ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
"УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТА КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра комп'ютерних наук

О.Ю. Мушинський, К.В. Тимофєєва, І. А Старчик

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО ВИКОНАННЯ ПРОЄКТУ ДРУГОГО РІВНЯ

здобувачами вищої освіти першого (бакалаврського) рівня вищої освіти
галузі знань 12 «Інформаційні технології»
за спеціальністю 122 «Комп'ютерні науки»

Київ – 2025

УДК 378. 147:004.4(072)

Розглянуто і затверджено на засіданні кафедри комп'ютерних наук, протокол № 10 від 29 квітня 2025 р.

Методичні рекомендації до виконання Проєкту другого рівня для здобувачів ступеня вищої освіти «бакалавр» галузі знань 12 (Інформаційні технології) спеціальності 122 (Комп'ютерні науки) / уклад. О.Ю. Мушинський, К.В. Тимофєєва, І. А Старчик. Київ, Університет «КРОК». 2025. 42 с.

Рецензенти: *Мічківський Сергій Миколайович*, кандидат економічних наук, доцент, старший науковий співробітник, директор навчально-наукового інституту інформаційних та комунікаційних технологій, завідувач кафедри комп'ютерних наук Університету економіки та права "КРОК";
Потапенко Едвард Сергійович, директор центру автоматизації та розробки департаменту цифровізації Університету економіки та права "КРОК".

Методичні рекомендації щодо виконання проєкту другого рівня розроблено для здобувачів вищої освіти спеціальності 122 «Комп'ютерні науки» відповідно до навчального плану другого курсу (4 семестр). Метою виконання проєкту є проєктування та реалізація програмного забезпечення із використанням процедурної та об'єктно-орієнтованої парадигм програмування, методів і алгоритмів обчислень, структур даних та механізмів управління. Робота над проєктом здійснюється на основі наукових доробок наукової школи «Управління інноваційним розвитком соціально-економічних систем в епоху економіки знань» (VARIORUM) та методичних матеріалів «Всеукраїнських освітніх курсів-стажування», розроблених компанією «Genesis».

ЗМІСТ

ВСТУП

Методичні рекомендації до виконання Проєкту другого рівня призначені для здобувачів вищої освіти спеціальності 122 «Комп'ютерні науки» освітнього рівня бакалавр. Документ розроблено відповідно до «Положення про практичну підготовку здобувачів вищої освіти Університету «КРОК», затвердженого рішенням Вченої Ради Університету (протокол № 4 від 07 квітня 2020 року), Закону України «Про вищу освіту», а також навчального плану освітньої програми «Комп'ютерні науки».

Практична підготовка здобувачів за напрямком Комп'ютерні науки» на другого курсі (4 семестр) здійснюється у вигляді проєктної практики. Вона передбачає встановлення відповідності засвоєних здобувачами рівня та обсягу засвоєних знань, умінь і навичок, необхідних для реалізації проєктів з розробки програмного забезпечення.

Метою проєктної практики є поглиблення практичних навичок здобувачів у сфері проєктування та створення програмного забезпечення, зокрема шляхом:

- застосування проєктного підходу та командної роботи для створення ІТ-продуктів;
- використання сучасних методологій розробки програмного забезпечення та технологічних інструментів;
- інтеграція знань про парадигми програмування, алгоритми, структури даних та моделі управління в реальних умовах розробки.

У процесі виконання Проєкту другого рівня здобувачі мають продемонструвати здатність:

1. до командної роботи при розробці програмного забезпечення протягом його життєвого циклу;
2. застосовувати проєктний підхід для виконання комплексних, взаємопов’язаних завдань в умовах обмеженості ресурсів, тимчасовості та невизначеності, з метою створення якісного програмного продукту;

3. проєктувати та розробляти програмне забезпечення із застосуванням різних парадигм програмування: узагальненого, об’єктно-орієнтованого, функціонального, логічного, з відповідними алгоритмами обчислень та структурами даних;

4. аналізувати предметні області різних галузей та формалізувати функціональні та нефункціональні вимоги;

5. використовувати UML-нотації для створення діаграм та опису архітектури програмного продукту;

6. проводити автоматизоване та ручне тестування компонентів програмного забезпечення.

Терміни виконання та захисту Проєкту визначаються згідно з графіком освітнього процесу.

Найкращі Проєкти здобувачів оприлюднюються на офіційному репозиторії відкритого доступу Університету економіки та права «КРОК».

1. ЗАГАЛЬНІ ТА КОНЦЕПТУАЛЬНІ ПОЛОЖЕННЯ З ВИКОНАННЯ ПРОЄКТУ

1.1 Загальні положення

Проєктна практика є невід'ємною складовою практичної підготовки здобувачів вищої освіти за освітньою програмою «Комп'ютерні науки» першого (бакалаврського) рівня. Особливістю Проєкту другого рівня є інтеграція проєктного підходу та командної роботи, що дозволяє студентам моделювати робочі процеси, наближені до сучасних прикладних практик ІТ-сфери.

Проєкт – це спеціально організована форма самостійної діяльності здобувачів, спрямована на вирішення певної практичної проблеми, оформлене у вигляді кінцевого програмного продукту, який можна побачити, осмислити та застосувати в реальних умовах.

Програмним продуктом вважається рішення, що:

- відповідає конкретному запиту, потребі або «болю» користувачів;
- має чітку технологічну реалізацію;
- створене за допомогою методів інформаційних технологій.

Проєкти другого рівня різняться залежно від мети, змісту робіт та очікуваних результатів і можуть виконуватися у форматі прикладного проєкту або сервісного проєкту. Сервісний проєкт спрямований на розробку ІТ-рішення для вирішення службових або організаційних завдань структурних підрозділів Університету (наприклад, ННІ інформаційних та комунікаційних технологій, Центр інформаційних технологій або Центр Автоматизації та Розробки). Ініціатором сервісного проєкту можуть виступати працівники Університету, представники ІТ-компаній та інші зацікавлені сторони. Ініціатор проєкту формує проєктну заявку (Додаток А) та узгоджує її з куратором проєктів.

У прикладному проєкті основною метою є створення повноцінного програмного продукту (додатку, вебсайту, прототипу системи, тощо) за

ініціативою здобувачів, які формують проблему, опис продукту (Додаток Б), мету та умови виконання проєкту.

