МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут **КНІТ**
Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи № 6
З дисципліни: *"Моделювання та аналіз програмного забезпечення"*
На тему: *"Дослідження предметної області та проектування системи за допомогою UML діаграм"*

Лектор:
доц. каф. ПЗ
Сердюк П.В.

Виконав:
ст. гр. ПЗ-22
Чаус Олег

Прийняв:
викл. каф. ПЗ
Микуляк А. В.

« ____ » _____ 2023 р.

∑= ____ _____.

Львів – 2023

Тема роботи: Дослідження предметної області та проектування системи за допомогою UML діаграм.

Мета роботи: Здобути навички дослідження предметної області та проектування системи за допомогою UML діаграм.

ТЕОРЕТИЧНІ ВІДОМОСТІ

UML (Unified Modeling Language) — уніфікована мова моделювання, що використовується розробниками програмного забезпечення для візуалізації процесів та роботи систем.

Це не мова програмування, скоріше набір правил та стандартів для створення діаграм. Вони дозволяють розробникам програмного забезпечення та інженерам «говорити однією мовою», не заглиблюючись у фактичний код свого продукту. Складання діаграм за допомогою UML — це чудовий спосіб допомогти іншим швидко зрозуміти складну ідею чи структуру.

Одне із завдань UML — служити засобом комунікації всередині команди та при спілкуванні з замовником. Давайте розглянемо можливі варіанти використання діаграм:

- Проектування. UML-діаграми стануть у пригоді при моделюванні архітектури великих проектів, в якій можна зібрати як великі, так і дрібніші деталі і намалювати каркас (схему) програми. По ньому пізніше буде будуватись код.
- Реверс-інжиніринг — створення UML-моделі з існуючого коду додатку, зворотна побудова. Може застосовуватися, наприклад, на проектах підтримки, де є написаний код, але документація неповна або відсутня.
- З моделей можна витягувати текстову інформацію і генерувати відносно читабельні тексти — документувати. Текст і графіка будуть доповнювати один одного.

UML-діаграми поділяються на 14 типів.

Рис. 1. Класифікація UML діаграм

ЗАВДАННЯ

1. Відповідно до завдання, яке ви виконуєте у рамках команди (3-4 студенти в команді) розробити: діаграму діяльності; діаграму класів (+- 10 класів для кожного студента). Врахувати у проектах інтерактивну взаємодію користувачів. Всі діаграми повинні бути складними, оскільки кінцевої реалізації проекту не потрібно, як і всіх задекларованих можливостей у діаграмах. Побудувати діаграму класів із використанням основних парадигм ООП. Для інкапсуляції логіки, створити щонайменше 2 модулі: модуль з класами, що відповідальні за внутрішню логіку програми (бізнес-логіку) та модуль інтерфейсу користувача. Модифікатори доступу класів і полів повинні надавати доступ лише до необхідних елементів, усі решта повинні бути невидимі розробнику інтерфейсу користувача.

2. Моя роль – QA

1) Розробити базовий test plan для компонент. Тест план повинен передбачати різні види тестування, які застосовні до компонент - performance, security, unit test, etc.

ХІД РОБОТИ

1. Отже моя команда – команда 5. До неї входять: я, Руденко А., Коваленко Д., Ковальов Р. Мій варіант – варіант 18 (Рейтинг матеріалів). Моє завдання – налаштування рейтингу матеріалів.
2. Розробив діаграму діяльності для своєї частини.

