НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

«На правах рукопису»
УДК _____

До захисту допущено:
Завідувач кафедри
_____ *Сергій СТІРЕНКО*
«___»_____ 2023 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-науковою програмою «Інженерія програмного забезпечення комп'ютерних та інформаційних систем»

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Спосіб та засоби контролю якості інформаційної системи керування навчальним закладом вищої освіти»

Виконала:
студентка VI курсу, групи ІМ-11мн
Сіваченко Марина Геннадіївна _____

Керівник:
проф. кафедри ОТ, д.т.н., проф.
Клименко Ірина Анатоліївна _____

Консультант з нормоконтролю:
проф. Кафедри ОТ, д.т.н., проф.
Жабін Валерій Іванович _____

Рецензент:
проф. кафедри ІПІ, д.т.н., проф.
Стеценко Інна Вячеславівна _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2023 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти — другий (магістерський)
Спеціальність — 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма — «Інженерія програмного забезпечення комп'ютерних та інформаційних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Сергій СТІРЕНКО*
(підпис) (ініціали, прізвище)
«_____» _____ 2023 р.

ЗАВДАННЯ
на магістерську дисертацію студентці
Сіваченко Марині Геннадіївні

1. Тема дисертації «Спосіб та засоби контролю якості інформаційної системи керування навчальним закладом вищої освіти» науковий керівник дисертації Клименко Ірина Анатоліївна д.т.н., професор.

Затверджені наказом по університету від «20» березня 2023 р. № 1275-С

2. Термін подання студентом дисертації:

3. Об’єкт дослідження: *процес контролю якості інформаційних систем керування*

4. Предмет дослідження: *спосіб та засоби контролю якості інформаційної систем керування*

5. Перелік завдань, які потрібно розробити: *дослідити способи та засоби контролю кості інформаційної системи керування навчальним закладом вищої освіти, здійснити контроль якості інформаційної системи керування навчальним закладом вищої освіти*

6. Консультанти розділів дисертації:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормконтроль	Жабін В. І.		

1	Клименко І. А.		
2	Клименко І. А.		
3	Клименко І. А.		
4	Клименко І. А.		

7. Дата видачі завдання

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів дисертації	Примітка
1	*Затвердження теми роботи*	*01.02.2023*	
2	*Вивчення та аналіз завдання*	*02.02.2023 — 10.02.2023*	
3	*Вивчення і аналіз існуючих підходів*	*11.02.2023 — 28.02.2023*	
4	*Розробка та моделювання способу*	*01.03.2023 — 15.03.2023*	
5	*Застосування розробленого способу*	*16.03.2023 — 20.04.2023*	
6	*Оформлення пояснювальної записки*	*21.04.2023 — 06.05.2023*	
7	*Попередній захист та проходження нормативного контролю*	*14.05.2023*	
8	*Захист*	*22.05.2023*	

Студентка _____ Марина СІВАЧЕНКО

Науковий керівник _____ Ірина КЛИМЕНКО

Пояснювальна записка

до магістерської дисертації

на тему: «Спосіб та засоби контролю якості інформаційної системи керування навчальним закладом вищої освіти»

Київ – 2023

ЗМІСТ

РЕФЕРАТ 7

ВСТУП 10

РОЗДІЛ 1 12

ОГЛЯД ПОНЯТТЯ ІНФОРМАЦІЙНО-КЕРУЮЧИХ СИСТЕМ ТА ПРОБЛЕМАТИКИ КОНТРОЛЮ ЯКОСТІ 12

1.1. Інформаційно-керуючі системи 12

1.2. Аналіз проблематики контролю якості 14

1.3. Постановка завдання магістерської дисертації 15

ВИСНОВКИ ДО РОЗДІЛУ 1 17

РОЗДІЛ 2 18

АНАЛІЗ ІСНУЮЧИХ СПОСОБІВ ТА ЗАСОБІВ КОНТРОЛЮ ЯКОСТІ 18

2.1. Життєвий цикл програмного забезпечення 18

2.2. Дефекти програмного забезпечення 24

2.3. Способи контролю якості 28

2.4. Артефакти контролю якості 38

ВИСНОВКИ ДО РОЗДІЛУ 2 42

РОЗДІЛ 3 43

РОЗРОБКА АРТЕФАКТІВ КОНТРОЛЮ ЯКОСТІ 43

3.1. Аналіз продукту 43

3.2. Стратегія 50

3.3. Цілі 54

3.4. Критерії 56

3.5. Ресурси 57
3.6. Середовище 58
3.7. Розклад та оцінка 60
3.8. Результати 60
ВИСНОВКИ ДО РОЗДІЛУ 3 82
РОЗДІЛ 4 83
АНАЛІЗ ТА ЗАГАЛЬНИЙ ЗВІТ 83
4.1. Аналіз плану здійснення контролю якості 83
4.2. Звіт з контролю якості 86
ВИСНОВКИ ДО РОЗДІЛУ 4 92
ВИСНОВКИ 94
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 96
ДОДАТОК А 100

РЕФЕРАТ

на магістерську дисертацію

виконану на тему: Спосіб та засоби контролю якості інформаційної системи керування навчальним закладом вищої освіти

студенткою: Сіваченко Мариною Геннадіївною

Робота складається з вступу та чотирьох розділів. Робота налічує: 90 аркушів основного тексту, 72 ілюстрацій, 4 таблиць. Список використаної літератури налічує 33 джерела.

Актуальність теми. Контроль якості програмного забезпечення, є одним з найважливіших етапів життєвого циклу розробки програмного забезпечення. Жоден продукт не випускається на загальний вжиток без контролю якості, адже продукт що працює несправно тільки зруйнує процеси та унеможливить використання та збереження будь-яких даних. Це актуально значним чином й для інформаційних систем керування навчальними закладами вищої освіти, адже такі системи відображають повністю робочі процеси закладів і в таких системах все тісно пов’язане, тож якщо будуть збої в одному з модулів продукту — це може мати негативні наслідки для інших. Аби уникнути цих проблем, виникає необхідність здійснювати контроль якості системи, аби досягти високих результатів та найкращої роботи системи.

Мета і завдання дослідження. Метою магістерської роботи є розроблення засобів для контролю якості інформаційної системи для виявлення дефектів на різних рівнях оцінки якості, що в загальному сприятиме підвищенню ефективності функціонування інформаційних систем. Завданням дослідження є розроблення засобів контролю якості інформаційної системи на базі удосконаленого набору артефактів тестування та контролю якості для забезпечення необхідного рівня якості інформаційної системи для керування навчальним закладом вищої освіти.

Об'єкт дослідження. Процес контролю якості інформаційних систем керування навчальним закладом вищої освіти.

Предмет дослідження. Спосіб та засоби контролю якості інформаційної систем керування навчальним закладом вищої освіти.

Методи дослідження. Для виконання завдання були здійснені глибокий аналіз засобів, що сприяють здійсненню контролю якості, різні техніки контролю якості, такі як мануальне тестування (функціональне, нефункціональне та їх підвиди), а також автоматизоване.

Наукова новизна одержаних результатів. Запропоновано підхід до реалізації контролю якості інформаційної системи, що базується на індивідуальних плані та стратегії тестування заснованих на унікальному поєднанні артефактів функціонального та нефункціонального тестування. Застосування запропонованого підходу дозволяє зменшити кількість дефектів готового продукту і в результаті — забезпечити гарантії якості інформаційної системи.

Запропонований спосіб контролю якості інформаційної системи, за рахунок нетипового використання комплексного та адаптивного набору артефактів контролю якості, дозволяють ефективно оцінити якість інформаційної системи, за рахунок пришвидшення тестування і комплексно різностороннього охоплення складових компонентів системи.

Особистий внесок здобувача. Магістерська дисертація — є самостійно виконаною роботою, яка включає в себе різні самостійні дослідження та аналіз, що відображають особистий підхід до вирішення проблем якості інформаційної системи управління навчальним закладом вищої освіти. Формування завдання та мети здійснювалось разом з науковим керівником.

Практична цінність. Розроблені набори артефактів та способів контролю з елементами аналізу та аргументації можна застосовувати до інших схожих продуктів, а отримані результати по проходженню контролю якості, застосувати

для розуміння що необхідно виправити і пошуку слабких місць в роботі системи, для можливості загального вжитку нею.

Публікації.

M. Sivachenko ARCHITECTURAL REVIEW AND CONCEPTUAL DEVELOPMENT OF FACULTY INFORMATION SYSTEM “KPI-CONNECT” / M. Sivachenko, D. Nguyen, Y. Butskyi, K. Hryshchenko, V. Kryvets, D. Kryvoshei // Information, Computing and Intelligent systems № 2 – 2021. https://doi.org/10.20535/2708-4930.2.2021.247770

Ключові слова. Інформаційно-керуюча система, контроль якості, тестування, дефект, артефатки контролю якості, юзкейс, тесткейс, «KPI Connect».

ВСТУП

Сучасне положення речей та розвиток технологій дає наснагу автоматизовувати різні процеси у різних установах, будь-то: лікарня, школа, танцювальна студія, освітні заклади тощо. Сучасні технології дають розробити корисні додатки, сервіси, програмне забезпечення за для різних цілей. Цілями можуть бути: облік даних про клієнтів, облік даних співробітників, зберігання документів, зберігання інформації про осіб, поліпшення ведення успішності осіб у різних установах та інше. Для кейса інформаційних систем керування навчальним закладом вищої освіти базовими компонентами що поліпшать керування закладом — є ведення обліку всіх особистостей, від працівників до студентів, ведення обліку документів різного типу, керування групами, навчальними планами, освітніми програмами, аби всі ці процеси були автоматизовані та були доступні в лічені кліки, аби зробити ці всі процеси максимально простими та зменшити оборот паперових документів їх копій та решти.

Проте як би добре не було мати у користуванні інформаційну систему керування навчальним закладом, що спростить та автоматизує велику кількість процесів, якщо інформаційна система була розроблена не якісно, не було досить часу на тестування, система не проходила особливо важливі стадії життєвого циклу розробки, та попри все це була випущена в реліз, то така система не те що не покращить процеси обліку даних різного типу чи решту процесів керування, на превеликий жаль така інформаційна система додасть тільки клопоту та погіршить ведення цих процесів. Основними причинами такого становища будуть дефекти, неточності в роботі, втрата даних, не відповідність вимогах, тобто через те, що система не проходила належної перевірки контролю якості, чи стратегія та концепція контролю якості була сформована не найкращим чином, через що були здійснені не всі перевірки що б допомогли переконатись, що система працює справно, та розроблена якісно згідно з чіткими вимогами клієнта.

Завданням даної роботи є дослідження різноманітних концепцій, методологій та засобів контролю якості, аби створити покращену методологію контролю якості інформаційної системи та покращити процеси контролю якості використавши найоптимальніший набір артефактів та засобів для здійснення контролю якості.

РОЗДІЛ 1
ОГЛЯД ПОНЯТТЯ ІНФОРМАЦІЙНО-КЕРУЮЧИХ СИСТЕМ ТА ПРОБЛЕМАТИКИ КОНТРОЛЮ ЯКОСТІ

1.1. Інформаційно-керуючі системи

Інформаційно-керуюча система (ІКС) — це комп'ютерна система, яка керує, обробляє, зберігає та отримує дані та інформацію в організації [1]. Інформаційно-керуючі системи автоматизують бізнес-процеси, покращують комунікацію та співпрацю, а також забезпечують підтримку керування даними користувачам системи.

Системи управління інформацією зазвичай складаються з апаратного забезпечення, програмного забезпечення, сховищ даних та мережевої інфраструктури [1].

Апаратні компоненти включають сервери, настільні комп'ютери, ноутбуки, планшети та мобільні пристрої, які використовуються для доступу та обробки даних. Компоненти програмного забезпечення можуть включати операційні системи, бази даних, програми та інші інструменти, які використовуються для обробки та керування даними. Компонент зберігання даних може включати локальні або хмарні рішення для зберігання даних, такі як бази даних, сховища даних або централізовані сховища даних. Компонент мережевої інфраструктури може включати маршрутизатори, комутатори, брандмауери та інші пристрої, які використовуються для підключення апаратних і програмних компонентів і забезпечення безпечного доступу до даних.

Інформаційні системи управління можна класифікувати на різні категорії залежно від їх функцій і цілей [2]. Деякі поширені типи інформаційно-керуючи систем включають:

- Системи обробки транзакцій — ці системи використовуються для обробки транзакцій і керування ними в режимі реального часу. Системи обробки транзакцій часто використовуються в роздрібній торгівлі, банківській справі та інших галузях, де відбуваються великі обсяги транзакцій.

- Інформаційні системи менеджменту — ці системи використовуються для надання менеджерам інформації для підтримки прийняття рішень. Інформаційні системи менеджменту часто включає інформаційні панелі, звіти та інші інструменти, які можна використовувати для аналізу даних і відстеження показників ефективності.
- Системи підтримки прийняття рішень — ці системи використовуються для надання менеджерам інструментів і методів для аналізу складних даних і прийняття рішень. Системи підтримки прийняття рішень часто включають інструменти моделювання та імітації, інструменти візуалізації даних та інші аналітичні інструменти.
- Системи бізнес-аналітики — ці системи використовуються для збору, обробки й аналізу великих обсягів даних із різних джерел. Система бізнес-аналітики часто включає інструменти сховища даних, аналізу даних і візуалізації даних.
- Системи планування ресурсів підприємства — ці системи використовуються для інтеграції та управління різними бізнес-процесами в одній системі. Система планування ресурсів підприємства часто включає модулі для бухгалтерського обліку, нарахування заробітної плати, управління запасами та інші функції.

Інформаційно-керуючу систему також можна налаштувати та інтегрувати з іншими системами для задоволення конкретних потреб організації. Наприклад, інформаційно-керуючу систему можна інтегрувати з системами управління взаємовідносинами з клієнтами, системами управління ланцюгами поставок або платформами електронної комерції, щоб забезпечити безперебійний досвід для користувачів і клієнтів.

Загалом інформаційно-керуючі системи відіграють вирішальну роль в управлінні даними та інформацією в організаціях, дозволяючи їм приймати кращі рішення, підвищувати ефективність і управляти даними організації.

1.2. Аналіз проблематики контролю якості

Проблеми з контролем якості можуть виникнути в будь-якій галузі чи організації, де продукти чи послуги постачаються клієнтам. У контексті розробки програмного забезпечення проблеми з контролем якості можуть мати серйозні наслідки, наприклад системні збої, порушення безпеки та невдоволення клієнтів. Тому вкрай важливо визначити та вирішити проблеми з контролем якості, щоб гарантувати, що програмне забезпечення відповідає очікуванням користувачів і бізнесу.

Однією з поширених проблем контролю якості при розробці програмного забезпечення є відсутність або неадекватне тестування. Тестування допомагає виявити дефекти та помилки в системі, які можна виправити перед випуском програмного забезпечення для користувачів. Без належного тестування програмне забезпечення може бути випущено з дефектами, що може спричинити проблеми для користувачів і зрештою зашкодити репутації компанії. Крім того, відсутність належного тестування може призвести до збільшення кількості помилок і дефектів, які пізніше буде складно та дорого виправити [3].

Іншою проблемою контролю якості є відсутність зв’язку та співпраці між різними командами, які беруть участь у процесі розробки програмного забезпечення. Це може призвести до непорозумінь, невиконання вимог і помилок у програмному забезпеченні. Щоб вирішити цю проблему, компанії можуть запровадити гнучкі методи розробки, такі як *Scrum* або *Kanban*, які наголошують на співпраці, комунікації та постійному вдосконаленні [4].

Безпека також є критичним питанням контролю якості при розробці програмного забезпечення. У сучасну цифрову епоху програмні додатки часто стають мішенню кіберзловмисників, які намагаються використати вразливі місця в системі, щоб отримати несанкціонований доступ або викрасти конфіденційні дані. Тому розробники програмного забезпечення повинні надавати пріоритет безпеці та впроваджувати безпечні методи кодування, такі як шифрування, автентифікація та контроль доступу [5].

Авжеж проблеми з контролем якості також можуть виникати через використання застарілої або несумісної технології, що може призвести до проблем із сумісністю, збоїв системи та проблем із продуктивністю. Щоб вирішити цю проблему, розробники програмного забезпечення повинні йти в ногу з останніми технологічними тенденціями та гарантувати, що їх програмне забезпечення сумісне з різними пристроями та платформами.

Підсумовуючи, питання контролю якості викликають серйозне занепокоєння у розробці програмного забезпечення, і для їх вирішення потрібен проактивний підхід, який наголошує на співпраці, спілкуванні та тестуванні. Впроваджуючи найкращі методи розробки програмного забезпечення та надаючи пріоритет контролю якості, компанії можуть постачати високоякісне програмне забезпечення, яке відповідає очікуванням їхніх клієнтів і покращує їхню репутацію на ринку.

1.3. Постановка завдання магістерської дисертації

Якість у інформаційно-керуючих системах є критичним питанням, яке пов’язане з точністю, повнотою, релевантністю та надійністю даних та інформації, які зберігаються та обробляються в інформаційній системі до яких потім можна отримати доступ. Виміри якості в інформаційно-керуючій системі включають наступне [6]:

- Якість даних. Цей вимір якості стосується точності, повноти, послідовності та своєчасності даних, які обробляються та якими керують в інформаційній системі. Дані мають бути точними та актуальними, щоб гарантувати надійність, ефективність та простоту у користуванні.
- Якість інформації. Цей вимір якості стосується актуальності, зручності використання та надійності інформації, що зберігає інформаційна система. Інформація має відповідати потребам зацікавлених сторін і представлена у форматі, який є легким для розуміння та використання.

- Якість системи — цей вимір якості стосується функціональності, надійності та продуктивності інформаційно-керуючої системи. Система повинна надійно та ефективно виконувати свої призначені функції та бути доступною, коли це необхідно.
- Якість обслуговування. Цей вимір якості стосується якості послуг підтримки, які надаються користувачам інформаційно-керуючої системи. Служби підтримки мають бути чуйними, корисними та ефективними у вирішенні будь-яких проблем або проблем, які можуть виникнути.

Окрім цих параметрів, є кілька інших факторів, які впливають на загальну якість іноформаційно-керуючої системи, зокрема:

- Відповідність — інформаційно-керуюча система має відповідати потребам за для яких була розроблена, чітко мати всі необхідні функції та формати, що були передбачені замовником, налічувати все необхідне аби керування даними організації було максимально ефективним та таким як очікувалось.
- Масштабованість — інформаційно-керуюча система повинна мати можливість масштабуватися від меншого та більшого у міру того, як потреби організації змінюються з часом.
- Сумісність — інформаційно-керуюча система повинна мати можливість працювати на різних девайсах, у різних операційних система для безперебійного обміну даними та інформацією.

Забезпечення якості в інформаційно-керуючих системах вимагає комплексного підходу, який передбачає планування, проектування, впровадження та моніторинг системи протягом тривалого часу. Це вимагає залучення різних зацікавлених сторін, включаючи IT-фахівців, бізнес-лідерів і кінцевих користувачів, щоб забезпечити відповідність системи потребам організації та її користувачів.

Загалом, параметри якості в інформаційно-керуючих системах мають вирішальне значення для забезпечення ефективності та надійності системи в обробці та управлінні даними та інформацією в організації.

ВИСНОВКИ ДО РОЗДІЛУ 1

В даному розділі було розглянуто поняття інформаційно-керуючих систем, проаналізована проблематика контролю якості в розглянутих системах та поставлено завдання магістерської дисертації.

В результаті дослідження можна зробити висновок про важливість контролю якості в інформаційно-керуючих системах, оскільки вони мають велике значення для різних організацій у забезпеченні ефективного управління інформацією.

Також було встановлено, що проблематика контролю якості в інформаційно-керуючих системах є актуальною та потребує подальших досліджень і вдосконалення наявних підходів.

Завдання магістерської дисертації полягає у розробці артефактів для контролю якості інформаційної системи керування навчальним закладом вищої освіти.

РОЗДІЛ 2
АНАЛІЗ ІСНУЮЧИХ СПОСОБІВ ТА ЗАСОБІВ КОНТРОЛЮ ЯКОСТІ

2.1. Життєвий цикл програмного забезпечення

Життєвий цикл програмного забезпечення (ЖЦПЗ) — це процес, який описує життєвий цикл створення, розробки та підтримки програмного забезпечення від початкової ідеї до відкладення проекту [7].

В свою чергу система життєвого циклу розробки має кілька переваг, таких як [7]:

- Структурований підхід. ЖЦР забезпечує структурований підхід до розробки програмного забезпечення, що дозволяє забезпечити якість та зменшити ризики помилок в процесі розробки.
- Управління проектом. Система ЖЦР допомагає управляти проектом з точки зору планування, ресурсів, затрат та часових рамок.
- Результативність. За допомогою системи ЖЦР можна досягнути кращої результативності в розробці програмного забезпечення, оскільки вона дозволяє контролювати весь процес розробки.
- Надійність. Система ЖЦР допомагає забезпечити надійність програмного забезпечення та зменшити ризики помилок та вразливостей.
- Підтримка. Система ЖЦР дозволяє забезпечити ефективну підтримку програмного забезпечення після випуску на ринок, оскільки вона забезпечує збір та аналіз даних про використання програмного забезпечення.
- Поліпшення якості. ЖЦР допомагає покращити якість програмного забезпечення, оскільки вона дозволяє виявляти помилки та недоліки на ранніх етапах розробки.

Зазвичай, життєвий цикл програмного забезпечення складається з наступних етапів (рис. 2.1) [8]:

- Планування. Цей етап включає в себе визначення вимог та функцій продукту, оцінку вартості та ресурсів, а також визначення часових рамок.
- Розробка. Під час цього етапу програмне забезпечення перебуває у стадії розробки та випробування. Цей етап може складатися з декількох ітерацій, залежно від складності проекту.
- Тестування. Після розробки програмного забезпечення, його тестують, щоб переконатися, що воно відповідає вимогам та функціям, визначеним на етапі планування.
- Випуск. Коли програмне забезпечення успішно пройшло тестування та відповідає вимогам, воно може бути випущене на ринок.
- Підтримка. Цей етап включає в себе виправлення помилок та недоліків у програмному забезпеченні, а також надання користувачам підтримки та допомоги.
- Відкладення. Цей етап відбувається, коли програмне забезпечення вже не використовується або виходить зі свого терміну служби.

Рис. 2.1. Життєвий цикл програмного забезпечення

2.1.1. Моделі життєвого циклу розробки програмного забезпечення

Каскадна модель (рис. 2.2) — це традиційна модель, яка передбачає послідовну розробку програмного забезпечення, починаючи з етапу збору вимог і закінчуючи етапом тестування та впровадження. Вона не дозволяє гнучкості управління змінами вимог у процесі розробки, оскільки кожна фаза повинна бути завершена, перш ніж почати наступну. Каскадна модель не дозволяє ефективно використовувати результати тестування для вдосконалення дизайну та вимог програмного забезпечення. Це може призвести до того, що помилки виявляться на пізніших стадіях розробки, коли виправлення їх буде важче та дорожче. Ця модель використовується, коли вимоги до програмного забезпечення добре визначені та незмінні протягом всього процесу розробки [9].

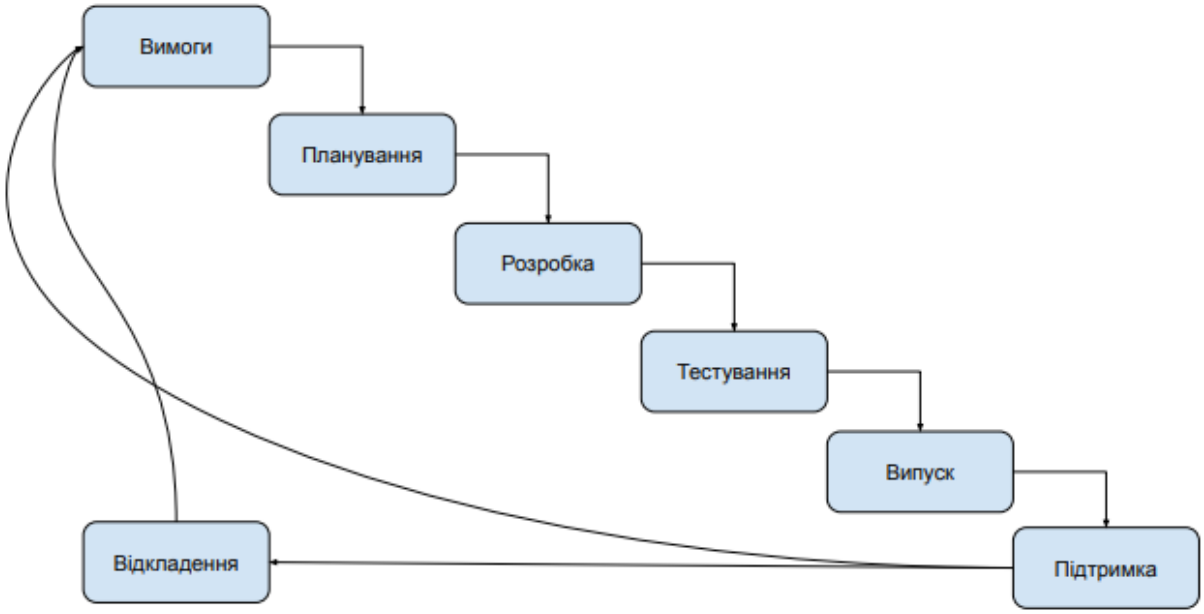

Рис. 2.2. Каскадна модель

Ітераційна модель (рис. 2.3) — передбачає ітераційну послідовність етапів розробки, кожен з яких складається зі свого власного планування, розробки, тестування та випуску.

Кожна ітерація (цикл) є повним життєвим циклом, що передбачає певний обсяг розробки програмного забезпечення. Після завершення кожної ітерації, розробники отримують зворотний зв'язок від клієнтів і користувачів, який використовується для покращення продукту в майбутньому.

Ітераційна модель є підходом до розробки програмного забезпечення, який дозволяє розробникам швидко відповідати на зміни вимог та забезпечує гнучкий підхід до розробки продукту. Однак, слід враховувати, що ця модель має свої обмеження та не є універсальним рішенням для всіх типів проектів [10].

Рис. 2.3. Ітераційна модель

Спіральна модель (рис. 2.4) — поєднує в собі ітераційну розробку з елементами управління ризиками. Ця модель полягає в виконанні послідовних ітераційних кругів вздовж спіралі, кожен з яких охоплює чотири основні етапи: планування, ризик-аналіз, розробка і оцінка.

