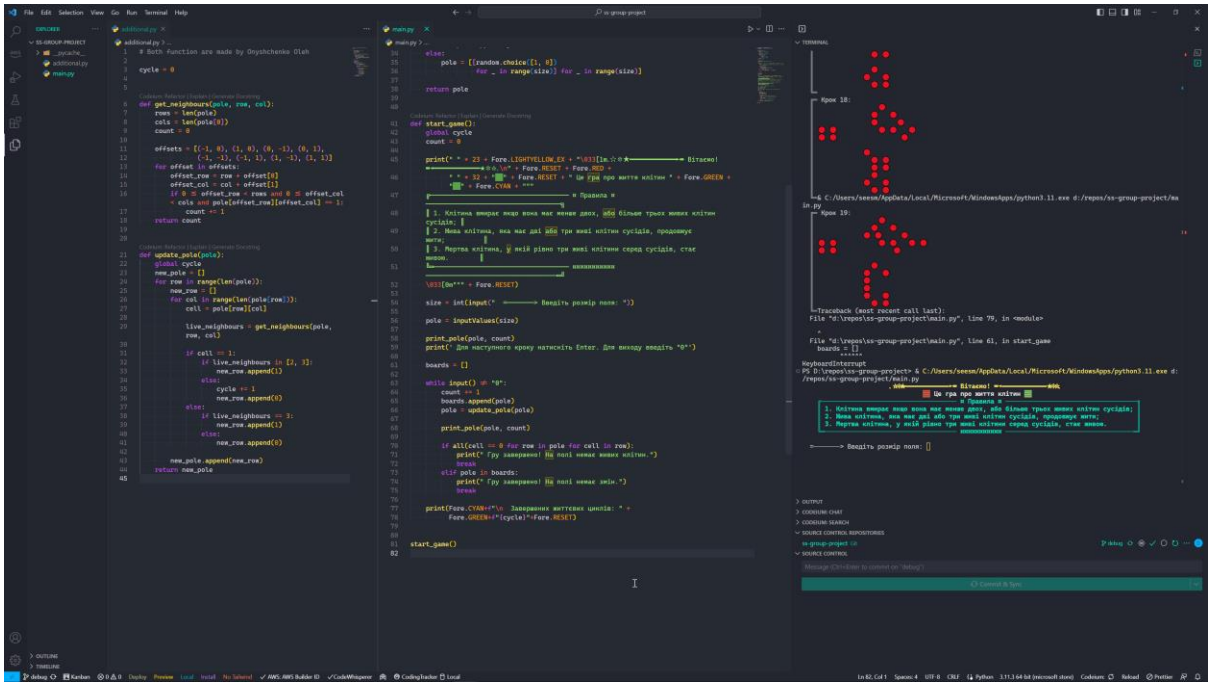
Роман відповідає за презентаційну частину програми: це меню, головна функція, а я відповідаю за реалізацію алгоритма