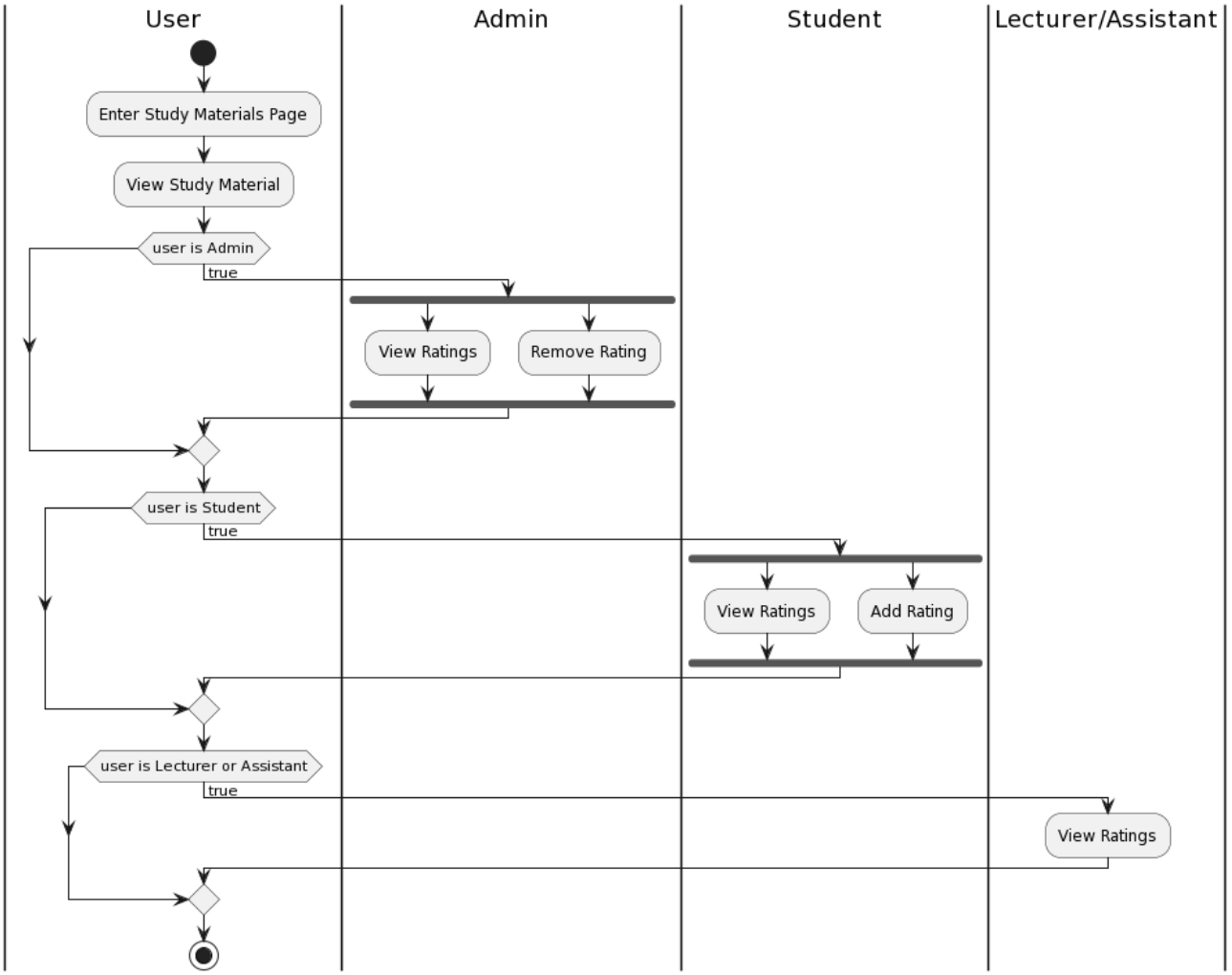

Рис. 1. Діаграма діяльності.

3. Розробив діаграму класів.

