Державний вищий навчальний заклад

Ужгородський національний університет

Факультет інформаційних технологій

**ЛАБОРАТОРНА РОБОТА № 3**

**Тема:** Каркас (структура) проєкту.

Виконав:

студент Ⅲ курсу

спеціальності «Інженерія

програмного забезпечення»

Кулешов Себастьян Сергійович

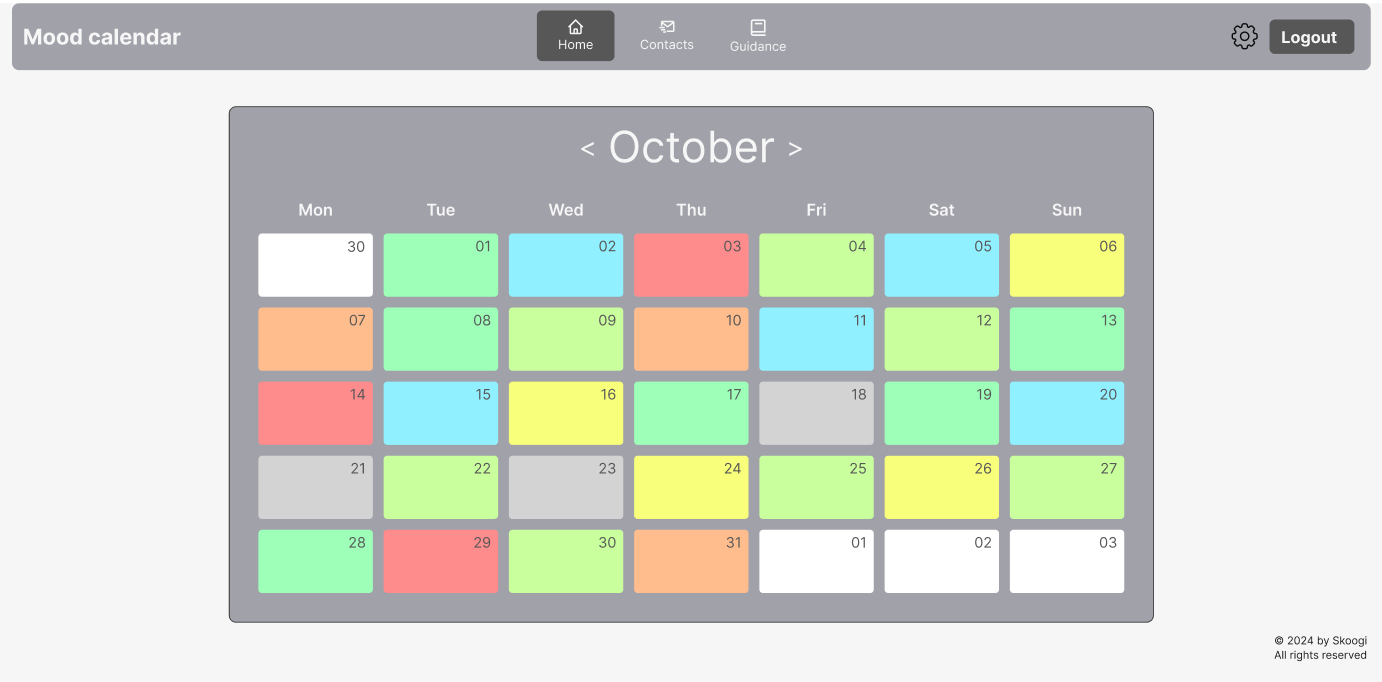
**Мета:** Створення та організація каркасу (структури) проєкту відповідно до обраної архітектури. Це включає налаштування директорій і файлів. Після цього всі зміни мають бути завантажені на GitHub з відповідним Pull Request для перегляду.

**Завдання до роботи:**

1. Детальний опис ідеї з відповідними ілюстраціями (необхідно, щоб зрозуміти структуру папок/файлів).
2. Представлення структури проєкту, директорій...
3. Обгрунтувати обране рішення (по архітектурі).
4. Представити стек проєкту.
5. Детальне обгрунтування у випадку відхилення від запропонованої на заняттях архітектури.
6. Опублікувати всі зміни на GitHub.
7. Створити Pull Request (нас додати як переглядачів (reviewers)).