Проєкти можуть виконуватися у командному форматі (рекомендовано) або індивідуально (лише за погодженням із керівником проєкту).

У разі командної роботи необхідно дотримуватись наступних вимог:

– кількість учасників не повинна перевищувати п’яти осіб;
– завдання мають бути чітко розподілені між учасниками;
– результати роботи здобувачів оцінюються разом.

Особливості реалізації проєкту та підготовки звіту з Проєкту описано у наступних розділах. Назва командного проєкту єдина для всіх учасників однієї проєктної команди.

У реалізації Проєкту другого рівня беруть участь такі основні ролі:

1. Команда проєкту (Виконавець) – здобувачі, які безпосередньо реалізують проєкт.

2. Керівник проєкту – відповідальна особа, яка забезпечує організацію та контроль за виконанням проєкту, оцінювання учасників проєкту, консультування щодо змісту проєкту та оформлення звітної документації. Відповідальність за перевірку проєкту, допуск до захисту та дотримання здобувачами академічної доброчесності несе Керівник Проєкту.

3. Куратор проєкту – це гарант освітньої програми або призначений ним науково-педагогічний працівник, який відповідальний за стратегічний та методичний супровід і координацію проєктної діяльності здобувачів. Його основні обов’язки включають:

– підготовка та оновлення методичних матеріалів;
– узгодження звітних документів за проєктами здобувачів;
– формування команд проєктів;
– моніторинг ходу виконання проєктів.

4. Координатор команди – це додаткова роль, яку може виконувати здобувач другого (магістерського) рівня або студент старших курсів бакалаврату.

Основні завдання координатора:

– організація взаємодії всередині команди;

– контроль за дотриманням графіка реалізації проєкту;

– узгодження та комунікацію з керівником проєкту щодо завдань;

– забезпечення належного оформлення проєктної документації.

Координатор команди не здійснює оцінювання і не приймає рішень щодо допуску до захисту.

5. Комісія з захисту проєкту – проводить захист проєктів та оцінює роботу здобувачів (детальніше в розділі 3). Склад комісії затверджується завідувачем кафедри. Комісія може рекомендувати публікацію результатів Проєкту на офіційному репозиторії відкритого доступу Університету економіки та права «КРОК», на конференціях або на інших ресурсах Університету.

Проєкт є завершальним етапом другого року навчання та має на меті систематизацію, закріплення і розширення теоретичних знань, умінь і навичок здобувачів, а також перевірку їхньої здатності до практичного застосування отриманих компетентностей у вирішенні професійних завдань. Встановлені параметри проєкту визначають його обсяг, формат виконання, склад команди, підхід до управління та перелік обов'язкових результатів (таблиця 1.1).

Таблиця 1.1 – Зміст Проєкту II рівня

Обсяг	270 годин/9 кредитів ECTS
Виконання	командне (можливе індивідуальне виконання при попередньому погодженні з Керівником Проєкту)
Команда	- Менеджер проєкту - Фронтенд-розробник - Бекенд-розробник - Дизайнер - Аналітик
Управління	V-модель та проєктний підхід
Обов'язкові складові результату виконання проєкту	- реліз програмного забезпечення; - звіт з проєкту; - презентація продукту; - спланований сценарій демонстрації роботи програмного забезпечення.

1.2 Теоретичні та концептуальні положення з виконання проєкту

1.2.1 Проєктний підхід

У сучасній практиці створення програмних продуктів одним із найефективніших методів є проєктний підхід. Він передбачає цілеспрямовану, структуровану та адаптивну організацію діяльності, яка забезпечує досягнення унікального результату з визначеною цінністю та якісними характеристиками, у межах обмежень часу, бюджету та ресурсів.

Кожен проєкт має відрізні ознаки:

– унікальність результату – продукт або рішення, створене вперше в конкретному контексті;

– тимчасовість – проєкт має чітко визначений початок і кінець;

– обмеженість ресурсів – бюджет, час, людські й технічні ресурси;

– невизначеність – ситуації, що виникають у процесі реалізації;

– одноразовість – результат створюється в рамках конкретної ініціативи.

Проєктний підхід реалізується через життєвий цикл проєкту (рис. 1.1), що включає наступні фази:

1. Ініціація – аналіз проблемної ситуації, формулювання ідеї проєкту, постановка мети, визначення очікуваних результатів, оцінка доцільності.

2. Планування – деталізація цілей проєкту, постановка завдань, розподіл ролей, створення графіка виконання, оцінка ресурсів та ризиків.

3. Реалізація – виконання запланованих робіт, створення проміжних і фінальних продуктів, комунікація в команді та з зацікавленими сторонами.

4. Завершення/Впровадження – перевірка досягнутих результатів, оцінка якості, передача створеного продукту у використання, оформлення звітності, фіксація набутого досвіду [1].

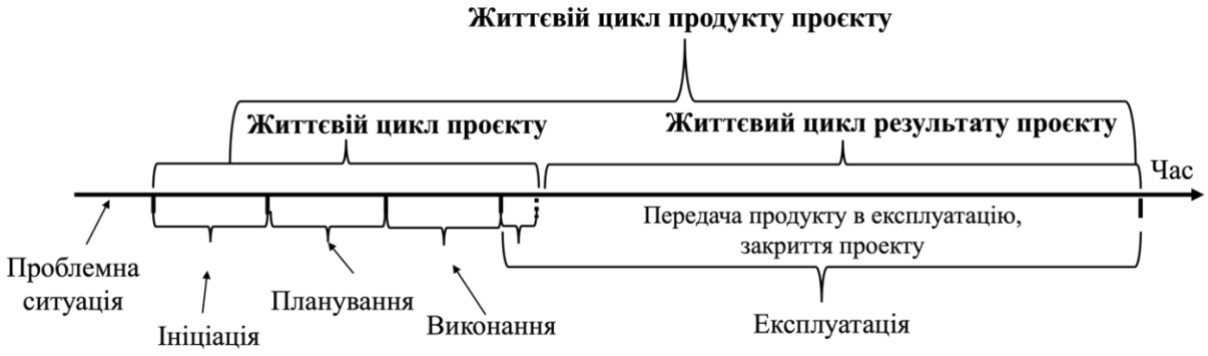

Рисунок 1.1 – Життєвий цикл продукту проєкту

В проєктному підході важливим є розуміння обмежень, в межах яких функціонує проєктна команда. Будь-який проєкт реалізується не в ідеальних умовах, а з урахуванням обмеженого часу, бюджету, доступних ресурсів і технічних та організаційних умов. Загальноприйнятим інструментом балансування між ключовими обмеженнями проєкту є трикутник управління проєктом (рис 1.2) (Project Management Triangle) [2].