Копії екранних форм з результатами виконання завдань

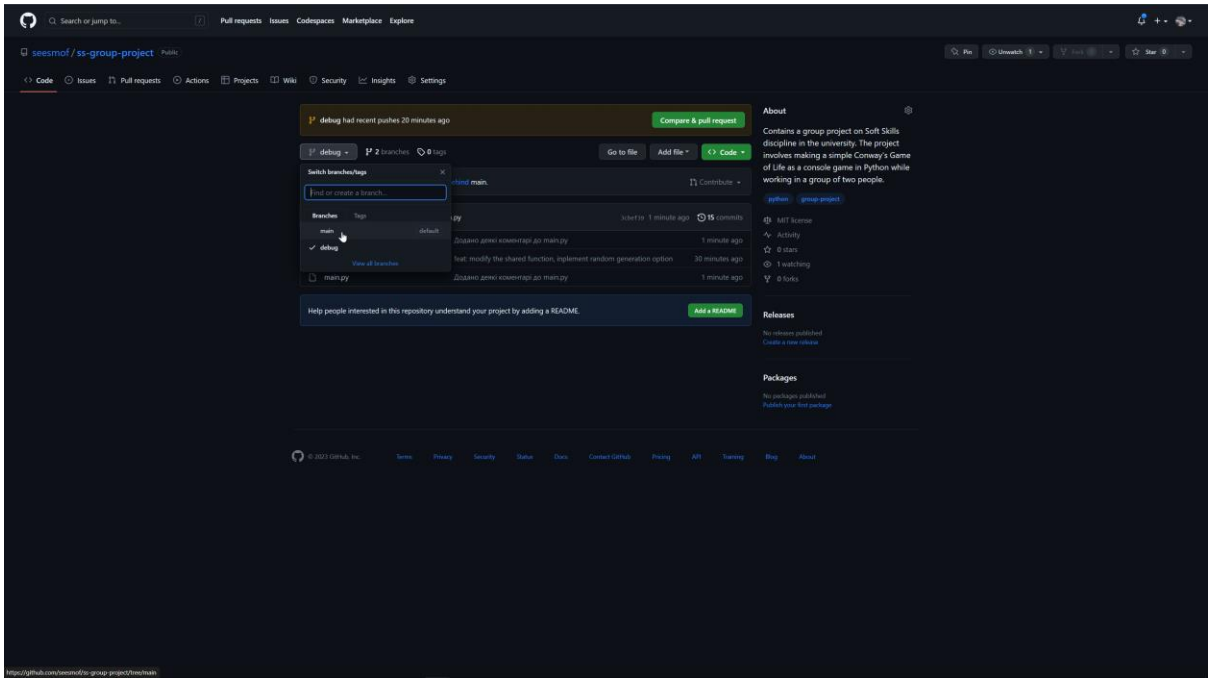
Розподіл зобов’язань



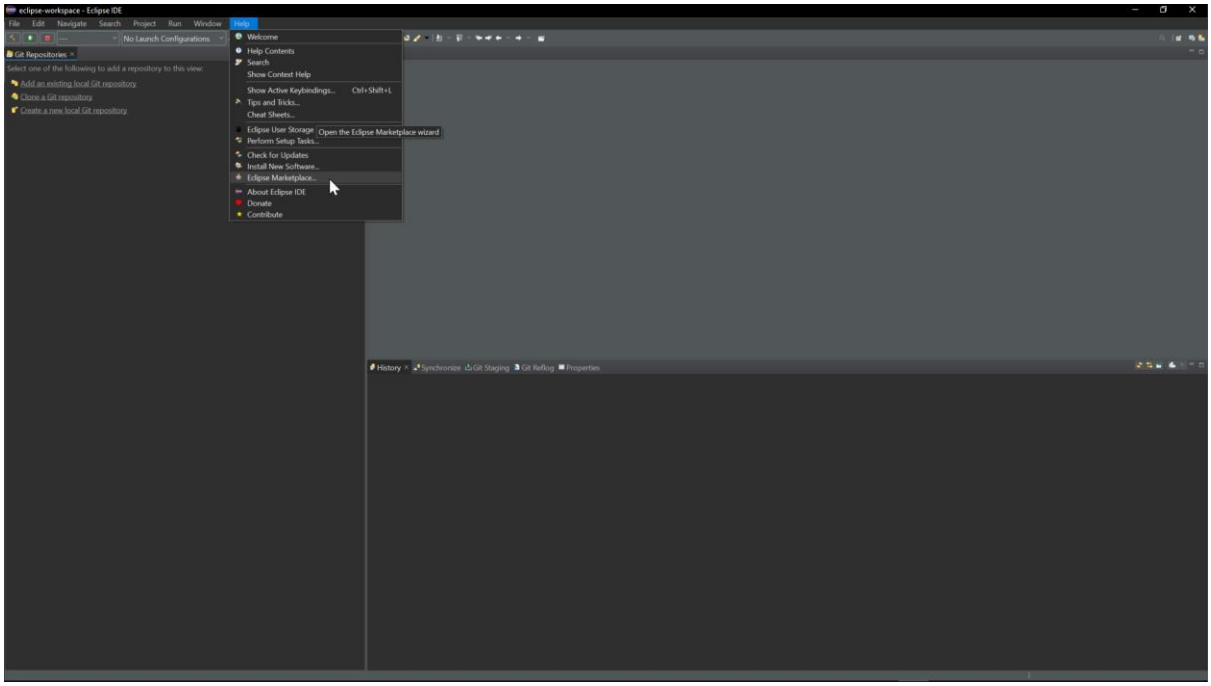
Завантаження файлів проєкту у GitHub репозиторію