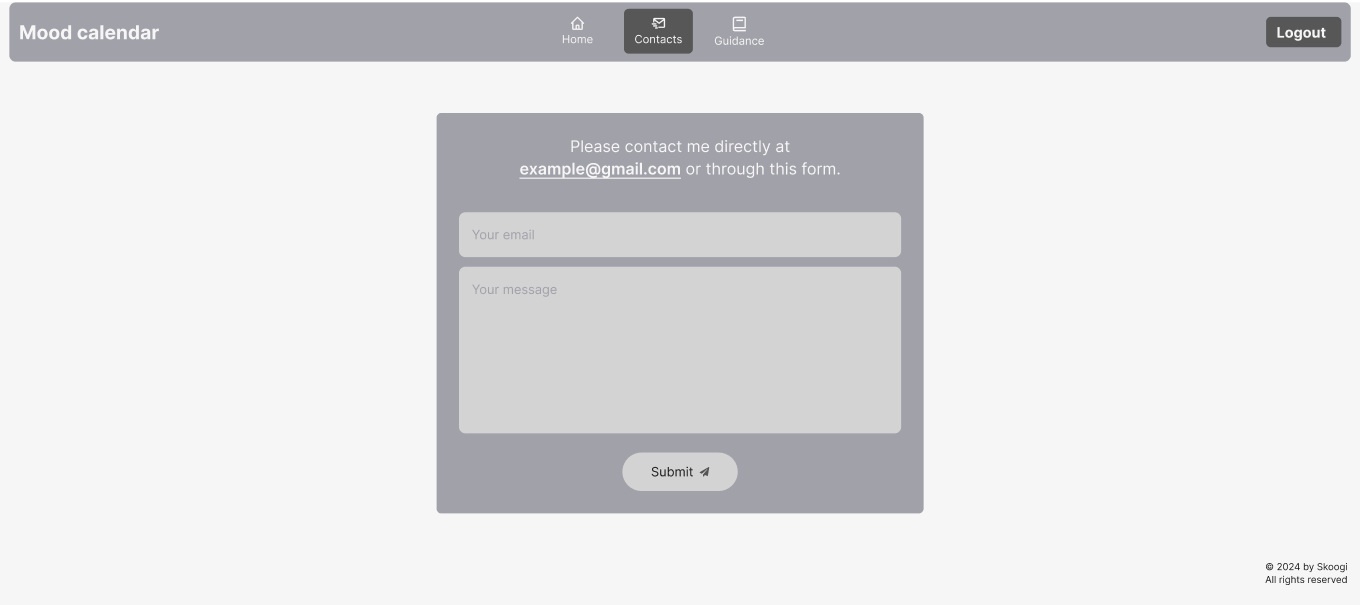
**Хід роботи**

**Завдання 1:** Детальний опис ідеї з відповідними ілюстраціями (необхідно, щоб зрозуміти структуру папок/файлів).



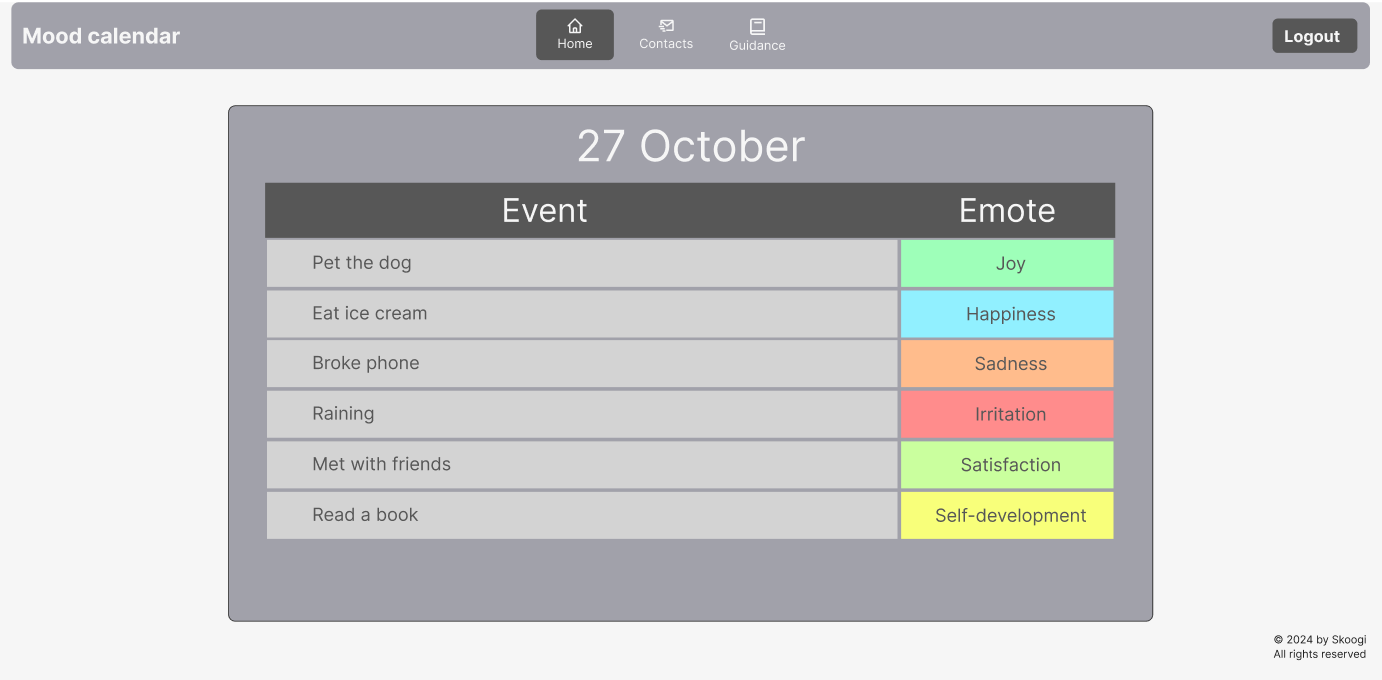