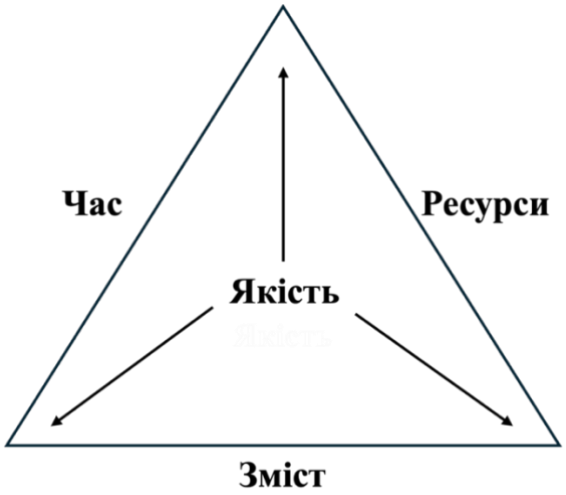

Рисунок 1.2 – Трикутник управління проєктом

Трикутник управління проєктом, або «трикутник обмежень» - це концептуальна модель, яка використовується командами проєктів для балансування між трьома основними змінними:

- час – скільки триватиме реалізація проєкту, які дедлайни поставлені, які етапи мають бути виконані у визначені терміни;

- вартість – бюджет проєкту, фінансові, людські та технічні ресурси, що доступні команді;
- зміст – обсяг робіт, функціональні й нефункціональні вимоги, те, що має бути реалізоване як результат.

Ці три елементи перебувають у взаємозалежності: зміна одного з них впливає на інші. Наприклад, якщо розширюється зміст проєкту (додаються нові функції), то, як правило, потрібне або більше часу, або більше ресурсів.

1.2.2 Процес розробки програмного забезпечення

Процес розробки програмного забезпечення це технічні дії які застосовуються протягом усього життєвого циклу програмного продукту для перетворення потреб і вимог зацікавлених сторін на ефективний, надійний та життєздатний продукт або послугу. До технічний дій відносять:

- аналіз проблем і потреб;
- визначення вимог;
- проєктування архітектури;
- реалізацію й інтеграцію;
- верифікацію та валідацію;
- експлуатацію й обслуговування;
- завершення життєвого циклу (утилізація, архівування тощо).

Підходи до розробки програмного забезпечення це засіб, який використовуються для створення та розвитку програмного продукту. В сучасних умовах використовуються два основні підходи до розробки ПЗ: предиктивний (прогнозований) та адаптивний (гнучкий) [3].

Предиктивний підхід орієнтований на ретельне попереднє планування усіх фаз проєкту. Він застосовуються в проєктах із чітко визначеними вимогами, стабільним середовищем та високими вимогами до документування.

Адаптивний підхід побудований на принципах гнучкості, ітеративності та постійного зворотного зв’язку. Він враховують зміну вимог у процесі розробки, що важливо для сучасних ринкових умов.

Для структурованого опису підходів з розробки ПЗ використовуються моделі життєвого циклу, що визначають послідовність виконання та взаємозв’язок процесів, дій і задач протягом життєвого циклу програмного продукту

До предиктивних моделей належать:

- каскадна модель (Waterfall) – послідовне проходження всіх фаз (вимоги → проєктування → розробка → тестування → впровадження);
- V-модель (V-Model) – розширення каскадної, в якій фази розробки поєднані з відповідними фазами тестування;
- спіральна модель – інтегрує ітеративність, аналіз ризиків та контроль якості.

До адаптивних (agile) моделей відносять:

- Scrum – фреймворк з короткими ітераціями (спринтами), чіткими ролями та щоденними зустрічами;
- Extreme Programming (XP) – орієнтований на високу якість коду та тісну співпрацю з замовником.

У межах виконання Проєкту другого рівня пропонується застосовувати V-модель (рис. 1.3.). Її ключові характеристики:

- всі фази розробки проходять зверху вниз по лівій частині “V” (аналіз вимог → системне проєктування → архітектурне проєктування → модульне проєктування → кодування);
- всі фази тестування йдуть знизу вгору по правій частині “V” (модульне тестування → інтеграційне тестування → системне тестування → приймальне тестування);
- кожна фаза проєктування має відповідну фазу тестування (горизонтальні зв’язки між лівою та правою сторонами “V”) [4].

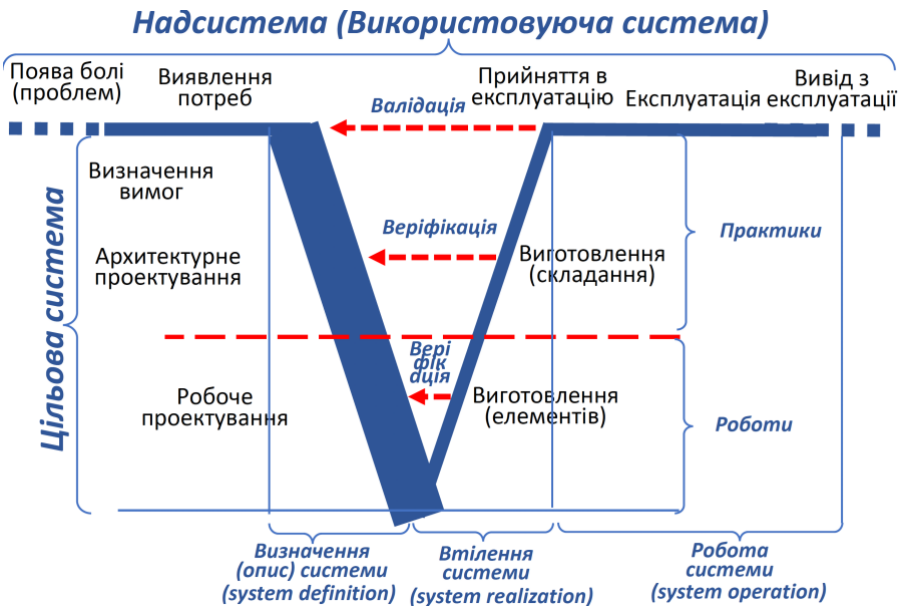

Рисунок 1.3 – V-модель розробки програмного забезпечення

1.2.3. Командна робота

Однією з ключових компетентностей під час виконання Проєкту другого рівня є здатність ефективно працювати в команді. Ця компетентність включає: домовлятися про спільну мету, розподіляти відповідальність, планувати та координувати діяльність, комунікувати, вирішувати конфлікти та забезпечувати якість результатів.