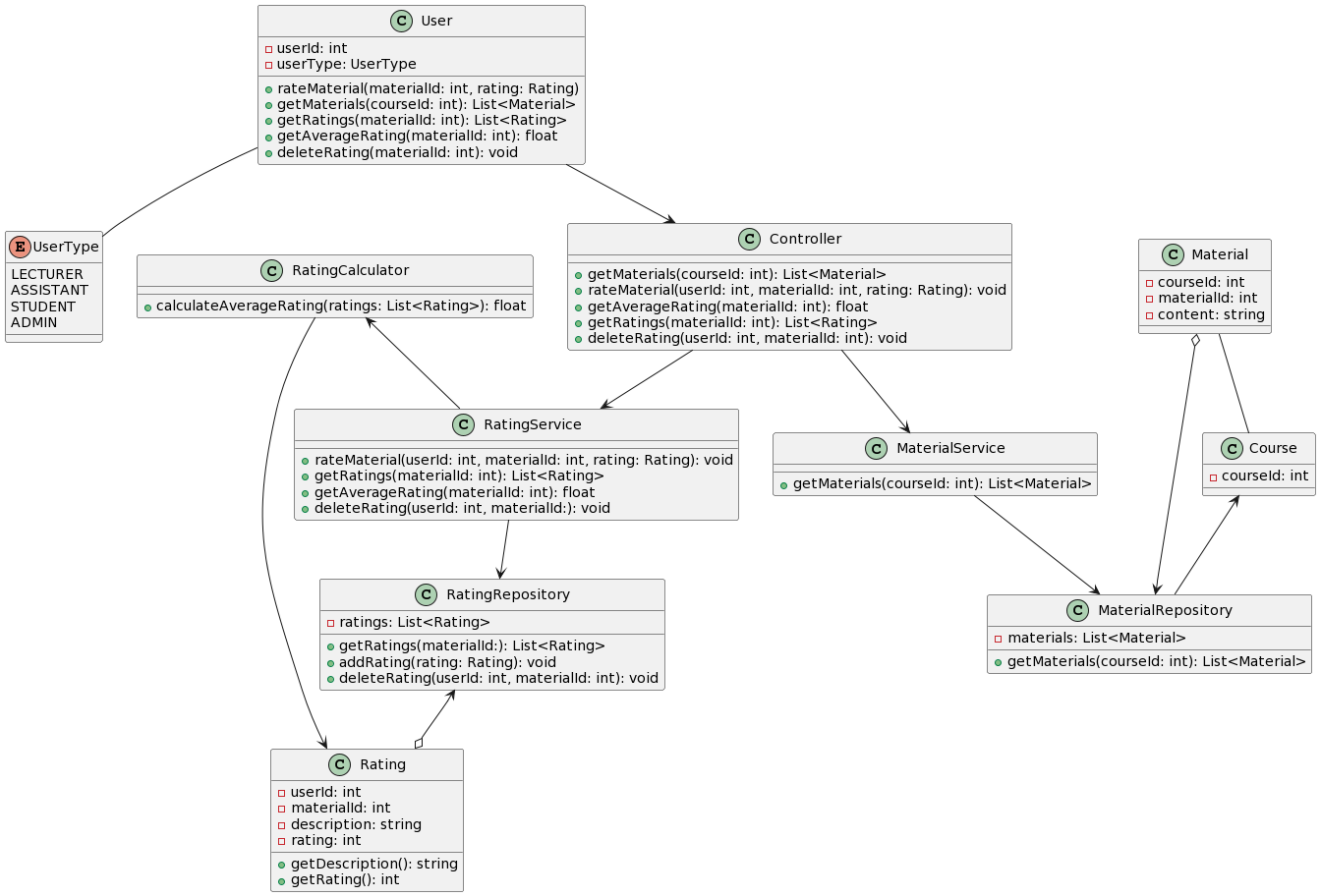

Рис. 2. Діаграма класів.

4. Розробив тестовий план.

1. ***Вступ (короткий зміст продукту, який тестується)***

Система “Комунікація” була розроблена для зручної роботи студентів та викладачів із спільною системою та взаємодії між ними у вигляді сповіщень, оцінок, розкладу та організації подій.

2. ***Завдання (все, що потрібно виконати згідно цього плану)***

Остаточним результатом тесту є два аспекти:

- Готове до впровадження програмне забезпечення;
- Набір стабільних тестових скриптів, які можуть бути використані повторно для виконання функціональних тестів.

Метою плану тестування є забезпечення якісного функціонування та стабільності системи шляхом виявлення потенційних дефектів та багів у програмному забезпеченні. Тестування буде спрямоване на перевірку наступних компонентів системи.

