Хмельницький національний університет

Кафедра комп’ютерних наук

**Лабораторна робота** **№3**

з предмету:

«Основи програмної інженерії та тестування програмного забезпечення»

Виконав:

студент групи КН-22-2

Бекешко Ю.Р.

Перевірив:

Собко О. В.

Хмельницький-2023

**Варіант 1:**

Знайти довжину дуги кола, якщо відомі радіус кола та центральний кут, який вона охоплює.

**Завдання:**

Завдання 1. Зареєструватися у веб-сервісі спільної роботи GitHub. Відкоригувати дані профілю, завантаживши фотографію.

Завдання 2. Створити власний приватний репозиторій на GitHub.

Завдання 3. Надати доступ до репозиторію викладачу, знайшовши користувача за логіном.

Завдання 4. Скачати та встановити настільний клієнт GitHub Desktop на робочий комп’ютер. Авторизуватися у настільному клієнті за допомогою створеного раніше профілю GitHub.

Завдання 5. Клонувати створений репозиторій на робочий комп’ютер.

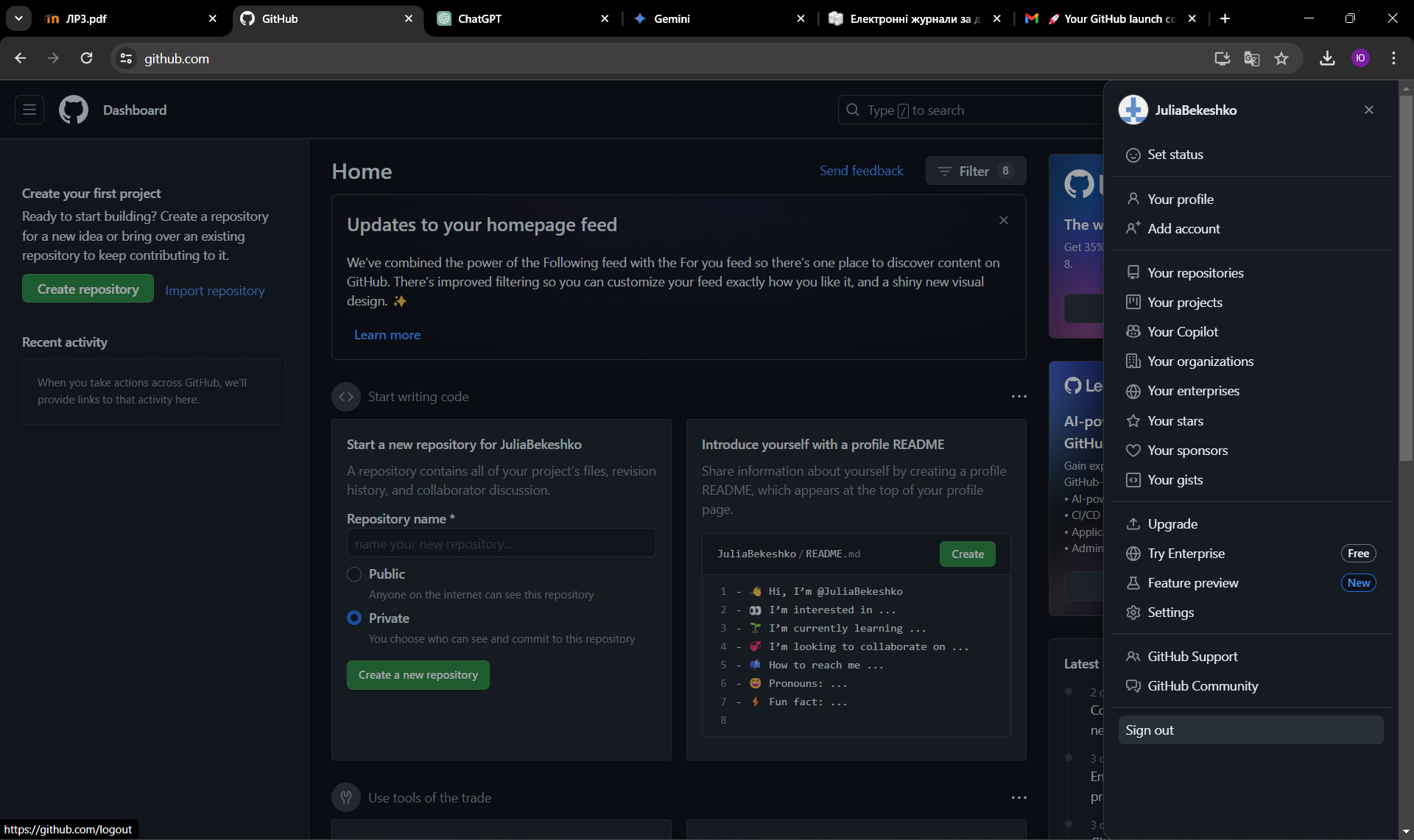
Завдання 6. Створити Windows Forms застосунок на мові програмування C# у папці локального репозиторію GitHub згідно варіанту. Варіант виконання завдання призначається викладачем індивідуально. Передбачити введення відомих даних користувачем з клавіатури, забезпечити пояснення задачі, яку вирішує застосунок.

Завдання 7. Завантажити створений програмний застосунок на віддалений сервер GitHub. За потреби внести зміни в код, створивши для цього окрему гілку, після чого виконати злиття.

Завдання 8. Оформити звіт з виконання лабораторної роботи, завантажити звіт на віддалений репозиторій GitHub.

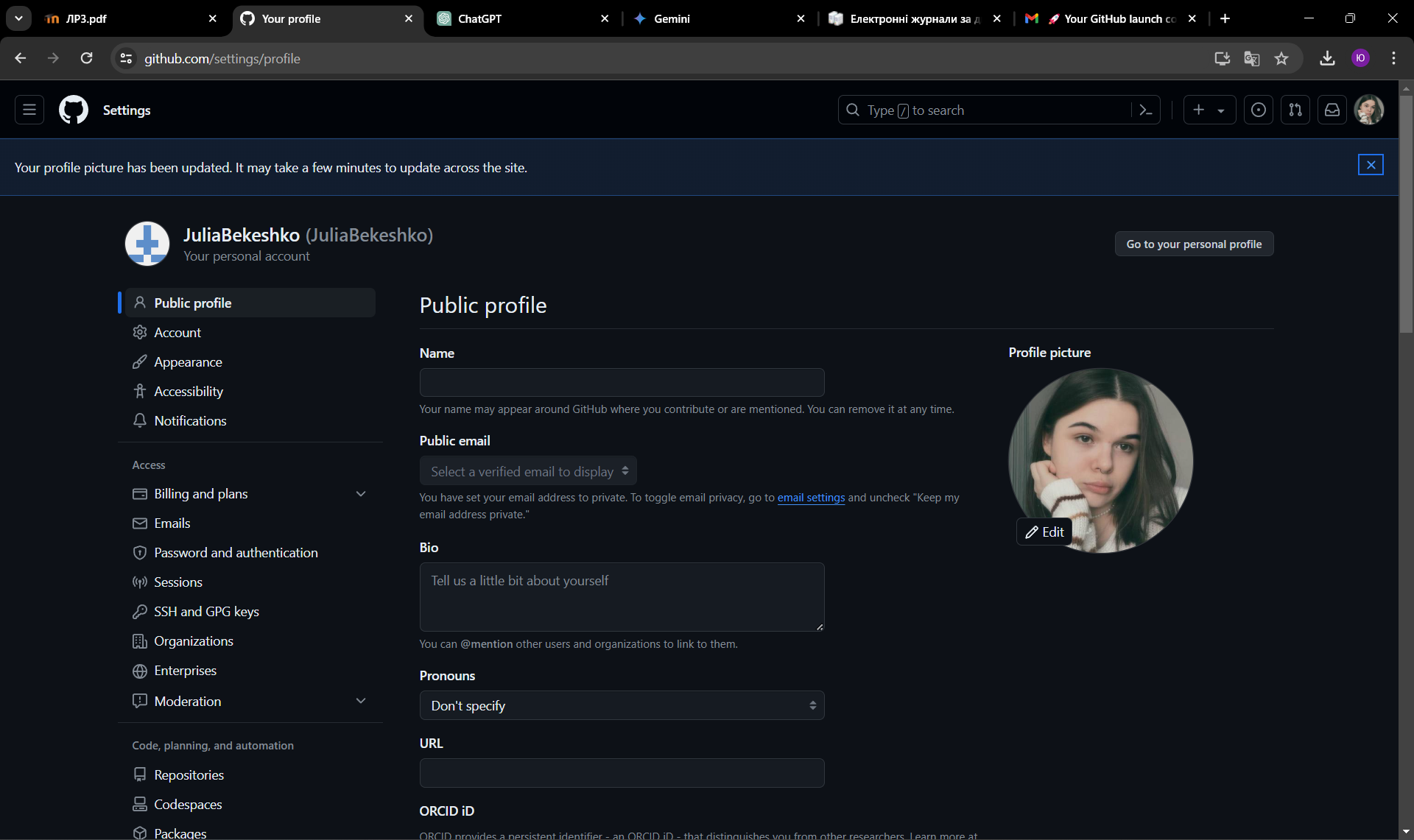
**Виконання завдання:**

Реєстрація у веб сервісі спільної роботи GitHub.