**Рис. 1.** Головна сторінка – “Home”.

Дана сторінка містить в собі календар, у якого кожен день фарбується кольором, який вираховується на основі подій, які сталися з користувачем, протягом дня.



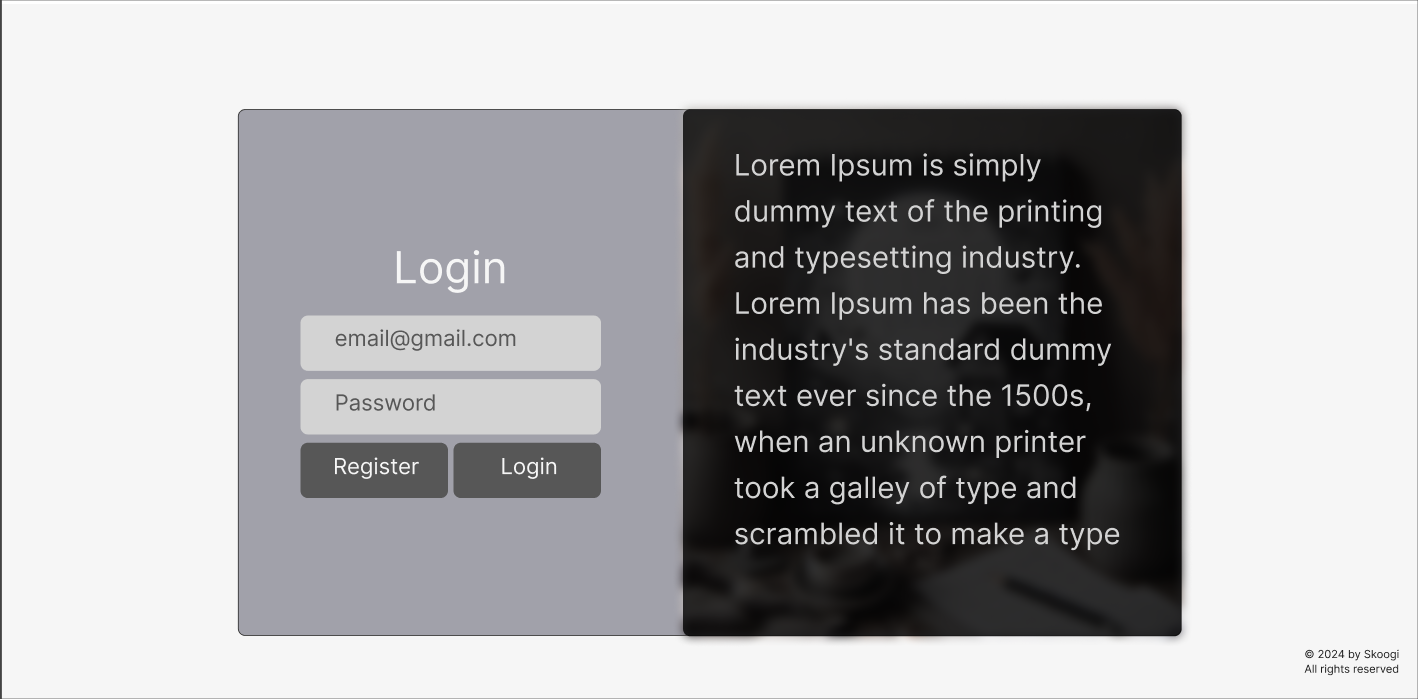
**Рис. 2.** Сторінка – “Contact us”.