1. Функціональна частина
    - З боку студента:
        - Реєстрація

        - Авторизація
        - Перегляд доступних матеріалів
        - Отримання сповіщень
        - Перегляд вмісту подій
        - Ввімкнення/вимкення сповіщення про події
        - Сповіщення інших студентів
        - Перегляд графіку
        - Оцінювання матеріалу
        - Додавання коментарів до матеріалу
        - Створення запитань на сторінці FAQ.
        - Надсилання/отримання повідомлень.
    - З боку викладача та асистента:
        - Авторизація
        - Створення подій
        - Редагування подій
        - Створення навчальних матеріалів
        - Редагування навчальних матеріалів
        - Створення запису на сторінці FAQ
        - Відповідь на запитання на сторінці FAQ
    - З боку адміністратора:
        - Авторизація
        - Редагування графіку
        - Створення сповіщень
        - Редагування сповіщень
        - Створення облікових записів
        - Видалення облікових записів
        - Створення запису на сторінці FAQ
        - Відповідь на запитання на сторінці FAQ
2. Графічна складова
        - Сторінка “Навчальні матеріали”
        - Сторінка “Події”
        - Сторінка “Створення подій”
        - Сторінка “Мій розклад”
        - Сторінка “Чат”
        - Сторінка “Коментарі до навчальних матеріалів”
        - Сторінка “FAQ”

Необхідно визначити цілі, тестові припущення, принципи тестування, тестові дані, обсяг і рівні тестування, після чого описати типи тестування.

Наступним кроком необхідно визначити критерії входу/виходу, цикли тестування, принципи за якимим відбуватимуться валідація та управління помилками, а також метрики тестів. Далі потрібно зазначити, яким чином буде відбуватись відстеження помилок та звітування, вказати засоби тестування (для кожного типу тестування), вимоги до оточення, і на прикінець зазначити ризики.

3. Стратегія тестування (опис загального підходу до тестування).

3.1. Цілі

Основні цілі тестування:

- Виявлення помилок та недоліків у роботі системи, зокрема функцій, які були згадані раніше, на початкових етапах розробки. Це досягається шляхом перевірки правильності виконання запланованих дій та отримання очікуваних результатів.
- Переконання в тому, що система відповідає вимогам, викладеним у специфікації та загальних вимогах до системи.
- Перевірка справності роботи системи в ситуаціях, які не передбачалися під час розробки.
- Перевірка безпеки системи щодо можливих атак зловмисників і здатність системи працювати під надмірним навантаженням.
- Надання замовнику інформації про якість системи та готовність системи до введення в експлуатацію.

3.2. Тестові припущення

- Тестувальне середовище буде працювати на найновіших підтримуваних версіях тестувальних застосунків.
- Система коректно працюватиме під середнім навантаженням (500 людей) з очікуваним часом відклику (2 сек.).
- Система відображатиме коректний вміст (текст, фото) для відповідної групи користувачів.
- Тестувальне середовище матиме безперебійний доступ до системи.

3.3. Принципи тестування

- Тестування буде спрямоване на досягнення бізнес-цілей, забезпечення ефективності витрат і високої якості.
- Процеси тестування будуть чітко визначені, але гнучкі, з можливістю змін за необхідністю.
- Етапи тестування будуть ґрунтуватися на попередніх, щоб уникнути зайвого дублювання чи повторення зусиль.

- Тестувальне середовище та дані будуть якомога більше емулювати production environment.

3.4. Тестові дані

Під час функціонального тестування буде використовуватись попередньо завантажена тестова інформація, яка буде використовуватись для тестувальних цілей.

- Дані студентів (логін та пароль)
- Дані викладачів.
- Дані адміністраторів.
- Дані асистентів.
- Список предметів (курсів).
- Список навчальних матеріалів.
- Список подій.
- Список відгуків.
- Список коментарів.
- Список запитань та відповідей.
- Список записів FAQ.

3.5. Обсяг і рівні тестування

Обсяг тестування: весь функціонал та графічна складова.

У межах обсягу тестування планується перевірка всіх функціональних можливостей, які передбачені вимогами до програмного забезпечення. Це охоплює тестування всіх основних функцій, підсистем, процесів та функціональних модулів, що становлять складову частину програми. Під час тестування функціоналу, здійснюється перевірка правильності реалізації кожної функції, відповідності вимогам, а також виявлення можливих дефектів, непередбачених сценаріїв та несправностей.