Спіральна модель може бути особливо ефективною в ситуаціях, коли вимоги змінюються, або коли є значні ризики, пов'язані з розробкою продукту. Вона також може бути корисною для проектів з великою складністю та довгим терміном розробки. Однак, вона не є панацеєю для всіх проектів, і вибір методології повинен залежати від конкретних потреб та вимог проекту [11].

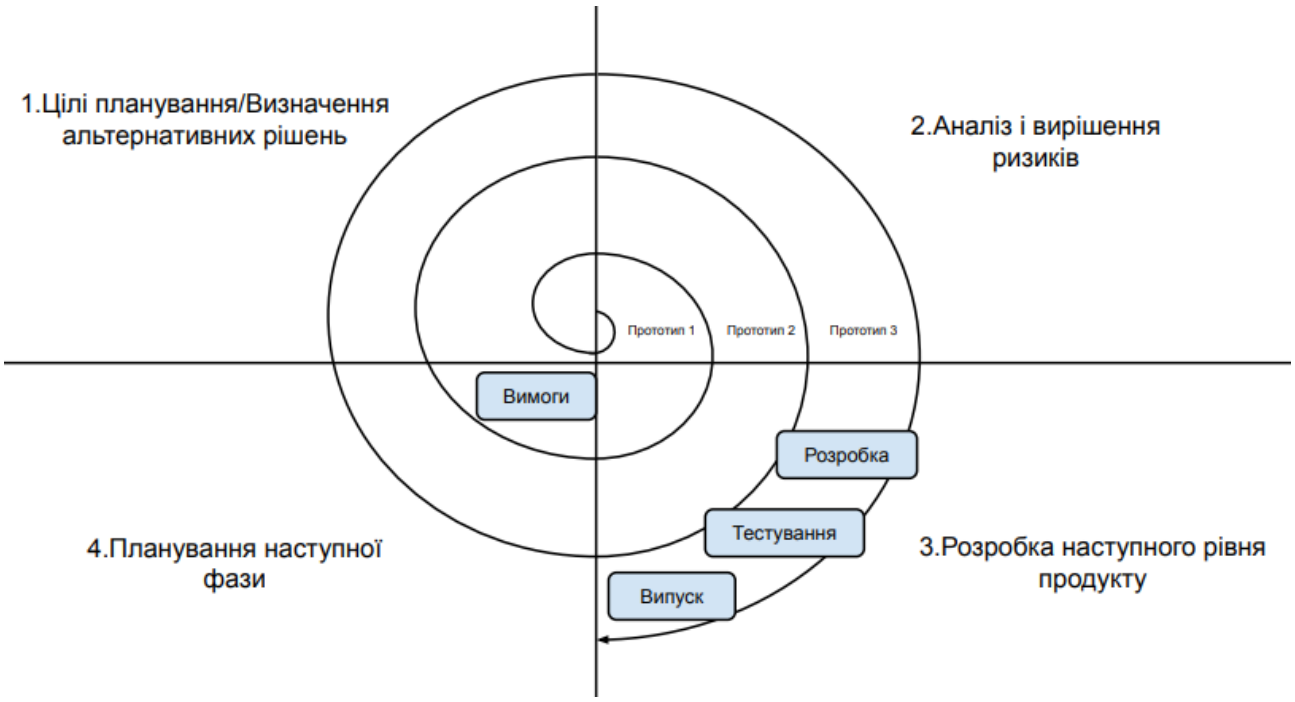

Рис. 2.4. Спіральна модель

Інкрементна модель життєвого циклу розробки програмного забезпечення передбачає поділ проекту на невеликі частини, які розробляються поетапно. Кожен етап є самостійним проектом, який розробляється та тестується окремо перед тим, як об'єднати його з попередніми етапами (рис. 2.5).

Інкрементна модель передбачає повторювані ітерації, протягом яких розробляються та тестуються невеликі блоки програмного забезпечення. Кожен блок додається до попередньо розробленого програмного забезпечення, щоб утворити більш повну версію продукту. Кожна ітерація може включати в себе різні етапи життєвого циклу, такі як аналіз вимог, проектування, розробка та тестування.

Модель дозволяє швидко розробляти та тестувати невеликі блоки програмного забезпечення, що забезпечує більшу гнучкість управління змінами вимог та можливість швидкого відгуку на зміни вимог клієнта.

Вона може бути менш ефективною для великих проектів, де необхідно розробляти багато блоків програмного забезпечення, оскільки це може затримати процес розробки. Крім того, ця модель може вимагати більшої координації між командами розробників, що може бути складним для організації [12].

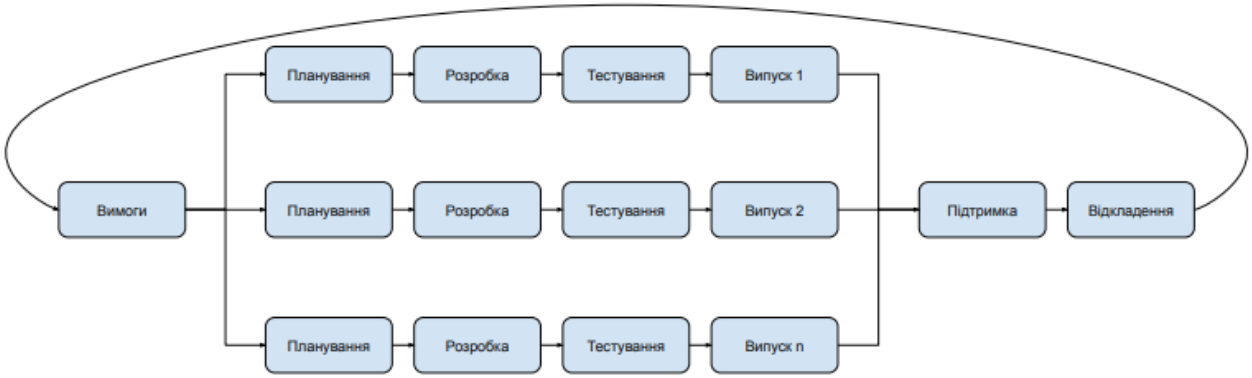

Рис. 2.5. Інкрементна модель

V-модель життєвого циклу розробки програмного забезпечення є моделлю, яка поєднує елементи каскадної моделі з ідеями тестування та верифікації (рис. 2.6).

Ця модель базується на підході, коли кожен етап розробки програмного забезпечення має відповідний етап тестування, що забезпечує більш високу якість продукту та раннє виявлення та виправлення помилок.

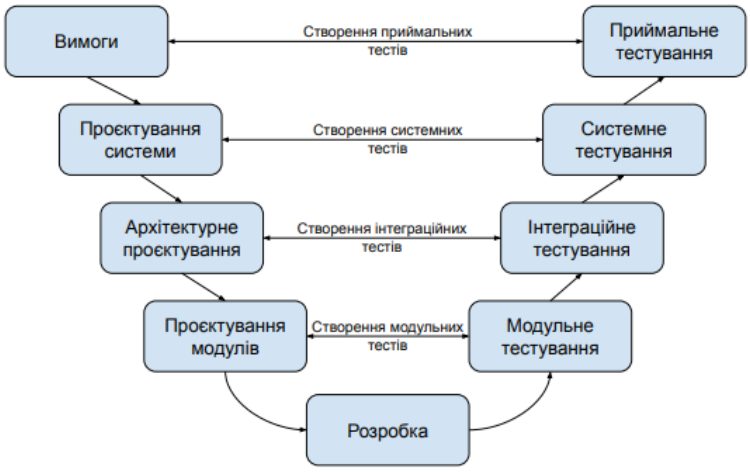

Рис. 2.6. V-модель

V-модель дозволяє зменшити ризик помилок в програмному забезпеченні, тому що кожен етап розробки має відповідний етап тестування, що забезпечує раннє виявлення та виправлення помилок. Крім того, ця модель дозволяє забезпечити високу якість продукту, оскільки перевірка відбувається на кожному етапі розробки. Вона передбачає строгий порядок виконання етапів, що не завжди підходить для динамічних проектів, де вимоги можуть змінюватися

протягом розробки. Крім того, відсутня можливість відстеження та виправлення помилок, які виявляються на пізніших етапах розробки [13].

2.2. Дефекти програмного забезпечення

Дефекти програмного забезпечення можуть бути серйозною проблемою для розробників, користувачів та бізнесу загалом. Вони можуть призвести до неправильної роботи програми, що може спричинити втрату даних, збій в роботі, або навіть загрожувати безпеці користувачів.

Отже, виявлення та виправлення дефектів є важливою частиною процесу розробки програмного забезпечення. Це допомагає забезпечити високу якість та надійність ПЗ, що позитивно впливає на його прийняття користувачами та ринком в цілому.

У програмному забезпеченні, терміни «помилка», «дефект» та «відмова» використовуються для позначення різних станів проблеми з ПЗ (рис. 2.7). Хоча ці терміни часто використовуються як синоніми, вони мають свої відмінності та важливість у процесі розробки та тестування програмного забезпечення.

Помилка (error) — це людський фактор або помилка в проектуванні, що призводить до неправильного виконання функцій програмного забезпечення. Помилка може виникнути на будь-якому етапі розробки ПЗ, від проектування до впровадження.

Дефект (defect) — це конкретний прояв помилки, який призводить до неправильної роботи програми або несправності її функцій. Дефект може виникнути внаслідок помилки або неочікуваної поведінки програми.

Відмова (failure) — це відсутність відповіді від програми або її неправильна робота під час виконання функції. Відмова може бути викликана дефектом або невідомою помилкою, іноді це може призвести до краху програми або втрати даних [14].

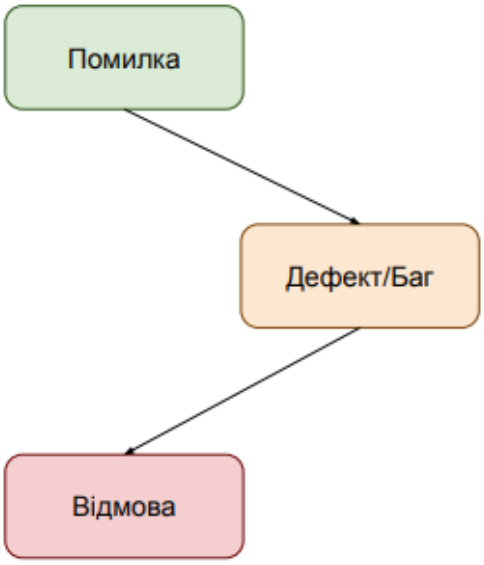

Рис. 2.7. Стани проблеми в програмному забезпеченні

2.2.1. Типи дефектів

Існує безліч різних типів дефектів в програмному забезпеченні, що можуть впливати на його роботу та функціональність. Деякі з найпоширеніших типів дефектів включають [15]:

- Синтаксичні помилки. Пов'язані зі структурою коду, такі як неправильне використання роздільників, відсутність крапки з комою, неправильні відступи тощо.
- Логічні помилки. Пов'язані з логікою програми, які не дозволяють їй працювати правильно. Це може бути неправильне рішення в певній ситуації, неправильна обробка вхідних даних або помилка в алгоритмі обчислень.
- Функціональні помилки. Пов'язані з неправильною реалізацією функціоналу програми. Наприклад, програма не повинна виконувати певних операцій, але вона це все ж робить.
- Помилки інтерфейсу користувача. Пов'язані з неправильною роботою інтерфейсу програми, що може призвести до незручностей для користувача або навіть до неможливості користування програмою.

- Помилки безпеки. Можуть призвести до порушення безпеки програми, включаючи злам інформаційної системи, зловживання правами користувача та інші.
- Помилки продуктивності. Впливають на швидкість та ефективність програми. Це може бути неправильне використання ресурсів системи, помилки в оптимізації коду, неправильне налаштування бази даних тощо.

Важливо виявляти та виправляти дефекти в ранній стадії розробки, щоб забезпечити якість та надійність програмного забезпечення та зменшити витрати на його тестування та підтримку в майбутньому. Для виявлення дефектів використовують різні методи та підходи, такі як юніт-тестування, функціональне тестування, тестування на відмову, тестування безпеки та інші.

Один з ефективних способів зниження кількості дефектів — це застосування практик розробки програмного забезпечення, таких як тестування на відповідність, автоматизоване тестування, контроль версій та код-рев'ю. Застосування цих практик може допомогти виявляти дефекти на ранніх стадіях розробки, а також забезпечувати відповідність коду до стандартів якості та безпеки [16].

Важливо розуміти, що дефекти є невід'ємною частиною розробки програмного забезпечення. Кожен дефект може відкрити нові можливості для поліпшення та оптимізації програми. Але одночасно, невиправлені дефекти можуть призвести до серйозних наслідків для користувачів та бізнесу, тому важливо надавати їм належну увагу та пріоритет у розробці програмного забезпечення.

2.2.2. Життєвий цикл дефекту

Життєвий цикл дефекту — це процес, який відбувається в процесі розробки програмного забезпечення, коли виявляється, що код містить помилку або дефект. Життєвий цикл дефекту складається з кількох етапів, кожен з яких

відображає різні аспекти виявлення, виправлення та тестування дефекту. Основні етапи життєвого циклу дефекту (рис. 2.8) [17]:

- Виявлення дефекту. Дефект може бути виявлений під час тестування, розробки або під час використання програмного забезпечення користувачами.
- Опис дефекту. Після виявлення дефекту, його необхідно описати якомога детальніше, вказавши приклади відтворення та симптоми.
- Класифікація дефекту. Дефекти можуть бути класифіковані за типом (наприклад, логічні помилки, синтаксичні помилки, проблеми з продуктивністю), за серйозністю (критичні, середньої важливості, незначні) та іншими параметрами.
- Розгляд дефекту. Розгляд дефекту може проводитися керівництвом, командою розробки, тестування або користувачами, які звітують про дефект.
- Виправлення дефекту. Розробники виправляють дефект, виходячи з опису та симптомів, наданого тим, хто виявив дефект. Після виправлення дефекту код повинен пройти процес тестування.
- Перевірка виправлення. Після виправлення дефекту код перевіряється, щоб переконатися, що дефект був виправлений та немає побічних ефектів.
- Закриття дефекту. Коли дефект успішно виправлено та перевірено, він закривається та відзначається в системі управління дефектами.
- Моніторинг виправлення. Дефект може повернутися, якщо виправлення було некоректним або якщо у коді виникли нові помилки. Тому важливо відслідковувати моніторинг виправлення та повторно перевіряти код на наявність дефектів.
- Аналіз виникнення дефекту. Після виправлення дефекту, команда може провести аналіз причин його виникнення, щоб уникнути подібних дефектів у майбутньому.

Ці етапи можуть бути виконані в різному порядку або бути повторними в залежності від конкретної ситуації та методів розробки програмного забезпечення. Важливо забезпечити ефективність та точність кожного етапу життєвого циклу дефекту для підтримки якості програмного забезпечення та задоволення потреб користувачів [17].

На діаграмі відображені основні статуси дефекту після того як він був виявлений у робочому процесі (рис. 2.9).

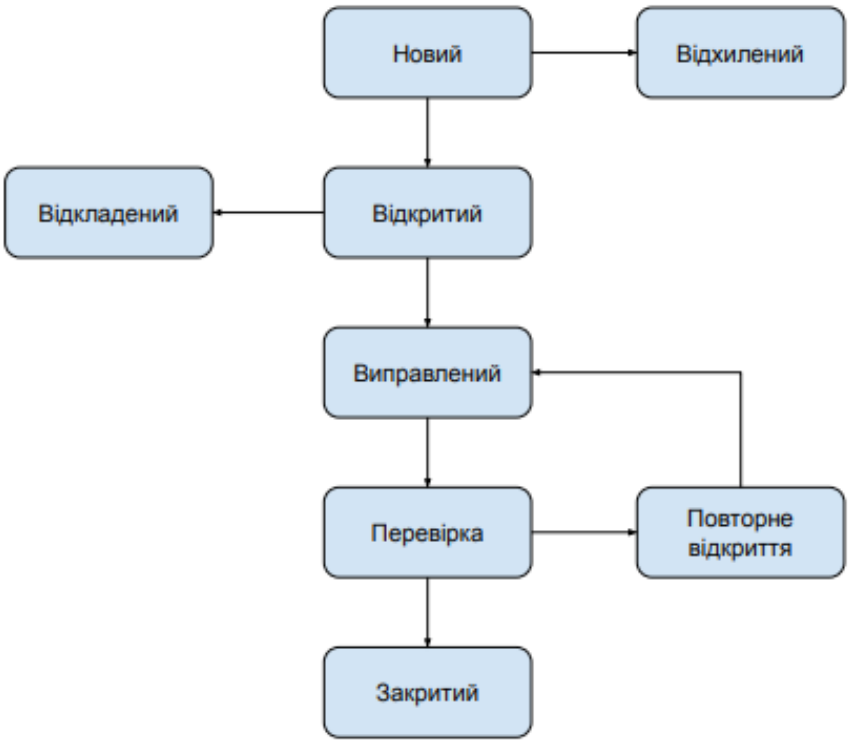

Рис. 2.9. Статуси дефекту в робочому процесі

2.3. Способи контролю якості

Якість програмного забезпечення є ключовим фактором для успіху будь-якого проекту. Якщо програмне забезпечення не функціонує належним чином, то це може призвести до збитків, які можуть стати критичними для бізнесу.

Контроль якості програмного забезпечення є необхідним етапом у процесі розробки, що дозволяє впевнитесь, що програмне забезпечення відповідає вимогам, є надійним, ефективним та безпечним.

Задля здійснення контролю якості використовують наступні загальні способи:

- Тестування
- Ревізія коду

2.3.1. Тестування

Тестування програмного забезпечення — це процес перевірки програмного забезпечення на відповідність вимогам та очікуванням користувачів. Цей процес включає в себе запуск програмного забезпечення з метою виявлення помилок та інших проблем, які можуть вплинути на роботу програми.

Основною метою тестування програмного забезпечення є забезпечення високої якості програмного забезпечення та зменшення ризику виникнення помилок у виробничому середовищі. Під час тестування програмного забезпечення, тестувальники перевіряють різні аспекти програмного забезпечення, такі як його функціональність, продуктивність, надійність, безпеку та сумісність з іншими системами.

Тестування програмного забезпечення може бути виконано вручну або автоматично, в залежності від потреб проекту та можливостей. Під час виконання тестів, тестувальники документують та повідомляють про знайдені проблеми, що допомагає розробникам виправити їх та забезпечити високу якість програмного забезпечення.

Тестування програмного забезпечення є важливою складовою процесу розробки програмного забезпечення та допомагає забезпечити, що програмне забезпечення відповідає вимогам користувачів та бізнесу, є надійним та безпечним в експлуатації [18].

Існують наступні базові принципи тестування [19]:

- Тестування показує наявність дефектів. Передбачає, що процес тестування призначений для виявлення дефектів та помилок в програмному

забезпеченні. Цей принцип базується на припущенні, що неможливо створити програмне забезпечення, яке не містить жодної помилки.

- Вичерпне тестування неможливе. Базується на тому, що програмне забезпечення може мати дуже велику кількість можливих шляхів взаємодії з користувачем, а також дуже велику кількість можливих комбінацій вхідних даних. Оскільки кількість можливих варіантів є нескінченною, неможливо вичерпно протестувати кожен можливий варіант. Важливо зосередитись на тестуванні найважливіших та найбільш імовірних варіантів взаємодії з програмним забезпеченням та вхідних даних, які найбільше можуть впливати на його роботу та найбільш імовірно можуть призвести до помилок.
- Раннє тестування. Тестування яке має початись як найраніше після того як було прийняте рішення почати розробку. Це означає, що тестування повинно починатися ще на етапі розробки прототипу, і продовжуватися на кожному етапі розробки.
- Скупчення дефектів. Передбачає те, що дефекти в програмному забезпеченні зазвичай згруповані разом в певних місцях (20% складових програмного забезпечення містять 80% дефектів). Цей принцип допомагає тестувальникам ефективніше виявляти та виправляти дефекти, зосереджуючись на тих частинах програми, де ймовірніше виникнення проблем.
- Парадокс пестицидів. Стверджує, що повторне використання одних і тих же тестових сценаріїв може призвести до того, що дефекти більше не виявляються, навіть якщо вони існують. Це пов'язано з тим, що тестувальники можуть стати «сліпими» до дефектів через часте використання одних і тих же тестів, і відповідно, не помічати їх.
- Тестування залежить від контексту. Стверджує, що тестування повинно відбуватися з урахуванням контексту, в якому розробляється програмний продукт і здійснюється його експлуатація. Контекст може включати такі

фактори, як бізнес-цілі, потреби користувачів, особливості апаратного та програмного забезпечення, в якому працює програмний продукт, технічні вимоги, терміни розробки та інші. Для кожного проекту необхідно розробляти свій унікальний підхід до тестування, який відповідає контексту проекту. Наприклад, тестування медичного програмного забезпечення буде відрізнятися від тестування ігрового програмного забезпечення, а тестування програмного забезпечення для малих підприємств буде відрізнятися від тестування програмного забезпечення для великих корпорацій.

- Відсутність помилок — омана. Засвідчує про те, що навіть у випадку коли максимальна частина дефектів виправлена може скластись так, що програмним забезпеченням все ще не можна буде користуватись. Така ситуація має право на існування, якщо вимоги були невірно складені, і не перевірені ретельно. В такому випадку як би програмне забезпечення не було добре перевірено на дефекти, випускати в реліз його не можна буде бо воно не відповідатиме вимогам клієнта.

Надалі будуть розглянуті різні типи тестування (рис. 2.10), які часто застосовуються під час контролю якості програмного забезпечення. Для досягнення найкращого результату під час контролю якості необхідно задіяти різні техніки тестування ті їх комбінації.

Мануальне тестування. Процес перевірки програмного продукту з використанням інтуїції, досвіду та знань, без використання автоматизованих засобів. Під час мануального тестування тестувальник виконує різні тест-кейси та сценарії з метою виявлення дефектів та переконання в тому, що програмний продукт працює так, як очікується.

Автоматизоване тестування. Виконання тестів з використанням автоматизованих засобів, які дозволяють створювати, виконувати та аналізувати результати тестування без участі людини. Автоматизоване тестування дозволяє

прискорити процес тестування, знизити його вартість та забезпечити більш високу якість програмного забезпечення [20].

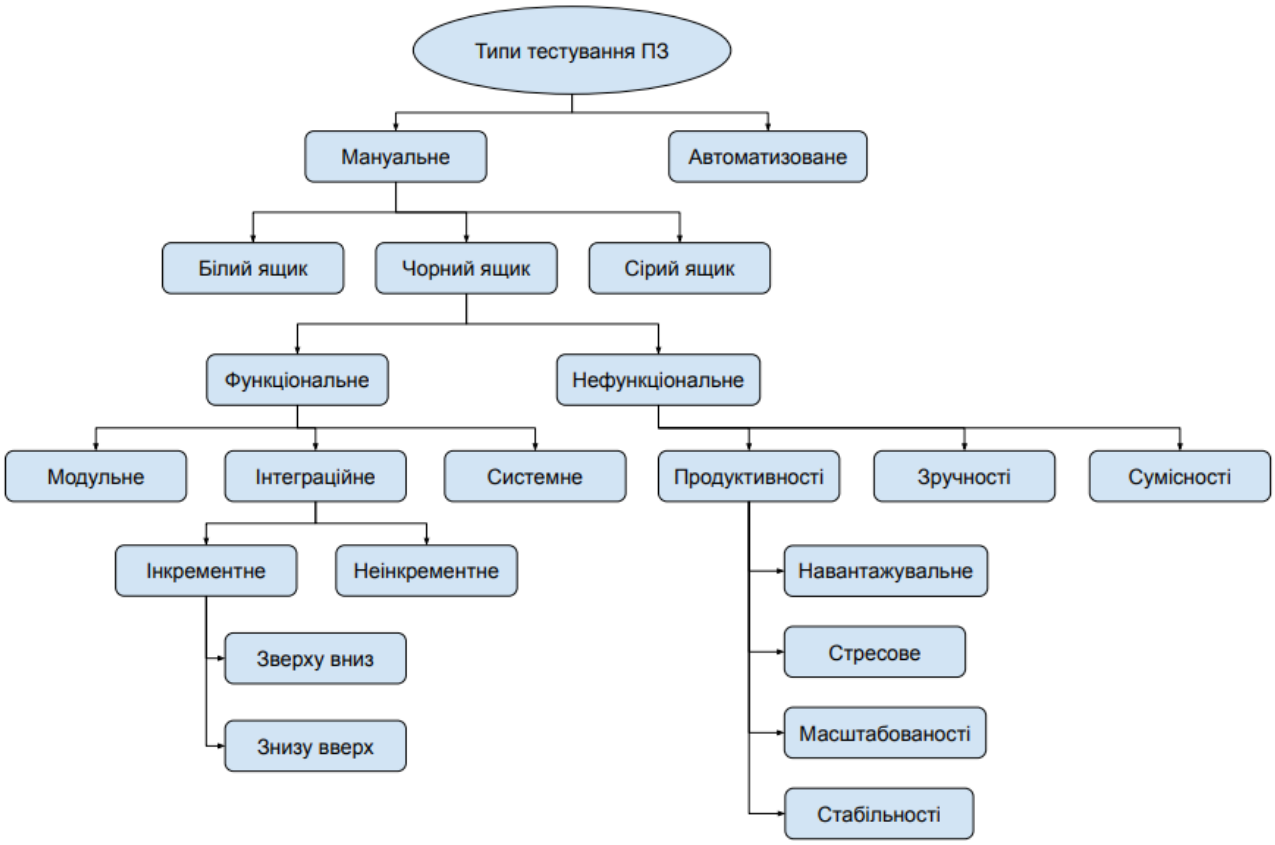

Рис. 2.10. Типи тестування

Тестування білого ящика (підтип мануального тестування). Метод тестування програмного забезпечення, коли тестувальник має повний доступ до внутрішніх деталей коду програми. У техніці білого ящика тестувальник виконує тестування на основі знань про структуру програми, тому він може перевірити, як добре працює кожна складова програми [21].

Тестування чорного ящика (підтип мануального тестування). Метод тестування, при якому тестувальник не має доступу до внутрішньої структури програми та не знає, як саме вона працює. Тестувальник працює з програмою як з чорним ящиком, тобто він тестує програму зовнішньо, перевіряючи її вхідні та вихідні дані. В даній техніці тестувальник перевіряє функціональність програми, здатність взаємодіяти з іншими системами та середовищами, а також здатність програми зберігати та оброблювати дані. Цей підхід зазвичай використовується

для тестування програмного забезпечення в цілому, після того, як програма вже була розроблена та підготовлена для тестування [21].