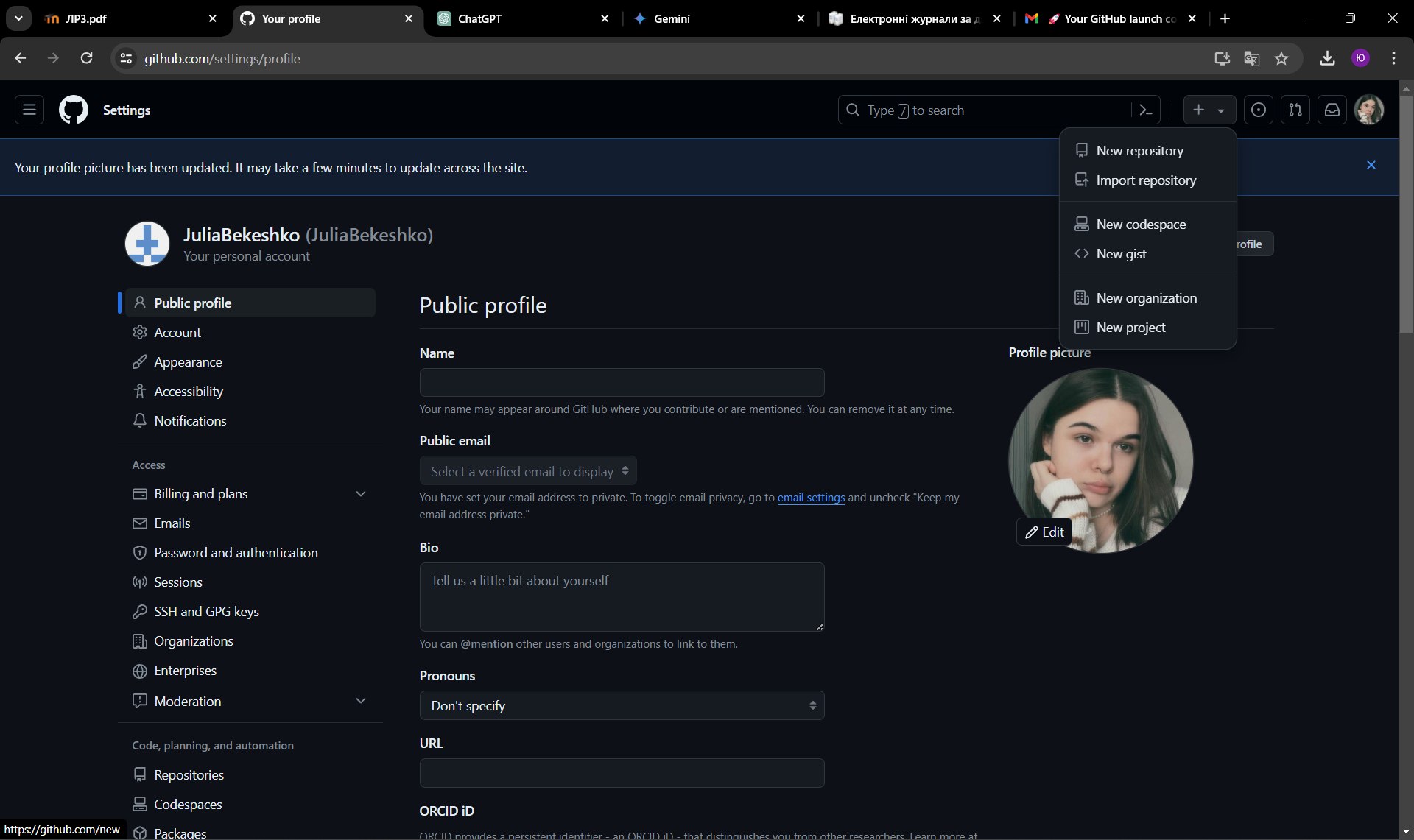
**Рисунок 1 – Перехід на сторінку налаштувань профілю користувача**

Коригування даних профілю, завантаження фотографії.

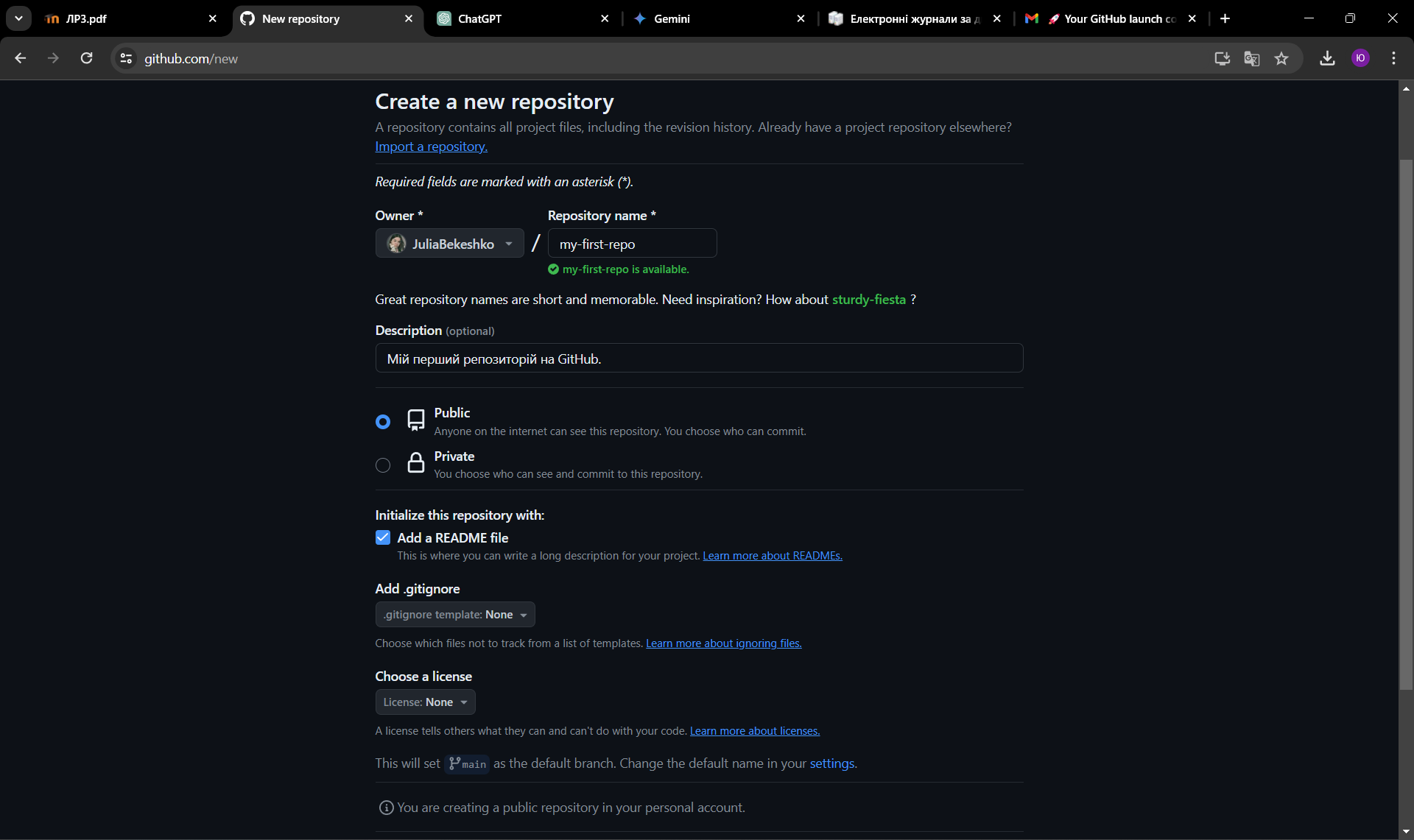


**Рисунок 2 – Завантаження фото користувача**

Було створено власний приватний репозиторій на GitHub. Для цього слід у верхньому правому куті будь-якої сторінки скористатия випадаючим меню та вибрати Новий репозиторій . Далі необхідно ввести коротку назву, яка запам’ятовується, для сховища. Наприклад, "my-first-repo. За бажанням можна додати опис сховища. Наприклад, «Мій перший репозиторій на GitHub». Виберіть видимість сховища. Далі необхідно ініціалізувати це сховище за допомогою README. Натиснути «Створити репозиторій».