Також в рамках обсягу тестування здійснюється перевірка графічної складової програмного продукту. Це охоплює тестування коректності відображення графічних елементів, забезпечення їхньої взаємодії з користувачем та дотримання графічних стандартів та дизайну. Тестування графічної складової також включає адаптивність програми до різних розмірів екранів та пристроїв, а також перевірку коректності відображення під різними операційними системами та веб-браузерами.

Рівні тестування: модульне, інтеграційне, системне, приймальне тестування.

3.6. Типи тестування

1. Модульне тестування

Метод тестування програмного забезпечення, який полягає в окремому тестуванні кожного модуля коду програми. Тестуватимуться окремі модулі програми:

- Авторизація
- Створення подій
- Створення сповіщень
- Перегляд вмісту подій
- Отримання сповіщень
- Редагування вмісту подій
- Створення навчальних матеріалів
- Редагування навчальних матеріалів
- Отримання списку доступних навчальних матеріалів
- Сортування списку навчальних матеріалів
- Рейтингування навчальних матеріалів
- Коментування навчальних матеріалів
- Перегляд коментарів навчальних матеріалів
- Створення питання
- Відповідання на питання
- Створення FAQ-запису
- Перегляд розкладу навчання
- Зміна розкладу в залежності від тижня/групи/семестру
- Надсилання повідомлення
- Отримання повідомлення

Очікувані результати: коректна робота всіх модулів системи, відповідність результатів до вимог продукту.

2. Інтеграційне тестування.

Фаза тестування програмного забезпечення, під час якої окремі модулі програми комбінуються та тестуються разом, у взаємодії. Виділені модулі програми:

- Модуль автентифікації
- Модуль сповіщень
- Модуль створення подій
- Модуль спілкування
- Модуль оцінювання

Інтеграційне тестування передбачає комбінування модулів.

- Модуль автентифікації:
    - Автентифікація як студент
    - Автентифікація як викладач
    - Автентифікація як адміністратор
    - Автентифікація як асистент
- Модуль сповіщень
    - Отримання сповіщень для відповідної групи
    - Отримання сповіщень на вибрану платформу
    - Створення/редагування сповіщень як викладач, адміністратор, асистент.
- Модуль подій:
    - Створення подій
    - Редагування подій
    - Перегляд подій
- Модуль спілкування (форум):
    - Надсилання/отримання повідомлень.
    - Створення кімнат
    - Додавання користувачів до кімнат
    - Вилучення користувачів з кімнат
- Модуль оцінювання
    - Оцінювання матеріалу як студент
    - Коментування матеріалу як студент
    - Створення запитань до матеріалу як студент
    - Відповідь на запитання як викладач/адміністратор/асистент

Очікувані результати: коректна взаємодія модулів програми.

3. Системне тестування

Метою системного тестування є виявлення протиріч між розробленою системою та первісними цілями її створення. Компонентами системного тестування є розроблена система ПЗ, кінцеві цілі і вся документація, яка додається до системи.

Буде застосоване до усіх модулів системи:

- Модуль автентифікації
- Модуль сповіщень

- Модуль створення подій
- Модуль спілкування
- Модуль оцінювання

Очікуваним результатом тестування системи є перевірка того, що інтегрована програмна система, включаючи всі модулі (автентифікація, сповіщення, генерація подій, комунікація та оцінка), відповідає визначеним цілям та вимогам.

4. Приймальне тестування

   Приймальне тестування - це завершальна фаза процесу тестування програмного забезпечення, яка проводиться з метою визначення відповідності розробленої системи вимогам та очікуванням кінцевих користувачів, зацікавлених сторін або клієнта. Воно зосереджене на перевірці готовності системи до розгортання та її здатності задовольняти визначені критерії прийнятності. Приймальне тестування зазвичай передбачає участь зацікавлених сторін і кінцевих користувачів, які оцінюють функціональність, зручність використання і загальну придатність системи для досягнення поставленої мети.