Тестування сірого ящика (підтип мануального тестування). Комбінація двох попередніх методів (чорного та білого ящика). Використовується, коли тестувальник повинен перевірити функціональність певної частини програми та її взаємодію з іншими частинами. Цей підхід дозволяє тестувальнику зосередитися на перевірці тієї частини програми, яка є найбільш важливою для клієнта або має найбільший ризик для функціонування програми в цілому [21].

Функціональне тестування (підтип техніки чорного ящика). Спрямоване на перевірку відповідності функціональних вимог системи вимогам, які були визначені клієнтом та задокументовані.

Нефункціональне тестування (підтип техніки чорного ящика). Тестування що спрямоване на перевірку тих властивостей, що не є реальними функціями програмного забезпечення (продуктивність, зручність, сумісність тощо) [22].

Модульне тестування (підтип функціонального тестування). Це тестування окремих функціональних одиниць програмного забезпечення (модулів). Кожен модуль тестується окремо з метою перевірки його відповідності вимогам та правильної роботи в ізоляції від інших модулів [23].

Інтеграційне тестування (підтип функціонального тестування). Орієнтоване на виявлення дефектів у взаємодії компонентів системи під час їх інтеграції. Цей тип тестування зазвичай виконується після модульного тестування, коли окремі модулі програми були вже протестовані і перевірені на відповідність вимогам [23].

Системне тестування (підтип функціонального тестування). Основні завдання полягають у тому, щоб перевірити систему на відповідність її вимогам та стандартам, виявити та виправити дефекти та недоліки, забезпечити якість та надійність системи в цілому [23].

Інкрементне тестування (підвид інтеграційного тестування). Тести виконуються поступово, по частинах, додаючи нові функції або компоненти до

продукту і перевіряючи, що вони працюють правильно і не впливають на роботу вже наявних частин продукту. Такий підхід дозволяє забезпечити більшу стабільність і надійність продукту на ранніх етапах розробки та зменшити витрати на налагодження програми в цілому. Має наступні підвиди [24]:

- Тестування зверху вниз (підвид інкрементного тестування). Використовується для тестування програмного забезпечення, починаючи з найвищого рівня ієрархії та доходячи до найнижчого. На кожному рівні розробник додає функціональність до вже наявного коду та тестує її до того, як перейти до наступного рівня. На кожному рівні тестування можуть бути використані заміщувачі або заглушки, які допоможуть замінити елементи, які ще не були розроблені на цей момент.
- Тестування знизу вверх (підвид інкрементного тестування). Спочатку тести виконуються на найнижчому рівні — окремих модулях програми, а потім, коли модулі успішно пройшли тестування, їх поєднують у більші блоки, які також тестуються, і так далі до того моменту, коли всі блоки об'єднуються в єдину систему і пройшли успішно тестування.

Неінкрементне тестування (підвид інтеграційного тестування). Тестування виконується поступово об’єднуючи модулі один за одним, воно може виконуватись одразу всіх, коли немає чіткого поняття дочірніх та батьківських модулів [24].

Тестування продуктивності (підвид нефункціонального тестування). Процес тестування програмного забезпечення з метою оцінки продуктивності та стабільності роботи системи під очікуваним навантаженням. Зазвичай тестування продуктивності виконується з використанням спеціальних інструментів, які дозволяють створювати великі навантаження на систему та аналізувати її реакцію на ці навантаження. Включає наступні види тестування [25]:

- Навантажувальне. Перевірка роботи системи за умови, коли на неї приходить велике навантаження.

- Стресове. Перевірка роботи системи за умови, коли на неї приходить дуже велике навантаження, що перевищує максимальне значення, передбачене вимогами.
- Масштабованості. Використовується для визначення, наскільки добре система або додаток може працювати з більшим обсягом даних, більшою кількістю користувачів, більшим обсягом операцій та транзакцій. Причому, при збільшенні навантаження на систему, її продуктивність та реакційність не повинні зменшуватися, адже вона повинна бути здатна працювати з більшим обсягом даних та користувачів.
- Стабільності. Перевірка роботи системи за тривалий період часу під постійним навантаженням.

Тестування зручності (підвид нефункціонального тестування). Тестування інтерфейсу користувача спрямоване на оцінку зручності використання програмного забезпечення для кінцевих користувачів. Основна мета цього тестування — переконатися, що програмне забезпечення є зрозумілим, легким в навчанні та використанні, а також забезпечує користувачеві комфортну роботу з ним. У процесі тестування зручності оцінюється, наскільки легко користувач може взаємодіяти з програмним забезпеченням, чи інтуїтивно зрозумілі його інтерфейс, ергономіка його елементів, зручність розташування елементів, доступність функцій, наявність документації тощо [26].

Тестування сумісності (підвид нефункціонального тестування). Спрямований на перевірку сумісності програмного забезпечення з різноманітними обладнанням, операційними системами, браузерами, базами даних, іншими додатками та сервісами. Цей вид тестування допомагає виявити можливі проблеми, що виникають під час взаємодії між різними компонентами програмного забезпечення. Наприклад, можливість взаємодії між браузерами та веб-сайтом, сумісність між операційними системами та додатками, тощо [26].

2.3.2. Ревізія коду

Ревізія коду — це процес перегляду програмного коду, який виконується іншим розробником або командою розробників з метою забезпечення якості та вдосконалення кодової бази.

Під час ревізії коду, інші кваліфіковані особи аналізують код, який був написаний розробником, та шукають помилки, недоліки, потенційні проблеми безпеки та можливість покращення кодової бази. Ревізія коду може бути виконана вручну або за допомогою автоматичних інструментів, які допомагають виявляти проблеми з кодом.

Існують наступні види ревізій коду:

- Офіційна. Тип ревізії коду, який проводиться в офіційному форматі та згідно з встановленими процедурами організації. Офіційна ревізія коду проводиться командою розробників або спеціальною групою, яка має досвід та знання в області кодування та ревізії коду. Цей тип ревізії коду передбачає попередню підготовку до проведення процедури ревізії, включаючи встановлення критеріїв, згоду з учасниками, організацію зустрічі та інше. Під час офіційної ревізії коду, код розглядається з точки зору його якості, ефективності, безпеки та зрозумілості, відповідно до встановлених критеріїв.
- Спрощена. Проводиться з метою швидкої перевірки коду на наявність відомих помилок, недоліків, або можливостей для покращення коду. Цей тип ревізії може бути проведений одним розробником або декількома розробниками, які мають достатній досвід та знання в області кодування. У спрощеній ревізії коду зазвичай використовуються певні методи та інструменти для автоматичної перевірки коду на наявність відомих проблем, такі як невикористаний код, потенційні помилки в безпеці, дублювання коду тощо. Однак, цей тип ревізії не передбачає глибокого аналізу коду та відшукування відсутніх відомих проблем.

- «Через плече». Розробник перевіряє код іншого розробника, дивлячись на його екран (через плече). Цей метод може використовуватись в малих командах або при роботі в парах, коли один розробник кодує, а інший перевіряє його роботу. Під час спрощеної ревізії коду «через плече» розробник що перевіряє, може надавати короткі коментарі, щоб покращити код, наприклад, вказати на можливі проблеми з безпекою, покращити читабельність коду або запропонувати альтернативний підхід до рішення задачі.
- Паралельне програмування. Тип коли два розробника працюють разом над однією задачею, сидячи за одним комп'ютером. Один розробник кодує, а інший перевіряє його роботу, а обидва розробники співпрацюють над вирішенням завдання. У процесі парного програмування, розробники можуть взаємодіяти та обговорювати рішення на льоту, надаючи короткі коментарі та вказівки один одному. Цей метод може допомогти забезпечити вищу якість програмного забезпечення, оскільки розробники можуть взаємно перевіряти свій код та запобігати появі дефектів.
- За допомогою інструментів. Це тип ревізії коду, при якому використовуються спеціальні інструменти для автоматизації процесу перевірки коду. Ці інструменти можуть включати різноманітні програми, такі як різноманітні аналізатори коду, статичні аналізатори, інструменти для автоматичного тестування тощо. Зазвичай цей тип ревізії коду використовується для перевірки великих об'ємів коду, оскільки ручна перевірка може бути недоцільною через обсяг і складність коду. Використання інструментів для автоматизації допомагає знизити ризик помилок таких як невизначені змінні, неправильні типи даних, помилки синтаксису, дублюючого коду, перевірки коду на відповідність стандартам та забезпечити вищу якість коду. Однак, використання

додаткових інструментів може бути складним і вимагати додаткової підготовки з боку розробників [27].

Ревізія коду є важливою складовою процесу розробки програмного забезпечення, оскільки вона дозволяє забезпечити високу якість кодової бази та зменшити кількість помилок у програмному забезпеченні. Крім того, ревізія коду може допомогти забезпечити відповідність коду вимогам, що встановлюються на етапі аналізу вимог, а також сприяє збільшенню ефективності розробки, зменшенню часу розробки та зниженню витрат на тестування та утримання програмного забезпечення.

2.4. Артефакти контролю якості

Артефакти контролю якості — це документи, файли або інші матеріали, які створюються під час розробки програмного забезпечення, що допомагають контролювати та оцінювати якість програмного забезпечення. Артефакти контролю якості можуть бути використані для оцінки якості програмного забезпечення на різних етапах його розробки, тестування та використання.

Вимоги до програмного забезпечення — це опис того, що програмне забезпечення має робити та які функції повинно мати для того, щоб задовольнити потреби користувачів. Обов’язково має бути визначене призначення програмного забезпечення, функціональні вимоги, вимоги до користувацького інтерфейсу, системні вимоги та нефункціональні вимоги [28].

План контролю якості програмного забезпечення - це документ, що визначає стратегію та методи контролю якості програмного забезпечення під час його розробки та тестування. Основною метою плану контролю якості є забезпечення високої якості програмного забезпечення та відповідності вимогам [29].

Характеристики плану контролю якості програмного забезпечення можуть включати [29]:

- Опис процесів контролю якості. Визначення процесів та методів контролю якості, включаючи методи валідації та верифікації.
- Опис тестової стратегії. Визначення тестових методів, тестових критеріїв, тестових скриптів та тестових сценаріїв.
- Опис процесів управління дефектами. Визначення процесів виявлення, документування та відстеження дефектів, а також їх виправлення та перевірки.
- Опис критеріїв прийняття продукту. Визначення критеріїв, за якими програмне забезпечення вважається відповідним вимогам та готовим до випуску.
- Вимоги до середовища. Визначення середовища, в якому буде проводитись контроль якості, зокрема необхідність спеціального обладнання або програмного забезпечення.
- Відповідальність. Визначення відповідальності за виконання певних завдань, а також ролі та обов'язки членів команди розробки та тестування.
- Звіти. Опис формату та частоти звітування про хід робіт та результати контролю якості.

Юзкейси — це техніка аналізу вимог до програмного забезпечення, яка дозволяє описати, як система повинна взаємодіяти з її користувачами або іншими системами в різних сценаріях використання. Кожен юзкейс описує конкретний сценарій взаємодії користувача з системою та описує, які дії має виконати система для того, щоб відповісти на запит користувача та забезпечити очікуваний результат [30].

Під час контролю якості програмного забезпечення юзкейси використовуються для перевірки того, чи відповідає програмне забезпечення вимогам та очікуванням користувачів. Кожен юзкейс може бути перевірений наявністю необхідного функціоналу, правильністю взаємодії системи з користувачем та іншими системами, а також правильністю обробки вхідних даних та генерацією очікуваних результатів [30].

Також, використання юзкейсів під час контролю якості допомагає забезпечити повноту тестування програмного забезпечення, оскільки для кожного юзкейсу можуть бути розроблені відповідні тестові сценарії. Юзкейси також допомагають забезпечити прозорість та співпрацю між різними учасниками проекту, оскільки вони надають однозначне та загальноприйняте розуміння функціоналу та поведінки системи [30].

Тесткейси — це детально описані кроки, які тестувальник повинен виконати, щоб перевірити певний функціонал програмного забезпечення та переконатися в його коректності та працездатності. Тесткейси зазвичай описуються у вигляді таблиць або списків кроків, які повинні бути виконані, щоб провести певний тест. Під час контролю якості програмного забезпечення тесткейси використовуються для автоматизованого або ручного тестування програмного забезпечення та перевірки його відповідності вимогам. Кожен тесткейс повинен бути детально протестований тестувальником, який повинен перевірити, чи функціонал працює правильно та відповідає очікуванням. Результати тестів записуються у спеціальні звіти, які використовуються для виявлення та виправлення помилок. Використання тесткейсів дозволяє забезпечити високу якість програмного забезпечення та знизити ризики його некоректної роботи у реальних умовах використання [31].

Дефект-звіт — це документ, що містить інформацію про виявлену помилку або дефект у програмному забезпеченні. До складу дефект-звіту можуть входити такі дані, як:

- Опис проблеми. Що саме сталося та як саме виявився дефект.
- Шляхи відтворення. Які кроки потрібно зробити, щоб повторити помилку.
- Результат. Який результат був отриманий, що саме не сподобалось або що було некоректно.
- Очікуваний результат. Який результат очікувався, якщо б програмне забезпечення працювало правильно.

Дефект-звіти є важливим інструментом під час контролю якості програмного забезпечення, оскільки вони дозволяють виявляти та документувати помилки в програмному забезпеченні. Крім того, дефект-звіти допомагають тестувальникам та розробникам спілкуватися між собою та вирішувати виниклі проблеми. Вони також використовуються для відстеження та контролю якості розробки програмного забезпечення, а також для забезпечення коректної та ефективної роботи тестувальників та розробників [32].

Чек-ліст (checklist) — це список завдань або критеріїв, які потрібно перевірити або виконати під час контролю якості програмного забезпечення. Це може бути список функціональних вимог, рекомендацій по дизайну, правил виконання тестів та інших завдань, які повинні бути виконані перед випуском програмного забезпечення.

Чек-листи використовуються для стандартизації процесів контролю якості програмного забезпечення та для забезпечення повноти та точності проведення тестування. Вони дозволяють забезпечити, що всі необхідні функції та вимоги були перевірені, а також допомагають уникнути пропусків в процесі тестування.

Чек-листи можуть бути створені для кожного етапу контролю якості програмного забезпечення, наприклад:

- Вимоги. Перевірка, чи всі вимоги були включені до документації та розуміються правильно.
- Проектування. Перевірка, чи всі аспекти дизайну були виконані, включаючи унікальні функції, користувацький інтерфейс та інші.
- Тестування. Перевірка, чи виконуються всі вимоги та функції, чи немає помилок та чи не порушуються правила тестування.
- Реліз. Перевірка, чи все необхідне включено до випуску програмного забезпечення та чи він готовий для випуску.

Використання чек-лістів допомагає забезпечити якість та ефективність процесу контролю якості програмного забезпечення, а також допомагає збільшити точність та повноту тестування [33].

ВИСНОВКИ ДО РОЗДІЛУ 2

В даному розділі було розглянуто життєвий цикл програмного забезпечення та його різні підходи з різними прикладами концепцій контролю якості. Була детально розглянута одна з найважливіших складових під час розробки програмного забезпечення, така як дефект, що своєю чергою тестувальники ставлять на меті віднайти під час здійснення контролю якості.

Зокрема були розглянуті різні засоби та способи за допомогою яких можна здійснити контроль якості, а саме такі як: тестування та ревізії коду з детальними описами та оглядами їх видів. Неодмінною частиною контролю якості є артефакти, що допомагають структурувати усю інформацію під час проведення контролю якості, задокументувати її та надати звіти відповідним особам задля виправлення дефектів, аналізу тощо.

Для забезпечення високої якості ПЗ необхідно враховувати різноманітні фактори, що можуть впливати на якість продукту, та використовувати різні методи та засоби контролю якості. Відповідність вимогам, тестування, ревізії коду, використання артефактів та інші методи можуть допомогти забезпечити якість програмного забезпечення і підвищити його корисність та ефективність.

РОЗДІЛ 3
РОЗРОБКА АРТЕФАКТІВ КОНТРОЛЮ ЯКОСТІ

В попередньому розділі було розглянуто різні артефакти контролю якості. Одним з таких артефактів є документ плану контролю якості який налічує в собі значну кількість артефактів контролю якості. План допоможе зібрати всю необхідну інформацію, визначити чіткі цілі та слідувати цьому задокументованому плану, та по виконанню чи необхідності внесення корективів доповнювати його.

План контролю якості може складатись з наступних розділів (рис. 3.1):

- Аналіз продукту
- Стратегія
- Цілі
- Критерії
- Ресурси
- Середовище
- Розклад та оцінка
- Результати

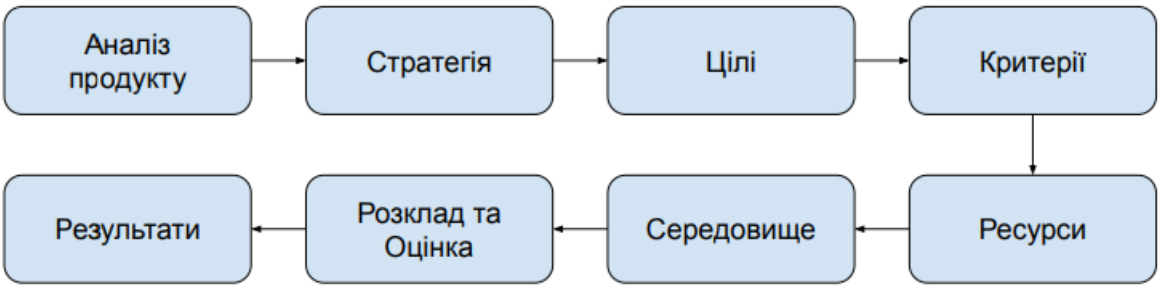

Рис. 3.1. Складові плану контролю якості

3.1. Аналіз продукту

Насамперед важливо проаналізувати задачі продукту, хто буде користуватись продуктом, яке апаратне/програмне забезпечення використовується в продукті.

Аби дізнатись всю необхідну інформацію можна застосувати наступні підходи:

- Переглянути вимоги до продукту та документацію
- Бесіда з клієнтом, дизайнером або розробниками
- Безпосередній огляд продукту

Оскільки задокументованих вимог до продукту немає, було прийнято рішення поспілкуватись з клієнтом, дизайнером та розробниками, аби мати насамперед загальне бачення — яке призначення цієї системи, хто нею має користуватись, чи є вимоги якісь безпосередньо до апаратного/програмного забезпечення, які функції мають бути у продукту, тонкощі роботи (якщо є), особливості дизайну та щось на що було б варто звернути увагу в першу чергу при здійсненні контролю якості.

Після безпосередньої розмови з більшою частиною учасників циклу розробки програмного забезпечення можна задокументувати вимоги що були визначені при обговоренні та приступити безпосередньо до самостійного огляду системи.

Насамперед для користування системою буде досить стабільного інтернет з’єднання, та браузеру, не потрібно додатково встановлювати якесь програмне забезпечення, що зазвичай може займати велику кількість пам’яті ноутбука або персонального комп’ютера, що є досить зручним підходом.

Першопочатково, як тільки переходити на основну сторінку можна побачити стилізовану сторінку представлення системи (рис. 3.2).

Так би мовити ознайомча сторінка з логотипом, назвою, яка дозволяє зрозуміти куди потрапив користувач перейшовши за посиланням. На сторінці присутня кнопкою «Почати», що надасть змогу увійти в систему та почати нею користуватись. Після натискання на кнопку початку користувач має змогу заповнити дані для входу та ввійти в систему.

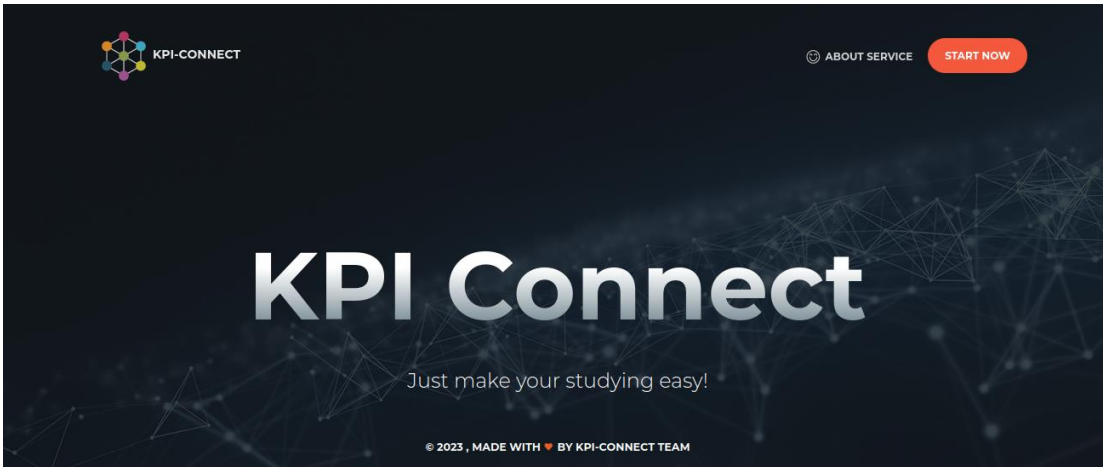

Рис. 3.2. Сторінка представлення системи

Після натискання на кнопку «Почати», постає один з найважливіших компонентів будь-яких систем, такий як процес аутентифікації та авторизації користувача в системі (рис. 3.3). Необхідно ввести пароль та логін, і лише після цього можна буде повноцінно користуватись опціями/функціями системи.

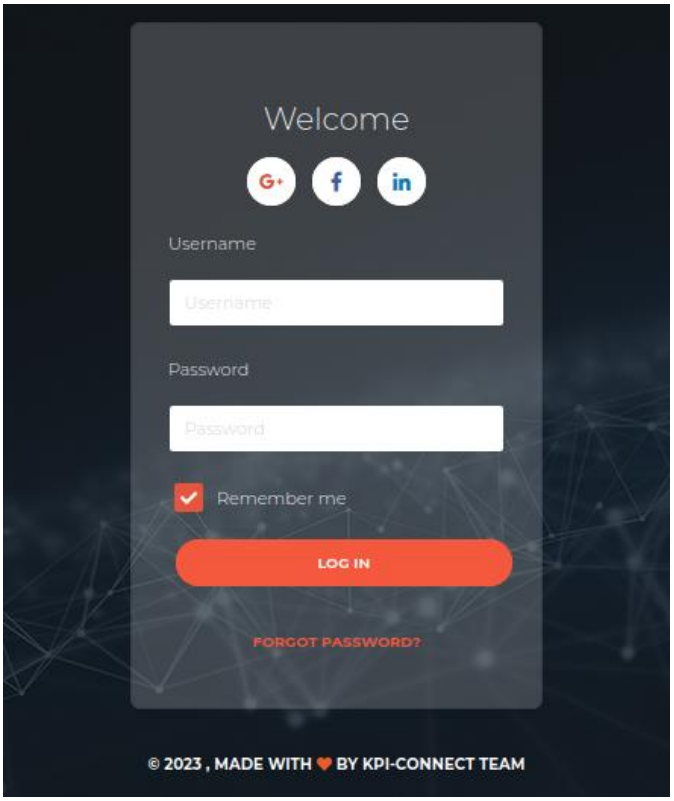

Рис. 3.3. Сторінка авторизації

Звісно сторінка так само налічує всі ці пункти в боковому меню (розгортається та виводить дочірні опції по натисканню на якийсь з пунктів). Окрім бокового меню, існує горизонтальне меню згори сторінки, що налічує

наступні пункти: «Згорнути бокове меню» (дозволяє сховати бокове меню), «Домашня сторінка» (сторінку представлення системи з кнопками «Повернутись до робочого простору»), «Додати ресурс» (дозволяє обрати ресурс системи який необхідно додати), «Мій профіль» (дозволяє вийти з системи), «Змінити мову» (дозволяє змінити мову системи).

Після входу в систему перед користувачем постає початкова сторінка системи після входу (рис. 3.4), що являє собою загальну панель з усіма пунктами меню, та опціями, що доступні для користувача в подальшому.

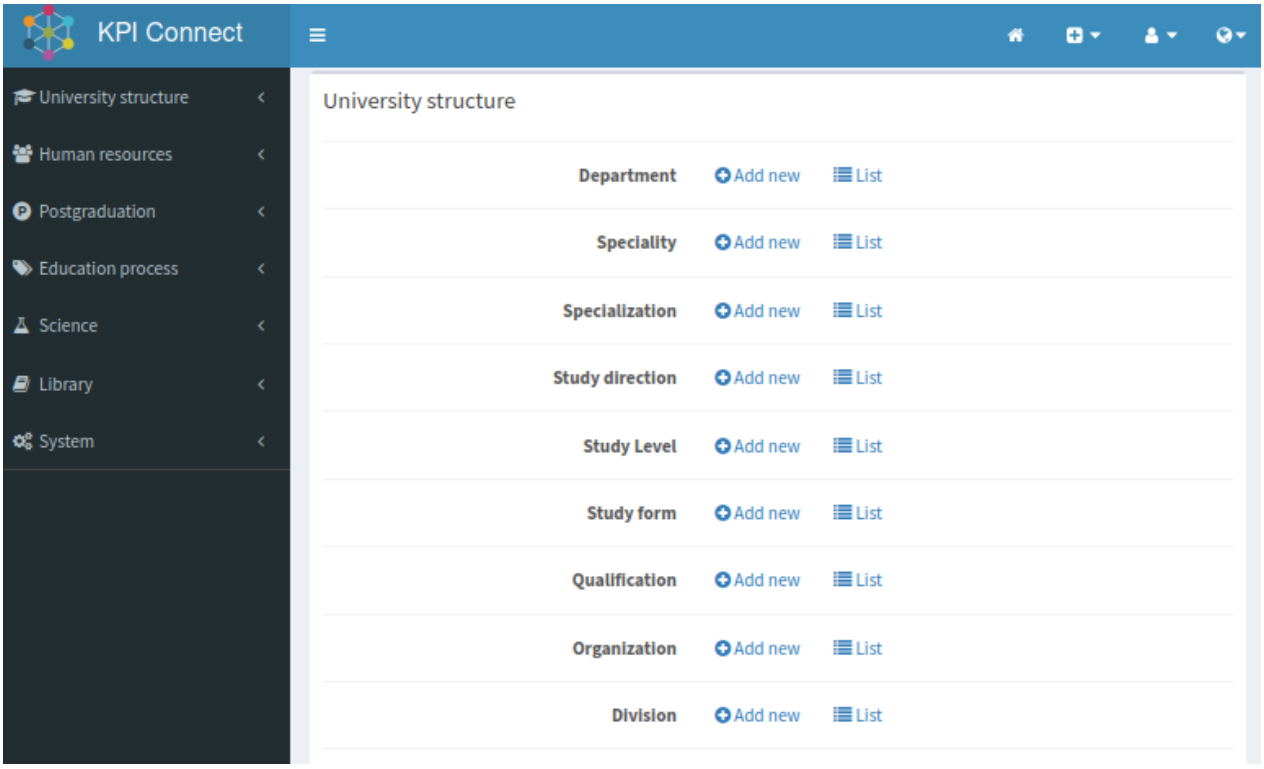

Рис. 3.4. Початкова панель системи

Наступним кроком є вже безпосередньо загальний огляд функцій що є в системі, проходження по пунктах меню, намагання взаємодії з системою, спроба додавання нових ресурсів, спроба скористатись додатковими опціями як зміна мови тощо. Зрештою після цього огляду можна зрозуміти як працює система, з якими даними, в чому полягає її основна задача, які з вимог поставлених до неї були виконані, які ні (на рівні поверхневого огляду: чи всі є функції, чи відповідає дизайн вимогам, якщо до дизайну були поставлені вимоги, чи зручно нею користуватись тощо).