Дана сторінка містить в собі форму, щоб надіслати повідомлення адміністрації.



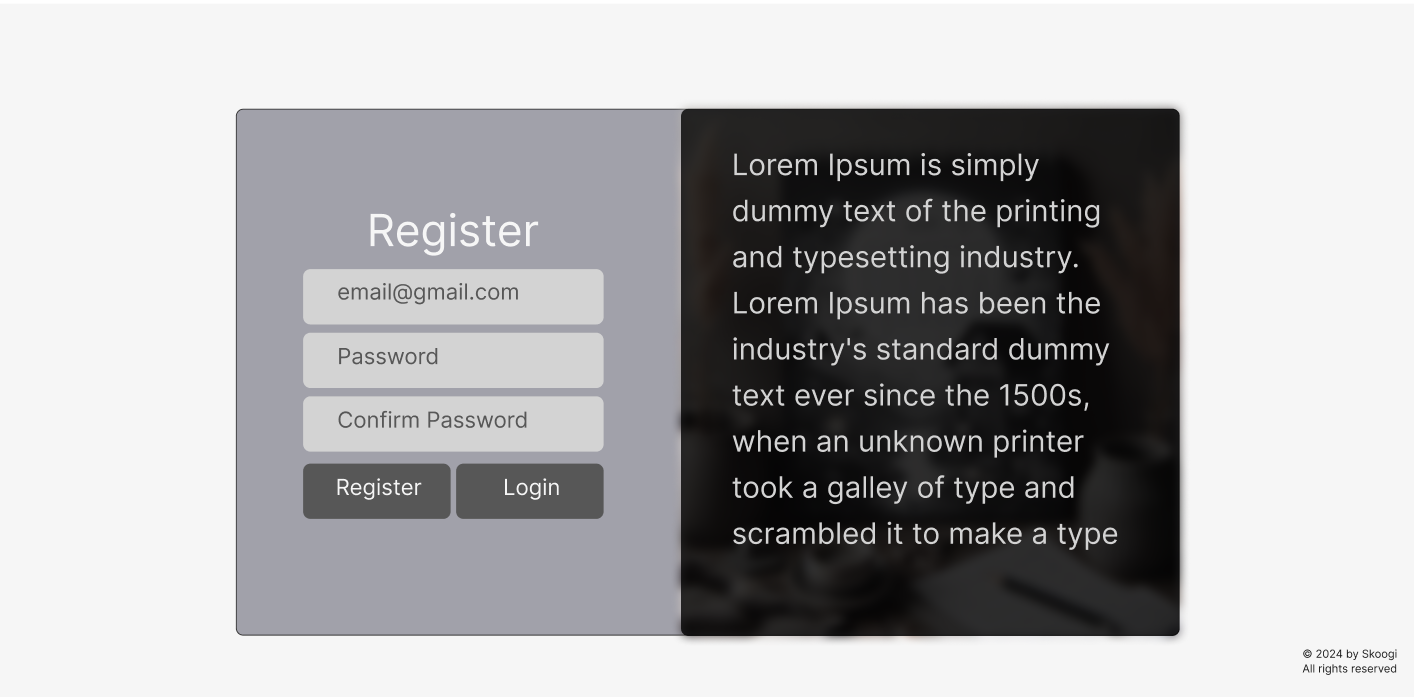
**Рис. 3.** Сторінка – “Day”.