   Очікувані результати:

   1. Відповідність розробленого програмного забезпечення специфікованим бізнес-вимогам та критеріям приймальності, які були визначені спільно зі зацікавленими сторонами або кінцевими користувачами.
   2. Валідність та правильність функціональності програмного забезпечення, забезпечуючи його здатність виконувати очікувані завдання та операції.
   3. Надійність та стабільність програмного забезпечення, забезпечуючи його працездатність без критичних помилок, перебоїв або непередбачених збоїв.
   4. Відповідність вимогам безпеки, забезпечуючи захист даних, доступ лише для авторизованих користувачів та запобігання потенційним загрозам безпеки.
   5. Зручність та ефективність використання програмного забезпечення для кінцевих користувачів, забезпечуючи інтуїтивно зрозумілий і дружній інтерфейс, простоту взаємодії та задоволення потреб користувачів.
   6. Забезпечення заздалегідь визначених рівнів продуктивності та швидкості, здатність обробляти

завдання в межах прийнятних часових обмежень та здатність масштабуватися залежно від потреб користувачів.

4. Стратегія виконання

4.1. Критерії входу/виходу

Критерії входу:

- Вимоги до розробленого програмного забезпечення
- Завершене програмування модуля ПЗ
- Коректність вхідних даних, що використовуються для тестування модуля ПЗ

Критерії виходу:

- Повне охоплення всіх модулів ПЗ тестуванням
- Відсутність помилок та некоректної роботи функціоналу в роботі ПЗ
- Відсутність вразливостей системи
- Стабільна робота ПЗ під час проведення тестування
- Відповідність ПЗ вказаним вимогам
- Всі заплановані тести виконані
- Дефекти зареєстровані та ліквідовані

4.2. Цикли тестування

- Розробка тестових сценаріїв
- Виконання тестування
- Аналіз результатів
- Ліквідування помилок
- Повторне тестування

4.3. Валідація та управління помилками

В рамках цього тестування можна виконати такі дії для перевірки:

- Перевірка функціональності: переконатися, що програма виконує очікувані функції та завдання.
- Перевірка відповідності вимогам: перевірити, чи відповідає програма встановленим вимогам та специфікаціям.
- Тестування взаємодії: перевірити, як програма взаємодіє з користувачем та іншими системами.
- Тестування безпеки: перевірити, наскільки програма захищена від потенційних загроз безпеки.
- Тестування продуктивності: оцінити продуктивність програми під навантаженням та у різних умовах.

Управління помилками у цьому тестуванні включатиме такі етапи:

- Реєстрація помилок: документування виявлених проблем та помилок.
- Пріоритезація помилок: визначення важливості та пріоритету для виправлення помилок.
- Вирішення проблем: виконання відповідних кроків для виправлення помилок або відхилених функцій.
- Тестування виправлень: перевірка виправлень, щоб забезпечити їх правильну роботу.
- Підтвердження виправлення: підтвердження, що помилки виправлені та програма працює належним чином.
- Звітність: підготовка звіту, який містить інформацію про виявлені помилки, їх вирішення та результати тестування.

4.4. Метрики тестів

Після виконання кожного набору тестів будуть визначені наступні метрики:

- Passed/Failed Test Cases. Буде використано для вираження відношення між тестами які виконались успішно, і тими які завершились помилкою. Ця метрика допоможе оцінити успішність проходження тестів.
- Not Run Test Cases. Буде використано для демонстрації кількість тестів, які необхідно виконати для даної системи. Ця метрика допоможе виявити причини невиконання тестів і способи їх вирішення.
- Open/Closed Bugs. Буде використано для вираження відношення між відкритими багами та закритими. Ця метрика допоможе оцінити швидкість виправлення багів, а також знайти причини, які вплинули на те, що баг не було виправлено.
- Reopened/Closed Bugs. Буде використано для вираження відношення між перевідкритими багами та закритими. Ця метрика зможе оцінити ефективність закриття бага, а також допоможе виявити причини, які впливають на те, що виправлення багів знаходиться на низькому рівні.
- Bugs by Severity/Priority. Буде використано для вираження загальної кількості багів за серйозністю/пріоритету. Ця метрика допоможе визначити якість коду, який надано для тестування.