****

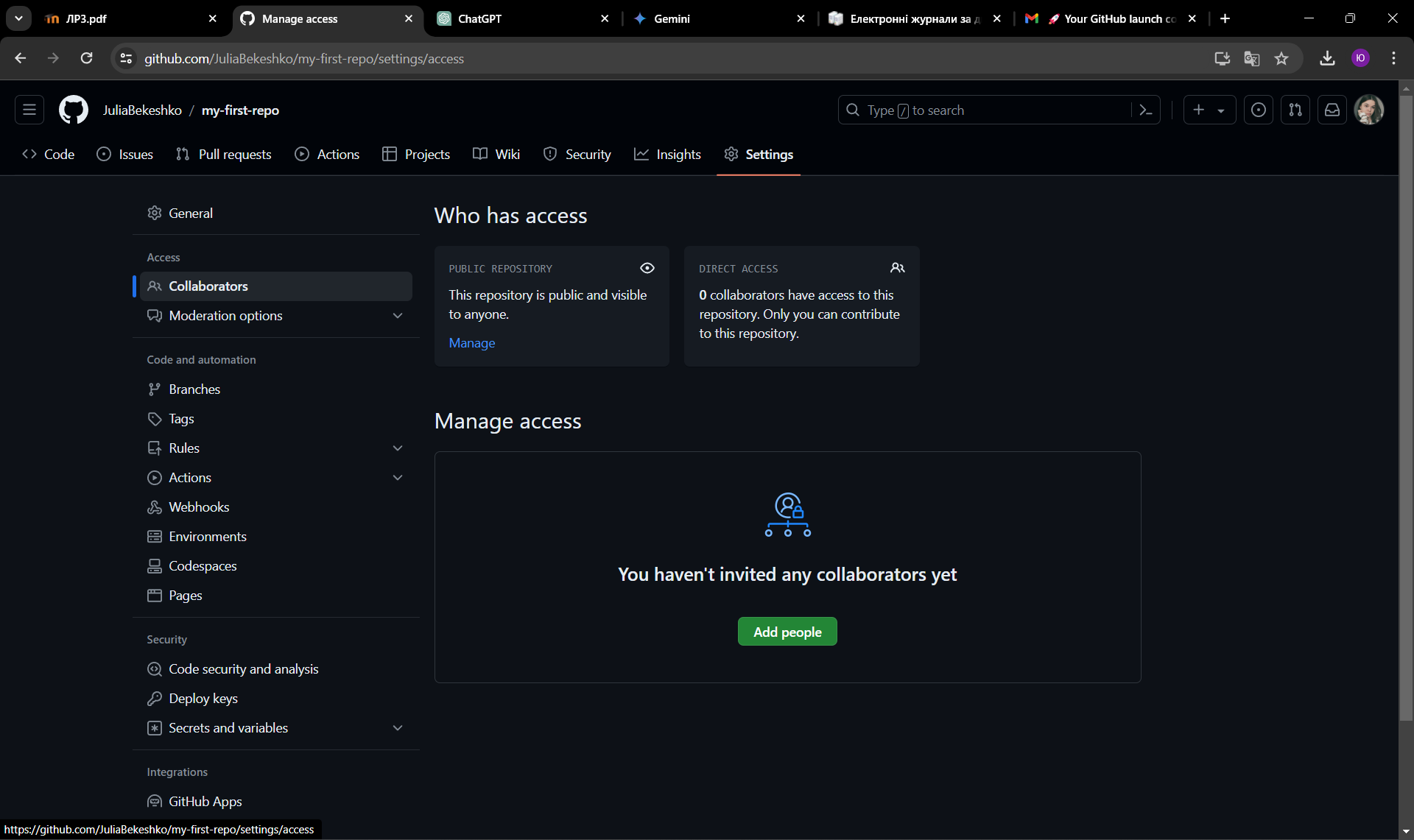
**Рисунок 3 – Створення нового репозиторію**

****

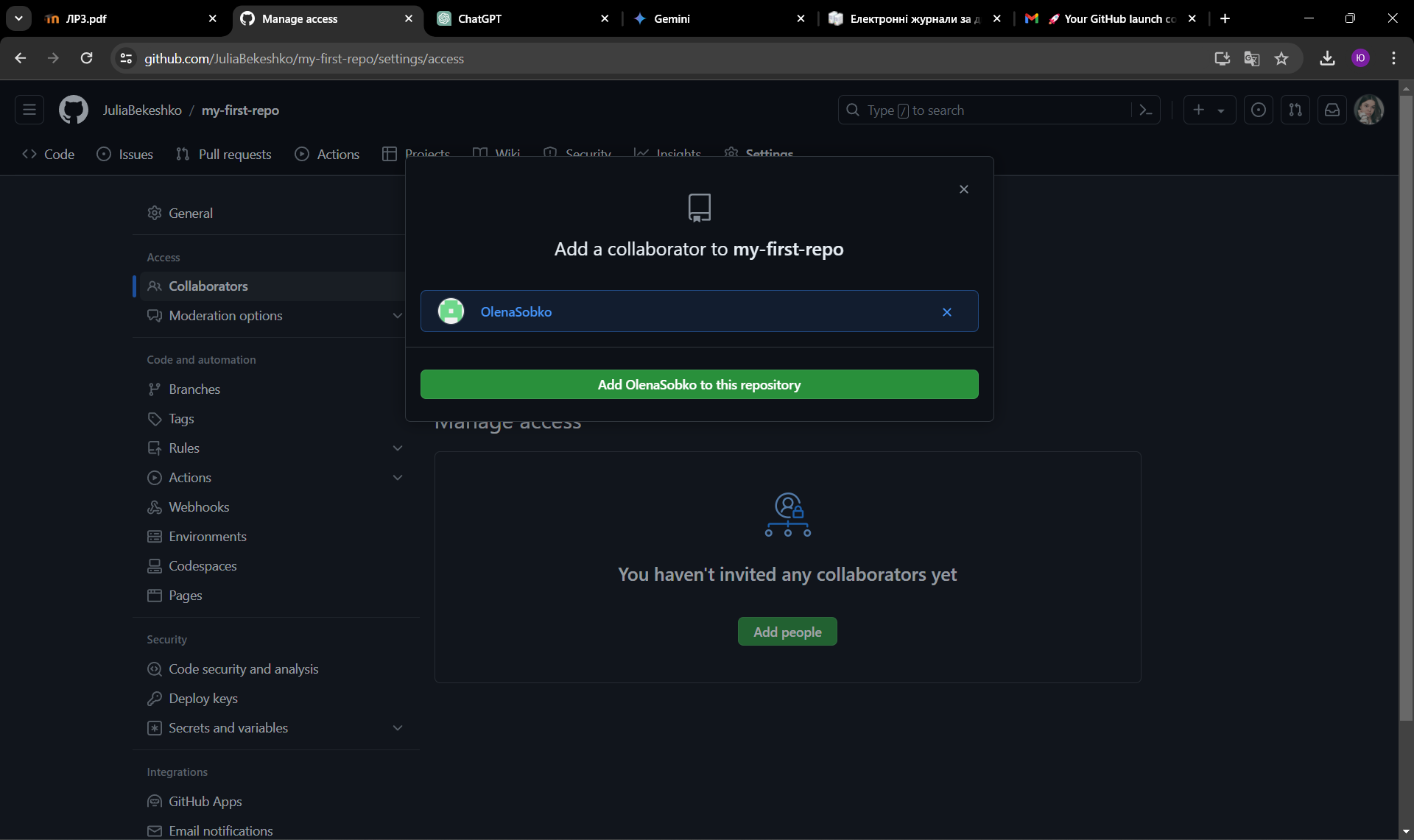
**Рисунок 4 – Меню створення нового репозиторію**

Було надано доступ до репозиторію викладачу, знайшовши користувача за логіном.

Для того, щоб надати доступ іншим користувачам до свого приватного репозиторію необіхдно запитати ім’я користувача особи, яку потрібно запросити як співавтора. На GitHub.com необхідно перейти на головну сторінку сховища. Під назвою свого сховища клацнути Налаштування. У розділі «Доступ» на бічній панелі натиснути «Співавтори». Натиснути «Додати людей» . У полі пошуку потрібно ввести ім’я особи, яку необхідно запросити, а потім клацнути ім’я у списку збігів. Користувач отримає електронний лист із запрошенням до сховища.