З огляду на меню, та вже коли системою спробували скористуватись, можна побачити закономірність, що всі опції фактично мають однакові цілі (взаємодія з ресурсом), працюють за однаковою схемою (можна додати ресурс, змінити ресурс, видалити ресурс чи переглянути), основною відмінністю звісно є те, що всі ресурси різні, і аби додати/оновити/видалити якийсь з них, будуть необхідні різні умови та поля до заповнення (рис. 3.5, 3.6, 3.7, 3.8, 3.9).

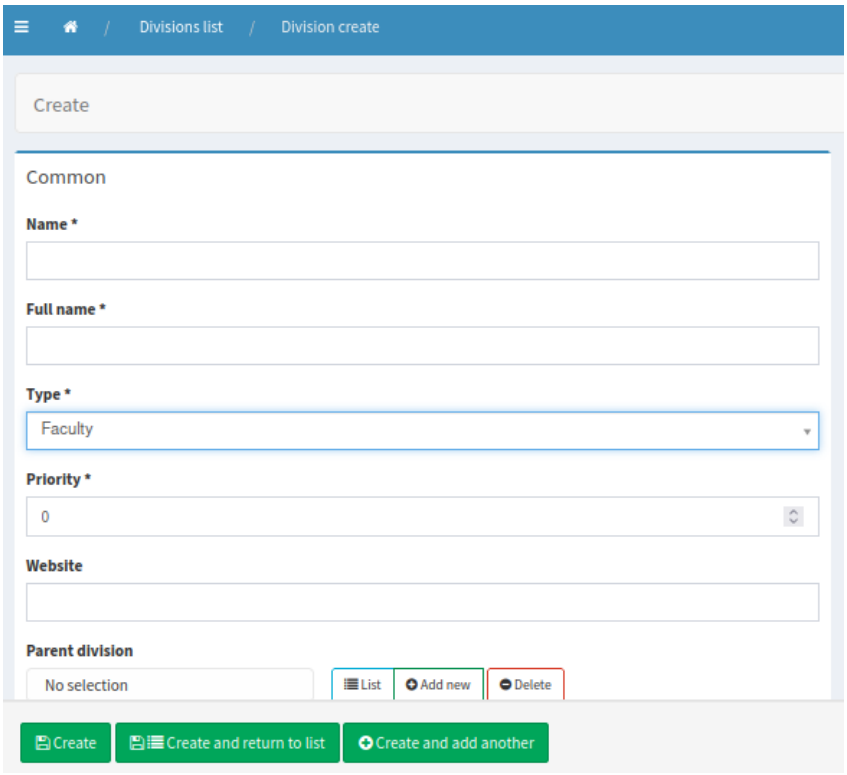

Рис. 3.5. Додати новий підрозділ

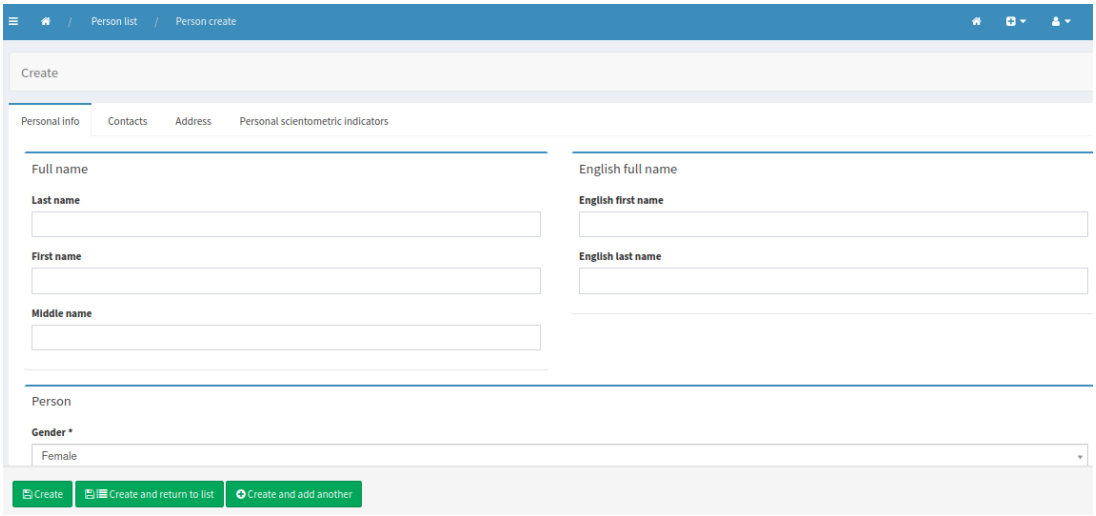

Рис. 3.6. Додати нову персону

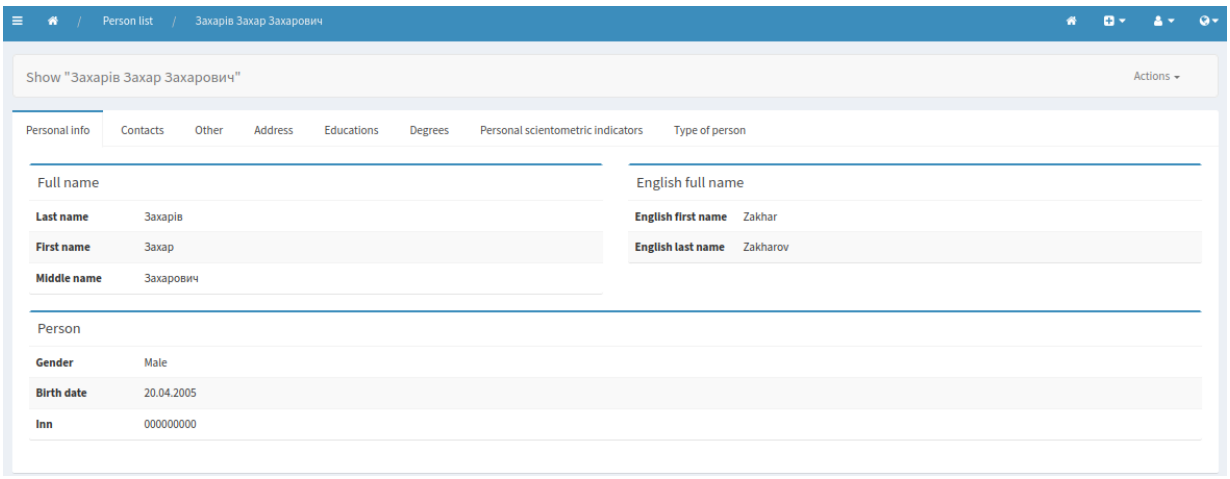

Рис. 3.7. Переглянути інформацію по існуючій персоні

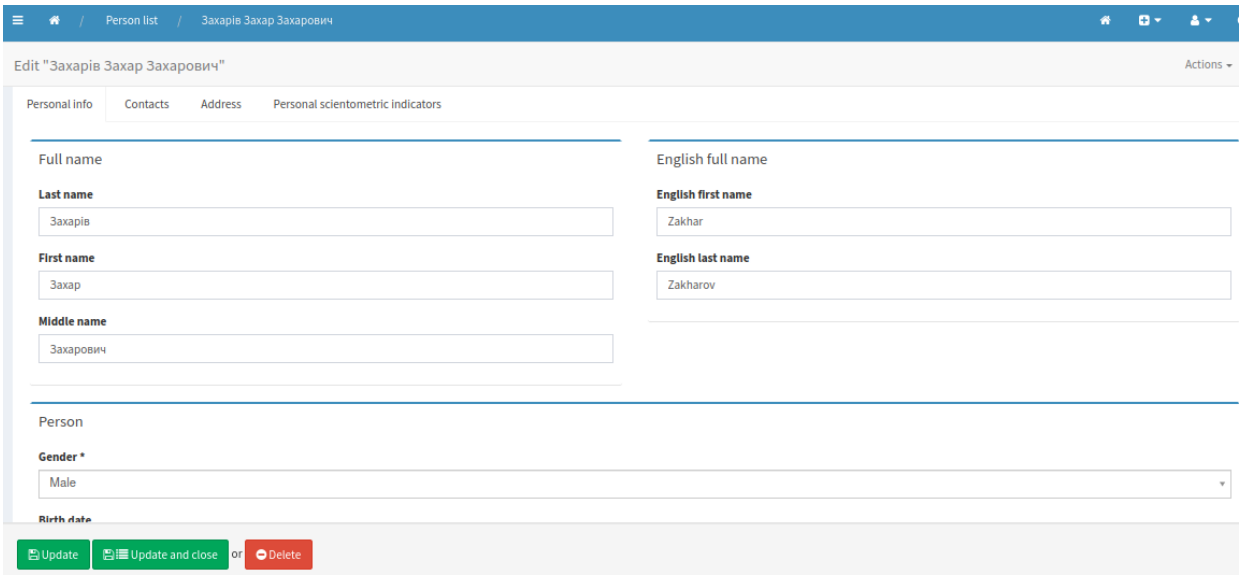

Рис. 3.8. Оновити інформацію існуючої персоні чи видалити

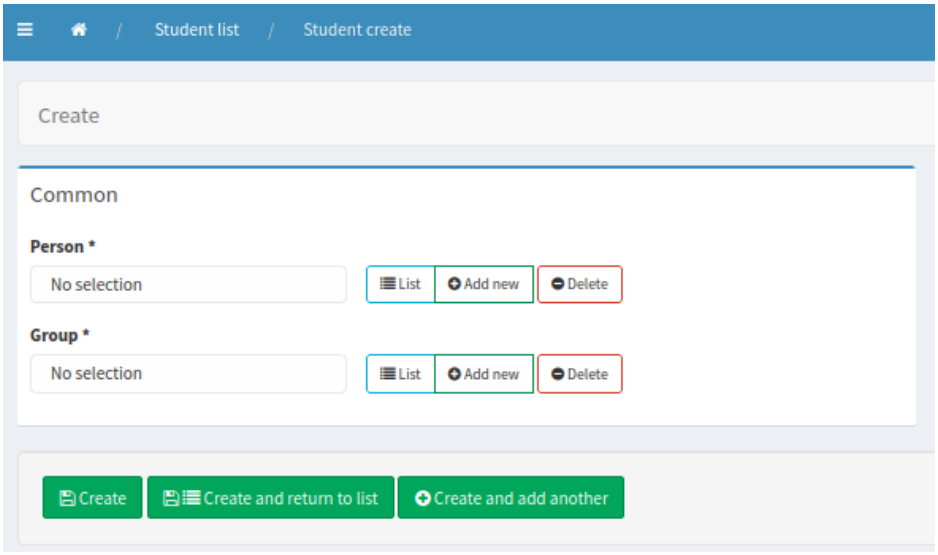

Рис. 3.9. Додати студента

Після здійснення усіх вищеперерахованих пунктів аби здійснити аналіз продукту, в такому випадку це бесіда з клієнтом, дизайнером, розробниками, а також огляд та попереднє поверхневе використання продукту для наочного розуміння, дало зрозуміти що собою являє загалом система та сформувати наступні основні вимоги до інформаційної системи управління вищим навчальним закладом:

- Задачі. Зручне керування різними ресурсами навчального закладу вищої освіти, швидкий доступ до перегляду, оновлення, редагування або ж видалення тих чи інших ресурсів (структури університету: кафедра, спеціальність, спеціалізація, напрямок навчання, рівень вищої освіти, форма навчання, кваліфікація, організація, підрозділ; кадрових ресурсів: персона, студент, аспірант, посада, вчене звання, наукова ступінь, почесне звання, напрямок роботи, освіта, персональні напрямки роботи за підрозділом, персональні посади за підрозділом, персональні почесні звання; процесу навчання: предмет, група, оцінка, навчальний план; науки: наукове видання, наукометричні показники, персональні наукометричні показники, науковий проєкт; бібліотеки: дисертація, документ; системи: користувач, подія, аудиторія, показник нормоконтролю, показник нормоконтролю за диплом, вкладення)
- Користувачі. Мати змогу користуватись системою мають адміністратор (той хто додаватиме всі необхідні нові дані, аби можна було користуватись ними в системі), та інші кадрові ресурси університету (студент, викладач тощо)
- Апаратне/програмне забезпечення. В обов’язковому порядку аби користуватись системою має бути стабільне підключення до мережі інтернет (кабельне з’єднання, мобільна мережа, wi-fi, супутниковий онлайн зв’язок), браузер що дозволить здійснити перегляд веб-сторінок (система являється веб-продуктом)

3.2. Стратегія

Наступний важливий етап — це розробка стратегії контролю якості. Цей документ допомагає визначитись з наступним (рис. 3.10):

- Покриття
- Види тестування до застосування
- Ризики
- Логістика

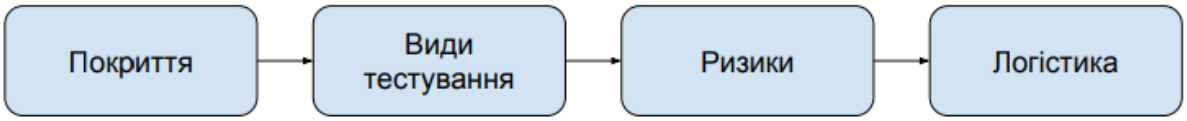

Рис. 3.10. Шлях формування стратегії

3.2.1. Покриття

Згідно вимог що були визначені на стадії аналізу продукту, а саме особлива увага була приділена функціям системи, тож обов'язково необхідно покрити всі функції системи та запевнитись в їх справності, валідація даних в формах, зовнішній інтерфейс веб-додатку. А також приділити окрему увагу до зручності, сумісності та навантаження на систему.

3.2.2. Види тестування до застосування

Під час стадії покриття було визначено на чому буде сконцентрований контроль якості інформаційної системи управління вищим навчальним закладом. Згідно цього можна визначитись з видами тестування які буде необхідно провести задля покриття всього обсягу критеріїв, а саме це буде як функціональні, так і нефункціональні види тестування:

- Функціональні
  - Модульне
  - Інтеграційне
  - Системне

- Нефункціональне
  - Зручності
  - Сумісності
  - Навантажувальне

3.2.3. Ризики

Ризики та їх пом’якшення можуть бути описані наступними (табл. 3.1).

Таблиця. 3.1

Ризики системи та їх можливі пом’якшення

ID	Ризик	Пом’якшення
R0	Не будуть покриті великі об’єми даних	Ретельна перевірка тестових даних, використання інструментів для генерації тестових даних.
R1	Не будуть враховані всі поведінкові характеристики	Провести тестування на різних версіях операційних систем та різних браузерах.
R2	Система перестане працювати раптово	Використання високодоступних серверів та мережевих систем, що забезпечують резервне копіювання даних та забезпечують підтримку роботи від різних географічних регіонів. По-друге, можна використовувати механізми моніторингу, що дозволять відстежувати працездатність системи та вчасно виявляти помилки.

Продовження таблиці 3.1.

ID	Ризик	Пом’якшення
R3	Проблеми з доменом	Завчасна перевірка терміну дії домену, його статус, чи не було домен заблоковано чи знято з реєстрації. Використання альтернативного доменного імені. Миттєве звернення до служби підтримки.
R4	Проблеми з хостингом	Вчасна сплата рахунків за хостингові послуги. Переконання в правильності налаштування та відповідності вимогам веб-продукту. Використання служби моніторингу. Вчасне звернення до служби підтримки. Робити резервне копіювання.
R5	Некоректна обробка даних системою	Виконання валідації даних на всіх етапах обробки. Застосування різних методів перевірки даних. Регулярно перевіряти дані на наявність помилок та внесення необхідних коректівів.
R6	Невідповідність вимогами	Уважно аналізувати вимоги системи. Задокументовувати вимоги та затверджувати їх.
R7	Неправильне зберігання даних	Застосування механізму аутентифікації та авторизації. Перевірка цілісності даних. Використання механізмів шифрування.

3.2.4. Логістика

При логістиці контролю якості продукту мають бути відповіді на наступні запитання:

- Хто буде тестувати?
- Коли почнеться процес контролю якості?

На перший погляд, відповісти на ці питання здається дуже легко, але зазначити що буде тестувати — тестувальник, та коли — по завершенню розробки не покриває ці питання взагалі.

Тож перш ніж відповісти на питання «Хто буде тестувати?», потрібно розуміти контроль якості чого потрібно проводити і відштовхуватись від цього, та підбирати тестувальника згідно цих вимог. Оскільки контроль якості чого саме необхідно проводити — відомо, та є певні вимоги до продукту, можна визначитись з навичками людини, що займатиметься контролем якості системи:

- Добре розуміння баз даних. Тестувальник повинен мати розуміння структури баз даних та їх взаємодії з іншими складовими системи.
- Увага до деталей. Тестувальник повинен бути дуже уважним до деталей та мати відповідальне ставлення до своєї роботи.
- Висока концентрація та точність. Тестувальник повинен бути в змозі зосереджуватися на своїй роботі протягом тривалого часу, не допускаючи помилок.
- Аналітичні навички. Тестувальник повинен бути здатним аналізувати дані та висновки, які він отримує в ході тестування системи.
- Гнучкість та відкритість до змін. Тестувальник повинен бути гнучким та відкритим до змін, що можуть виникнути в ході проєкту.
- Комунікативність. Тестувальник повинен бути здатним ефективно комунікувати з іншими членами команди розробників, менеджерами та користувачами, щоб допомогти зрозуміти та вирішити проблеми.

- Знання контролю якості. Тестувальник повинен мати досвід у процесі контролю якості веб-продукту та розуміти методологій контролю якості та процесів, що стосуються розробки програмного забезпечення.

Відповісти на наступне запитання «Коли почнеться процес контролю якості?», можна запевнившись що всі наступні пункти були виконані:

- Є вимоги до системи
- Обрана людина що буде займатись контролем якості
- Налаштоване середовище для здійснення контролю якості

Після визначення усіх необхідних аспектів логістики контролю якості, можна розуміти, що система готова перейти у стадію контролю якості.

3.3. Цілі

Під час визначення цілей контролю якості необхідно ще раз уважно перечитати вимоги, переглянути види тестування що необхідно застосувати під час контролю якості інформаційної системи управління вищим навчальним закладом та згідно цього все скомпонувати та отримати набір з чіткими цілями до контролю якості.

Всі цілі були сформовані у вигляді картки думок в ієрархічному порядку, з розмежуванням, та розфарбуванням в різні кольори.

Оскільки рішення досить велике та має багато цілей для контролю якості, до представлення надається 2 види картки думок:

- Частково згорнута (рис. 3.11). Зображує що саме буде тестуватись без конкретизації, з додатковими відмітками на гілках, скільки пунктів до контролю якості сховано.
- Повністю розгорнута (рис. 3.12). Зображує повну картину цілей до контролю якості.

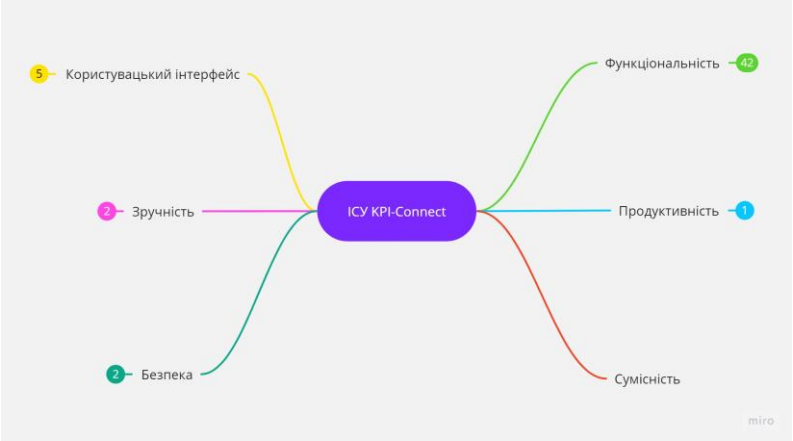

Рис. 3.11. Частково згорнута картка думок

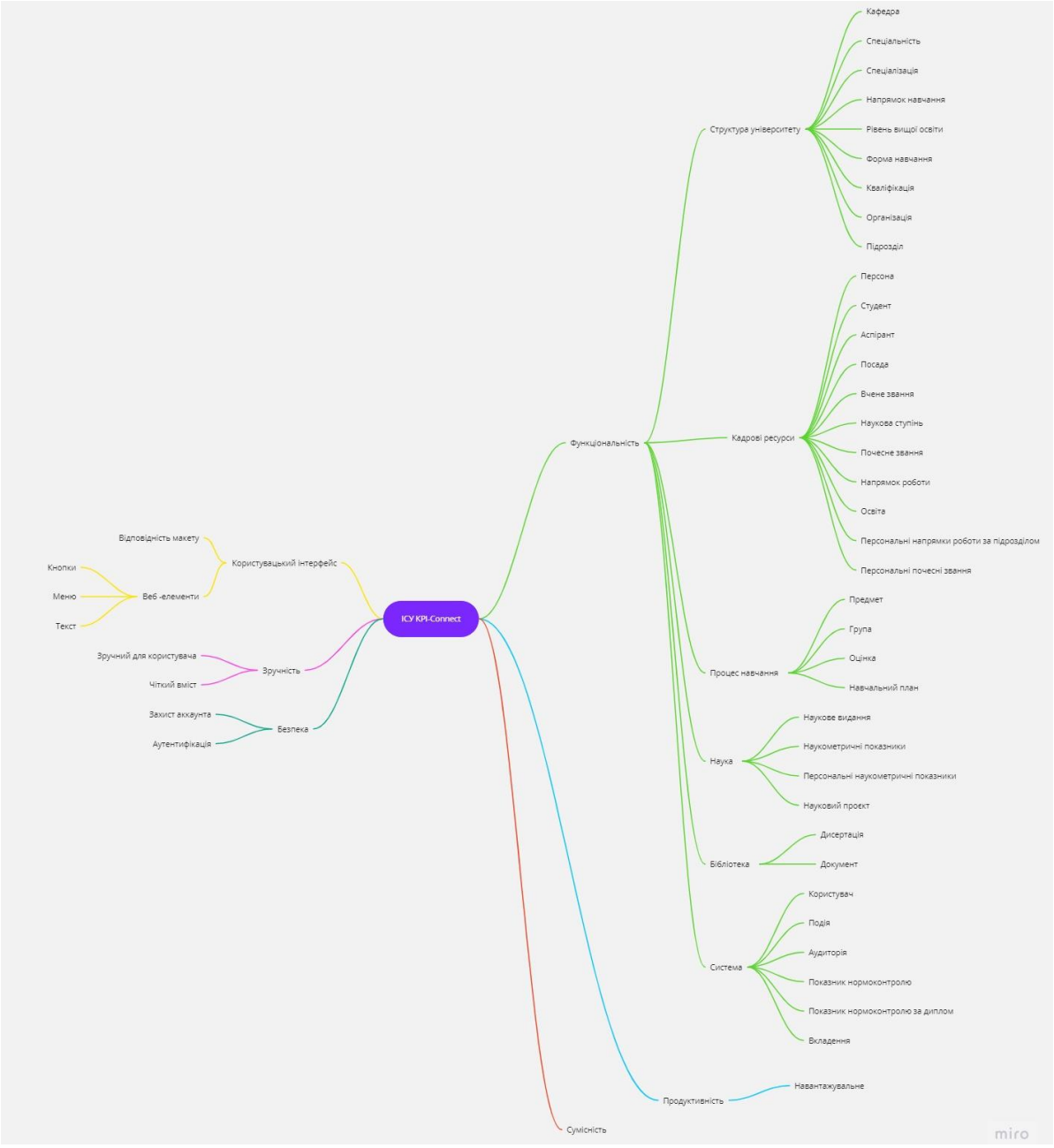

Рис. 3.12. Повністю розгорнута картка думок

3.4. Критерії

Критерій застосовується аби оцінити чи потрібно призупинити та відкласти процес контролю якості, чи можна його завершити (рис. 3.13).

- Критерій призупинення. Оскільки модулі системи працюють за схожим принципом, і зустрічається десь помилка, велика ймовірність того, що у решті модулів що працюють за таким же принципом буде ця помилка, потрібно слідкувати після контролю якості кожного модуля, якщо один модуль налічує велику кількість помилок, необхідно призупинити тестування конкретно цього модуля та схожих на нього, але можна продовжити контроль якості решти модулів чи або ж перейти до інших цілей, що не будуть залежні від модулів для яких було призупинено контроль якості.
- Критерій завершення. Коли всі тесткейси, що були заплановані будуть перевірені. Якщо завершився період що був відведений на контроль якості.