Дана сторінка містить в собі форму для заповнення подій, які сталися з користувачем протягом дня. Він може описати, що з ним сталось та, яку емоцію викликає у нього ця подія.



**Рис. 4.** Сторінка – “Login”.

Дана сторінка включає в себе логін користувача.



**Рис. 5.** Сторінка – “Register”.

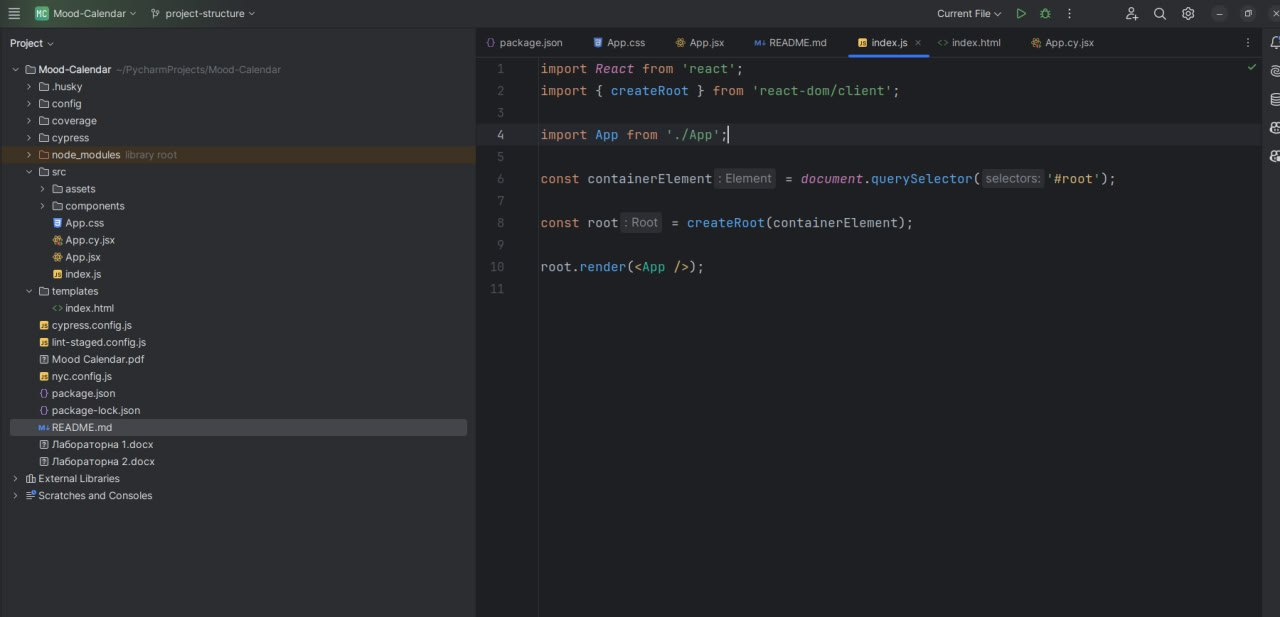
Дана сторінка включає в себе реєстрацію користувача.



**Рис. 6.** Модальне вікно – “Contact us”.

Дане модальне вікно відповідає за те, як кожна емоція асоціюється з кожним кольором для окремого користувача.

**Завдання 2:** Представлення структури проєкту, директорій.



**Рис. 7.** Структура проєкту.

**Завдання 3:** Обгрунтувати обране рішення (по архітектурі).

Компонентна структура: У директорії src файли організовані так, що кожен компонент відповідає за власну логіку та відображення, а стилі винесені в окремі файли, як-от App.css. Такий підхід покращує структуру коду, знижує залежність між компонентами, робить його більш зрозумілим і легшим для тестування.

Конфігураційні файли: Директорії .husky, config і cypress відповідають за налаштування автоматизації процесів та тестування, що підвищує якість і надійність коду. Використання Cypress для тестування end-to-end гарантує, що функціонал веб-додатку працює коректно з точки зору користувача.