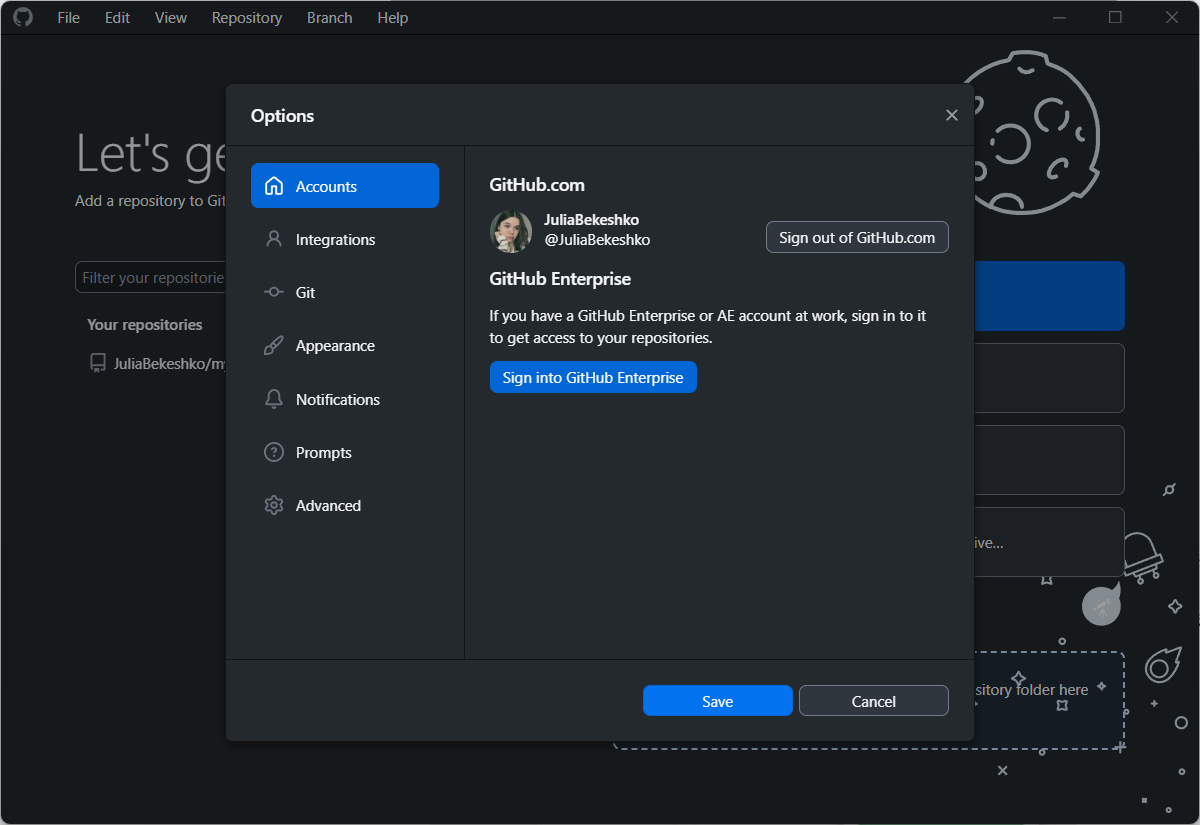
**Рисунок 5 – Меню надання доступу до репозиторію користувачам**

****

**Рисунок 6 – Запрошення користувача до репозиторію**

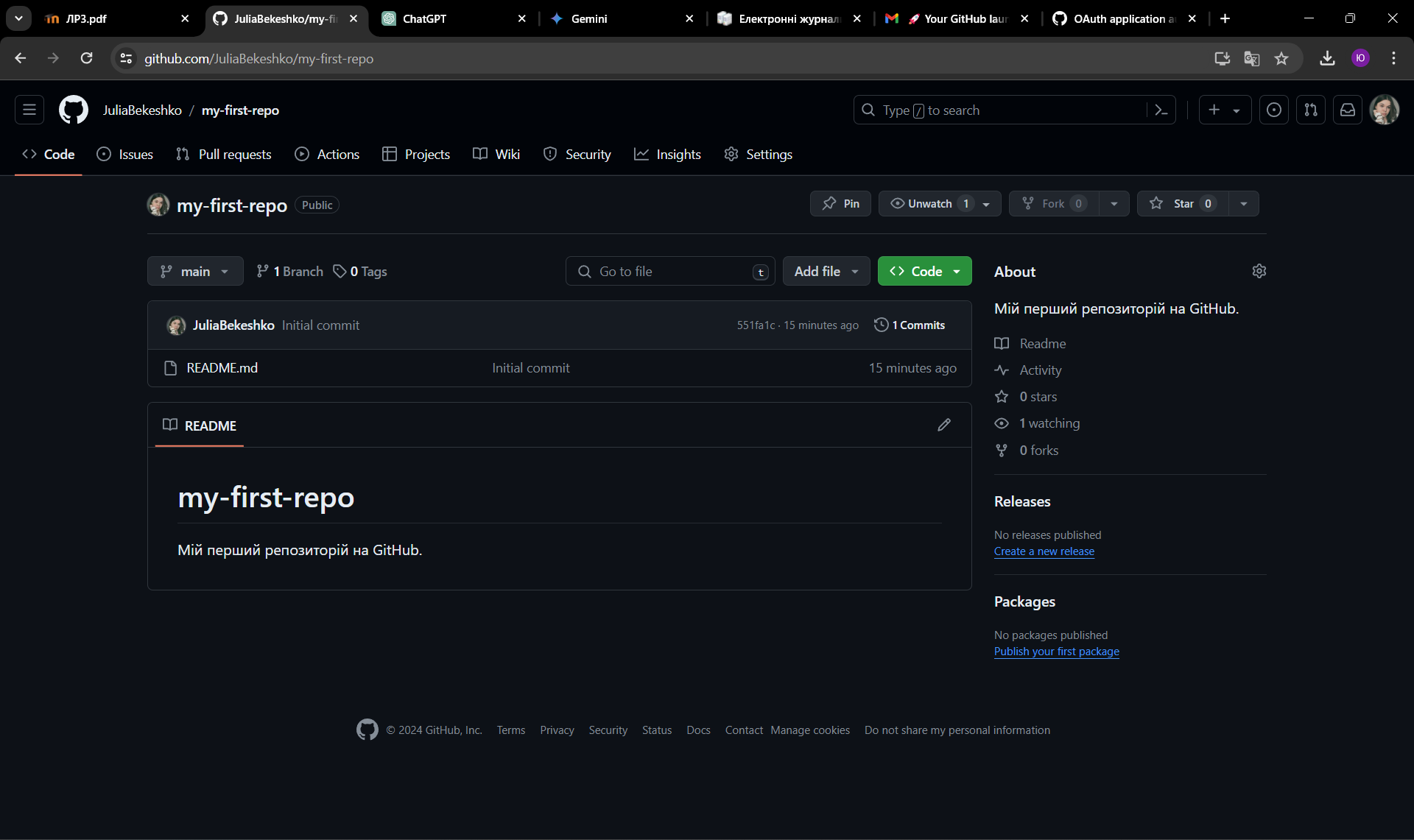
Було встановлено GitHub Desktop на персональний комп’ютер.

Після встановлення GitHub Desktop було виконано авторизацію за допомогою облікового запису GitHub. Авторизація на настільному клієнті за допомогою профілю GitHub.