Команда проєкту це самоорганізоване об’єднання взаємозалежних осіб, які діють в межах єдиної методології та спільної мети, спрямованої на створення програмного продукту [5]. Самоорганізація команди не скасовує академічної відповідальності: кожен учасник має виконувати власні зобов’язання і робити внесок у спільний результат.

З метою підвищення ефективності командної роботи перед початком Проєкту Куратор проєкту може запропонувати здобувачам пройти вхідне анкетування. Анкетування використовується для:

– визначення наявних навичок (програмування, тестування, UI/UX, аналітика, DevOps тощо);

– розуміння попереднього досвіду командної розробки (навчальні/комерційні проєкти, роль у команді, інструменти);

– виявлення організаційних обмежень (доступність у часі, комунікаційні канали, технічні ресурси).

Результати анкетування можуть враховуватися Куратором під час формування команд для збалансування складу (за компетенціями та ролями). Здобувачі, які не пройшли анкетування у визначений термін, можуть бути розподілені до команд на розсуд Куратора проєкту.

У межах проєкту ролі можуть бути гнучкими, однак відповідальність має бути визначена. Розподіл ролей у команді та закріплення відповідальних осіб за кожною роллю визначається і зазначається у завданні на Проєкт. Типово команда розподіляє (або поєднує) такі ролі: менеджер проєкту; фронтенд-розробник; бекенд-розробник; дизайнер; аналітик.

Під час виконання проєкту керівник проєкту може запропонувати команді пройти психометричні опитувальники або тести типологій особистості з метою сформувати індивідуальні профілі сильних сторін та зон розвитку та покращити розподіл ролей і спосіб комунікації в команді. Для системної побудови командної взаємодії рекомендується застосовувати інтегровану модель управління взаємодією в проєктах (рис 1.4) [6].

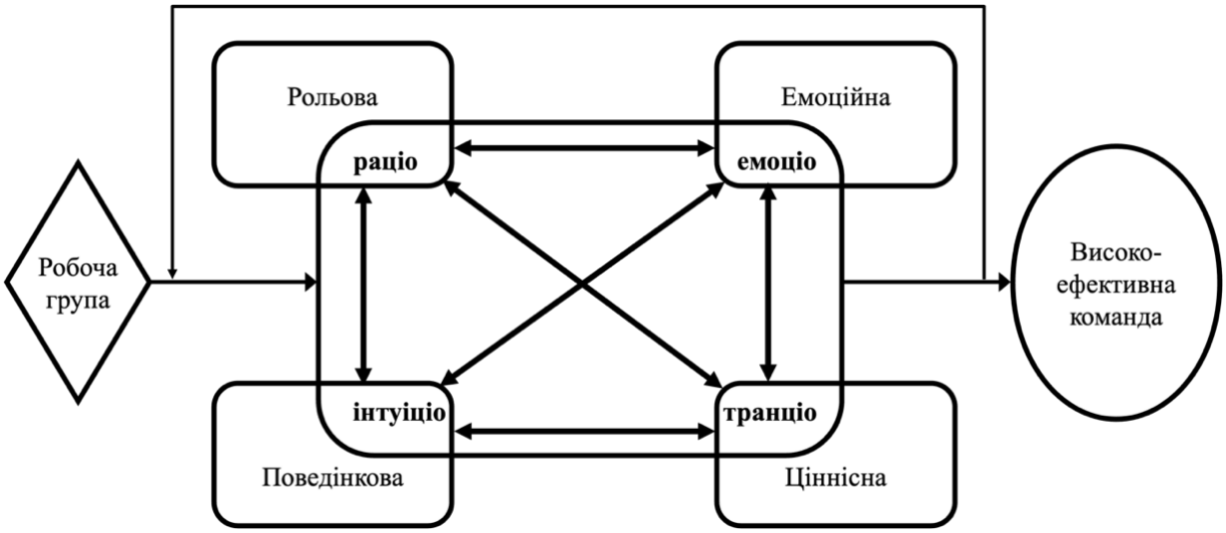

Рисунок 1.4 – Інтегрована модель управління взаємодією в проєктах

2. ПОРЯДОК ВИКОНАННЯ ПРОЄКТУ ТА СТРУКТУРА ЗВІТУ

2.1 Етапи виконання Проєкту

Структура проєктної практики та очікувані результати кожного етапу наведені в табл. 2.1.

Таблиця 2.1 – Структура практики та очікувані результати

№	*Етап*	*Термін виконання*	*Результат*
1	Формування проєктної команди	03.11.20XX-12.01. 20XX	Затверджений склад проєктної команди
2	Пошук ідеї продукту	03.11. 20XX - 12.01. 20XX	Документ з описом продуку
3	Затвердження завдання на проєкт	12.02. 20XX - 18.02. 20XX	Документ з технічним завданням проєкту
4	Аналіз ринку та конкурентів	03.03. 20XX - 04.05. 20XX	Документ з аналізом конкурентів
5	Проєктування Use-case діаграм програмного продукту	27.04. 20XX - 03.05. 20XX	Файл з діаграмами
6	Проєктування UI/UX дизайну продукту	04.05. 20XX - 10.05. 20XX	Макет продукту у Figma
7	Розробка програмного забезпечення	11.05. 20XX - 24.05. 20XX	Розроблений програмний продукт
8	Тестування програмного забезпечення	25.05. 20XX - 31.05. 20XX	Перелік помилок до виправлення
9	Підготовка звіту по проєкту	28.04. 20XX - 30.05. 20XX	Звіт з проєкту
10	Підготовка презентації	26.05. 20XX - 15.06. 20XX	Презентація проєкту
11	Захист проєктів	16.06. 20XX - 20.06. 20XX	Підсумкова оцінка за виконання Проєкту

Під час практики виконується завдання на проєкт що затверджується керівником проєкту (Додаток Г). Головною метою проєктної практики є досягнення цілей проєкту шляхом послідовного виконання визначених етапів:

Етап 1. Формування проєктної команди.