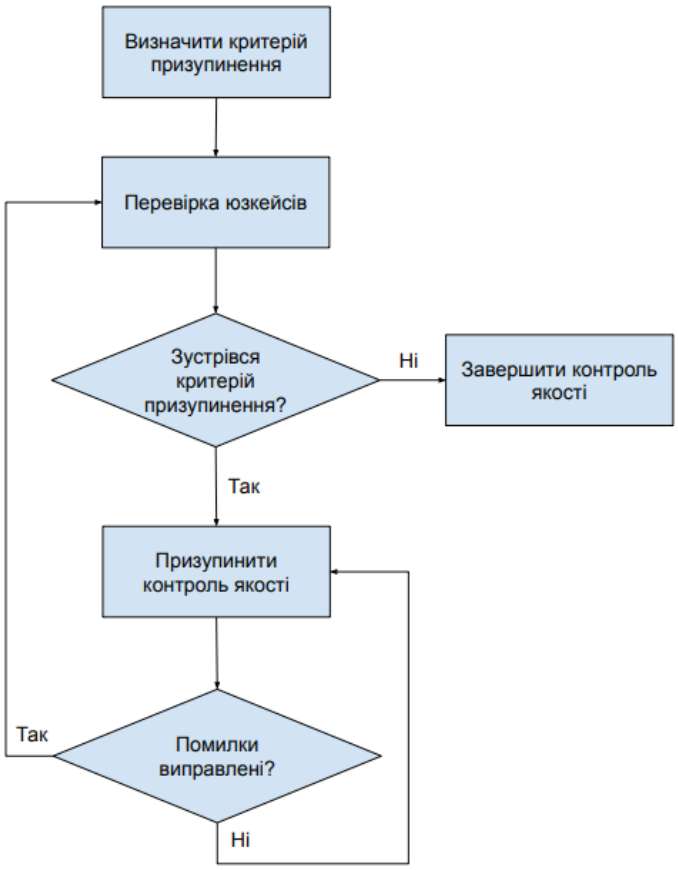

Рис. 3.13. Алгоритм застосування критеріїв

3.5. Ресурси

Планування ресурсів дозволяє визначити потреби в ресурсах, оцінити їх доступність, аби можна було провести контроль якості належним чином.

Розподіл ресурсів зображений у формі таблиці.

- Людських ресурсів (табл. 3.2). Визначає учасників процесу та їх завдання під час контролю якості.
- Системні ресурси (табл. 3.3). Визначає необхідне обладнання та засоби для здійснення контролю якості веб-додатку.

Таблиця. 3.2

Людські ресурси

Учасник	**Завдання**
Тестувальник	Аналіз відповідності вимог до продукту з розробленим продуктом. Опис як буде здійснюватись контроль якості. Виконання тесткейсів. Документування та подання дефектів. Комунікація з різними учасниками у вирішенні питань. Слідкуванні за часом, що відведений на контроль якості.
Розробник	Перевірка продукту у справності та відповідності вимогам зі своєї сторони, перед передачею тестувальникам. Виправлення дефектів. Комунікація з різними учасниками у вирішенні питань. У разі необхідності налаштування продукту на девайсах тестувальників.

Таблиця. 3.3

Системні ресурси

Ресурс	**Опис**
Комп’ютер, ноутбук	Девайс з яких користувачі переважно користуватимуться веб-застосунком.
Мережа інтернет	Оскільки ПЗ — є веб-додатком, необхідне підключення пристрою до мережі інтернет, щоб ним користуватись.
Браузер	Додаток що дозволяє відкрити веб-додаток
Додаткові інструменти/засоби для розгортання системи	Засоби що необхідні для локального запуску додатка, під’єднання до бази даних (*PhpStorm*, *php*, *docker*, *composer*, *postgreSql* тощо).
Засоби для контролю якості системи	Програмне забезпечення/засоби що необхідні для виконання контролю якості (*Apache JMeter*, *IntelliJ IDEA*, *Selenium*, *excel* тощо)

3.6. Середовище

Для здійснення контролю якості інформаційної системи керування навчальним закладом вищої освіти, необхідно попередньо розгорнути систему локально.

Щоб розгорнути систему локально, необхідно виконати наступні дії:

- Завантажити *Git*
- Завантажити *Docker*
- Завантажити *Php 7.4.x*
- Завантажити *Composer 2.2.x*
- Склонувати проєкт з *Git*
- Оновити файл /*etc/hosts* наступними даними (перевизначити хости):

- *127.0.0.1 admin.kpi-connect.test*
- *127.0.0.1 api.kpi-connect.test*
- *127.0.0.1 kpi_connect_postgresql*
- *127.0.0.1 kpi_connect_minio*

- Запустити *Docker*
- Відкрити іншу консоль та продовжити роботу там, завантажити всі пакети за допомогою *composer install* (всі параметри можна залишати за замовчуванням, окрім хосту бази даних — 127.0.0.1)
- Виконати файл *./build.sh*, аби завантажити всі бандли та виконати міграції
- Створити користувача адміністратора

Після того як система готова до локального запуску, проєкт можна запускати. Запускається проєкт в 2 етапи (в різних консолях), спочатку необхідно запустити *Docker — docker-compose up*, надалі необхідно виконати — *php bin/console server:run '*:8000'*. Після цього можна вже випробовувати систему. По завершенню роботи з системою необхідно припинити процеси в кожній з консолей командою *ctrl+C*.

Аби зручно було переглядати код та базу даних, варто завантажити середовище розробки *phpStrom*, яке дозволить як переглядати код, так і підключитись до бази даних.

Середовище веб-додатка має клієнт-серверну архітектуру (рис. 3.14).

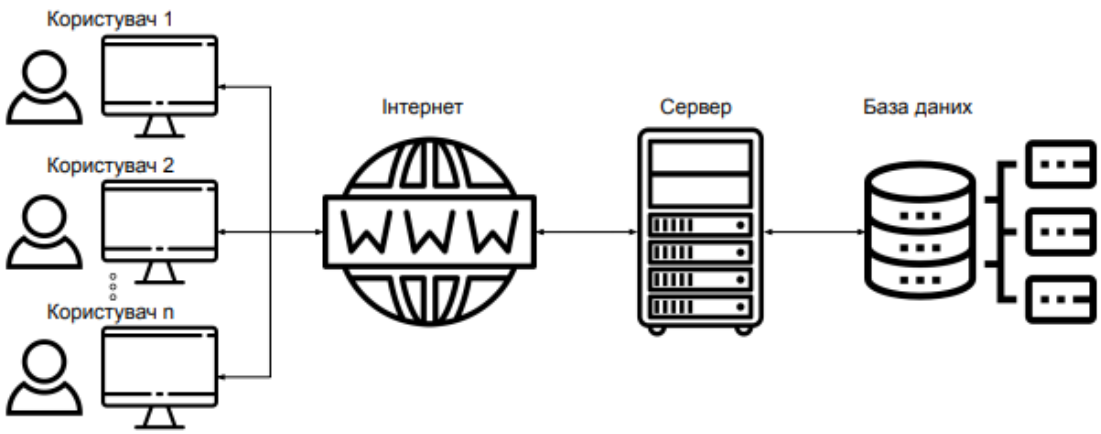

Рис. 3.14. Модель клієнт-серверної архітектури

3.7. Розклад та оцінка

На цьому етапі відбувається розподіл часу на всі задачі що необхідно виконати під час контролю якості. Перше та найскладніше це створення, розробка юзкейсів та тесткейсів, тобто розписати всі випадки які потрібно буде перевірити при здійсненні контролю якості системи. Наступний етап — етап виконання та перевірки розроблених тесткейсів, юзкейсів тощо, що також буде займати достатній відсоток часу аби обробити всі випадки. Після перевірки всіх випадків наступає етап складання звіту, де описуються результати по проходженню тесткейсів та всі дефекти що були знайдені впродовж випробовування. І завершальний етап це формування та надання клієнту всіх результатів з проведення контролю якості (звіти, документи тощо).

Цей документ можна сформувати у вигляді таблиці з завданнями та виділеними на ці завдання часом (табл. 3.4).

Таблиця 3.4

Розклад та оцінка завдань контролю якості

Завдання	**Орієнтовна часова оцінка**
Специфікація контролю якості	150 годин
Виконання тестів	120 годин
Результати виконання тестів	20 годин
Звіт з контролю якості	35 годин

3.8. Результати

Ця складова плану налічує перелік артфеактів контролю якості, які необхідно мати по завершенню контролю якості системи. В даному випадку цей чекліст складатиметься з наступного:

- Юзкейси
- Структурні макети сторінок
- Документація до бази даних

- Тесткейси та тестові дані
- Звіт виконання тест кейсів
- Звіт по дефектах
- Загальний звіт з контролю якості

Важливо зазначити те що в цей чекліст входять деякі пункти які безпосередньо не відносяться до контролю якості, такі як структурні макети сторінок та документація до бази даних, але оскільки цих документів не було, було прийнято рішення також їх розробити аби також перевірити структурну складову дизайну чи відповідатиме вона вимогам та документацію до бази даних, аби перевірити чи вірно зберігаються в ній дані та було розуміння до того з якими даними оперує система.

3.8.1. Юзкейси

Юзкейсти — це документування сценаріїв користувача, які можливі в системи. За допомогою юзкейсів можна визначити функціональну відповідність системи вимогам, зручність, правильність порядку дій тощо.

Юзкейси оформлюються у вигляді таблиць на кожну дію в системи, та складаються з наступних пунктів:

- Ідентифікатор
- Дія
- Опис
- Передумова
- Порядок дій
- Постумова
- Коментарі

Юзкейс входу користувача в систему (рис. 3.15) — описує сценарій за яким користувач може здійснити вхід в систему згідно заданих вимог.

ID	**UC-0LI**	
Дія	**Вхід в систему**	
Опис	Здійснення входу в систему для подальшої можливості користування її функціоналом	
Передумова	Бути зареєстрованим в системі	
Порядок дій	• Перейти на сторінку входу • Натиснути на кнопку "Start Now"	
	• Заповнии форму (Username, Password) • Натиснути кнопку "LOG IN"	• Здійснити вхід за допомогою (google, facebook, linkedIn) • Обрати обліковий запис в якому зараз безпосередньо або обрати інший запис та заповнити дані для входу • Натиснути кнопку входу
Постумова	Користувач успішно здійснив вхід в систему та має змогу користуватись функціоналом системи	
Коментарі	• Користувач може невірно ввести дані для входу після чого необхідно буде ввести їх правильно повторно Користувач має моливісь відновити свій пароль якщо він його забув натиснувши на кнопку "FORGOT PASSWORD" • Користувач має змогу запам'ятати себе в системі відмітивши пункт "Remember me", аби в подальшому при вході не заповнювати повторно поля Username та password аби здійснити вхід	

Рис. 3.15. Юзкейс входу в систему

Юзкейс створення персони (рис. 3.16) — описує сценарій, за яким користувач може створити персону для подальшого оперування з цим ресурсом. Описує специфіку та послідовність кроків, опис полів що необхідно ввести аби створити персону належним чином тощо.

ID	**UC-0HRP**	
Дія	**Створення Персони**	
Опис	Опція що слугує створенню профілю Персони та збереження всієї необхідної інформація про людину та подальшої влаємодії з ним в системі.	
Передумова	Користувач виконав вхід в систему "KPI-Connect".	
Порядок дій	• Оберіть опцію "Кадрові ресурси" з початкового меню. • Знайдіть в списку пункт "Персона". • Оберіть опцію "Додати новий".	• З бокового меню розгорніть опцію "Кадрові ресурси". • Натисніть "Персона" аби переглянути список Персон. • В правому верхньому кутку оберіть "Додати новий".
	• Заповніть особисту інформацію: ПІБ, ПІБ англійською, Стать, Дата народження, Ідентифікаційник код. • Заповніть контаки: Мобільний телефон, Робочій телефон, Адреса поштової скриньки. • Можливість визначити особу як "Сторонню" та заповнити місце роботи. • Заповніть адресу: Країна, Область, Місто, Район, Вулиця, Поштовий індекс. • Заповніть персональні наукометричні показники: Scopus ідентифікатор, Індекс Хірша (в системі Scopus), Кількість цитувань (Scopus), Web of Science ідентифікатор, Індекс Хірша (в системі Web of Science), Кількість цитувань (Web of Science), ORCID, Google Scholar ідентифікатор, Індекс Хірша за останні 5 років (в системі Google Scholar), Індекс Хірша (в системі Google Scholar), Кількість цитувань (Google Scholar), Кількість цитувань за останні 5 років (Google Scholar). • Стровіть персону натиснувши "Створити та редагувати", "Створити й повернутись до списку", "Створити та додати новий".	
Постумова	"Персона" була створена і користувач може оперувати нею в системі.	
Коментарі	Більшість параметрів для створення особи необов'язкові, але після створення їх можна змінити.	

Рис. 3.16. Юзкейс створення персони

Юзкейс редагування персони (рис. 3.17) — описує сценарій, за яким користувач може редагувати різні дані персони що були заповнені або не заповнені на стадії створення персони.

ID	**UC-1HRP**	
Дія	**Редагування Персони**	
Опис	Опція що слугує для редагування профілю Персони в системі.	
Передумова	Користувач виконав вхід в систему "KPI-Connect". В системі створений профіль Персони що необхідно відредагувати.	
Порядок дій	• Оберіть опцію "Кадрові ресурси" з початкового меню. • Знайдіть в списку пункт "Персона". • Оберіть опцію "Список".	• З бокового меню розгорніть опцію "Кадрові ресурси". • Натисніть "Персона" аби переглянути список Персон.
	• Знайдіть в списку Персону для редагування. • Натисніть опцію "Редагувати" в колонці списку "Дія". • Відредагуйте необхідні поля (для редагвання доступні всі поля що задаються при створення Персони). • Підтвердіть ваші дії натиснувши "Зберегти" або"Зберегти й повернутись до списку".	
Постумова	Інформація у профілі Персони оновлена згідно попередньо внесених змін.	
Коментарі	Для редагвання доступні всі поля що задаються при створення Персони.	

Рис. 3.17. Юзкейс редагування персони

Юзкейс видалення персони (рис 3.18) — описує сценарій, за яким користувач має можливість видалити персону, що унеможливить подальшу роботу з видаленим ресурсом.

ID	**UC-2HRP**	
Дія	**Видалення Персони**	
Опис	Опція яка слугує для видалення необхідної Персони(Персон).	
Передумова	Користувач виконав вхід в систему "KPI-Connect". В системі створений профіль Персони що необхідно видалити.	
Порядок дій	• Оберіть опцію "Кадрові ресурси" з початкового меню. • Знайдіть в списку пункт "Персона". • Оберіть опцію "Список".	• З бокового меню розгорніть опцію "Кадрові ресурси". • Натисніть "Персона" аби переглянути список Персон.
	• Відмітьте у першій колонці списку персон - персону (персон) що необхідно видалити. • Натисніть кнопку "Видалити" в кінці списку аби видалити одрану персону(персон).	• Знайдіть в списку Персону для видлення. • Натисніть опцію "Редагувати" в колонці списку "Дія". • Підтвердьте видалення персони натиснувши кномку "Видалити".
Постумова	Профіль Персони був успішно видалений.	
Коментарі	Усіх Пресон можна видалити одразу, відмітивши позначку в кінці списку Персон на опції «Застосувати до усіх» і натиснувши кнопку «Видалити».	

Рис. 3.18. Юзкейс видалення персони

Юзкейс створення викладача (рис. 3.19) — описує сценарій, за яким користувач може додати викладача в систему для подальшої взаємодії з цим ресурсом.

ID	**UC-0HRT**	
Дія	**Ствrення Викладача**	
Опис	Опція яка слугує для створення Викладача в системі	
Передумова	Користувач виконав вхід в систему "KPI-Connect". Необхідно створити Персону в системі аби призначити її Викладачем	
Порядок дій	• Оберіть опцію "Кадрові ресурси" з початкового меню. • Знайдіть в списку пункт "Викладач". • Оберіть опцію "Додати новий".	• З бокового меню розгорніть опцію "Кадрові ресурси". • Натисніть "Викладач" аби переглянути список Студентів. • В правому верхньому кутку оберіть "Додати новий".
	• Заповніть персональну інформацію оберавши Персону яку необхідно призначити викладачем натиснувши кнопку "Список" або якщо Персони немає скористайтесь кнопкою "Додати новий". • Заповніть інформацію по роботі: аудиторії (необхідно обрати зі списку, або створити нові скориставшись кнопкою "Додати новий"). • Заповніть звання: науковий ступінь, вчене звання (необхідно вибрати зі списку, або створити нові скориставшись кнопкою "Додати новий"). • Заповніть наукометричні показники: персональні наукометричні показники (необхідно вибрати зі списку, або створити нові скориставшись кнопкою "Додати новий"). • Стровіть Викладача натиснувши "Створити та редагувати", "Створити й повернутись до списку", "Створити й додати новий".	
Постумова	"Викладач" був створений і користувач може оперувати ним в системі.	
Коментарі	Більшість параметрів для створення викладача необов'язкові, але після створення їх можна змінити.	

Рис. 3.19. Юзкейс створення викладача

Юзкейс редагування викладача (рис. 3.20) — описує сценарій, за яким користувач може відредагувати дані про викладача, що були заповнені чи незаповнені на стадії створення викладача.

ID	**UC-1HRT**	
Дія	**Редагування Викладача**	
Опис	Опція що слугує для редагування інформації про Викладача в системі.	
Передумова	Користувач виконав вхід в систему "KPI-Connect". В системі створений Викладач якого необхідно відредагувати.	
Порядок дій	• Оберіть опцію "Кадрові ресурси" з початкового меню. • Знайдіть в списку пункт "Викладач". • Оберіть опцію "Список".	• З бокового меню розгорніть опцію "Кадрові ресурси". • Натисніть "Викладач" аби переглянути список Викладачів.
	• Знайдіть в списку Викладача для редагування. • Натисніть опцію "Редагувати" в колонці списку "Дія". • Відредагуйте необхідні поля (для редагвання доступні всі поля що задаються при створення Викладача). • Підтвердіть ваші дії натиснувши "Зберегти" або"Зберегти й повернутись до списку".	
Постумова	Інформація Викладача оновлена згідно попередньо внесених змін.	
Коментарі	Для редагвання доступні всі поля що задаються при створення Викладача.	

Рис. 3.20. Юзкейс редагування викладача

Юзкейс видалення викладача (рис. 3.21) — описує сценарій, за яким користувач може видалити викладача з системи.

ID	**UC-2HRT**	
Дія	**Видалення Викладача**	
Опис	Опція яка слугує для видалення необхідного Викладача (Викладачів).	
Передумова	Користувач виконав вхід в систему "KPI-Connect". В системі створений Виклада, якого необхідно видалити.	
Порядок дій	• Оберіть опцію "Кадрові ресурси" з початковго меню. • Знайдіть в списку пункт "Викладач". • Оберіть опцію "Список".	• З бокового меню розгорніть опцію "Кадрові ресурси". • Натисніть "Викладач" аби переглянути список Викладачів.
	• Відмітьте у першій колонці списку викладачів - викладача (викладачів), яких необхідно видалити. • Натисніть кнопку "Видалити" в кінці списку аби видалити обраного викладача (викладачів).	• Знайдіть в списку Викладача для видлення. • Натисніть опцію "Редагувати" в колонці списку "Дія". • Підтвердьте видалення викладача натиснувши кнопку "Видалити".
Постумова	Викладач був успішно видалений.	
Коментарі	Усіх Викладачів можна видалити одразу, відмітивши позначку в кінці списку Викладачів на опції "Застосувати до усіх" і натиснувши кнопку "Видалити".	

Рис. 3.21. Юзкейс видалення викладача

Юзкейс створення студента (рис. 3.22) — описує сценарій, за яким користувач може додати студента в систему для подальшої взаємодії з ресурсом студент.

ID	**UC-0HRS**	
Дія	**Ствренння Студента**	
Опис	Опція яка слугує для створення Студента в системі	
Передумова	Користувач виконав вхід в систему "KPI-Connect". Необхідно створити Персону в системі аби призначити її Студентом. Необхідно створити Групу аби призначити її студенту.	
Порядок дій	• Оберіть опцію "Кадрові ресурси" з початковго меню. • Знайдіть в списку пункт "Студент". • Оберіть опцію "Додати новий".	• З бокового меню розгорніть опцію "Кадрові ресурси". • Натисніть "Студент" аби переглянути список Студентів. • В правому верхньому кутку оберіть "Додати новий".
	• Заповніть загальну інформацію. Оберіть Персону, яку необхідно призначити студентом натиснувши кнопку "Список", що відноситься до поля "Персона", або якщо Персони немає скористайтесь кнопкою "Додати новий". Оберіть групу, яку необхідно призначити студенту натиснувши кнопку "Список", що відноситься до поля "Група", або якщо Персони немає скористайтесь кнопкою "Додати новий".	
Постумова	Студент був створений і користувач може оперувати ним в системі.	
Коментарі	Всі поля обов'язкові до заповнення, але їх можна потім переобрати в режимі редагування.	

Рис. 3.22. Юзкейс створення студента

Юзкейс редагування студента (рис. 3.23) — описує сценарій, за яким користувач може змінити дані про студента, що були визначені під час створення студента.

ID	**UC-1HRS**	
Дія	**Редагування Студента**	
Опис	Опція що слугує для редагування інформації про Студента в системі.	
Передумова	Користувач виконав вхід в систему "KPI-Connect". В системі створений Студент якого необхідно відредагувати.	
Порядок дій	• Оберіть опцію "Кадрові ресурси" з початкового меню. • Знайдіть в списку пункт "Студент". • Оберіть опцію "Список".	• З бокового меню розгорніть опцію "Кадрові ресурси". • Натисніть "Студент" аби переглянути список Студентів.
	• Знайдіть в списку Студента для редагування. • Натисніть опцію "Редагувати" в колонці списку "Дія". • Відредагуйте необхідні поля (для редагвання доступні всі поля що задаються при створення Студента). • Підтвердіть ваші дії натиснувши "Зберегти" або "Зберегти й повернутись до списку".	
Постумова	Інформація Студента оновлена згідно попередньо внесених змін.	
Коментарі	Для редагвання доступні всі поля що задаються при створення Студента.	

Рис. 3.23. Юзкейс редагування студента

Юзкейс видалення студента (рис. 3.24) — описує сценарій, за яким користувач може видалити студента з системи.

ID	**UC-2HRS**	
Дія	**Видалення Студента**	
Опис	Опція яка слугує для видалення необхідного Студента (Студентів).	
Передумова	Користувач виконав вхід в систему "KPI-Connect". В системі створений Студент, якого необхідно видалити.	
Порядок дій	• Оберіть опцію "Кадрові ресурси" з початкового меню. • Знайдіть в списку пункт "Студент". • Оберіть опцію "Список".	• З бокового меню розгорніть опцію "Кадрові ресурси". • Натисніть "Студент" аби переглянути список Студентів.
	• Відмітьте у першій колонці списку студентів - студента (студенів), яких необхідно видалити. • Натисніть кнопку "Видалити" в кінці списку аби видалити обраного студентів (студенів).	• Знайдіть в списку Студента для видлення. • Натисніть опцію "Редагувати" в колонці списку "Дія". • Підтвердьте видалення студента натиснувши кнопку "Видалити".
Постумова	Студент був успішно видалений.	
Коментарі	Усіх Студентв можна видалити одразу, відмітивши позначку в кінці списку Студентів на опції "Застосувати до усіх" і натиснувши кнопку "Видалити".	

Рис. 3.24. Юзкейс видалення студента

Юзкейс створення кафедри (рис. 3.25) — описує сценарій, за яким користувач може створити кафедру в системі та оперувати нею.

ID	**UC-0USDEP**	
Дія	**Створення Кафедри**	
Опис	Опція що слугує створенню структурної частини університету - Кафедри.	
Передумова	Користувач виконав вхід в систему "KPI-Connect". Необхідно створити Підрозділ в системі аби призначити його Кафедрою.	
Порядок дій	• Оберіть опцію "Структура університету" з початкового меню. • Знайдіть в списку пункт "Кафедра". • Оберіть опцію "Додати новий".	• З бокового меню розгорніть опцію "Сртуктура університету". • Натисніть "Кафедра" аби переглянути список Персон. • В правому верхньому кутку оберіть "Додати новий".
	• Визначте Підрозділ, який необхідно призначити Кафедрою натиснувши кнопку "Список", що відноситься до поля "Підрозділ", або якщо Підрозділу немає скористайтесь кнопкою "Додати новий". • Визначте Спеціальність, яку необхідно призначити Кафедрі натиснувши кнопку "Список", що відноситься до поля "Спеціальність", або якщо Спецільаності немає скористайтесь кнопкою "Додати новий".	
Постумова	"Кафедра" була створена і користувач може оперувати цим ресурсом в системі.	
Коментарі	Всі поля обов'язкові до заповнення, але їх можна потім переобрати в режимі редагування.	

Рис. 3.25. Юзкейс створення кафедри

Юзкейс редагування кафедри (рис. 3.26) — описує сценарій, за яким користувач може відредагувати вже існуючу кафедру в системі.

ID	**UC-1USDEP**	
Дія	**Редагування Кафедри**	
Опис	Опція що слугує для редагування Каферди в системі.	
Передумова	Користувач виконав вхід в систему "KPI-Connect". В системі створена Кафедра яку необхідно відредагувати.	
Порядок дій	• Оберіть опцію "Структура університету" з початкового меню. • Знайдіть в списку пункт "Кафедра". • Оберіть опцію "Список".	• З бокового меню розгорніть опцію "Структура університету". • Натисніть "Кафедра" аби переглянути список Студентів.
	• Знайдіть в списку Кафедру для редагування. • Натисніть опцію "Редагувати" в колонці списку "Дія". • Відредагуйте необхідні поля (для редагвання доступні всі поля що задаються при створення Кафедри). • Підтвердіть ваші дії натиснувши "Зберегти" або "Зберегти й повернутись до списку".	
Постумова	Інформація Кафедри оновлена згідно попередньо внесених змін.	
Коментарі	Для редагвання доступні всі поля що задаються при створення Кафедри.	

Рис. 3.26. Юзкейс редагування кафедри

Юзкейс видалення кафедри (рис. 3.27) — описує сценарій, слідуючи якому користувач може видалити існуючу кафедру.

ID	**UC-3USDEP**	
Дія	**Видалення Кафедри**	
Опис	Опція яка слугує для видалення необхідної Кафедри (Кафедр).	
Передумова	Користувач виконав вхід в систему "KPI-Connect". В системі створена Кафедра, яку необхідно видалити.	
Порядок дій	• Оберіть опцію "Структура університету" з початкового меню. • Знайдіть в списку пункт "Кафедра". • Оберіть опцію "Список".	• З бокового меню розгорніть опцію "Структура університету". • Натисніть "Кафедра" аби переглянути список Кафедр.
	• Відмітьте у першій колонці списку Кафедр - Кафедру (Кафедри), які необхідно видалити. • Натисніть кнопку "Видалити" в кінці списку аби видалити обрану Кафедру (Кафедри).	• Знайдіть в списку Кафедру для видлення. • Натисніть опцію "Редагувати" в колонці списку "Дія". • Підтвердьте видалення студента натиснувши кнопку "Видалити".
Постумова	Кафедра була успішно видалена.	
Коментарі	Усі Кафедри можна видалити одразу, відмітивши позначку в кінці списку Кафедр на опції "Застосувати до усіх" і натиснувши кнопку "Видалити".	