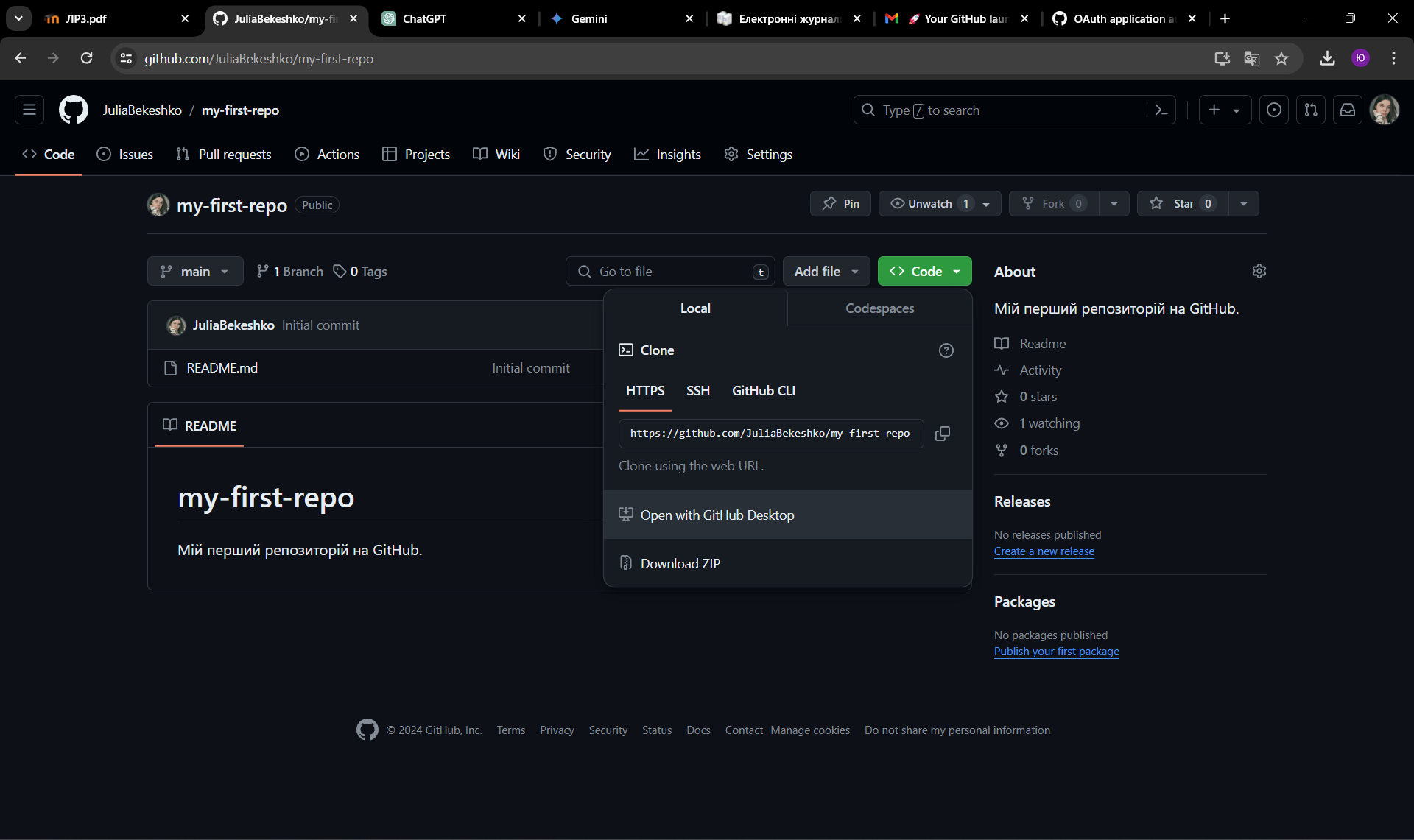
**Рисунок 7 – Авторизація на настільному клієнті за допомогою профілю GitHub**

Далі було клоновано існуючий репозиторія з віддаленого сервера. Для того, щоб клонувати вже існуючий репозиторій з віддаленого сервера GitHub на локальний, необхідно виконати наступні кроки. Увійти до GitHub.com та GitHub Desktop перед початком клонування. На GitHub.com перейти на головну сторінку репозиторію. Клацнути «Code» над списком файлів.

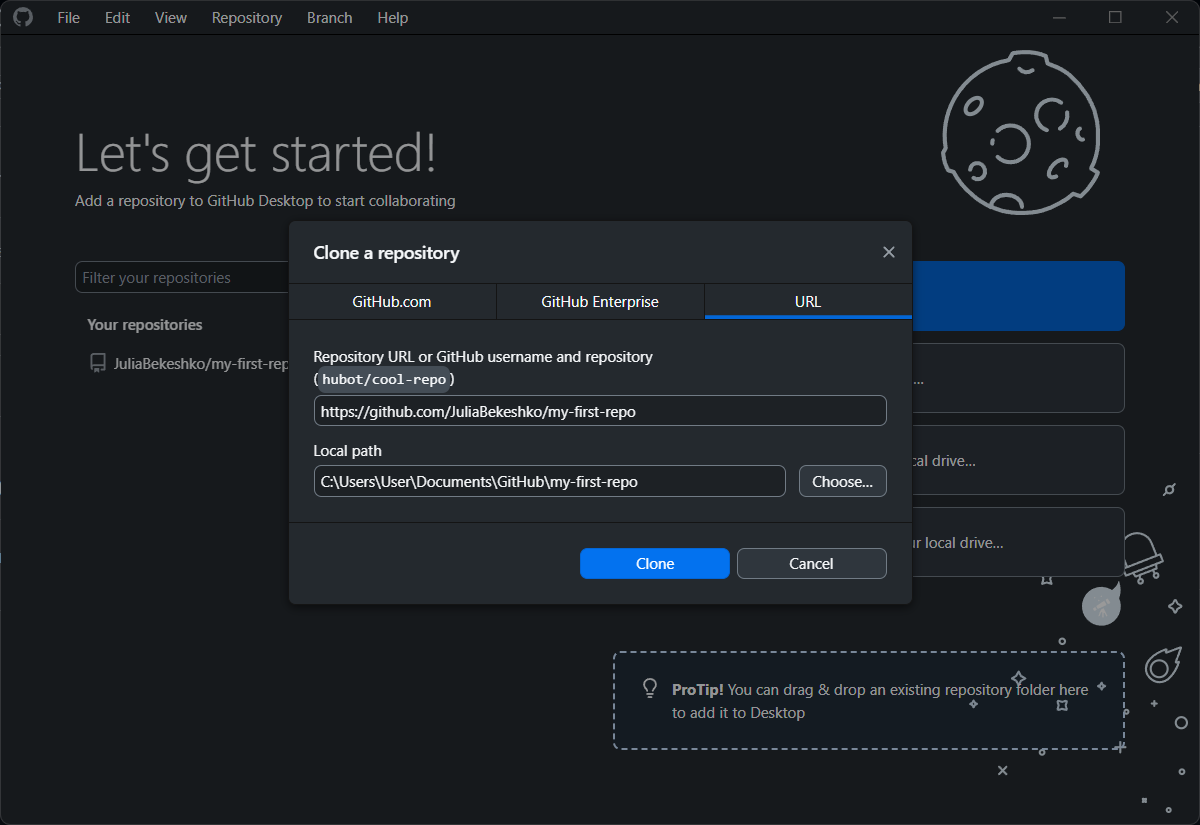
****

**Рисунок 8 – Клонування існуючого репозиторію у GitHub Desktop через сайт**

Далі натиснути кнопку «Open» за допомогою GitHub Desktop, щоб клонувати та відкрити репозиторій у GitHub Desktop. Натиснути кнопку «Choose» і, використовуючи вікно засобу пошуку, перейти локальним шляхом, куди потрібно клонувати репозиторій. Клацнути «Clone».