Після виконання кожного тесту будуть визначені наступні метрики:

- К-сть знайдених помилок;
- Час витрачений на тест;
- Успішність тестування;

● Результативність тестування.

4.5. Відстеження помилок та звітування

JIRA є продуктивною платформою управління помилками та проектами, розробленою компанією Atlassian. Вона надає розширені можливості для ефективного відстеження помилок, керування завданнями, спринтами та проектами. За допомогою JIRA ви можете:

- Легко створювати та контролювати помилки.
- Ефективно керувати завданнями та задачами.
- Здійснювати спільну роботу та забезпечувати комунікацію між учасниками проекту.
- Отримувати звіти та аналітику для розуміння стану проекту.

Моніторинг помилок повинен включати наступні кроки:

- Виявлення помилок під час тестування.
- Створення звіту.
- Виправлення помилок.
- Підготовка звіту про виконану роботу.
- Оновлення та відстеження програмного забезпечення.

Звіти про помилки повинні містити такі дані:

- Опис помилки.
- Рівень тестування, на якому була виявлена помилка.
- Модуль, де була виявлена помилка.
- Середовище та умови, за яких була виявлена помилка.
- Приклад помилки (скрішоти, логи).
- Пріоритет виправлення помилки.
- Особа, яка виявила помилку.

5. Засоби тестування (для кожного типу тестування)

Модульне тестування:

- Середовище для написання тестів для модулів NUnit
- Середовище модульного тестування: Jest
- Баг-трекер: JIRA.

Інтеграційне тестування:

- Середовище тестування NUnit.
- Середовище тестування Cypress.
- Баг-трекер: JIRA.

Системне тестування:

- Selenium.
- Postman.
- Cypress.
- JIRA.

Приймальне тестування:

- Cucumber
- SpecFlow
- Appium (for mobile development)

6. Вимоги до оточення

Для успішного проведення тестування на всіх рівнях необхідно виконати наступні вимоги:

- **Середовище для тестування**: Наявність окремого середовища, яке відтворює умови реального виробничого середовища з встановленими необхідними залежностями та конфігураціями.
- **Доступ до засобів тестування**: Наявність необхідного програмного забезпечення, фреймворків та інструментів для проведення тестування на різних рівнях.
- **Коректні вхідні дані для тестування**: Наявність набору тестових даних, які охоплюють різні сценарії та випадки використання, забезпечуючи повноту тестування.
- **Доступ до обмеженого функціоналу програми для тестувальників**: Забезпечення можливості отримання спеціальних облікових записів або прав доступу, які дають змогу тестувальникам перевірити функціональність, недоступну для звичайних користувачів.
- **Наявність тестової документації**: Забезпечення наявності актуальної документації, включаючи плани тестування, тест-кейси та інструкції для тестувальників, що описують очікувані результати та процедури тестування.
- **Моніторинг та звітність**: Забезпечення можливості моніторингу та збору даних про проведені тести, їх результати та відхилення, що дозволяє здійснювати контроль якості тестування та виявляти проблеми у процесі.

7. Ризики

Наступні проблеми можуть впливати на процеси тестування:

- зміни та модифікації програмного продукту, які не були попередньо заплановані та обговорені з тестовою командою;

- зміни в вимогах до програмного забезпечення, які не були обговорені з тестовою командою передчасно;
- затримки у виправленні помилок;
- затримки у постачанні нових збірок тестовій команді.

ВИСНОВКИ

Здобув навички дослідження предметної області та проектування системи за допомогою UML діаграм. Спроектував діаграму діяльності та класову для частини системи. Отримав досвід роботи у команді. Це дало мені можливість не тільки поглибити свої знання в галузі розробки програмного забезпечення, але й навчитися ефективно спілкуватися та працювати в команді. Я зрозумів важливість збору вимог, аналізування проекту та вислуховування думок інших учасників команди для досягнення успішного результату.