Рис. 3.27. Юзкейс видалення кафедри

Юзкейс створення групи (рис. 3.28) — описує сценарій, за яким користувач може додати нову групу в систему.

Дія	**Ствренння Групи**	
Опис	Опція яка слугує для створення Групи в системі	
Передумова	Користувач виконав вхід в систему "KPI-Connect". Необхідно створити Навчальний план до якого має бути прив'язана група. Необхідно створити Викладача, аби призначити його як Керівника групи.	
Порядок дій	• Оберіть опцію "Процес навчання" з початкового меню. • Знайдіть в списку пункт "Група". • Оберіть опцію "Додати новий".	• З бокового меню розгорніть опцію "Процес навчання". • Натисніть "Група" аби переглянути список Груп. • В правому верхньому кутку оберіть "Додати новий".
	• Визначте Префікс Групи • Визначте Номер Групи • Визначте Навчальний план, до якого має бути прив'язана група натиснувши кнопку "Список", що відноситься до поля "Навчальний план", або якщо Навчального плану немає скористайтесь кнопкою "Додати новий". • Оберіть Викладача, який буде призначений Керівником Групи натиснувши кнопку "Список", що відноситься до поля "Керівник", або якщо Викладача немає скористайтесь кнопкою "Додати новий". • Оберіть Студентів, які будуть в цій групі, одрати студентів можна натиснувши для поле до заповнення та обравши з списку. або якщо в системі немає жодного Студента скористайтесь кнопкою "Додати новий".	
Постумова	Група була створена і користувач може оперувати нею в системі.	
Коментарі	Всі поля обов'язкові до заповнення окрім поля Студенти, але їх можна потім переобрати в режимі редагування. Список студентів має множинний вибір.	

Рис. 3.28. Юзкейс створення групи

Юзкейс редагування групи (рис 3.29) — описує сценарій, за яким користувач може редагувати створену в системі групу.

ID	**UC-1EPGRO**	
Дія	**Редагування Групи**	
Опис	Опція що слугує для редагування інформації про Групу в системі.	
Передумова	Користувач виконав вхід в систему "KPI-Connect". В системі створена Група яку необхідно відредагувати.	
Порядок дій	• Оберіть опцію "Процес навчання" з початкового меню. • Знайдіть в списку пункт "Група". • Оберіть опцію "Список".	• З бокового меню розгорніть опцію "Процес навчання". • Натисніть "Група" аби переглянути список Студентів.
	• Знайдіть в списку Група для редагування. • Натисніть опцію "Редагувати" в колонці списку "Дія". • Відредагуйте необхідні поля (для редагвання доступні всі поля що задаються при створенні Групи). • Підтвердіть ваші дії натиснувши "Зберегти" або "Зберегти й повернутись до списку".	
Постумова	Інформація Групи оновлена згідно попередньо внесених змін.	
Коментарі	Для редагвання доступні всі поля що задаються при створення Групи.	

Рис. 3.29. Юзкейс редагування групи

Юзкейс видалення групи (рис. 3.30) — описує сценарій, виконавши який користувач може видалити існуючу в системі групу.

ID	**UC-2EPGRO**	
Дія	**Видалення Групи**	
Опис	Опція яка слугує для видалення необхідної Групи (Груп).	
Передумова	Користувач виконав вхід в систему "KPI-Connect". В системі створена Група, якого необхідно видалити.	
Порядок дій	• Оберіть опцію "Процес навчання" з початкового меню. • Знайдіть в списку пункт "Група". • Оберіть опцію "Список".	• З бокового меню розгорніть опцію "Процес Навчання". • Натисніть "Група" аби переглянути список Студентів.
	• Відмітьте у першій колонці списку Груп - Групу (Групи), яких необхідно видалити. • Натисніть кнопку "Видалити" в кінці списку аби видалити обрану Групу (Групи).	• Знайдіть в списку Студента для видлення. • Натисніть опцію "Редагувати" в колонці списку "Дія". • Підтвердьте видалення студента натиснувши кнопку "Видалити".
Постумова	Група була успішно видалена.	
Коментарі	Усі Групи можна видалити одразу, відмітивши позначку в кінці списку Грцп на опції "Застосувати до усіх" і натиснувши кнопку "Видалити".	

Рис. 3.30. Юзкейс видалення групи

Юзкейс створення дисертації (рис. 3.31) — описує сценарій, за яким користувач може додати в систему сутність дисертації.

ID	**UC-LDIS**	
Дія	**Ствренння Дисертації**	
Опис	Опція яка слугує для створення Дисертації в системі.	
Передумова	Користувач виконав вхід в систему "KPI-Connect". Необхідно створити Вид дисертації в системі.	
Порядок дій	• Оберіть опцію "Бібліотека" з початкового меню. • Знайдіть в списку пункт "Дисертація". • Оберіть опцію "Додати новий".	• З бокового меню розгорніть опцію "Бібліотека". • Натисніть "Дисертація" аби переглянути список Дисертацій. • В правому верхньому кутку оберіть "Додати новий".
	• Заповніть загальну інформацію: • Назва • Назва англійською • Оберіть Вид дисертації, який необхідно призначити натиснувши кнопку "Список", що відноситься до поля "Вид дисертації", або якщо немає необхідного в визначеному списку, скористайтесь кнопкою "Додати новий". • Оберіть Викладача, яку необхідно призначити Керівником Дисертації натиснувши кнопку "Список", що відноситься до поля "Керівник", або якщо Викладача немає скористайтесь кнопкою "Додати новий". • Заповніть інформацію з контролю якості за Дисертацією: • Номоконтроль • Процент плагіату • Чи завершена перевірка на плагіат • Статус керівника • Статус нормоконтролю • Статус плагіату	
Постумова	Дисертація була створений і користувач може оперувати нею в системі.	
Коментарі	Більша частина полів не обов'язкові до заповнення, але їх можна потім переобрати в режимі редагування. Вид дисертації за замовчуванням - Дипломний проєкт (бакалавра).	

Рис. 3.31. Юзкейс створення дисертації

Юзкейс редагування дисертації (рис 3.32) — описує сценарій, пройшовши який, користувач може відредагувати необхідні дані про дисертацію що була попередньо створена в системі.

ID	**UC-1LDIS**	
Дія	**Редагування Дисертації**	
Опис	Опція що слугує для редагування інформації про Дисертацію в системі.	
Передумова	Користувач виконав вхід в систему "KPI-Connect". В системі створена Дисертація яку необхідно відредагувати.	
Порядок дій	• Оберіть опцію "Бібліотека" з початкового меню. • Знайдіть в списку пункт "Дисертація". • Оберіть опцію "Список".	• З бокового меню розгорніть опцію "Бібліотека". • Натисніть "Дисертація" аби переглянути список Студентів.
	• Знайдіть в списку Дисертацію для редагування. • Натисніть опцію "Редагувати" в колонці списку "Дія". • Відредагуйте необхідні поля. • Підтвердіть ваші дії натиснувши "Зберегти" або "Зберегти й повернутись до списку".	
Постумова	Інформація Дисертації оновлена згідно попередньо внесених змін.	
Коментарі	Для редагвання доступні всі поля що задаються при створенні Дисертації.	

Рис. 3.32. Юзкейс редагування дисертації

Юзкейс видалення дисертації (рис. 3.33) — описує сценарій, за яким користувач може видалити дисертацію що існує в системі.

ID	**UC-2LDIS**	
Дія	**Видалення Дисертації**	
Опис	Опція яка слугує для видалення необхідної Дисертації (Дисертацій).	
Передумова	Користувач виконав вхід в систему "KPI-Connect". В системі створена Дисертація, яку необхідно видалити.	
Порядок дій	• Оберіть опцію "Бібліотека" з початкового меню. • Знайдіть в списку пункт "Дисертація". • Оберіть опцію "Список".	• З бокового меню розгорніть опцію "Бібліотека". • Натисніть "Дисертація" аби переглянути список Студентів.
	• Відмітьте у першій колонці списку Дисертацій - Дисертацію (Дисертації), які необхідно видалити. • Натисніть кнопку "Видалити" в кінці списку аби видалити обрану Дисертацію (Дисертації).	• Знайдіть в списку Дисертацію для видлення. • Натисніть опцію "Редагувати" в колонці списку "Дія". • Підтвердьте видалення Дисертації натиснувши кнопку "Видалити".
Постумова	Студент був успішно видалений.	
Коментарі	Усі Дисертації можна видалити одразу, відмітивши позначку в кінці списку Дисертацій на опції "Застосувати до усіх" і натиснувши кнопку "Видалити".	

Рис. 3.33. Юзкейс видалення дисертації

Всі юзкейси оформлені в веб-додатку *google tables,* для зручного орієнтування по всіх юзкейсах було прийнято рішення розподілити їх згідно основних пунктів меню в системі (кожен пункт меню — окрема тека) та на кожний під-пункт системи виділити свою сторінку в документі таблиці (рис.3.34). Також юзкейси були задокументовані англійською мовою.

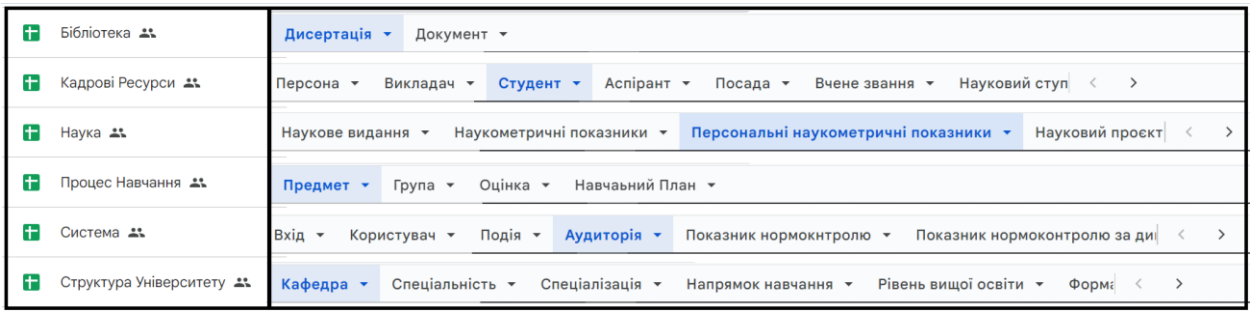

Рис. 3.34. Структура зберігання розроблених юзкейсів

3.8.2. Структурні макети сторінок

Структурний макет сторінки після входу в систему який налічує меню загального плану з управляння різними ресурсами, в особливості перегляду списку та додання нового (рис. 3.35).

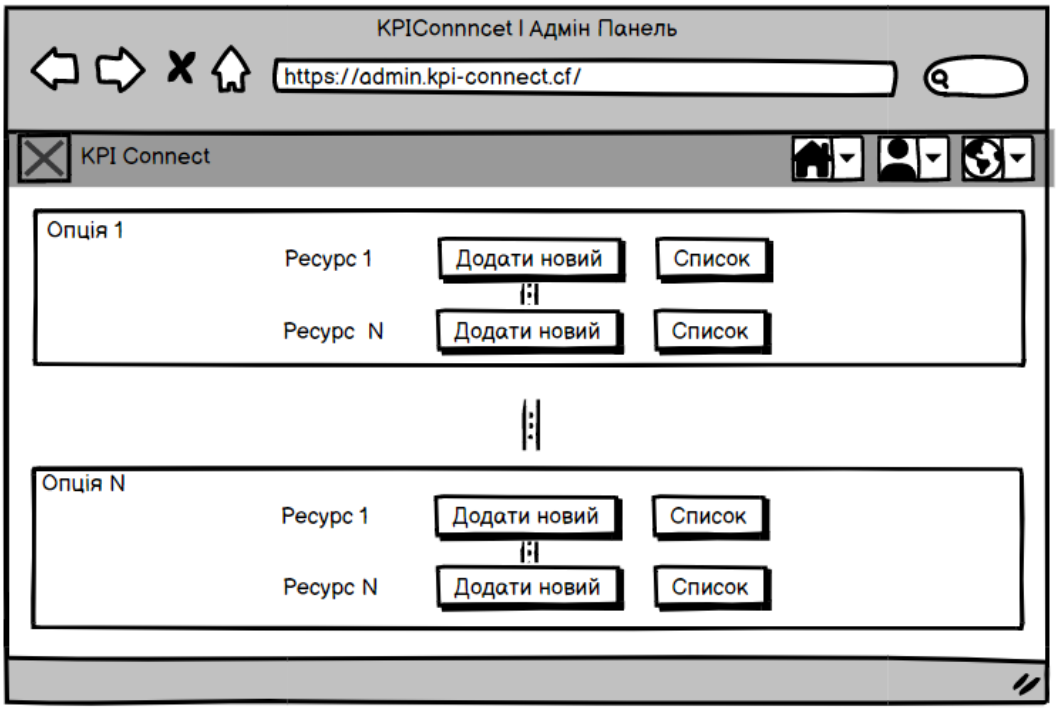

Рис. 3.35. Початкова сторінка — загальне меню

Структурний макет сторінки з списком ресурсів певного типу (рис. 3.36). Сторінка налічує список ресурсів одного типу у вигляді таблиці, загальне меню переміщується в бокове меню, де по кліку на певну опцію можна переглянути її дочірні пункти. На цій же сторінці даються можливості до показу/редагування/додавання ресурсу.

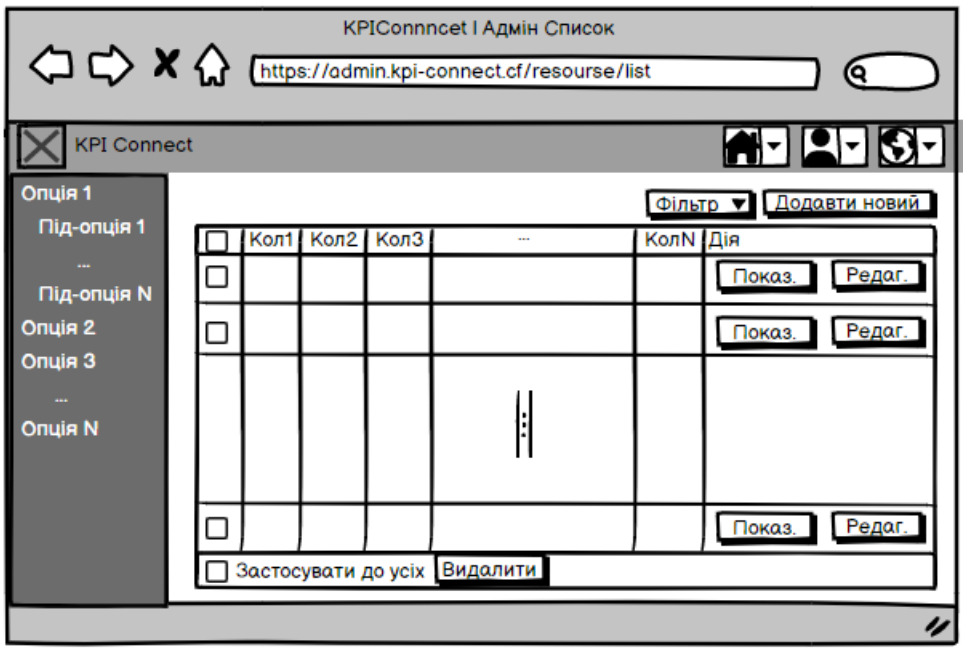

Рис. 3.36. Сторінка списку певних ресурсів системи

Наступна сторінка — сторінка додання якогось ресурсу/сутності в систему (рис. 3.37).

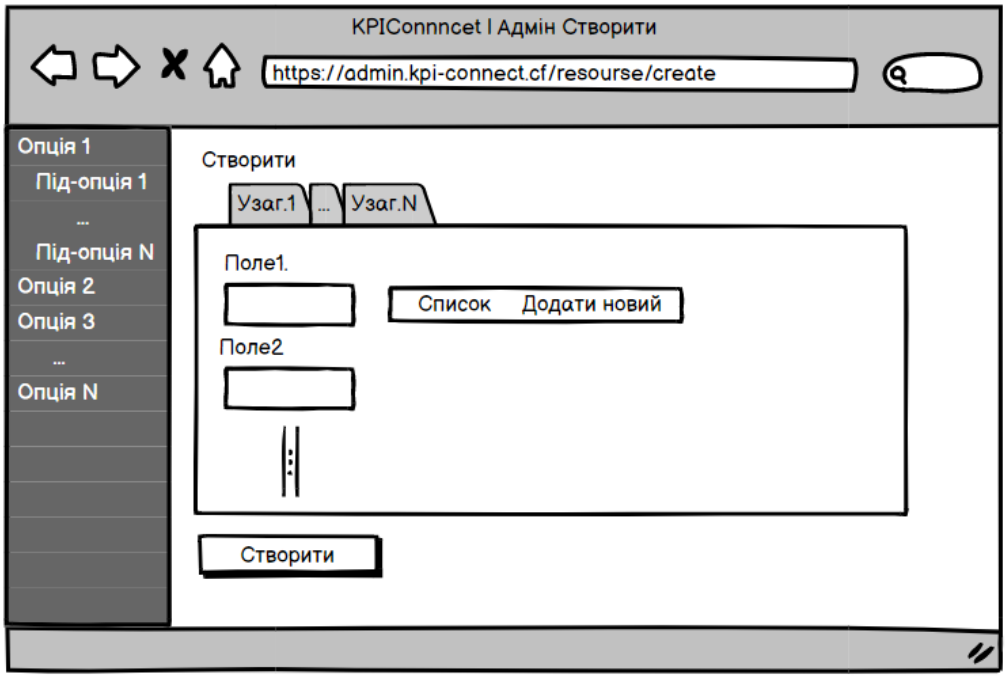

Рис. 3.37. Сторінка додання нового ресурсу в систему

Сторінка редагування одного з ресурсів дуже схожа на сторінку створення, адже дозволено редагувати всі ті самі поля що і доступні при створенні якогось ресурсу, а також можливість видалити цей ресурс (рис. 3.38).

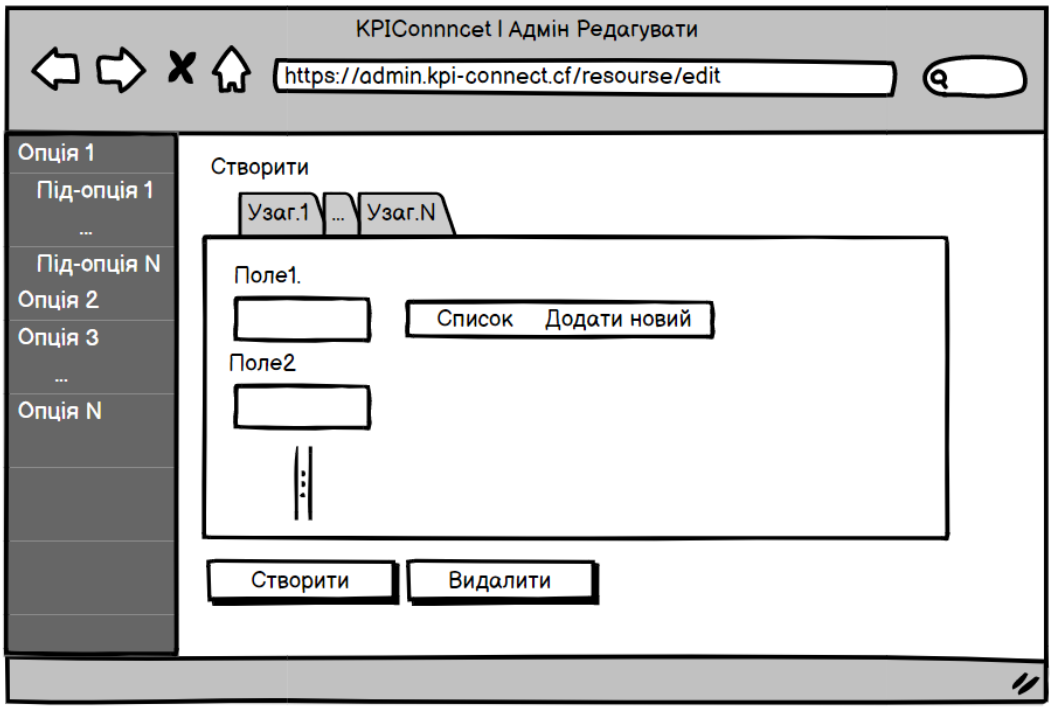

Рис. 3.38. Сторінка редагування певного ресурсу системи

Сторінка показу інформації (рис. 3.39) — сторінка що відображає інформацію обраного ресурсу в загальному вигляді.

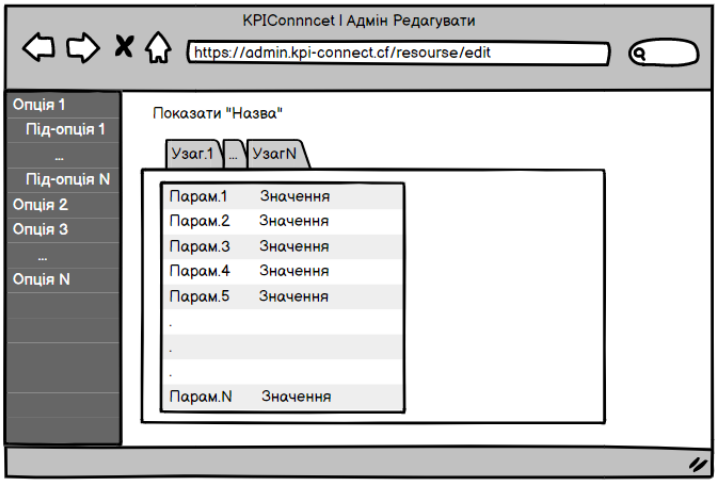

Рис. 3.39. Сторінка показу інформації ресурсу

3.8.3. Документація до бази даних

Цей документ налічує опис бази даних, а саме, з яких таблиць складається база даних, які колонки має, які зв'язки мають різні таблиці (рис. 3.40).

Загалом система складається з 62 таблиць. І кожна таблиця має зв'язок з іншою, тому було прийнято рішення для швидкої орієнтації по даних, зробити документ з описом. Що дійсно в подальшому стало в нагоді.

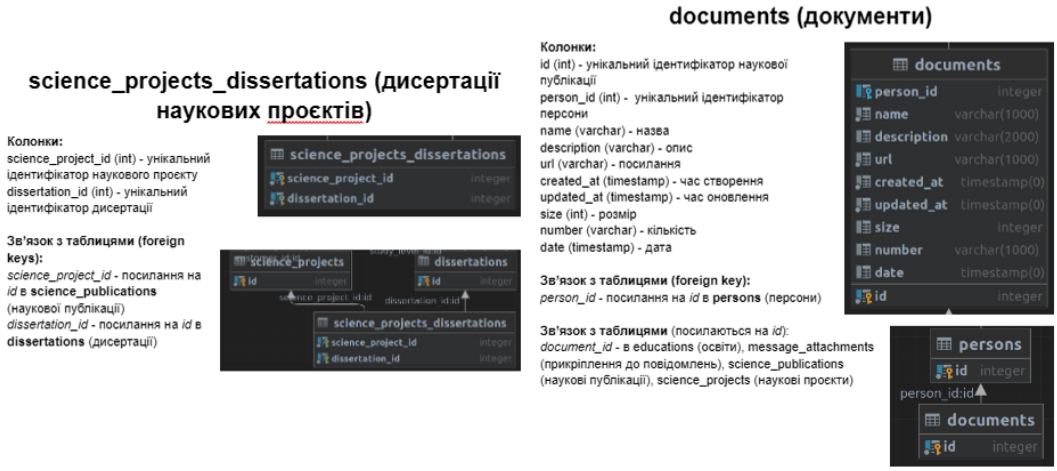

Рис. 3.40. Приклад опису кількох таблиць бази даних

Документ оформлений у веб-додатку *google documents*. Документ має зміст для швидкої переходу на необхідну таблицю та ознайомлення з інформацією (рис. 3.41).

Рис. 3.41. Зміст документа з описом таблиць

3.8.4. Тесткейси

Тесткейси допомагають оцінити всі функції системи з різних сторін, якщо вони вірно складені, та задокументувати які тесткейси були пройдені, які ні, і на основі цього створити звіти дефектів та передати розробникам на виправлення.

Тесткейси входу користувача в систему (рис. 3.42) — перевіряють різні варіації, за допомогою яких користувач може чи не може зайти в систему.

ID	Тесткейс	Тестові дані	Очікуваний результат	Статус	Примітка
TC-0LI	Вхід в систему методом заповнення username & password	username: "" password: ""	Нездійснення входу в систему	Пройдено	Переводить користувача на перше незаповнене поле
TC-1LI		username: "admin@gmail.com" password: ""	Нездійснення входу в систему	Пройдено	
TC-2LI		username: "" password: "123"	Нездійснення входу в систему	Пройдено	
TC-3LI		username: "admin@gmail.com" password: "123"	Здійснення входу в систему	Пройдено	
TC-4LI		незареєстрований користувач	Нездійснення входу в систему	Пройдено	Застереження що автентифікаційні дані невірні
TC-5LI		username - вірний password - невірний	Нездійснення входу в систему	Пройдено	
TC-6LI	Вхід в систему за допомогою сторонніх сервісів (google, facebook, linkedIn)	google account - зареєстрований користувач	Здійснення входу в систему	Не пройдено	Опція не працює
TC-7LI		facebook - зареєстрований користувач	Здійснення входу в систему	Не пройдено	
TC-8LI		linkedIn- зареєстрований користувач	Здійснення входу в систему	Не пройдено	
TC-9LI	Вхід якщо забув пароль	username - зареєствореного користувача	Відправити на пошту лист підтвердження, та заміна паролю	Не пройдено	

Рис. 3.42. Тесткейси входу користувача в систему

Тесткейси для ресурсу кафедра (рис. 3.43) — перевірка різних взаємодій що дозволені системою з ресурсом кафедри.