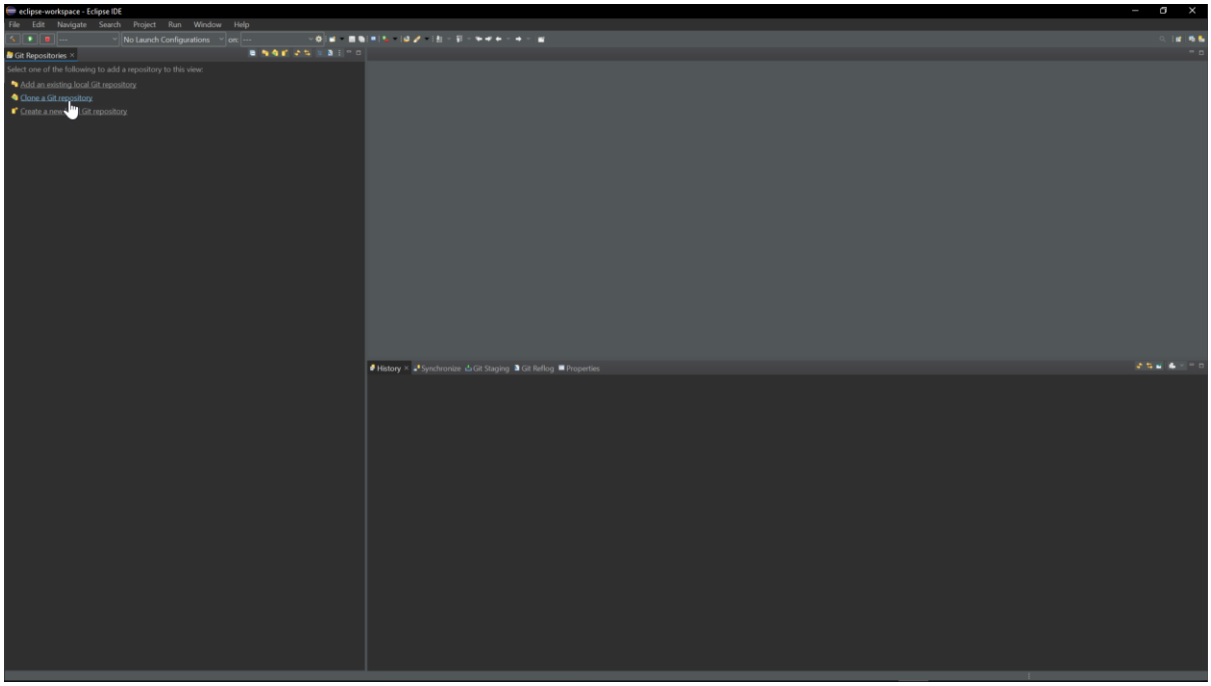


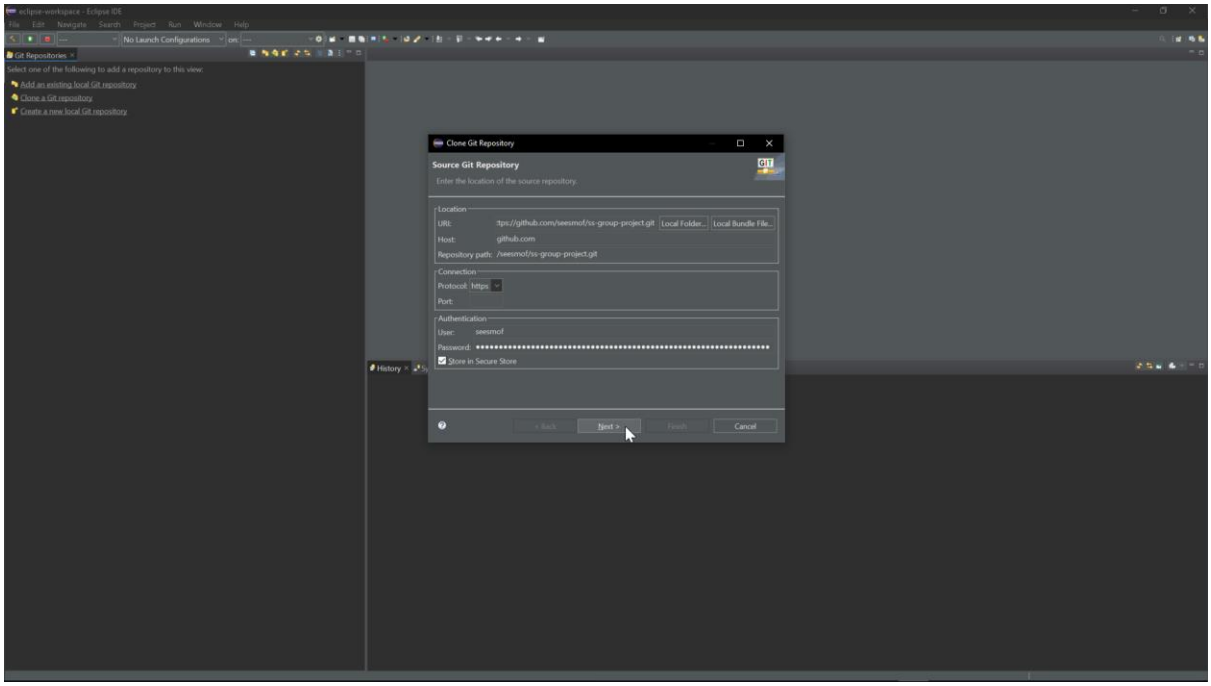
Створення гілок

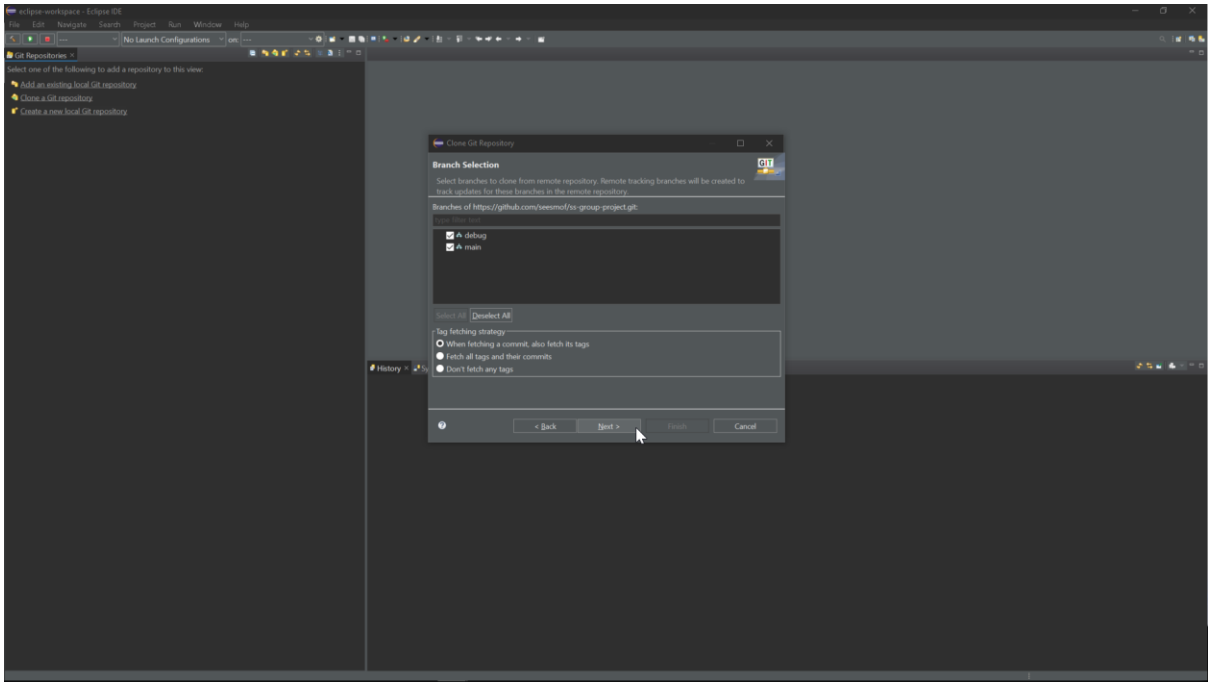


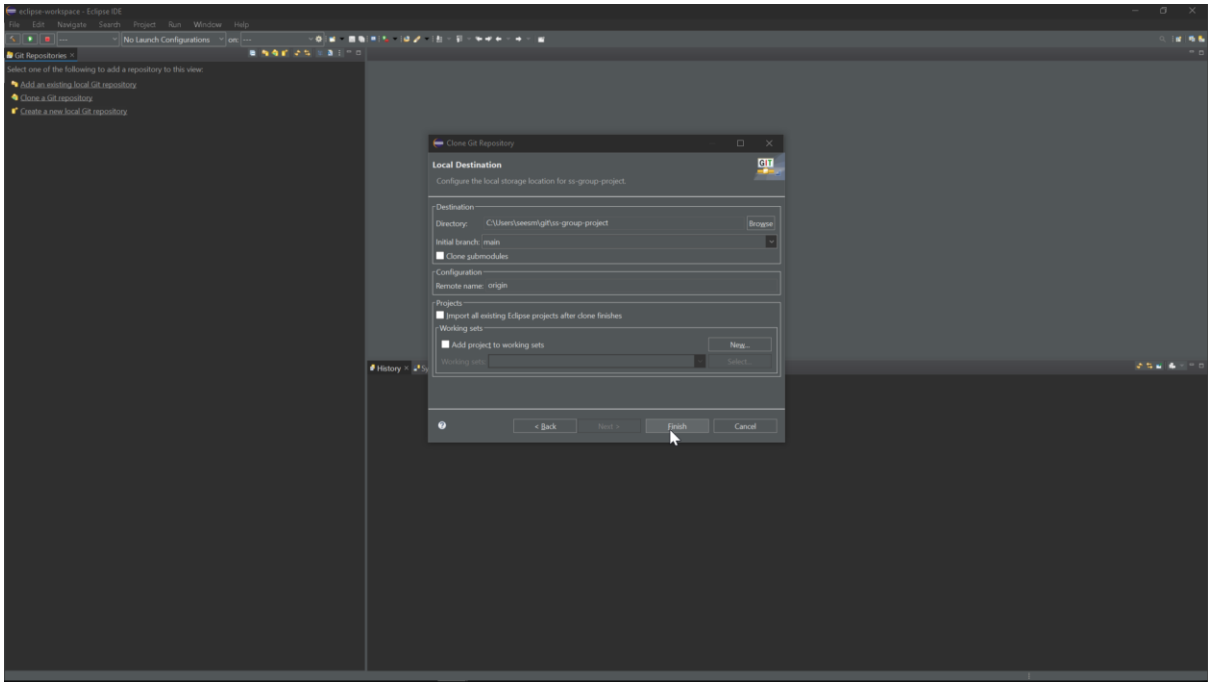
Налаштування Eclipse

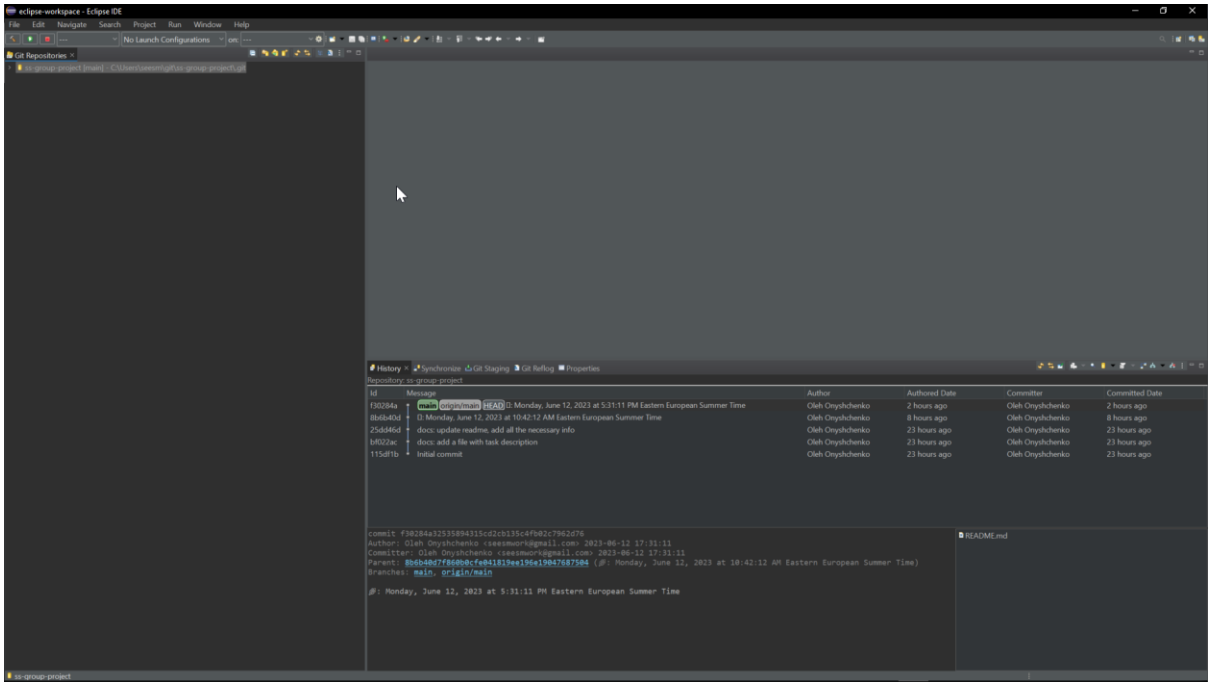



eclipse-workspace - Eclipse IDE
File Edit Navigate Search Project Run Window Help
No Launch Configurations
Git Repositories
Select one of the following to add a repository to this view:
Add an existing local Git repository
Clone a Git repository
Create a new local Git repository
History Synchronize Git Staging Git Reflog Properties


eclipse-workspace - Eclipse IDE
File Edit Navigate Search Project Run Window Help
No Launch Configurations
Git Repositories
Select one of the following to add a repository to this view:
Add an existing local Git repository
Clone a Git repository
Create a new local Git repository
Clone Git Repository
Source Git Repository
Enter the location of the source repository.
Location
URL: ttps://github.com/seesmof/ss-group-project.git
Local Folder...
Local Bundle File...
Host: github.com
Repository path: /seesmof/ss-group-project.git
Connection
Protocol: https
Port:
Authentication
User: seesmof
Password:
Store in Secure Store
< Back
Next >
Finish
Cancel
History