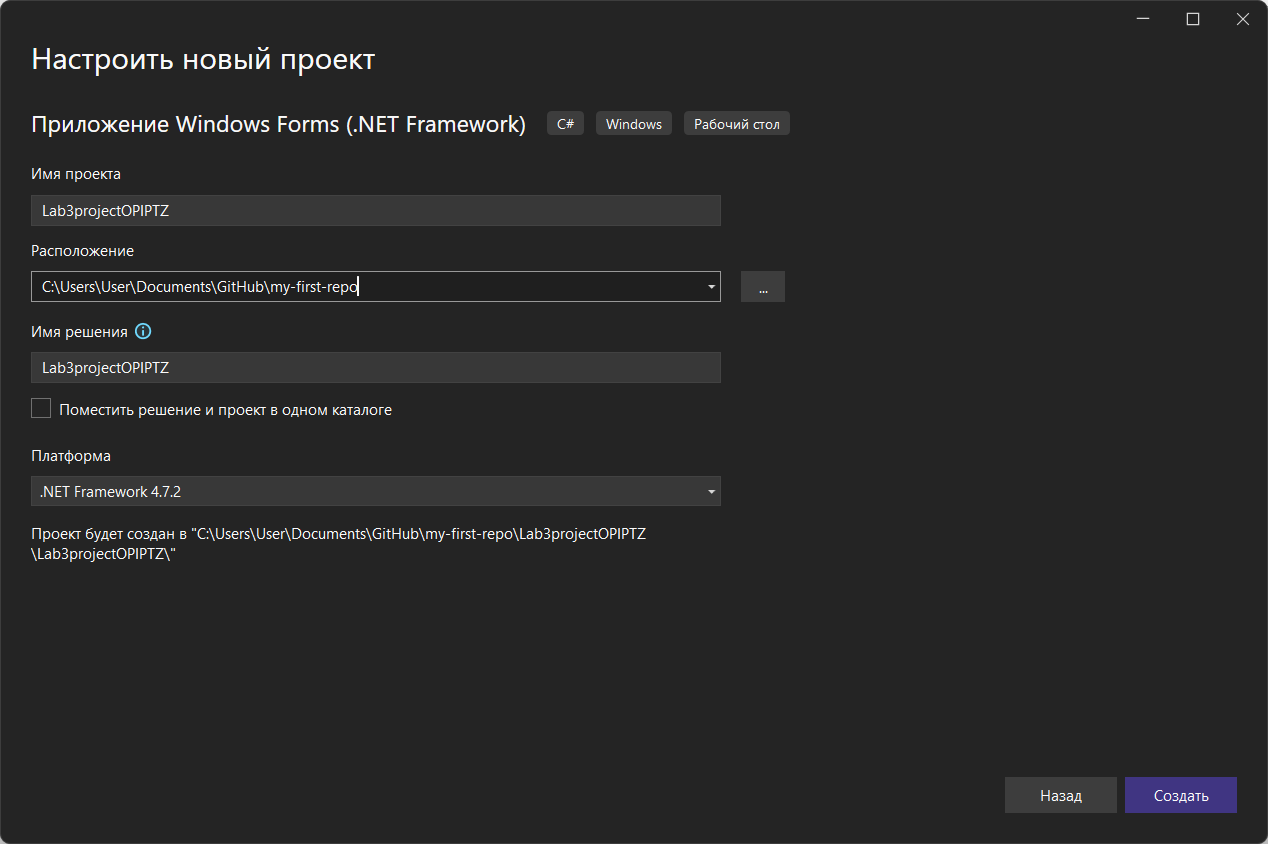


**Рис. 10 – Відкриття репозиторію у GitHub Desktop**

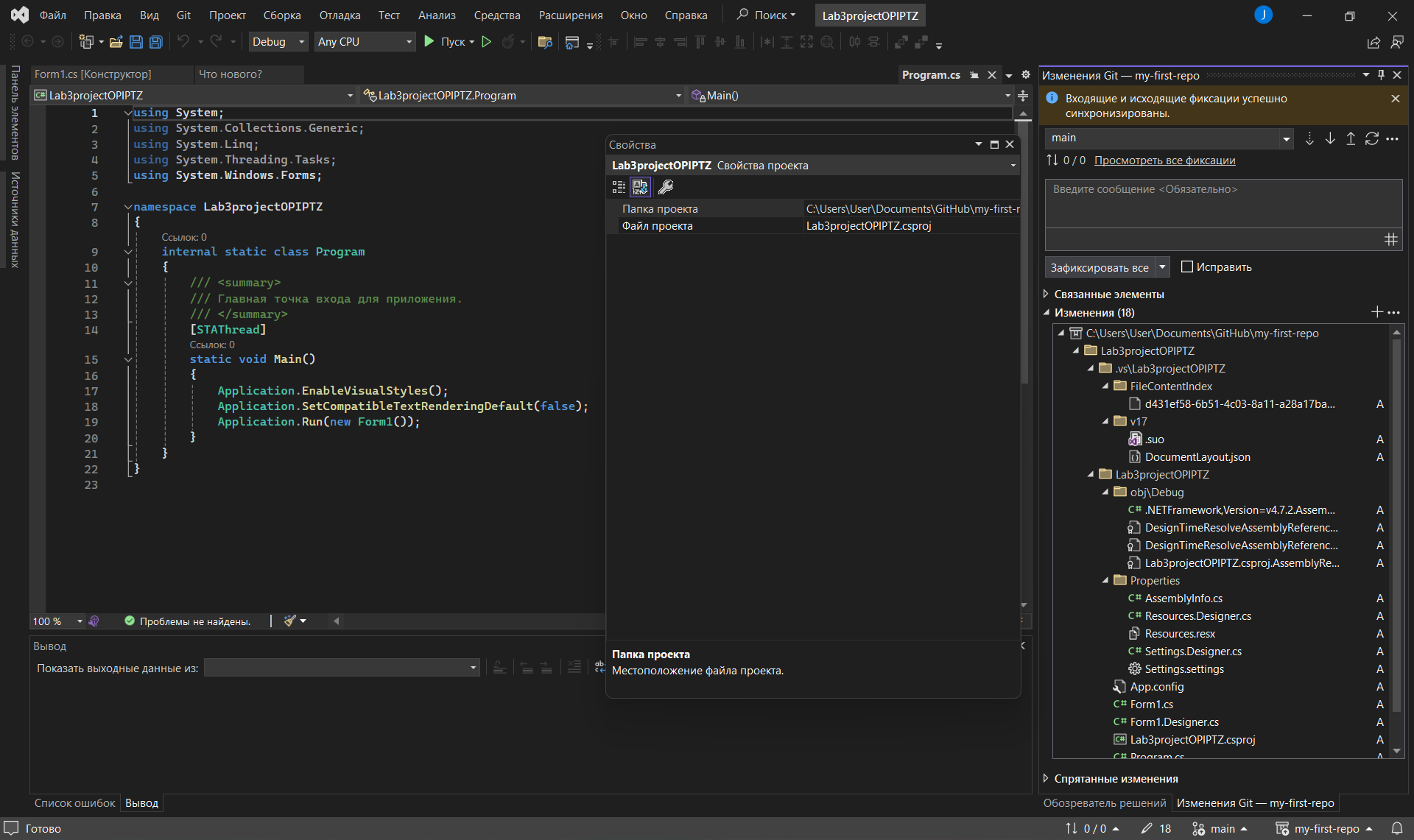
****

**Рисунок 11 – Вибір локального шляху для репозиторію у GitHub Desktop**

Створення проєкту Microsoft Visual Studio у локальному репозиторії GitHub. Для створення нового проєкту у локальному репозиторії GitHub необхідно запустити Microsoft Visual Studio та натиснути «Create new project». Далі обрати необхідний тип застосунку та у вікні, що відкриється вказати необхідні дані для створення нового проєкту. Важливо вказати папку для створення проєкту за адресою, за якою був клонований репозиторій на попередньому кроці. Настинути кнопку «Create» та синхронізувати.



**Рисунок 12 – Створення проєкту у Microsoft Visual Studio**



**Рисунок 13 – Синхронізація середовища розробки з віддаленим репозиторієм GitHub**

Було створено застосунок який знаходить довжину дуги кола, якщо відомі радіус кола та центральний кут, який вона охоплює. Радіус кола та центральний кут вводяться користувачем з клавіатури.

Лістинг програми:

namespace Lab3projectOPIPTZ

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void CalculateButton\_Click(object sender, EventArgs e)

{

try

{

// Отримання значень з TextBox

double radius = double.Parse(radiusTextBox.Text);

double angleDegrees = double.Parse(angleTextBox.Text);

// Переведення кута в радіани

double angleRadians = angleDegrees \* Math.PI / 180.0;

// Обчислення довжини дуги

double arcLength = radius \* angleRadians;

// Виведення результату

resultLabel.Text = $"Довжина дуги кола: {arcLength:F2}";

}

catch (FormatException)

{

MessageBox.Show("Будь ласка, введіть коректні значення для радіусу та кута.");