ID	Тесткейс	Тестові дані	Очікуваний результат	Статус	Примітка
TC-0USDEP	Створення кафедри (поля підрозділ, спеціальності)	Нічого не вибрано	Кафедра не створена	Passed	Застереження не було, система впала з помилкою
TC-1USDEP		Обрати будь-який розділ зі список Спеціальність не обирати	Кафедра створена	Passed	
TC-2USDEP		Обрати будь-який розділ зі списку, та спецільність	Кафедра створена	Passed	
TC-3USDEP		Додати нову кафедру та спецільаність	Кафедра створена	Passed	Заплутаний алгоритм, вікна відкриваються одне поверх іншого, якщо не вистачає даних аби створити щось нове
TC-4USDEP		Обрати підрозділ що є вже кафедрою	Кафедра не створена	Passed	Застереження не було, система впала з помилкою
TC-5USDEP		Обрати підрозділ що вже ідентифікований як якийсь конретний підрозділ, але не кафедра	Кафедра не створена	Falied	Був створений, в кафедрі відображається, але в списку всіх розділів ні
TC-6USDEP	Редагування кафедри	Очистити поле підрозділу	Кафедра не відредагована	Passed	Застереження не було, система впала з помилкою
TC-7USDEP	Видалення кафедри	Обрати кафедру, до якої щось вже прив'язано	?		Кафедра не видалена Застереження не було, система впала з помилкою
TC-8USDEP	Видалення всіх кафедр	Обрати всі з врахуванням яка має зв'язки	?		Кафеди не видалена Застереження не було, система впала з помилкою
TC-9USDEP		Обрати всі без зв'язків	Кафедри видалені	Passed	
TC-10USDEP	Перегляд кафедри		Показана інформація по каферді	Passed	

Рис. 3.43. Тесткейси для ресурсу кафедра

Після перевірки кількох форм ресурсів, було зрозуміло що все створюється видаляється та може редагуватись адже це автогенеровані *API,* які

працюватимуть однаково для різних ресурсів (опції доступні) тож слід сконцентруватись на валідації даних в формах та те, наскільки вони валідні з точки зору внесення їх в базу даних, та прийнятих шаблонів.

Тесткейси для форми ресурсу спеціальність (рис. 3.44) — розглянуті різні випадки валідації полів що необхідно заповнити для створення спеціальності.

ID	Тесткейс	Тестові дані	Очікуваний результат	Статус	Примітка
TC-0USSPEC_Y	Поле Назва	dfvd1231!@#$%^&*()_+~	Некоректна Назва	Falied	Доступні до заповнення всі спец символи та латиниця
TC-1USSPEC_Y		назва що більша 100 символів	Некоректна Назва	Passed	Застереження не було, система впала з помилкою Поле вводу не має обмеження
TC-3USSPEC_Y		назва що по дивжині = 100 символоам	Некоректна Назва	Passed	З точки зору організації таблиць коектна, з точки зору назви спеціальності ні
TC-4USSPEC_Y		заповнити пропусками	Некоректна Назва	Passed	Застереження не було, система впала з помилкою
TC-5USSPEC_Y	Поле Ідентифікатор	dfvd1231!@#$%^&*()_+~	Некоректий ідентифікатор	Falied	Доступні до заповнення всі спец символи та латиниця
TC-6USSPEC_Y		назва що більша 255 символів	Некоректий ідентифікатор	Passed	Застереження не було, система впала з помилкою Поле вводу не має обмеження
TC-7USSPEC_Y		назва що по дивжині = 255 символоам	Некоректий ідентифікатор	Passed	З точки зору організації таблиць коректна, з точки зору назви спеціальності ні
TC-8USSPEC_Y		ввести такий номер що вже є напрямку навчання	Некоректий ідентифікатор	Passed	Застереження не було, система впала з помилкою
TC-9USSPEC_Y		заповнити пропусками	Некоректий ідентифікатор	Passed	Застереження не було, система впала з помилкою

Рис. 3.44. Тесткейси форми ресурсу спеціальність

Після здійснення певних тесткейсів було прийнято рішення переглянути код, реалізації з валідаціями полів в формах, адже велика частина форм не пройшла належної перевірки на безпосередню валідацію даних.

Переглянувши код де реалізовані перевірки для форм до заповнення можна помітити що зроблені лише базові речі такі як обов’язковість полів, а також для деяких полів підказки що засвідчують що необхідно вписати в це поле, і валідація деяких полів таких як дати, телефон та поштова скринька.

Приклад прописування параметрів для валідації в системі для різних полів (рис. 3.45, 3.46).

```
->add(PersonInterface::FIELD_WORK_PHONE_NUMBER, TextType::class, [
    'required' => false,
    'attr' => [
        'data-mask' => '+38 (000) 000-00-00',
        'placeholder' => '+38 (___) ___-__-__'
    ],
])
```

Рис. 3.45. Задання поля телефону

```
->add(PersonInterface::FIELD_LAST_NAME, TextType::class, [
    'required' => false
])
->add(PersonInterface::FIELD_FIRST_NAME, TextType::class, [
    'required' => false
])
->add(PersonInterface::FIELD_MIDDLE_NAME, TextType::class, [
    'required' => false
])
->end()
->with('tab.label_full_name_eng', ['class' => 'col-md-6'])
->add(PersonInterface::FIELD_FIRST_NAME_ENGLISH,TextType::class,[
    'required' => false
])
->add(PersonInterface::FIELD_LAST_NAME_ENGLISH,TextType::class,[
    'required' => false
])
->end()
->add(PersonInterface::FIELD_GENDER, ChoiceType::class, [
    'choices' => array_flip(Person::AVAILABLE_GENDER_VALUES),
    'choice_translation_domain' => $this->getTranslationDomain()
])
->add(PersonInterface::FIELD_BIRTH_DATE, BirthdayType::class, [
    'required' => false,
    'choice_translation_domain' => $this->getTranslationDomain()
])
->add(PersonInterface::FIELD_INN, TextType::class, [
    'required' => false
])
```

Рис. 3.46. Задання полів для форми персональної інформації персони

За своєю структурою тесткейси макета сторінок мають дещо інший вигляд аніж тесткейси функцій, перевірки полів, взаємодії різних модулів тощо.

Тесткейс відповідності макета для початкової сторінки (рис.3.47) — має усі визначені макетом компоненти, але також має надлишкові.

Контейнер	Тип елементу	Статус
Меню (згори)	Кнопка посилання (Логотип + назва)	Passed
	Кнопка (Домашня сторінка)	Passed
	Кнопка/меню (профіль)	Passed
	Кнопка/меню (зміна мови)	Passed
Основна секція	Список (структура університету)	Passed
	Список (кадрові ресурси)	Passed
	Список (процес навчання)	Passed
	Список (наука)	Passed
	Список (бібліотека)	Passed
	Список (система)	Passed
Зайві елементи		
Бокове меню	Кнопки/меню	
Меню (згори)	Кнопка (навігації)	
	Кнопка/меню (всіх ресурсів)	

Рис. 3.47. Тесткейс дизайну початкової сторінки з загальним меню

Тесткейси дизайну сторінки певних ресурсів (рис. 3.48) — містить таблицю, містить всі необхідні елементи, а також додаткові, які не були передвизначені макетом, але під час процесу були визначені як необхідні.

Контейнер	Тип елементу	Статус
Бокове меню	Кнопки/меню з переходом на ресурс	Passed
Основна секція	Кнопка/меню (фільтр)	Passed
	Кнопка (додати новий)	Passed
	Таблиця	Passed
Таблиця	Колонки характеристик	Passed
	Колонка дії з кнопками показати, редагувати	Passed
	Колонка вибору	Passed
	Кнопка (видалити)	Passed
	Відмітка (застосувати до усіх)	Passed
Зайві елементи - затверджені як необхідні		
Таблиця	меню пагінації	Passed
	Кнопка розміронсті сторінки	Passed

Рис. 3.48. Тесткейси дизайну сторінки певних ресурсів системи

Також буди написані авто-тести для перевірки роботи кнопок, відкривання меню, входу на різні сторінки, входу в систему тощо (рис. 3.49, 3.50).

```
    @BeforeTest
    public void setUp() {...}

    @AfterTest
    public void tearDown() { driver.quit(); }

    @Test
    public void testPerson() {
        WebElement pDropdown = driver.findElement(By.cssSelector("[href='/admin/persons/list']"));
        pDropdown.click();

        WebElement pageTitle = driver.findElement(By.cssSelector(".nav.navbar-top-links.breadcrumb .active"));
        String expectedPageTitle = "Person list";
        String actualPageTitle = pageTitle.getText();
        assert(expectedPageTitle.equals(actualPageTitle));
    }

    @Test
    public void testTeacher() {
        WebElement tDropdown = driver.findElement(By.cssSelector("[href='/admin/teachers/list']"));
        tDropdown.click();

        WebElement pageTitle = driver.findElement(By.cssSelector(".nav.navbar-top-links.breadcrumb .active"));
        String expectedPageTitle = "Teacher list";
        String actualPageTitle = pageTitle.getText();
        assert(expectedPageTitle.equals(actualPageTitle));
    }
```

Рис. 3.49. Автотести для бокового меню та валідності переходу по натисканню

```
@BeforeTest
public void setUp() {...}

@AfterTest
public void tearDown() {
    driver.quit();
}

@Test(priority = 1)
public void testButtonDropdownUser() {
    WebElement profileDropdown = driver.findElement(By.xpath("/html/body/div/header/nav/div[2]/ul/li[3]/a"));
    profileDropdown.click();

    WebElement openedMenu = driver.findElement(By.cssSelector(".dropdown.user-menu.open"));
    assert(openedMenu.isDisplayed());
}

@Test(priority = 2)
public void testLogOut() {
    WebElement logoutButton = driver.findElement(By.xpath("/html/body/div/header/nav/div[2]/ul/li[3]/ul/li[4]/ul/li/a"));
    logoutButton.click();

    WebElement startButton = driver.findElement(By.cssSelector("[href='/auth/login']"));
    String expectedButtonText = "START NOW";
    String actualButtonText = startButton.getText();
    assert(expectedButtonText.equals(actualButtonText));
}
```

Рис. 3.50. Автотести для кнопки/меню профілю користувача

Для тесту навантаження були розроблені різні кейси з поступовим навантаженням на систему (рис. 3.51).

ID	Тесткейс	Результат
TC-1LT	Користувачі: 100 Час: 60сек Повтори: 1	100%
TC-2LT	Користувачі: 200 Час: 60сек Повтори: 1	100%
TC-3LT	Користувачі: 400 Час: 60сек Повтори: 1	100%
TC-4LT	Користувачі: 800 Час: 60сек Повтори: 1	99,90%
TC-5LT	Користувачі: 1600 Час: 60сек Повтори: 1	99,70%
TC-6LT	Користувачі: 3200 Час: 60сек Повтори: 1	99%
TC-7LT	Користувачі: 1600 Час: 300сек Повтори: 1	99,80%
TC-8LT	Користувачі: 3200 Час: 300сек Повтори: 1	99,10%
TC-9LT	Користувачі: 1600 Час: 600сек Повтори: 1	100%
TC-10LT	Користувачі: 3200 Час: 600сек Повтори: 1	99,90%
TC-11LT	Користувачі: 3200 Час: 1800сек Повтори: 1	100%
	Користувачі: 6000 Час: 1800сек	

Рис. 3.51. Тесткейси навантаження на систему

Тестування навантаження здійснювалось за допомогою програми *JMeter*, що дозволяє виконати тестування навантаження, основними параметрами до встановлення є кількість користувачів та час (рис. 3. 52).

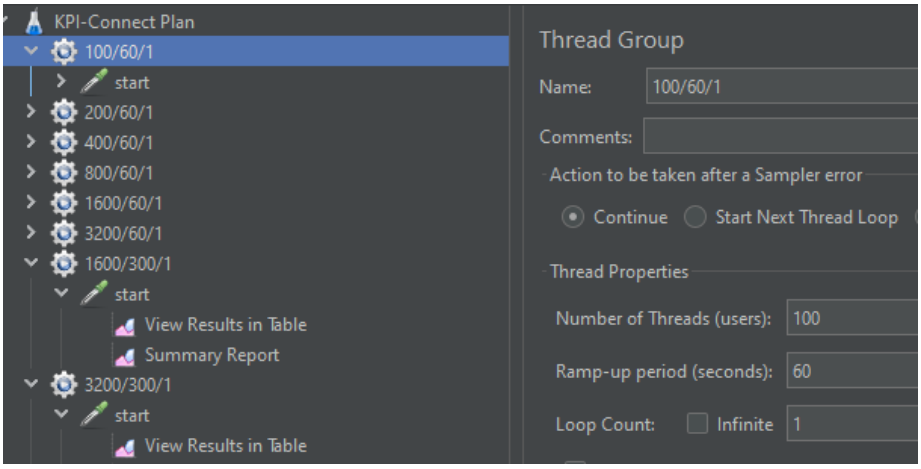

Рис. 3.52. Приклад налаштування параметрів для здійснення тестування

ВИСНОВКИ ДО РОЗДІЛУ 3

В даному розділі було здійснено комплексний огляд продукту та аналіз продукту. Були задокументовані вимоги до системи у вигляді юзкейсів, що водночас і стали корисними тесткейсами. Була розроблена стратегія контролю якості, а саме визначено що буде покриватись тестами, які види тестувань будуть застосовуватись та ризики. Були визначені цілі контролю якості та задокументовані у вигляді картки думок, з поділом на групи. Були сплановані критерії призупинення контролю якості та завершення. Були визначені ролі та задачі під час контролю якості. Відбулось також і налаштування середовища до контролю якості. Структурні макети сторінок були розроблені, що є також складовою частиною вимог. Була описана база даних усі її таблиці, що допомогло заглибитись та розібратись у структурі системи та зв’язках ресурсів. Зрештою був здійснений контроль якості під час якого також були розроблені такі артефакти як різні тесткейси, авто-тести тощо.

РОЗДІЛ 4
АНАЛІЗ ТА ЗАГАЛЬНИЙ ЗВІТ

4.1. Аналіз плану здійснення контролю якості

Насамперед першим в здійсненні контролю якості є визначений перелік вимог від клієнта. Маючи задокументований перелік вимог, набагато простіше оглядати продукт та одразу визначати чи відповідає він вимогам на поверхневому рівні. Це можуть бути наступні речі: функції, дизайн, зручність тощо. Найкраще за все під час огляду буде занотовувати все, наприклад якщо є явна невідповідність вимогам (не вистачає функцій, чи навіть їх більше, не така кольорова гама, відхилення від прийнятого дизайну), а також відмічати моменти які або не зрозумілі, або недоречні, чи які можна було б покращити з точки зору зручності користування тощо. Зробивши огляд продукту згідно вимог, та все задокументувавши, можна визначити критичні невідповідності та створити пул питань до розробників, дизайнерів чи інших відповідних осіб, щоб отримати на ці питання відповіді та продовжити аналізувати продукт. Насамперед довідатись про те хто буде користуватись продуктом, задачі продукту та вимоги до апаратного/програмного забезпечення можна з вимог до продукту, усі ці речі, це базові поняття які визначаються на перших стадіях розробки програмного забезпечення, аби було зрозуміло навіщо ж цей продукт, хто буде ним користуватись, та які будуть вимоги до користувацьких пристроїв. Згідно цих базових речей можна продовжувати заглиблюватись у наступні ітерації визначення вимог щодо дизайну, функцій та решти специфічних складових продукту що буде розроблятись. Звісно не є новиною, що перш ніж розроблювати програмний продукт, вимоги до нього могли бути обговорені поверхнево, у найгіршому випадку не задокументовані ніяким чином, тож на стадії контролю якості це вилізе боком і тестувальники не будуть мати поняття того, контроль якості чого ж зрештою потрібно перевіряти, та яким чином, якщо вимог нема, якщо є суто кінцевий продукт. В такому випадку чи немалих зусиль доведеться прикласти аби у цьому розібратись, приділити велику кількість часу спілкуванню

з розробниками, клієнтами, дизайнерами аби назбирати всі вимоги проаналізувати власне продукт та приступити до контролю якості, що в кінцевому випадку займе чималу кількість часу. Під час аналізу продукту необхідно було додатково задокументувати вимоги до веб-додатку, ознайомитись безпосередньо з додатком та подивитись в загальному плані як працює система, і у разі питань — спілкування безпосередньо з розробником чи клієнтом.

Визначення того що необхідно покрити є зазвичай всі основні аспекти системи, саме тому покривати в обов'язковому порядку було вирішено функції системи — тому що це основа, те без чого система не може працювати повноцінно, від неправильного функціонування слідує неправильна поведінка та неможливість користування. Наступне що було обрано — це валідація форм, оскільки це інформаційна система і система зберігає велику кількість інформації, а відповідно цю інформацію додає користувач, тож для правильного зберігання даних, має бути належним чином перевірена валідація всіх полів форм до заповнення, аби інформація подавалась та зберігалась в системі правильно. Обов'язково так само перевірити інтерфейс веб-додатку, адже інтерфейс це те що справляє перші враження на користувача, його зручність, його справність, те від чого буде залежати чи захоче користувач використовувати цей веб-додаток, адже яким би багатофункціональним не був додаток, але він не зручний та незрозумілий до використання, тим більше користувачі будуть намагатись знайти щось схожого плану, але більш зручне.

Вибір типів тестування обов'язково був взятий модульний, інтеграційний та системний. Модульний, адже структурно система розбита на різні модулі (ресурси), інтеграційний — адже модулі в системі можуть взаємодіяти між собою та бути пов'язаними, системний — тому що відповідність вимогам це найважливіший критерій оцінювання під час проходження контролю якості, адже в першу чергу система розроблюється згідно бажань замовника. Нефункціональні тести, такі як тести зручності, сумісності та навантаження були обрані, адже наступним після замовника, є користувачі, які будуть користуватись

системою. Для користувачів дуже важлива зручність, щоб все було максимально просто. Не менш важлива і сумісність, адже користувачі мають різні вподобання до браузерів та не є виключенням що користувач захоче зайти в систему з телефону. Тест навантаження дасть показник по тому, як система працює в залежності від кількості користувачів що користуються нею в один момент часу, або впродовж певного часу.

Критерій призупинення контролю якості було передвизначено не до повного призупиненні, а до призупинення конкретних модулів, адже немає потреби продовжувати тест модуля, які мають однакові шаблони, але аби не гаяти час необхідно продовжувати далі тестувати інші модулі що незалежні від цього, аби контроль якості був менш ресурсо-затратний та розтягнутий на більший проміжок часу якщо в тому немає потреби.

Критерій зупинки тестування (тобто завершення процесу після якого необхідно передавати результати клієнту) коли були здійснені тесткейси і всі дефекти були виправлені.

Вибір та складання артефактів на основі яких необхідно робити звіт визначались наступним чином. В першу чергу необхідно було створити юзкейси. Юзкейси мають в даному випадку комплексне призначення, адже вони виступають як вимогами від замовника, що вже будуть в задокументованому вигляд, так і поведінковими тесткейсами які одразу можна тестувати під час ознайомлення з системою. Наступний артефакт це макети сторінок, були розроблені загальні макети сторінок які зустрічаються у системі, аби перевірити наповненість сторінки необхідними елементами згідно вимог що так само стане частиною документації. Так само було прийнято рішення робити опис бази даних, адже база даних містить велику кількість таблиць та складності зв’язків, а це в будь-якому разі збільшує ймовірність виявлення дефектів. Також створення опису бази даних, допоможе ще більше проаналізувати систему, взаємодію сутностей та зв’язків між таблицями. І звісно ж базовий артефакт — це тесткейси. Тесткейси, які налічують сценарії різні для різних елементів, дій, форм,

поведінок системи, це те що допомагає задокументувати все перевірені дії належним чином, та у разі віднайдення дефектів оформлювати в дефект звіти та надавати розробникам для виправлення.

4.2. Звіт з контролю якості

В першу чергу до загального звіту з контролю якості мають бути включені усі артефакти що були розроблені впродовж здійснення контролю якості та неодмінно звіти з дефектами які були виявлені та які необхідно виправити.

Найперший дефект що зустрівся це дефекти при вході в систему за допомогою сторонніх сервісів таких як *google account*, *facebook*, *linkedIn* (рис. 4.1).

Дефект ID	ID	D-1A
	Назва	Вхід за допомогою google account
Огляд дефекту	Опис	Немає можливості входу в систему за допомогою google account
	Скріншот	googleAccountLoginIssue.png
Деталі дефекту	Кроки для відтворення	Перехід на сайт системи, натиснути кнопку що відповідає за вхід за допомогою google account
	Очікуваний результат	Користувач ввів облікові дані та зайшов в систему
	Фактичний результат	Користувач не зміг ввести свої облікові дани через помилку
Відстеження дефекту	Серйозність	Major
	Пріоритетність	Medium

Дефект ID	ID	D-2A
	Назва	Вхід за допомогою facebook
Огляд дефекту	Опис	Немає можливості входу в систему за допомогою facebook
	Скріншот	facebookLoginIssue.png
Деталі дефекту	Кроки для відтворення	Перехід на сайт системи, натиснути кнопку що відповідає за вхід за допомогою facebook
	Очікуваний результат	Користувач ввів облікові дані та зайшов в систему
	Фактичний результат	Користувач не зміг ввести свої облікові дани через помилку
Відстеження дефекту	Серйозність	Major
	Пріоритетність	Medium

Дефект ID	ID	D-3A
	Назва	Вхід за допомогою linkedIn
Огляд дефекту	Опис	Немає можливості входу в систему за допомогою linkedIn
	Скріншот	linkedInLoginIssue.png
Деталі дефекту	Кроки для відтворення	Перехід на сайт системи, натиснути кнопку що відповідає за вхід за допомогою facebook
	Очікуваний результат	Користувач ввів облікові дані та зайшов в систему
	Фактичний результат	Користувач не зміг ввести свої облікові дани через помилку
Відстеження дефекту	Серйозність	Major
	Пріоритетність	Medium

Рис. 4.1. Опис дефекту при вході в систему за допомогою сторонніх сервісів

Наступний дефект що був виявлений, це можливість надавати підрозділу водночас кількох типів (рис. 4.2).

Дефект ID	ID	D-1DIV
	Назва	Призначення одному підрозділу кількох типів
Огляд дефекту	Опис	Система дозволяє призначати одному і тому самому підрозділу набувати водночас різних типів
	Скріншот	divisionMultiTypesIssue.png
Деталі дефекту	Кроки для відтворення	Створити будь-який ресурс підрозділ, на основі цього підрозілу створити наприклад кафедру та факультет
	Очікуваний результат	Заборона на визначення підрозділу ще одного типу
	Фактичний результат	Підрозділ набув одночасно двох типів
Відстеження дефекту	Серйозність	Critical
	Пріоритетністи	High

Рис. 4.2. Опис дефекту призначення одному підрозділу кількох типів

Наступний дефект він загальний для різних модулів системи, адже застереження можуть відбуватись в різних модулях системи і було виявлено, що в якійсь частині функцій застереження відображаються належним чином із зрозумілим виглядом для користувача, але в різних випадках система просто падає і виводиться сторінка з кодом про помилку, що користувачу не потрібно знати (рис. 4.3).

Дефект ID	ID	D-2DIV
	Назва	Застереження неможливості створення ресурсу що вже має цей тип
Огляд дефекту	Опис	Застереження немає, користувач бачить просто код помилки
	Скріншот	divisionisameTypeWarning.png
Деталі дефекту	Кроки для відтворення	Створити будь-який ресурс підрозділ з певним типом, на основі цього підрозілу підрозділ такого самого типу
	Очікуваний результат	Підрозділ не створений та користувачу виведене застереження
	Фактичний результат	Підрозділ не створений та користувач отримав сторінку з кодом помилки що не інформативно для нього
Відстеження дефекту	Серйозність	Major
	Пріоритетністи	Enhancement

Рис. 4.3. Опис дефекту виведення помилки системи, а не застереження для користувача

Наступний розповсюджений дефект що зустрівся в дуже багатьох формах на створення — це проблеми з валідацією полів для створення різних ресурсів (рис. 4.4, 4.5, 4.6, 4.7).

Дефект ID	ID	D-0PEC_Y
	Назва	Валідація ідентифікатору спеціальності
Огляд дефекту	Опис	Поле ідентифікатору спеціальності може бути заповнене різними непередбачиними символами окрім цифр
	Скріншот	specialityIdIssue.png
Деталі дефекту	Кроки для відтворення	Перети на форму створення ресурсу спеціальність, заповнити поле ідентифікатору будь-якими символами окрім цифр, сторвити ресурс спецільності
	Очікуваний результат	Застереження користувачу, що ідентифікатор вказаний неправильно, або задорона на введення інших символів окрім цифр
	Фактичний результат	Жодних застережень, ресуср спеціальності створений
Відстеження дефекту	Серйозність	Major
	Пріоритетність	High

Рис. 4.4. Опис дефекту валідації ідентифікатора спецільності

Дефект ID	ID	D-0HR_P
	Назва	Валідація ідентифікаційного коду
Огляд дефекту	Опис	Поле ідентифікаційного коду персони може бути заповнене різними непередбачиними символами окрім цифр
	Скріншот	personIdIssue.png
Деталі дефекту	Кроки для відтворення	Перейти на форму створення ресурсу персона, заповнити поле ідентифікаційного коду будь-якими символами окрім цифр, сторвити ресурс персони
	Очікуваний результат	Застереження користувачу, що ідентифікаційний код вказаний неправильно, або заборона на введення інших символів окрім цифр
	Фактичний результат	Жодних застережень, ресуср спеціальності персона
Відстеження дефекту	Серйозність	Major
	Пріоритетність	High

Рис. 4.5. Опис дефекту валідації ідентифікаційного коду персони

Дефект ID	ID	D-1HR_P
	Назва	Валідація поштовогу індексу
Огляд дефекту	Опис	Поле поштового індексу персони може бути заповнене різними непередбачиними символами окрім цифр
	Скріншот	zipCodeIssue.png
Деталі дефекту	Кроки для відтворення	Перейти на форму створення ресурсу персона, заповнити поле поштовогу індексу будь-якими символами окрім цифр, сторвити/редагувати ресурс персони
	Очікуваний результат	Застереження користувачу, що поштовий індекс вказаний неправильно, або заборона на введення інших символів окрім цифр
	Фактичний результат	Жодних застережень, поштовий індекс невалідного формату був записаний в базу даних
Відстеження дефекту	Серйозність	Major
	Пріоритетність	High

Рис. 4.6. Опис дефекту валідації поштового індексу персони

Дефект ID	ID	D-2HR_P
	Назва	Валідація ідентифікатра Scopus
Огляд дефекту	Опис	Поле ідентифікатору Scopus персони може бути заповнене різними непередбачиними символами окрім цифр
	Скріншот	scopusIdIssue.png
Деталі дефекту	Кроки для відтворення	Перейти на форму ресурсу персона, заповнити поле ідентифікатора Scopus будь-якими символами окрім цифр, сторвити/редагувати ресурс персони
	Очікуваний результат	Застереження користувачу, що ідентифікатор вказаний неправильно, або заборона на введення інших символів окрім цифр
	Фактичний результат	Жодних застережень, ідентифікатор Scoups невалідного формату був записаний в базу даних
Відстеження дефекту	Серйозність	Major
	Пріоритетність	High

Рис. 4.7. Опис дефекту валідації ідентифікатору *Scopus*

Наступний дефект що був знайдений — це зайве поле що зберігається в таблиці, але не має там зберігатись (рис. 4.8).