Git-ігнорування та лінтинг: Файл .gitignore виключає тимчасові або зайві файли з репозиторію, забезпечуючи чистоту версійного контролю. ESLint і Prettier (.eslintrc.js та .prettierrc) допомагають підтримувати високі стандарти кодування та єдиний стиль у всьому проєкті.

Тестування та покриття коду: Інструменти лінтингу (lint-staged.config.js) і тестування коду (директорія coverage, файл nyc.config.js) забезпечують стабільність і надійність додатка на кожному етапі розробки, допомагаючи швидко виявляти та виправляти помилки.

**Завдання 4:** Представити стек проєкту.

React: Використовується для створення інтерфейсу користувача, дозволяючи будувати динамічні та інтерактивні веб-сторінки через компонентний підхід. Файл App.jsx є прикладом компонента React, що відповідає за рендеринг основного контенту додатка.

CSS (App.css): Стилізація компонентів здійснюється за допомогою CSS, що надає повний контроль над зовнішнім виглядом додатка.

Node.js та npm: Менеджер пакетів npm (файли package.json та package-lock.json) використовується для керування залежностями і запуску скриптів, необхідних для налаштування середовища розробки.

Cypress: Інструмент для автоматизованого тестування (директорія cypress), що дозволяє перевіряти правильність роботи додатка через браузер.

ESLint і Prettier: Інструменти для аналізу та форматування коду (.eslintrc.js, .prettierrc) забезпечують дотримання єдиних правил і стандартів під час розробки, що підвищує якість коду.

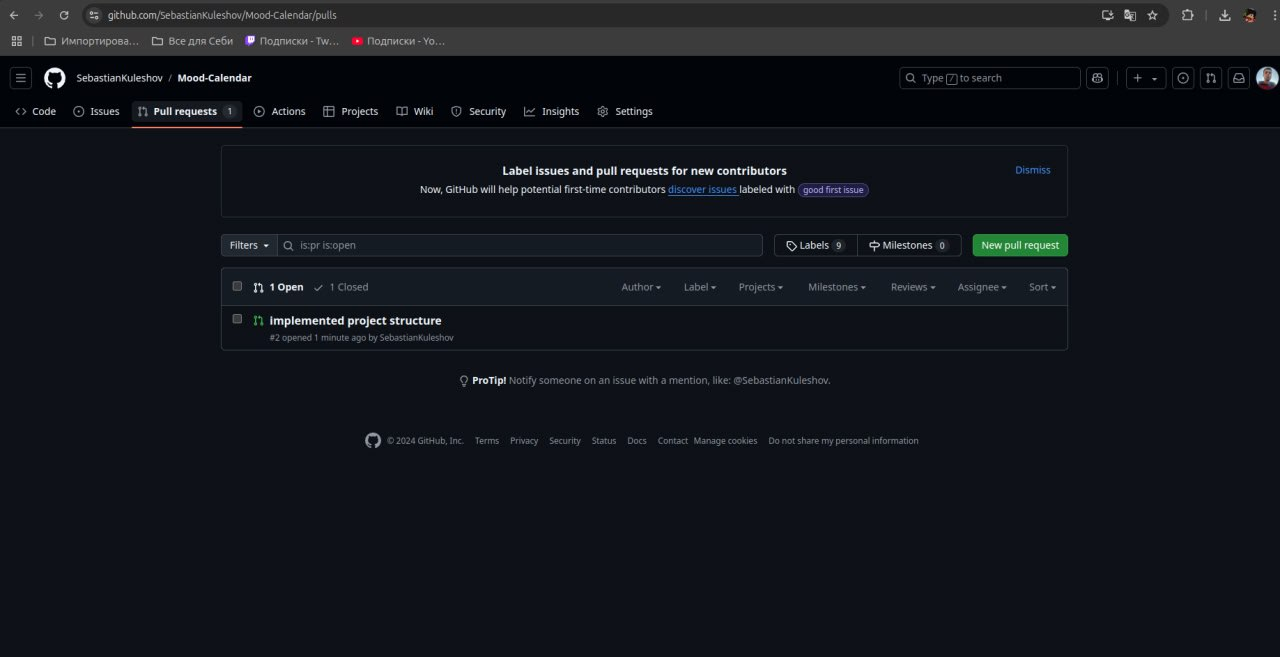
Husky: Використовується для налаштування Git hooks, що автоматизує процеси перевірки коду (наприклад, запуск тестів або лінтингу перед комітом), підвищуючи продуктивність команди та покращуючи якість коду.