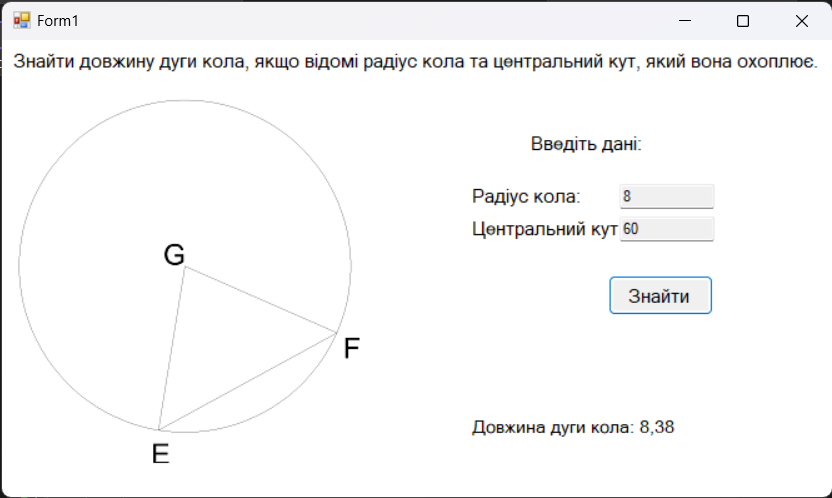
}

}

}

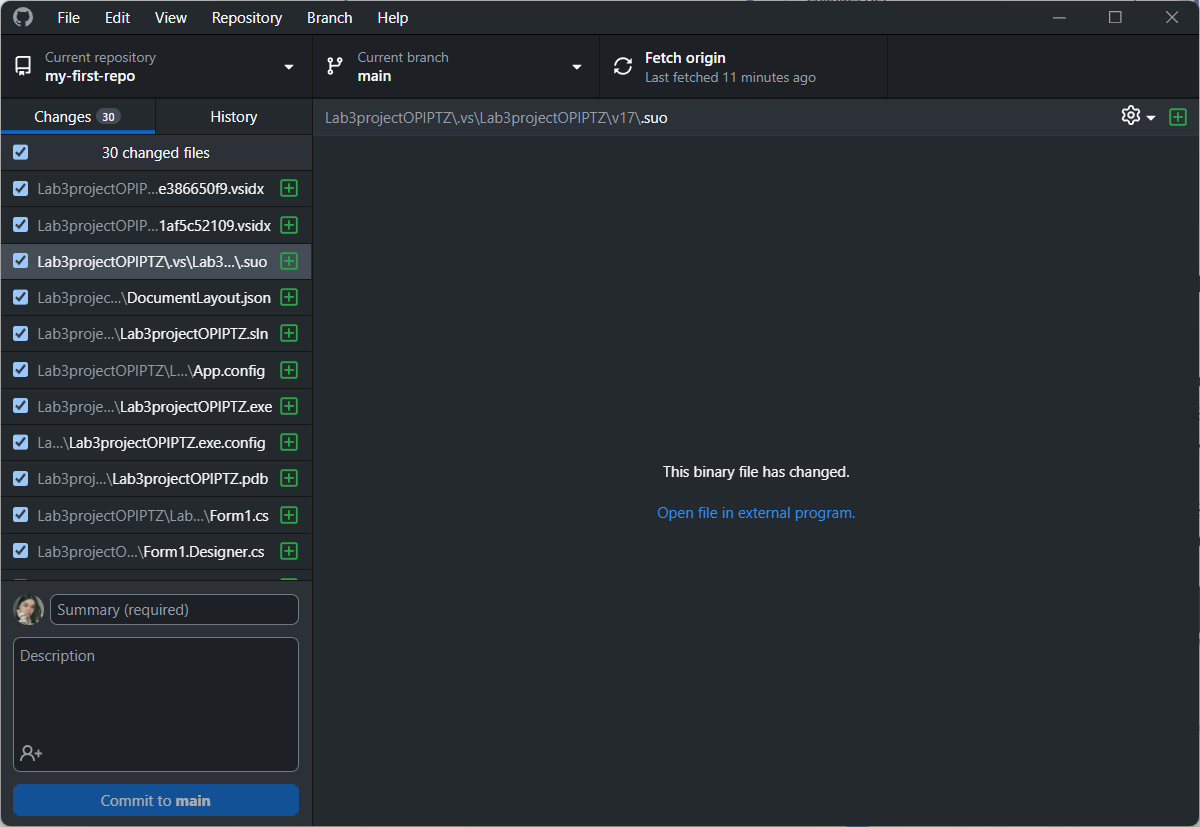
}

Світлина виконання програми:



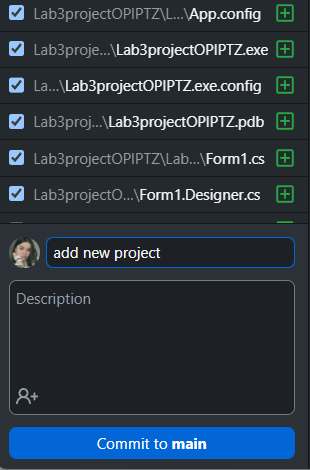
**Рисунок 14 – Виконання програми**

Завантаження створеного ПЗ на віддалений сервер GitHub. Після створення проєкту, в GitHub Desktop з’явиться вікно, в якому відображаються зміни, які стались в локальному репозиторії

****

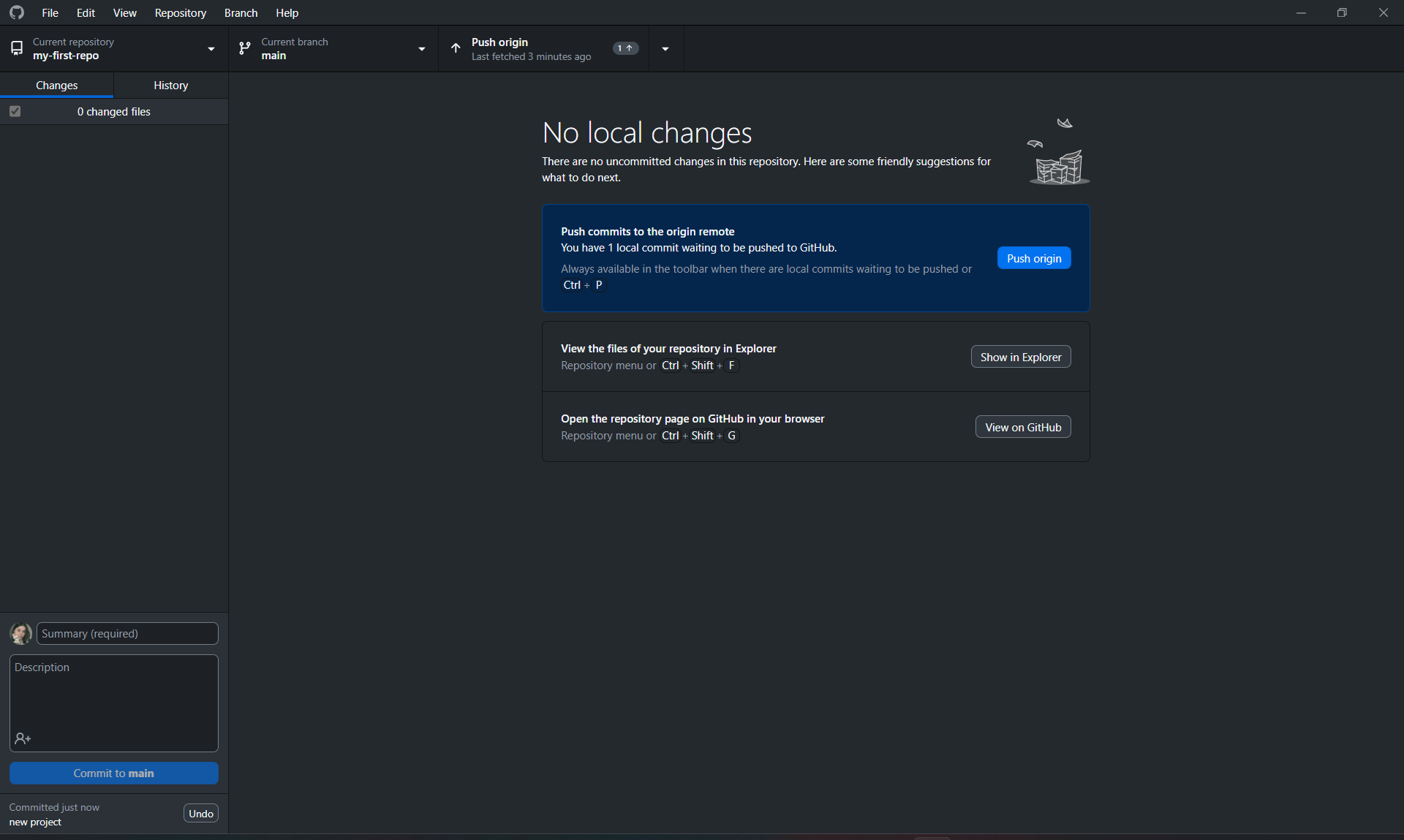
**Рисунок 15 – Відображення змін у локальному репозиторії**

Далі ці зміни було зафіксовано за допомогою команди Commit. У нижньому лівому кутку вікна необхідно ввести інформацію про те, які зміни були внесені (обов’язково) та залишити коментар (не обов’язково), після чого натиснути «Commit to main».