Дефект ID	ID	D-0T_P
	Назва	Зайва колонка (honor_degree) в таблиці (persons)
Огляд дефекту	Опис	Таблиця де зберагається інформація про персон налічує зайву колонку, яка набуває завжди значення null так як в формі персони ніколи не заповнюється
	Скріншот	tablePersonsColumnHonorDegreeIssue.png
Деталі дефекту	Кроки для відтворення	Спробувати створити персону, та додати їй почесне звання, перейти в таблицю персон та подивитись на колонку honor_degree
	Очікуваний результат	Колонка набуде значення
	Фактичний результат	Колонка не набула ніякого значення, значення було записане в іншу таблицю, презначену спеціально для цієї іформації
Відстеження дефекту	Серйозність	Minor
	Пріоритетність	Medium

Рис. 4.8. Опис дефекту з зайвою колонкою в таблиці персон

Наступний дефект в базі даних що був віднайдений — це критичний дефект, адже в таблиці яка зберігає інформацію про авторів та публікації, зв’язки переплутані та посилаються на неправильні колонки в інших таблицях (рис. 4.9).

Дефект ID	ID	D-0T_S_P_A
	Назва	Переплутані зв'язки в таблиці (science_publications_authors)
Огляд дефекту	Опис	Таблиця де зберагається інформація про авторів та публікації, має переплутані зв'язки для колонок
	Скріншот	tablePersonsColumnHonorDegreeIssue.png
Деталі дефекту	Кроки для відтворення	Перейти в таблицю авторів та публікацій, зайти в папку ключів, передевитись який ключ на який посилається (author_id -> publication_id, publication_id -> athor_id)
	Очікуваний результат	(author_id -> person_id, publication_id -> publication_id)
	Фактичний результат	author_id -> publication_id, publication_id -> athor_id
Відстеження дефекту	Серйозність	Critical
	Пріоритетність	High

Рис. 4.9. Опис дефекту зв’язків в таблиці авторів та їх публікацій

Для представлення відповідності системи вимогам, загальні результати функціональних можливостей системи були оформлені у вигляді матриці простежування (рис. 4.10).

Вимога	Тесткейси												
UI_MP	TC-0MP	TC-1MP	TC-2MP	TC-3MP	TC-4MP	TC-5MP	TC-6MP	TC-7MP	TC-8MP	TC-9MP	TC-10MP	TC-11MP	TC-12MP
UI_RP	TC-0RP	TC-1RP	TC-2RP	TC-3RP	TC-4RP	TC-5RP	TC-6RP	TC-7RP	TC-8RP	TC-9RP	TC-10RP	TC-11RP	
UI_ARP	TC-0ARP	TC-1ARP	TC-2ARP	TC-3ARP	TC-4ARP	TC-5ARP	TC-6ARP	TC-7ARP	TC-8ARP	TC-9ARP	TC-10ARP		
UI_ERP	TC-0ERP	TC-1ERP	TC-2ERP	TC-3ERP	TC-4ERP	TC-5ERP	TC-6ERP	TC-7ERP	TC-8ERP	TC-9ERP	TC-10ERP	TC-11ERP	
UI_SRP	TC-0ERP	TC-1ERP	TC-2ERP	TC-3ERP	TC-4ERP	TC-5ERP	TC-6ERP	TC-7ERP					
UC-HRP	TC-0HRP	TC-1HRP	TC-2HRP	TC-3HRP	TC-4HRP	TC-5HRP	TC-6HRP	TC-7HRP	TC-8HRP	TC-9HRP	TC-10HRP	TC-11HRP	TC-12HRP
UC-HRT	TC-0HRT	TC-1HRT	TC-2HRT	TC-3HRT	TC-4HRT	TC-5HRT	TC-6HRT	TC-7HRT	TC-8HRT	TC-9HRT	TC-10HRT	TC-11HRT	
UC-HRS	TC-0HRS	TC-1HRS	TC-2HRS	TC-3HRS	TC-4HRS	TC-5HRS							
UC-HRJP	TC-0HRJP	TC-1HRJP	TC-2HRJP	TC-3HRJP	TC-4HRJP	TC-5HRJP	TC-6HRJP						
UC-HRAD	TC-0HRAD	TC-1HRAD	TC-2HRAD	TC-3HRAD	TC-4HRAD	TC-5HRAD	TC-6HRAD	TC-7HRAD	TC-8HRAD	TC-9HRAD			

Рис 4.10. Приклад частини матриці простежування функціональних вимог

ВИСНОВКИ ДО РОЗДІЛУ 4

В даному розділі було проаналізовано здійснення аналізу продукту, аналіз побудови плану, описано що було досягнуто під час здійснення контролю якості інформаційної системи керування навчальним закладом вищої освіти.

В загальному звіті було подано детальні описи дефектів що необхідно виправити. Кожен дефект має свої наслідки, якісь з них були критичні, якісь несуттєві, але звісно ж мають бути виправлені всі без винятку, але порядок виконання буде залежати від критичності. Всі функціональні вимоги що були визначені в системі, були розроблені, та відповідають юзкейсам, окрім деяких юзкейсів на здійснення входу в систему за допомогою сторонніх сервісів. З точки зору роботи функцій були знайдені дефекти. Основна частина дефектів що була знайдена стосується саме валідації даних для форм, обробки застережень для користувача. Також було звернено увагу, та зазначено в звіті, що неоднозначності під час редагування певних ресурсів, в особливості що деякі ресурси можуть мати множинні типи, їх можна перевизначати, але якщо мають зв’язки з іншими ресурсами і лежать в основі створення інших, то видалити такі ресурси не можна. З точки зору дизайну всі макети виконані в повній мірі, в одній структурі сторінок було знайдено зайві елементи, але вони не є критичними та можуть продовжувати залишатись на сторінці, всі решта сторінок відповідають структурним макетам. За допомогою авто-тестів було перевірено веб-елементи сторінок та правильність поведінки, всі веб-елементи сторінки справні та мають правильну поведінку. З точки зору зручності система є досить простою для взаємодії, однак існує декілька нюансів на які варто звернути увагу, такі як редагування одного ресурсу зі сторінки іншого, це є досить заплутано і не зрозуміло, адже при створенні якогось ресурсу може довестись створити допоміжний для нього ресурс і при цьому відкриється безліч вікон що накладатимуться одне на одного. З точки зору сумісності веб-додатку, веб-додаток сумісний з усіма стандартними браузерами, все відображається коректно і більше того, не псується при зменшенні чи

збільшенні масштабів вікна, а також веб-додатком вільно можна користуватись на телефоні, адже продумана верстка не тільки для ПК чи ноутбуків.

ВИСНОВКИ

Магістерська дисертація присвячена розробці засобів для контролю якості інформаційної системи управління навчальним закладом вищої освіти, вдосконаленого набору артефактів тестування для виявлення дефектів на різних рівнях оцінки якості, що сприятиме підвищенню ефективності функціонування інформаційної системи та забезпечуватиме необхідний рівень якості системи.

- Аналіз життєвого циклу інформаційних систем та огляд різних технік тестування обґрунтували проблематику забезпечення якості інформаційних систем та визначили завдання магістерської дисертації.
- Були розроблені індивідуальні план та стратегія тестування безпосередньо для об’єкту контролю якості (інформаційної системи керування закладом вищої освіти). Були використані відомі види тестування такі як функціональне, нефункціональне, але в унікальній комбінації із застосуванням та розробкою артефактів в нестандартному та нетиповому ключі для контролю якості. Застосування запропонованого підходу дозволило зменшити час тестування та кількість дефектів у різних рівнях та структурних модулях готового продукту і в результаті — забезпечити гарантії якості інформаційної системи.
- Розроблений спосіб включає застосування функціональних тестів (модульне, інтеграційне, системне) та нефункціональні (зручності, сумісності, навантаження). Сукупність цих технік дає здійснити ефективно контроль якості в особливості інформаційних систем керування навчальними закладами вищої освіти через їх складну архітектуру та щільну зв’язність модулів. Були включені потужні артефакти для тестування, що застосовуються в незвичний спосіб та виконують одразу кілька функцій в процесі контролю якості, що дає здійснити контроль якості швидше, не витрачати час на зайві перевірки яких можна уникнути, охопити більше компонент системи та з різних сторін.

- Контроль якості та тестування інформаційної системи «KPI Connect» визначили, що система придатна до використання. Було надано загальний звіти щодо того на що потрібно звернути увагу аби покращити систему. В результаті контролю якості були виявлені наступні критичні недоліки, які можуть мати значні негативні наслідки: помилки в базі даних, обробка помилок зі сторони користувацького інтерфейсу і необхідність валідації полів, аби дані були правильно та в належному вигляді збережені та аби допускалось менше помилок під час користування та створення ресурсів системи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Martin M. What is MIS? Introduction & Definition [Електронний ресурс] / Matthew Martin. – 2023. – Режим доступу до ресурсу: https://www.guru99.com/mis-definition.html
2. Classifications of information systems and their components [Електронний ресурс] // KNOWCOMPUTING – Режим доступу до ресурсу: https://www.knowcomputing.com/what-are-types-and-components-of-information-system/
3. Reddy G. Most Important Task in Software [Електронний ресурс] / G C Reddy. – 2020. – Режим доступу до ресурсу: https://www.gcreddy.com/2020/08/most-important-task-in-software-testing.html
4. Team Management – Meaning and Importance [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://emeritus.org/in/learn/team-management-meaning-and-importance/
5. Kumar Jena B. What Is Software Security and What Makes It So Important Now? [Електронний ресурс] / Baivab Kumar Jena. – 2022. – Режим доступу до ресурсу: https://www.simplilearn.com/tutorials/cyber-security-tutorial/what-is-software-security
6. Markgraf B. Characteristics of a Good Management Information System [Електронний ресурс] / Bert Markgraf. – 2019. – Режим доступу до ресурсу: https://smallbusiness.chron.com/characteristics-good-management-information-system-59060.html
7. Alexandra. What Is SDLC? Understand the Software Development Life Cycle [Електронний ресурс] / Alexandra. – 2023. – Режим доступу до ресурсу: https://stackify.com/what-is-sdlc/
8. V J. How the software development life cycle works [Електронний ресурс] / JANANI V. – 2021. – Режим доступу до ресурсу: https://www.linkedin.com/pulse/how-software-development-life-cycle-works-janani-v

9. Lutkevich B. DEFINITION waterfall model [Електронний ресурс] / B. Lutkevich, S. Lewis. – 2022. – Режим доступу до ресурсу: https://www.techtarget.com/searchsoftwarequality/definition/waterfall-model
10. SDLC Iterative Model [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.in/sdlc/iterative-model
11. KUMAR PAL S. Software Engineering | Spiral Model [Електронний ресурс] / SAYAN KUMAR PAL. – 2023. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/software-engineering-spiral-model/
12. Martin M. Incremental Model in SDLC: Use, Advantage & Disadvantage [Електронний ресурс] / Matthew Martin. – 2023. – Режим доступу до ресурсу: https://www.guru99.com/what-is-incremental-model-in-sdlc-advantages-disadvantages.html
13. SDLC - V-Model [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/sdlc/sdlc_v_model.htm
14. Error vs Defect vs Failure [Електронний ресурс] – Режим доступу до ресурсу: https://tuskr.app/learn/error-defect-failure
15. Lutkevich B. bug [Електронний ресурс] / Ben Lutkevich. – 2021. – Режим доступу до ресурсу: https://www.techtarget.com/searchsoftwarequality/definition/bug
16. Agarwal R. Techniques To Prevent Software Bugs [Електронний ресурс] / Richa Agarwal. – 2022. – Режим доступу до ресурсу: https://testsigma.com/blog/techniques-to-prevent-software-bugs/
17. What Is Defect/Bug Life Cycle In Software Testing? Defect Life Cycle Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.softwaretestinghelp.com/bug-life-cycle/
18. Hamilton T. What is Software Testing? Definition [Електронний ресурс] / Thomas Hamilton. – 2023. – Режим доступу до ресурсу: https://www.guru99.com/software-testing-introduction-importance.html

19. Hamilton T. 7 Principles of Software Testing with Examples [Електронний ресурс] / Thomas Hamilton. – 2023. – Режим доступу до ресурсу: https://www.guru99.com/software-testing-seven-principles.html
20. Hamilton T. What is Automation Testing? Test Tutorial [Електронний ресурс] / Thomas Hamilton. – 2023. – Режим доступу до ресурсу: https://www.guru99.com/automation-testing.html
21. Ishita. Types of Testing Techniques: Black, White and Grey Box [Електронний ресурс] / Ishita. – 2022. – Режим доступу до ресурсу: https://securityboulevard.com/2022/08/types-of-testing-techniques-black-white-and-grey-box/
22. Kinsbruner E. What Is Non-Functional Testing? [Електронний ресурс] / Eran Kinsbruner. – 2023. – Режим доступу до ресурсу: https://www.perfecto.io/blog/what-is-non-functional-testing#:~:text=The%20difference%20between%20functional%20testing,how%20well%20the%20application%20works
23. What is functional testing? Definition, types & examples [Електронний ресурс] – Режим доступу до ресурсу: https://katalon.com/resources-center/blog/functional-testing
24. Integration Testing: Types, Approaches, Tools, Examples [Електронний ресурс] // KnowledgeHut. – 2023. – Режим доступу до ресурсу: https://www.knowledgehut.com/blog/software-testing/integration-testing
25. Gillis A. performance testing [Електронний ресурс] / Alexander Gillis. – 2023. – Режим доступу до ресурсу: https://www.techtarget.com/searchsoftwarequality/definition/performance-testing
26. Suhani J. What is Non-Functional Testing? [Електронний ресурс] / Jain Suhani. – 2023. – Режим доступу до ресурсу: https://www.browserstack.com/guide/what-is-non-functional-testing

27. Introduction to Software Engineering/Quality/Code Review [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Quality/Code_Review
28. Software Requirements Document: Definition, Steps and Template Included! [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://blog.bit.ai/software-requirements-document/
29. Hamilton T. TEST PLAN: What is, How to Create [Електронний ресурс] / Thomas Hamilton. – 2023. – Режим доступу до ресурсу: https://www.guru99.com/what-everybody-ought-to-know-about-test-planing.html
30. Brush K. use case [Електронний ресурс] / Kate Brush. – 2022. – Режим доступу до ресурсу: https://www.techtarget.com/searchsoftwarequality/definition/use-case
31. Bose S. How to write Test Cases? [Електронний ресурс] / Shreya Bose. – 2023. – Режим доступу до ресурсу: https://www.browserstack.com/guide/how-to-write-test-cases
32. Serguts A. How to Write a Bug Report That Will Make Your Engineers Love You [Електронний ресурс] / Anna Serguts. – 2023. – Режим доступу до ресурсу: https://orangesoft.co/blog/how-to-write-a-bug-report
33. The QA Software Testing Checklists [Електронний ресурс] – Режим доступу до ресурсу: https://www.softwaretestinghelp.com/software-testing-qa-checklists/

ДОДАТОК А

```
package tests;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;
public class LogIn {
    WebDriver driver;
    @BeforeMethod
    public void setUp() {
        System.setProperty("webdriver.chrome.driver", "E:\\selenium-
drivers\\chromedriver.exe");
        driver = new ChromeDriver();
        driver.get("https://admin.kpi-connect.ugrid.org/auth/login");
    }
    @AfterMethod
    public void tearDown() {
        driver.quit();
    }
    @Test
    public void testLoginWrongCreds() {
        WebElement emailInput = driver.findElement(By.name("_username"));
        emailInput.sendKeys("wrongEmail@gmail.com");
        WebElement passwordInput = driver.findElement(By.name("_password"));
        passwordInput.sendKeys("WrongPassword");
```

```
        WebElement signInButton = driver.findElement(By.cssSelector(".btn.btn-
danger.btn-block.btn-round"));
        signInButton.click();
        WebElement authHeader = driver.findElement(By.tagName("h3"));
        String expectedHeader = "Welcome";
        String actualHeader = authHeader.getText();
        assert(expectedHeader.equals(actualHeader));
    }
    @Test
    public void testLoginWrongPass() {
        WebElement emailInput = driver.findElement(By.name("_username"));
        emailInput.sendKeys("admin@gmail.com");
        WebElement passwordInput = driver.findElement(By.name("_password"));
        passwordInput.sendKeys("WrongPassword");
        WebElement signInButton = driver.findElement(By.cssSelector(".btn.btn-
danger.btn-block.btn-round"));
        signInButton.click();
        WebElement authHeader = driver.findElement(By.tagName("h3"));
        String expectedHeader = "Welcome";
        String actualHeader = authHeader.getText();
        assert(expectedHeader.equals(actualHeader));
    }
    @Test
    public void testLoginCorrectCreds() {
        WebElement emailInput = driver.findElement(By.name("_username"));
        emailInput.sendKeys("admi@gmail.com");
        WebElement passwordInput = driver.findElement(By.name("_password"));
        passwordInput.sendKeys("CorrectPass");
```

```
        WebElement signInButton = driver.findElement(By.cssSelector(".btn.btn-
danger.btn-block.btn-round"));
        signInButton.click();
        WebElement dashboardHref =
driver.findElement(By.cssSelector("[href='/admin/dashboard']"));
        String expectedHref = "https://admin.kpi-connect.ugrid.org/admin/dashboard";
        String actualHref = dashboardHref.getAttribute("href");
        assert(expectedHref.equals(actualHref));
    }
}
package tests.topNavBar;
  @AfterMethod
    public void tearDown() {
        driver.navigate().refresh();
    }
    @AfterTest
    public void end() {
        driver.quit();
    }
    @Test(priority = 1)
    public void testButtonDropdownLanguage() {
        WebElement profileDropdown =
driver.findElement(By.xpath("/html/body/div/header/nav/div[2]/ul/li[4]/a"));
        profileDropdown.click();
        WebElement openedMenu =
driver.findElement(By.cssSelector(".dropdown.lang-menu.open"));
        assert(openedMenu.isDisplayed());
    }
```

```
    @Test(priority = 2)
    public void testEnglish() {
        WebElement profileDropdown =
driver.findElement(By.xpath("/html/body/div/header/nav/div[2]/ul/li[4]/a"));
        profileDropdown.click();
        WebElement logoutButton =
driver.findElement(By.cssSelector("[href='/admin/lang/en']"));
        logoutButton.click();
        WebElement title = driver.findElement(By.tagName("h3"));
        String expectedTitle = "University structure";
        String actualTitle = title.getText();
        assert(expectedTitle.equals(actualTitle));
    }
    @Test(priority = 2)
    public void tesUkrainian() {
        WebElement profileDropdown =
driver.findElement(By.xpath("/html/body/div/header/nav/div[2]/ul/li[4]/a"));
        profileDropdown.click();
        WebElement logoutButton =
driver.findElement(By.cssSelector("[href='/admin/lang/uk']"));
        logoutButton.click();
        WebElement title = driver.findElement(By.tagName("h3"));
        String expectedTitle = "Структура університету";
        String actualTitle = title.getText();
        assert(expectedTitle.equals(actualTitle));
    }
 @Test(priority = 1)
    public void testButtonDropdownUser() {
```

```
        WebElement profileDropdown =
driver.findElement(By.xpath("/html/body/div/header/nav/div[2]/ul/li[3]/a"));
        profileDropdown.click();
        WebElement openedMenu =
driver.findElement(By.cssSelector(".dropdown.user-menu.open"));
        assert(openedMenu.isDisplayed());
    }
    @Test(priority = 2)
    public void testLogOut() {
        WebElement logoutButton =
driver.findElement(By.xpath("/html/body/div/header/nav/div[2]/ul/li[3]/ul/li[4]/ul/li/a
"));
        logoutButton.click();
        WebElement startButton =
driver.findElement(By.cssSelector("[href='/auth/login']"));
        String expectedButtonText = "START NOW";
        String actualButtonText = startButton.getText();
        assert(expectedButtonText.equals(actualButtonText));
    }
  @Test
    public void testHome() {
        WebElement languageDropdown =
driver.findElement(By.xpath("/html/body/div/header/nav/div[2]/ul/li[1]/a"));
        languageDropdown.click();
        WebElement button =
driver.findElement(By.cssSelector("[href='/admin/dashboard']"));
        String expectedButtonText = "GO TO CABINET";
        String actualButtonText = button.getText();
        assert(expectedButtonText.equals(actualButtonText));
```

```
    }
 @Test
    public void testPerson() {
        WebElement pDropdown =
driver.findElement(By.cssSelector("[href='/admin/persons/list']"));
        pDropdown.click();
        WebElement pageTitle = driver.findElement(By.cssSelector(".nav.navbar-top-
links.breadcrumb .active"));
        String expectedPageTitle = "Person list";
        String actualPageTitle = pageTitle.getText();
        assert(expectedPageTitle.equals(actualPageTitle));
    }
    @Test
    public void testTeacher() {
        WebElement tDropdown =
driver.findElement(By.cssSelector("[href='/admin/teachers/list']"));
        tDropdown.click();
        WebElement pageTitle = driver.findElement(By.cssSelector(".nav.navbar-top-
links.breadcrumb .active"));
        String expectedPageTitle = "Teacher list";
        String actualPageTitle = pageTitle.getText();
        assert(expectedPageTitle.equals(actualPageTitle));
    }
    @Test
    public void testJobPosition() {
        WebElement jpDropdown =
driver.findElement(By.cssSelector("[href='/admin/job_positions/list']"));
        jpDropdown.click();
```

```
        WebElement pageTitle = driver.findElement(By.cssSelector(".nav.navbar-top-
links.breadcrumb .active"));
        String expectedPageTitle = "Job position list";
        String actualPageTitle = pageTitle.getText();
        assert(expectedPageTitle.equals(actualPageTitle));
    }
    @Test
    public void testAcademicDegree() {
        driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);
        WebElement adDropdown =
driver.findElement(By.cssSelector("[href='/admin/academic_degrees/list']"));
        adDropdown.click();

        WebElement pageTitle = driver.findElement(By.cssSelector(".nav.navbar-top-
links.breadcrumb .active"));
        String expectedPageTitle = "Academic degree list";
        String actualPageTitle = pageTitle.getText();
        assert(expectedPageTitle.equals(actualPageTitle));
    }
    @Test
    public void testScienceDegree() {
        driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);
        WebElement sdDropdown =
driver.findElement(By.cssSelector("[href='/admin/science_degrees/list']"));
        sdDropdown.click();
        WebElement pageTitle = driver.findElement(By.cssSelector(".nav.navbar-top-
links.breadcrumb .active"));
        String expectedPageTitle = "Science degree list";
        String actualPageTitle = pageTitle.getText();
```

```
        assert(expectedPageTitle.equals(actualPageTitle));
    }
    @Test
    public void testHonorDegree() {
        driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);
        WebElement hdDropdown =
driver.findElement(By.cssSelector("[href='/admin/honor_degrees/list']"));
        hdDropdown.click();
        WebElement pageTitle = driver.findElement(By.cssSelector(".nav.navbar-top-
links.breadcrumb .active"));
        String expectedPageTitle = "Honor degree list";
        String actualPageTitle = pageTitle.getText();
        assert(expectedPageTitle.equals(actualPageTitle));
    }
    @Test
    public void testWorkDirection() {
        driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);
        WebElement wdDropdown =
driver.findElement(By.cssSelector("[href='/admin/work_directions/list']"));
        wdDropdown.click();
        WebElement pageTitle = driver.findElement(By.cssSelector(".nav.navbar-top-
links.breadcrumb .active"));
        String expectedPageTitle = "Work direction list";
        String actualPageTitle = pageTitle.getText();
        assert(expectedPageTitle.equals(actualPageTitle));
    }
    @Test
    public void testEducation() {
        driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);
```

```
        WebElement eDropdown =
driver.findElement(By.cssSelector("[href='/admin/educations/list']"));
        eDropdown.click();
        WebElement pageTitle = driver.findElement(By.cssSelector(".nav.navbar-top-
links.breadcrumb .active"));
        String expectedPageTitle = "Education list";
        String actualPageTitle = pageTitle.getText();
        assert(expectedPageTitle.equals(actualPageTitle));
    }
    @Test
    public void testPersonDivisionWorkDirection() {
        driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);
        WebElement pdvdDropdown =
driver.findElement(By.cssSelector("[href='/admin/personDivisionWorkDirections/list'
]"));
        pdvdDropdown.click();
        WebElement pageTitle = driver.findElement(By.cssSelector(".nav.navbar-top-
links.breadcrumb .active"));
        String expectedPageTitle = "Person division work directions list";
        String actualPageTitle = pageTitle.getText();
        assert(expectedPageTitle.equals(actualPageTitle));
    }

    @Test
    public void testPersonDivisionJobPosition() {
        driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);
        WebElement pdjpDropdown =
driver.findElement(By.cssSelector("[href='/admin/personDivisionJobPositions/list']")
);
```

```
        pdjpDropdown.click();
        WebElement pageTitle = driver.findElement(By.cssSelector(".nav.navbar-top-
links.breadcrumb .active"));
        String expectedPageTitle = "Person division job positions list";
        String actualPageTitle = pageTitle.getText();
        assert(expectedPageTitle.equals(actualPageTitle));
    }
    @Test
    public void testPersonalHonorDegree() {
        driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);
        WebElement phdDropdown =
driver.findElement(By.cssSelector("[href='/admin/personalHonorDegrees/list']"));
        phdDropdown.click();
        WebElement pageTitle = driver.findElement(By.cssSelector(".nav.navbar-top-
links.breadcrumb .active"));
        String expectedPageTitle = "Personal honor degree list";
        String actualPageTitle = pageTitle.getText();
        assert(expectedPageTitle.equals(actualPageTitle));
    }
```