На початку реалізації проєкту команда визначає необхідні ролі та призначає відповідальних осіб за типами робіт проєкту. Після цього проводиться перша командна зустріч, під час якої відбувається знайомство учасників, уточнюються мета проєкту та очікувані результати. Також команда узгоджує правила взаємодії, зокрема визначає канали комунікації, частоту проведення зустрічей, формат звітності, а також принципи роботи з репозиторієм і завданнями.

Етап 2. Вибір ідеї та ініціація Проєкту.

Другий етап передбачає генерування ідеї та ініціацію проєкту. Формування ідеї та визначення вимог до продукту здійснюється залежно від типу проєкту.

У разі виконання сервісного проєкту здобувачі ознайомлюються з проєктною заявкою і реалізують проєкт відповідно до визначених у ньому вимог. Для прикладного проєкту здобувачі самостійно пропонують ідею продукту, обґрунтовують її цінність та визначають базові параметри майбутнього рішення.

У межах прикладного проєкту команда проводить установчу зустріч у форматі брейнштормінгу, за результатами якої формується опис продукту (Додаток Б). Такий опис включає назву продукту, його мету та створювану цінність, основні функції (ключовий функціонал), унікальну пропозицію порівняно з альтернативами, формат реалізації (застосунок, веб-сервіс, гра або інше), визначення цільової аудиторії, а також проблеми, які вирішує продукт.

Етап 3. Аналітика: ринок, конкуренти та уточнення вимог.

Наступний етап передбачає проведення аналітичної роботи, зокрема дослідження ринку, конкурентного середовища та уточнення вимог до продукту. На цьому етапі команда здійснює аналіз ринку та конкурентів,

визначаючи наявні альтернативи, їх сильні та слабкі сторони, а також можливості для диференціації власного рішення. Одночасно уточнюються потреби цільової аудиторії, на основі чого формується перелік вимог до продукту. Завершальним кроком є визначення мінімально життєздатного функціоналу (MVP) та встановлення пріоритетів його реалізації.

Етап 4. Проєктування: сценарії використання та UX/UI

На цьому етапі команда здійснює проєктування сценаріїв взаємодії користувача з продуктом, зокрема шляхом побудови use-case діаграм або застосування інших погоджених способів їх опису. Паралельно розробляється UI/UX дизайн, що включає створення прототипів або макетів основних екранів, визначення логіки переходів між ними, а також узгодження загального стилю інтерфейсу.

Етап 5. Розробка програмного забезпечення.

На даному етапі команда здійснює реалізацію продукту відповідно до визначених вимог і затверджених дизайнерських рішень, забезпечуючи при цьому керованість змін, узгодженість роботи учасників та інтеграцію окремих компонентів у цілісну систему.

Етап 6. Тестування та забезпечення якості

На цьому етапі команда проводить комплексну перевірку продукту, виявляє, фіксує та усуває дефекти, а також здійснює оцінювання відповідності реалізованого рішення встановленим вимогам.

Етап 7. Підсумкові матеріали та захист Проєкту

Завершальним етапом проєктної практики є оформлення звіту за результатами виконання проєкту, підготовка презентації та демонстраційних матеріалів для його захисту, а також фінальне узгодження отриманих результатів із вимогами, визначеними у завданні на проєкт.

2.2 Структура та вимоги до звіту

Результати виконання Проєкту другого рівня оформлюється у вигляді звіту який має відображати хід виконання робіт, отримані результати,

прийняті рішення й обґрунтування застосованих підходів комп’ютерних наук. Під час написання звіту необхідно дотримуватися вимог до змісту, наведених у таблиці 2.2.

Таблиця 2.2 – Настанови до обов’язкових структурних елементів звіту

Назва розділу	*Назва підрозділу*	*Вимоги та обов’язкові елементи*
Титульний лист	-	Наявність
Завдання на проєкт	-	Наявність
Календарний план	-	Наявність
Анотація	-	Дуже стислий опис проєкту викладений у 5 реченнях. 5 ключових слів
Зміст	-	Найменування та номери перших сторінок усіх розділів та підрозділів
Вступ	-	Актуальність теми; мета проєкту, об’єкт дослідження, предмет дослідження, завдання дослідження, практична цінність проєкту
1. Концептуалізація продукту	1.1 Опис ідеї продукту	Генерація та перевірка ідеї продукту
	1.2 Аналіз предметної області та конкурентів	Порівняльна таблиця конкурентів; SWOT-аналіз розроблюваного продукту; Бізнес модель Lean Canvas продукту
	1.3 Формування цілей і вимог до продукту	Мета створення продукту (SMART); вимоги до продукту; вимоги до функцій продукту; системні вимоги
2. Проєктування продукту	2.1 Моделювання поведінки продукту	Use case діаграми
	2.2 Моделювання структури продукту	Опис структури бази даних
	2.3 Дизайн продукту /Архітектура продукту	UI\UX-дизайн продукту (макет у Figma) / Опис архітектурного стилю (патерну) продукту

3. Реалізація продукту	3.1 Особливості реалізації	Опис засобів реалізації програмного забезпечення
	3.2 Тестування	Модульне тестування, Функціональне тестування
	3.3 Документація продукту	Інструкція для користувача
Висновки	-	Стислий виклад результатів проєкту
Список використаних джерел	-	Перелік посилань у порядку згадування по тексту мовою оригіналу за стилем APA
Додатки	-	Лістинг коду, посилання на Github та демонстрацію роботи програмного продукту

Рекомендований обсяг пояснювальної записки – 20–30 сторінок (без урахування додатків).

Звіт умовно поділяється на три частини: вступна частина, основна частина та додатки.

Вступна частина звіту повинна містити такі структурні елементи:

– титульний аркуш;

– завдання на проєкт;

– календарний план;

– анотація та ключові слова;

– зміст;

– перелік умовних позначень, символів, одиниць, скорочень і термінів (за необхідністю).

Основна частина містить:

– вступ;

– змістовну частину звіту (три розділи);

– висновки;

– перелік джерел посилання.

Перший розділ звіту починається з опису проблемної ситуації та обґрунтування запропонованої ідеї (концепції) її вирішення. Проводиться аналіз зовнішнього та внутрішнього середовища за допомогою SWOT-аналізу (таблиця 2.3) та детального аналізу конкурентів (таблиця 2.4). На підставі проведеного аналізу моделюється бізнес модель продукту за шаблоном Lean Canvas (рис 2.1).

