

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

_____ УНІВЕРСИТЕТ

Кафедра _____

Дисципліна «_____»

Лабораторна робота №4

«Організація розроблення SmartNotes з використанням GitHub Projects та Git»

Виконав(ла): студент(ка) _____ курсу, групи _____

_____ (ПІБ)

Перевірив(ла): _____

«_____» _____ 2025 р.

Оцінка: _____

м. _____ — 2025

1 МЕТА РОБОТИ

Опанувати інструменти GitHub Projects, Kanban-підхід, систему контролю версій Git та засоби Code Review для організації розроблення застосунку SmartNotes.

2 ЗАВДАННЯ

1. Створити Kanban-дошку в GitHub Project та зафіксувати основні картки робіт.
2. Створити віддалений репозиторій на GitHub — [kristi-terabyte/it-project](https://github.com/kristi-terabyte/it-project).
3. Вибрати засіб розроблення коду (Visual Studio Code з розширеннями GitHub Copilot та Git Graph).
4. Розгорнути вихідний код SmartNotes, використовуючи Git, гілки та Pull Request, синхронізований із Kanban-картками.
5. Підготувати звіт з описом виконаних кроків, скріншотами дошки та історією комітів.

3 ВИКОНАННЯ ЗАВДАННЯ

3.1 Створення Kanban-дошки

У GitHub Projects створено дошку «SmartNotes Roadmap» з колонками

«Backlog», «In Progress», «Code Review», «Done». Для неї описано картки KAN-01...KAN-05 (ініціалізація репозиторію, CLI ядро, пошук, документація, рев'ю). Під час розроблення картки зміщувалися між колонками синхронно з прогресом гілок, що дозволило відслідковувати стан задач в режимі реального часу.



Рисунок 1 — Kanban-дошка GitHub Projects

3.2 Репозиторій та інструменти

На GitHub створено репозиторій `kristi-terabyte/it-project`. Основне IDE — Visual Studio Code з розширеннями GitHub Pull Requests та Git Graph, що дозволяє переглядати дифи безпосередньо в редакторі. Локальна копія ініціалізована командами:

```
echo "# it-project" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/kristi-terabyte/it-project.git
git push -u origin main
```

Основне IDE — Visual Studio Code, що забезпечило інтеграцію з Git, форматування коду та перегляд Pull Requests.

3.3 Вихідний код та Pull Requests

Функціонал SmartNotes реалізовано як CLI-додаток (`smartnotes/app.py`) з шаром зберігання (`smartnotes/storage.py`). Дані записуються у JSON-файл `data/notes.json`, який створюється автоматично, а команди `add`, `list`, `search`, `delete` надають базове керування нотатками.

Робочий процес ґрунтується на гілках і Pull Request. Після завершення фічі створюється PR, який прив'язується до канбан-картки й проходить рев'ю (рис. 3). Табл. 1 ілюструє ключові гілки:

Таблиця 1 — Приклади гілок та Pull Request

Гілка	Задача	PR/Статус
feature/cli-foundation	Каркас CLI, JSON-сховище	PR #1 «SmartNotes CLI foundation» — merged
feature/search-service	Повнотекстовий пошук і фільтри	PR #2 «Search & filters» — merged
feature/reminders	Нагадування та push-сповіщення	PR #3 «Reminder service» — in review
feature/docs-report	Оновлення README та звіту	PR #4 «Documentation set» — draft

Огляд коду виконувався в інтерфейсі GitHub (diff, коментарі, зміна статусів Kanban). Скриншот перевірки Pull Request наведено у файлі `screenshots/pr_overview.png`.

3.4 Історія комітів

Кожен крок фіксувався комітами з префіксами, що співвідносяться з картками Kanban (наприклад, `[KAN-01] init project`, `[KAN-03] search filters`). На рисунку 2 подано фрагмент історії (`git log --oneline --graph`), що демонструє злиття гілок і відстеження прогресу.



Рисунок 2 — Історія комітів у гілці `main`

4 ВИСНОВКИ

Під час лабораторної роботи №4 організовано командний процес розроблення SmartNotes: створено Kanban-дошку, репозиторій GitHub, налаштовано інструменти розробки та виконано серію Pull Request. Практика показала ефективність відслідковування задач через GitHub Projects і важливість рецензій коду перед інтеграцією у гілку `main`.