eclipse-workspace - Eclipse IDE
File Edit Navigate Search Project Run Window Help
No Launch Configurations
Git Repositories
Select one of the following to add a repository to this view:
Add an existing local Git repository
Clone a Git repository
Create a new local Git repository
Clone Git Repository
Branch Selection
Select branches to clone from remote repository. Remote tracking branches will be created to track updates for these branches in the remote repository.
Branches of https://github.com/seesmol/ss-group-project.git:
type filter text
debug
main
Select All
Deselect All
Tag fetching strategy
When fetching a commit, also fetch its tags
Fetch all tags and their commits
Don't fetch any tags
History
< Back
Next >
Finish
Cancel


eclipse-workspace - Eclipse IDE
File Edit Navigate Search Project Run Window Help
No Launch Configurations
Git Repositories
Select one of the following to add a repository to this view:
Add an existing local Git repository
Clone a Git repository
Create a new local Git repository
Clone Git Repository
Local Destination
Configure the local storage location for ss-group-project.
Destination
Directory: C:\Users\seesm\git\ss-group-project
Browse
Initial branch: main
Clone submodules
Configuration
Remote name: origin
Projects
Import all existing Eclipse projects after clone finishes
Working sets
Add project to working sets
New...
Working sets:
Select...
History
< Back
Next >
Finish
Cancel



Оформлений програмний документ

Програмний документ «Керівництво оператора» знаходиться у однойменному додатку до файлу.

Висновки

Таким чином, ми навчилися розробляти програмне забезпечення, працюючи в команді.

Контрольні запитання

2.5.1 Обґрунтуйте необхідність оформлення програмної документації

Необхідність оформлення програмної документації полягає в тому, що вона є важливою складовою процесу розробки програмного

забезпечення. Вона дозволяє зберігати та передавати інформацію про програму, її функції, вимоги, алгоритми, структуру та інші характеристики.

Оформлення програмної документації допомагає забезпечити належний рівень якості програмного забезпечення, зменшити ризики помилок та недоліків, а також забезпечити можливість подальшого розвитку та підтримки програми. Крім того, програмна документація є необхідною для взаємодії між розробниками, тестувальниками та користувачами програмного забезпечення.

Таким чином, оформлення програмної документації є важливим етапом у процесі розробки програмного забезпечення, який допомагає забезпечити якість та ефективність роботи програми, а також сприяє зручності її використання та підтримки.

2.5.2 Які основні документи оформлюються при розробленні програмного забезпечення?

Основні документи, які оформлюються при розробленні програмного забезпечення, це технічне завдання, специфікація, опис програми, текст програми, керівництво користувача та керівництво програміста.

2.5.4 З написання якого документу починається процес розроблення програмного забезпечення?

Процес розроблення програмного забезпечення починається з написання технічного завдання (ТЗ). Технічне завдання містить вимоги до програмного забезпечення, його функціональні та нефункціональні вимоги, умови експлуатації, вимоги до інтерфейсу користувача та інші

характеристики. ТЗ є основним документом, який визначає напрямок розробки програмного забезпечення та вимоги до нього. Він є основою для подальшого проектування, розробки та тестування програмного забезпечення.

2.5.5 Відповідно до якого стандарту оформляється документ «Технічне завдання»?

Документ "Технічне завдання" оформлюється відповідно до стандарту ДСТУ 3582-97 "Технічне завдання на розробку програмного забезпечення". Цей стандарт визначає вимоги до структури, змісту та оформлення технічного завдання на розробку програмного забезпечення. Він містить рекомендації щодо визначення вимог до програмного забезпечення, його функціональних та нефункціональних вимог, умов експлуатації, вимог до інтерфейсу користувача та інших характеристик. Дотримання стандарту ДСТУ 3582-97 дозволяє забезпечити належний рівень якості технічного завдання та зручність його використання в процесі розробки програмного забезпечення.

2.5.6 Які розділи містить технічне завдання?

Технічне завдання містить кілька розділів, а саме: вступ, підстави для розробки, призначення розробки, вимоги до програми чи програмного виробу, вимоги до програмної документації, техніко-економічні показники, стадії та етапи розробки, порядок контролю та приймання.