Таблиця 2.3 - Шаблон SWOT-аналізу

Внутрішнє середовище	Внутрішнє середовище
S – переваги (Strengths)	O – можливості (Opportunities)
W – недоліки (Weaknesses)	T – загрози (Threats)

Таблиця 2.4 - Шаблон аналізу конкурентів

Конкуренти	**Назва компанії та посилання**	**Ключові функції**	**Кількість вебплатформи трафіку / місяць (SimilarWeb)**	**Присутність у соціальних мережах (залінкувати, якщо такі є)**	**Переваги продукту**	**Недоліки продукту**
Конкурент 1						
Конкурент 2						
Конкурент 3						
Конкурент 4						

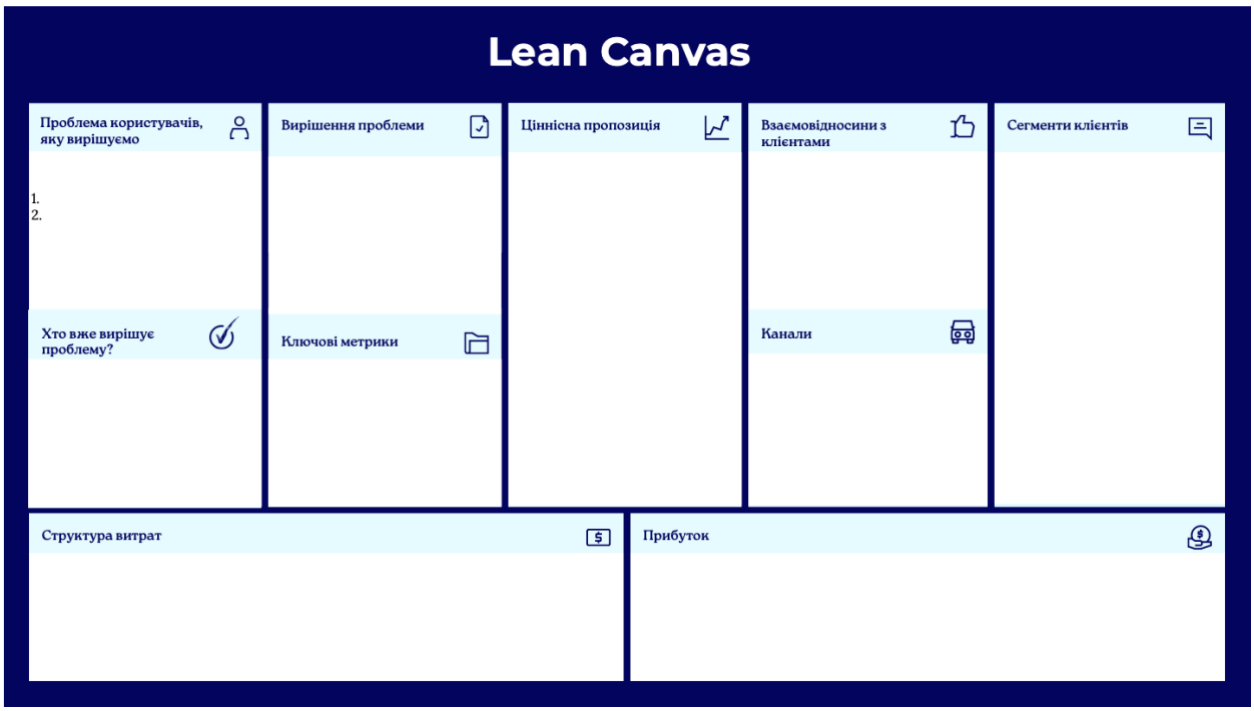

Рисунок 2.1 – Шаблон Lean Canvas

У підрозділі формування цілей і вимог до продукту вказується мета створення продукту, вимоги до продукту та системні вимоги. Мету продукту рекомендується формулювати за методом SMART (Specific, Measurable, Achievable, Relevant, Time-bound):

- specific (конкретна). Мета сформульована чітко та однозначно (що саме потрібно отримати, для кого і в яких умовах);
- measurable (вимірювана). Визначені критерії/показники, за якими можна перевірити досягнення мети (бажано з числовими значеннями);
- achievable (досяжна). Мета реалістична в межах ресурсів команди, наявних компетентностей і відведеного часу виконання Проєкту;
- relevant (узгоджена). Мета відповідає проблемі, яку вирішує продукт, і є логічно узгодженою з цілями проєкту та очікуваною цінністю для цільової аудиторії/замовника;
- time-bound (обмежена в часі). Визначено часові межі досягнення мети (період виконання, етапи, дедлайни або дата отримання результату).

Вимоги до продукту розділяються на функціональні та нефункціональні. Функціональні вимоги визначають, які саме функції та можливості має

реалізовувати продукт, тобто описують поведінку системи та взаємодію з користувачем або іншими системами. Нефункціональні вимоги характеризують якісні властивості продукту та обмеження його функціонування, зокрема продуктивність, надійність, безпеку, зручність використання, масштабованість, сумісність та інші характеристики, що впливають на ефективність і умови експлуатації системи.

Метою етапу визначення вимог є складення точного списку вимог, який можна використовувати в якості вхідних даних для подальшого аналізу системи (створення функціональних, структурних і поведінкових моделей). Такий опис вимог в результаті призводить до стадії проектування.

Опис вимог повинен містити:

1. умови або можливості, необхідні користувачеві для вирішення поставлених проблем або для досягнення цілей;

2. функції і обмеження, які повинна мати система або системні компоненти, щоб виконати контракт замовника на розробку;

3. положення і регламенти, які встановлені використаними стандартами і відображені у специфікаціях або інших формальних документах на систему;

4. умови, можливості і обмеження середовища, необхідного для проектування і виконання системи.

Другий розділ звіту присвячений проєктуванню продукту проєкту. На цьому етапі команда розробляє Use Case діаграми, що відображають, як користувачі та інші зовнішні учасники взаємодіють із системою, і відповідають ідеї та вимогам, сформульованим у першому розділі. Use case описує способи використання системи для досягнення мети конкретного користувача.