****

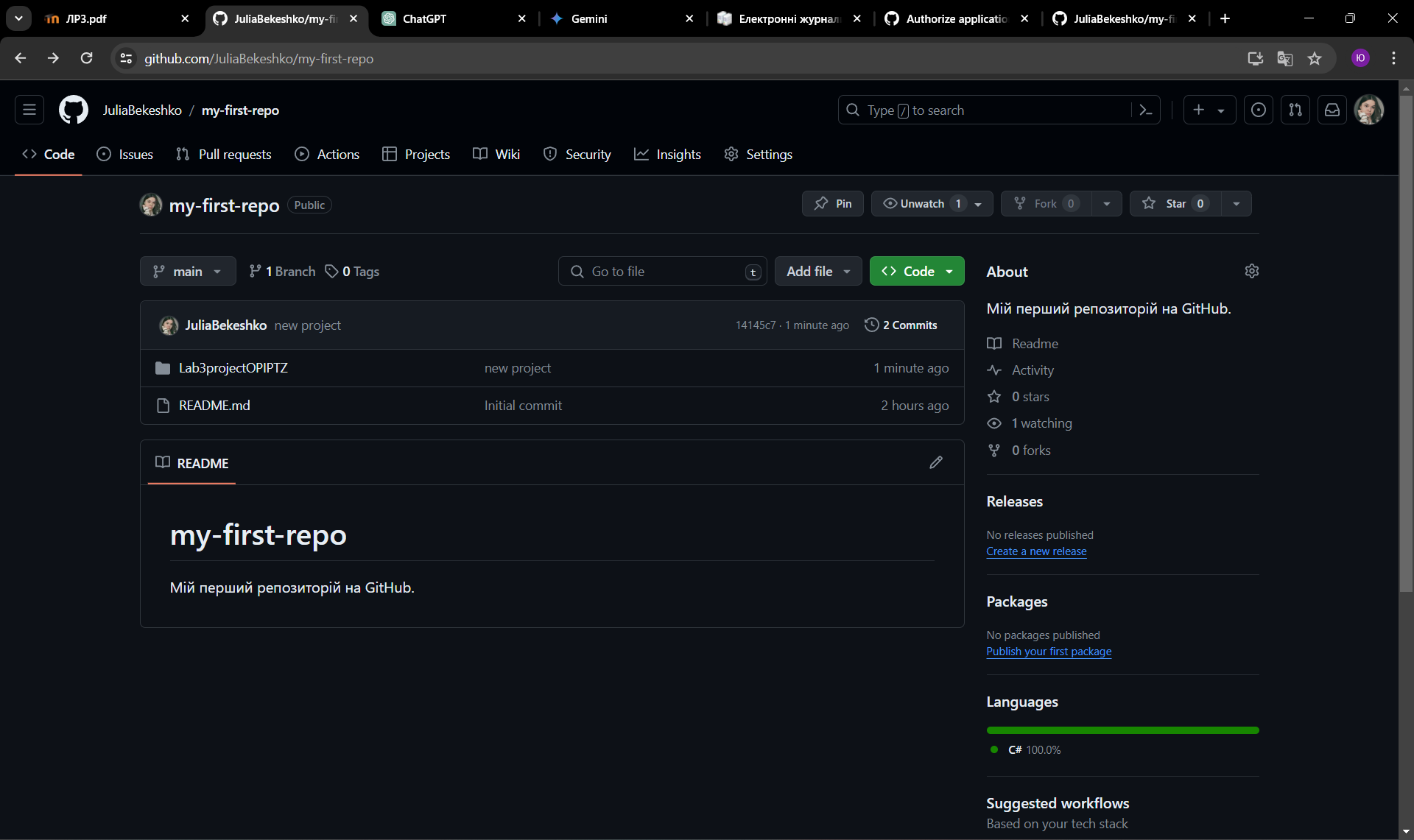
**Рисунок 16 – Фіксація змін за допомогою команди Commit**

Після цього відкриється головне вікно клієнта та з’явиться повідомлення про те, що є локальний коміт, який очікує публікації на віддаленому сервері. Необхідно ці зміни відправити на віддалений сервер, натиснувши на кнопку “Push origin”, тоді усі користувачі, які мають доступ до віддаленого репозиторію можуть побачити зміни, що були внесені розробником.

****

**Рисунок 17 – Відправка змін з локального репозиторія на віддалений**

Перейшовши на сайт GitHub та відкривши репозиторій можна побачити хто, коли і які зміни вніс

****

**Рисунок 20 – Демонстрація змін на віддаленому репозиторії**

**Висновок:**

У ході лабораторної роботи було сформовано навички користування інструментами GitHub та GitHub Desktop для керування версіями програмного коду та спільної роботи над проектами. Було освоєно створення репозиторіїв, роботу з гілками, внесення змін, злиття коду та відстеження проблем. Ці навички є важливими для ефективного управління проектами розробки програмного забезпечення та сприяють покращенню колаборації в команді розробників.

**Контрольні питання**

1. Що таке система керування версіями?

Система керування версіями – це інструмент для зберігання та управління різними версіями файлів, що може використовуватися для спільної роботи над проєктами. СКВ дозволяє зберігати всі версії файлів, зроблені під час розробки, забезпечує можливість повернення до попередніх версій та об'єднання змін, внесених різними людьми у один файл.

1. Які типи систем керування версіями є?

Локальні СКВ (Local VCS) – це прості СКВ, які використовуються на одному комп'ютері для ведення історії змін у файлах. Наприклад, RCS (Revision Control System), SCCS (Source Code Control System).

Централізовані СКВ (Centralized VCS) – ці СКВ мають один центральний репозиторій, до якого з'єднані всі розробники. Зміни завантажуються та вилучаються через центральний сервер. Наприклад, CVS (Concurrent Versions System), Subversion (SVN).

Розподілені СКВ (Distributed VCS) – репозиторій розподілений між всіма розробниками, кожен з них має свою копію. Це дозволяє працювати навіть без підключення до центрального сервера. Наприклади: Git, Mercurial, Bazaar.

1. Які СКВ входять в сімейство Git?

Git – це потужна система контролю версій, яка широко використовується в розробці програмного забезпечення. Однак Git – це не просто одна система. Це сімейство варіацій, які пропонують різні функції та можливості.

GitLab – це веб-платформа для керування проєктами розробки програмного забезпечення та система управління версіями, яка базується на GIT. Вона надає розробникам та командам інструменти для спільної роботи, управління кодом, автоматизації процесів збірки та розгортання, відстеження задач та багів, а також багато інших функцій, необхідних для розробки програмного забезпечення.

GitHub служить для розміщення репозиторіїв Git і надає розробникам засоби для поставки якіснішого коду: функції командного рядка, проблеми (ланцюжки обговорень), запити на витягування, перевірка коду та колекція безкоштовних і платних додатків у GitHub Marketplace.

1. Які компоненти та команди Git ви знаєте?

**Репозиторій:** Сховище для файлів проекту та історії їх змін.

**Гілка:** Копія коду з репозиторію, що використовується для розробки нових функцій без впливу на головну версію.

**Fork:** Копія репозиторію, створена для внесення змін без прямого впливу на вихідний код.

**Fetch:** Отримання оновлень з віддаленого репозиторію без їх злиття з локальною гілкою.

**Pull:** Отримання оновлень з віддаленого репозиторію та їх злиття з локальною гілкою.

**Commit:** Зафіксована зміна в коді, що додається до історії репозиторію.

**Push:** Відправлення змін з локального репозиторію на віддалений (наприклад, GitHub).

**Merge:** Об'єднання двох гілок коду в одну.

1. З якою ціллю створюються гілки?

Для створення нових функцій або внесення зміни до існуючого коду без впливу на головну версію проєкту.

1. За допомогою якої команди можна отримати на локальний комп’ютер останні зміни з віддаленого серверу?

git pull: Отримає та об'єднає зміни (рекомендується).

git fetch: Завантажить зміни, але не об'єднає (перегляд перед злиттям).

1. Якою командою можна внести зміни на віддалений сервер?

Для відправки локальних змін на віддалений Git-сервер використовується команда git push

1. Що таке конфлікт злиття? Як його вирішити?

Конфлікт злиття (Merge Conflict) – це ситуація, коли Git не може автоматично злити зміни з двох гілок, оскільки вони змінюють один і той же файл у різних місцях, або коли в одному файлі з'являється конфліктна секція зі змінами з обох гілок. У такому випадку потрібно відкрити конфліктний файл та вирішити конфлікт вручну, об'єднавши зміни з обох гілок таким чином, щоб вони не суперечили одна одній. Після того, як конфлікт вирішено, потрібно зробити коміт зі змінами та завершити процес злиття.

1. Які переваги використання GitHub для спільної роботи над проєктами?

GitHub – це потужний інструмент для спільної роботи над проєктами, який пропонує контроль версій, співпрацю, прозорість та ефективність.