Тестування покриття (NYC): Інструмент, який збирає статистику про те, наскільки код проєкту покритий тестами (файл nyc.config.js), що допомагає забезпечити високу надійність додатка та зменшити кількість помилок.

**Завдання 5:** Детальне обгрунтування у випадку відхилення від запропонованої на заняттях архітектури.

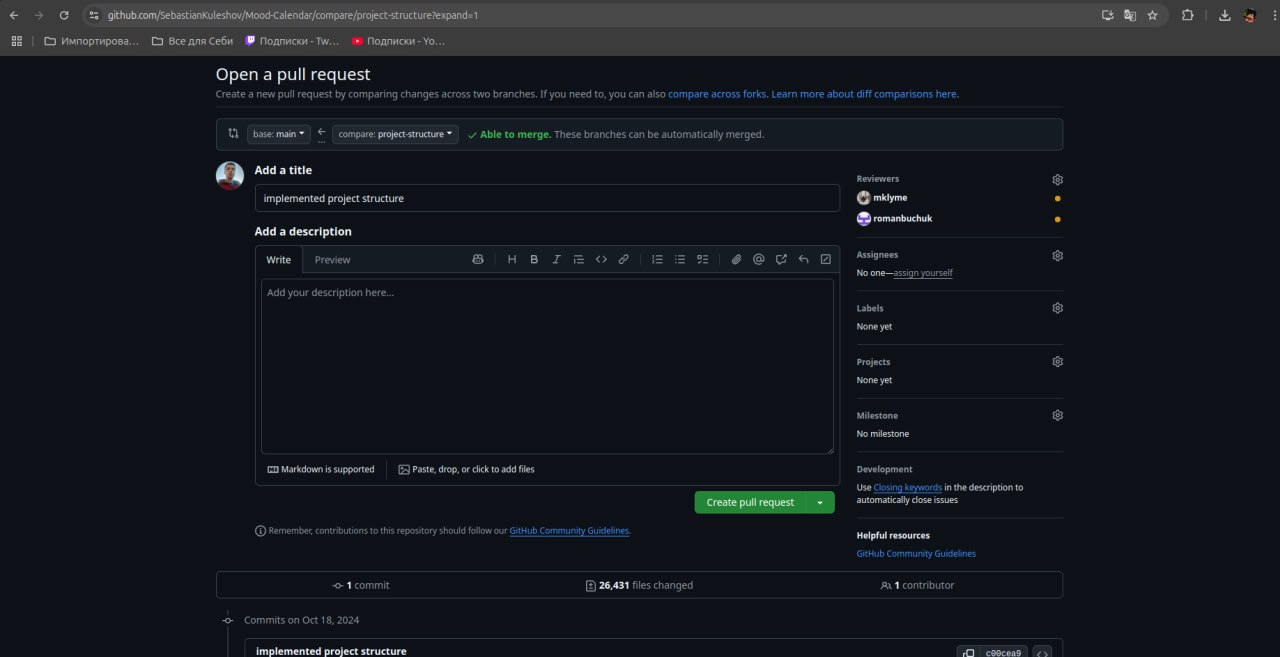
У даній лабораторній роботі відхилень від запропонованої на заняттях архітектури не було.

**Завдання 6:** Опублікувати всі зміни на GitHub.



**Рис. 8.** Публікація на GitHub.

**Завдання 7:** Створити Pull Request (нас додати як переглядачів (reviewers)).



**Рис. 9.** Публікація на GitHub.

**Висновки:** Під час виконання лабораторної роботи було створено каркас проєкту з продуманою та структурованою організацією директорій і файлів, яка відповідає вимогам проєкту. Обрано оптимальний стек технологій для ефективної реалізації завдань. Усі внесені зміни були опубліковані на GitHub, а для перегляду й перевірки створено Pull Request із призначенням reviewers для отримання зворотного зв'язку.