Спочатку команда визначає, що саме вважається системою в межах проєкту (межі системи). Далі визначаються актори, будь-які сутності з поведінкою, що взаємодіють із системою (людина, організація, інше програмне забезпечення тощо). Актор на діаграмі відображає роль, яку він

виконує у взаємодії із системою. Для кожного актора формулюється ціль — причина, з якої актор використовує систему, і цінність, яку він отримує у разі успіху. Саме цілі є основою для переліку use cases. Кожна значуща ціль оформлюється як окремий use case (овал усередині меж системи).

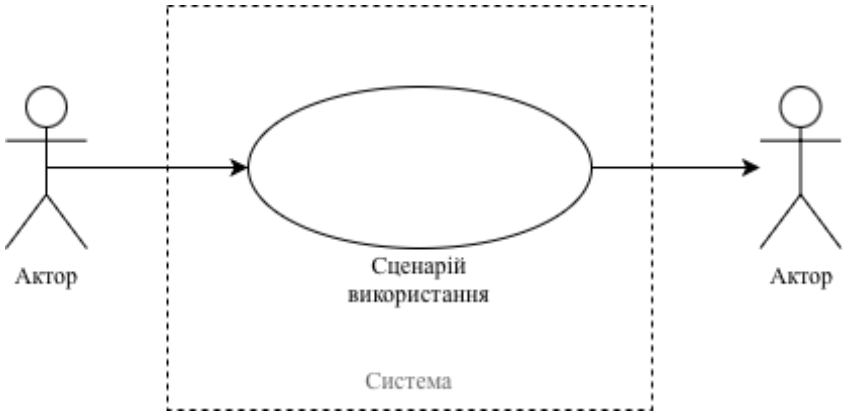

Рисунок 2.2 – Базові елементи Use case

У підрозділах 2.2 та 2.3. слід подати опис рішень щодо даних, дизайну та архітектури програмного продукту.

Необхідно описати реалізацію сховища даних (модель даних/сутності, зв’язки, ключові таблиці або колекції, принципи зберігання та доступу) та обґрунтувати вибір типу сховища даних (реляційна СУБД / NoSQL / файлове сховище / хмарні сервіси тощо) з урахуванням вимог продукту.

Якщо продукт передбачає розробку інтерфейсу (вебсайт, вебзастосунок, мобільний застосунок), необхідно розробити макет (прототип) продукту в Figma та подати у звіті посилання на макет та коротко описати ключові екрани, навігацію і логіку взаємодії користувача з продуктом.

Якщо продукт передбачає переважно або виключно серверну (backend) частину, необхідно описати обраний архітектурний стиль (моноліт / мікросервіси / модульна архітектура / serverless тощо) та коротко обґрунтувати вибір стилю відповідно до вимог проєкту (складність домену, потреби масштабування, незалежність компонентів, вимоги до розгортання та підтримки).

У розділі «Реалізація продукту» наводиться поетапний опис виконаної роботи з реалізації програмного продукту: від підготовки та налаштування середовища розробки до отримання готового, протестованого програмного забезпечення, придатного до розгортання та експлуатації. Опис має відображати логіку виконання робіт і підтверджувати, що результат відповідає вимогам, сформульованим у попередніх розділах.

У підрозділі «Особливості реалізації» необхідно:

– зазначити обрані технології, бібліотеки, фреймворки, СУБД (за наявності), інструменти розробки та тестування;

– обґрунтувати вибір засобів реалізації відповідно до вимог продукту та обмежень проєкту (складність задачі, продуктивність, сумісність, наявні компетенції команди, можливість розгортання тощо);

– за потреби вказати цільове середовище виконання (ОС, платформа, хмарна інфраструктура/хостинг);

– налаштування середовища розробки (репозиторій, структура проєкту, базові залежності, конфігурація);

– реалізацію ключових модулів/компонентів відповідно до архітектурного рішення;

– інтеграцію компонентів (взаємодія модулів, робота з даними, API, авторизація тощо — залежно від типу продукту);

– підготовку продукту до запуску/розгортання (конфігурації, збірка, запуск у тестовому середовищі).

Опис має бути достатнім для розуміння того, що саме реалізовано, у якій послідовності та яким чином підтверджено працездатність.

У підрозділі «Тестування» необхідно описати виконані роботи з перевірки якості програмного забезпечення. Обов’язковим елементом є модульне тестування (Unit testing), яке використовується для перевірки коректності роботи окремих функцій/класів/модулів.

У звіті слід зазначити:

– що саме тестувалося (які модулі/функції є критичними);

– інструмененти/фреймворк тестування, що використовувався;

– загальні результати тестування (кількість/перелік тестів, статус виконання, за можливості — рівень покриття).

Детальний алгоритм проведення модульного тестування та приклади оформлення результатів подаються в Додатку Д.

У підрозділі «Документація продукту» необхідно подати опис функціоналу програмного продукту у вигляді детально опису реалізованих сценаріїв використання з ілюстраціями (скріншоти) ключових екранів/станів системи або приклади запитів/відповідей API (для backend-рішень).

3. ПІДГОТОВКА ДО ЗАХИСТУ ТА КРИТЕРІЇ ОЦІНЮВАННЯ ПРОЄКТУ

3.1 Підготовка до захисту

Захист Проєкту здійснюється згідно графіку освітнього процесу. Здобувачі зобов'язані завчасно підготувати звіт з проєкту та презентацію для захисту. Усі матеріали подаються через електронну платформу Moodle. У випадку недотримання строків завантаження звітної документації, здобувач не допускається до процедури захисту.

Керівнику проєкту необхідно зробити перевірку звіту на наявність обов'язкових структурних елементів (таблиця 2.2) та на дотримання принципів академічної доброчесності.

Захист Проєктів відбувається публічно у присутності комісії по захисту проєктів. У разі виконання Проєкту в команді, команда самостійно обирає спосіб представлення результатів: виступати спільно або делегувати основного доповідача. Тривалість презентації повинна становити 7-10 хвилин.

Презентація має містити наступні слайди:

1. титульний слайд;
2. глосарій;
3. актуальність та проблематика;
4. опис предметної області ;
5. мета та завдання проєкту;
6. вимоги до продукту;
7. ролі задіяні в проєкті;
8. вибір технологій та засобів реалізації;
9. дизайн продукту (за наявності);
10. інтерфейс і екрани продукту;
11. демонстрація роботи продукту;
12. напрями розвитку продукту;
13. висновки.

3.2 Критерії оцінювання

Оцінювання здійснюється на основі комплексного аналізу поданих матеріалів і виступу на захисті. При цьому враховується якість підготовленої звітної документації, рівень командної роботи, обґрунтованість прийнятих рішень, відповідність результатів поставленим завданням, а також дотримання академічної доброчесності. Важливим елементом оцінки є вміння здобувачів аргументовано презентувати результати своєї роботи та відповідати на питання комісії.

Оцінювання здійснюється за трьома показниками:

- якість продукту;
- пояснювальна записка;
- презентація та відповіді на питання.

Підсумкова оцінка розраховується за формулою:

$G = 0{,}5 \cdot G_{продукт} + 0{,}3 \cdot G_{записка} + 0{,}2 \cdot G_{презентація}$

Для оцінювання рівня з $G_{продукт}$, $G_{записка}$ та $G_{презентація}$ використовується 10-бальна оцінна шкала.

Таблиця 3.1 - Критерії оцінювання

Рівні оцінної шкали	*Оцінка*	*Інтерпретація рівня досягнення результатів навчання*	*Коментар*
Високий рівень	10	чудово	Істотно перевершує очікування, зроблено значно більше, ніж вимагалось результатами навчання
	9	відмінно	Перевершує очікування, зроблено більше, ніж вимагалось результатами навчання
середній рівень	8	добре	Зроблено все, що вимагалося в повному обсязі
	7	майже добре	Зроблено майже все, що вимагалося
достатній рівень	6	більш ніж задовільно	У роботі є недоліки та/або завдання виконано не повністю
	5	задовільно	Зроблено мало або в роботі є значні недоліки та/або помилки
недостатній рівень	4	майже задовільно	Зроблено дуже мало або в роботі є дуже значні недоліки та/або помилки
	1-3	незадовільно	Зроблено критично недостатньо або не те, що було потрібно
	0	незадовільно	Здобувач не здав документацію та/або є порушення академічної доброчесності

ЛІТЕРАТУРА

1. Рач В.А., Россошанська О.В., Медведєва О.М. Управління проєктами: практичні аспекти реалізації стратегій регіонального розвитку: навчальний посібник/ за ред. В. А. Рача. Київ: К.І.С., 2010. 276 с. URI: https://dspace.krok.edu.ua/handle/krok/5339

2. Pinto J., Ika L. Project success. In Huemann M., Turner R. (Eds.), *The handbook of project management (6th ed.)* 2023 pp. 42–57. Routledge. DOI: https://doi.org/10.4324/9781003274179

3. Project Management Institute. (2021). A guide to the project management body of knowledge (PMBOK® Guide) (7th ed.). Project Management Institute.

4. Forsberg, K., H. Mooz, et al. (2005). Visualizing project management : models and frameworks for mastering complex systems. Hoboken, Wiley.

5. Петрова І. Л., Мушинський О. Ю. Інноваційні підходи до управління командами у гібридному середовищу. *Теоретичні та прикладні питання економіки*. 2024. № 49. DOI: https://doi.org/10.17721/tppe.2024.49.12

6. Мушинський, О., Тимофєєва, К. Управління командною взаємодією в проєктах на основі моделей особистості. *Вчені записки Університету «КРОК»*. 2025 (4(80), 273–281. https://doi.org/10.31732/2663-2209-2025-80-273-281

7. Jacobson I., Cockburn A. Use-Case Foundation. URL: https://www.ivarjacobson.com/publications/use-case-foundation

8. Методичні вказівки до оформлення здобувачами вищої освіти звітів з проєктних практик та кваліфікаційних робіт виконаних за освітніми програмами навчально-наукового інституту інформаційних та комунікаційних технологій / уклад. Мічківський С.М., Балдик Д.О., Мушинський О.Ю., Тимофєєва К.В.; Київ, Університет «КРОК», 2025

ДОДАТОК А

Проєктна заявка для сервісних проєктів

1. Назва проєкту

2. Інформація про заявника

Прізвище, ім’я, по-батькові, посада, посилання на сторінку профілю.

3. Підрозділ Університету

Вказати для якого підрозділу реалізується проєкт.

4. Короткий опис проєкту (до 300 слів)

Коротко опишіть концепцію реалізації проєкту:

– в чому полягає суть проєкту;

– яка ваша цільова аудиторія;

– як цей проєкт вплине на розвиток вашого підрозділу.

5. Очікувані результати проєкту

Освітні результати проєкту

Здобувачі повинні отримати…

Практичні результати проєкту

У таблиці вкажіть:

– цілі, які ви прагнете досягнути шляхом реалізації проєкту;

– завдання, які потрібно виконати для досягнення конкретної цілі;

– індикатори досягнення цілі – одиниця, якою можна виміряти виконання завдання і досягнення цілі.

Таблиця – Практичні результати проєкту

Цілі проєкту	**Завдання**	**Індикатори досягнення цілі**

6. Планування реалізації проєкту

Опис робіт які потрібно виконати.

Які ролі потрібні для реалізації проєкту/ які вимоги до здобувачів.

Критерії успішності проєкту.

ДОДАТОК Б

Опис продукту

Команда (якщо виконуєте в команді)

Ім’я ПРІЗВИЩЕ

Ім’я ПРІЗВИЩЕ

Ім’я ПРІЗВИЩЕ

Виконавець (якщо виконуєте індивідуально)

Ім’я ПРІЗВИЩЕ

Категорія	**Твій продукт**
Назва	
Мета	
Основні функції	
Унікальна пропозиція	
Формат **(застосунок / веб / гра / тощо)**	
Цільова аудиторія	
Проблеми, що вирішує продукт	

ДОДАТОК В

Титульний лист

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТА КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра комп'ютерних наук

ПРОЄКТ ДРУГОГО РІВНЯ

НАЗВА:
«**……**»

Ступінь вищої освіти – бакалавр
Спеціальність – 122 «Комп'ютерні науки»
Освітня програма «Комп'ютерні науки»

ЗВІТ З ПРОЄКТУ

Команда проєкту (якщо виконуєте в команді)
Ім'я ПРІЗВИЩЕ
Ім'я ПРІЗВИЩЕ
Ім'я ПРІЗВИЩЕ
Виконавець проєкту (якщо виконуєте індивідуально)
Ім'я ПРІЗВИЩЕ

Керівник проєкту: Ім'я ПРІЗВИЩЕ

м. Київ – 202_ рік

ДОДАТОК Г

Завдання на проєкт

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

ЗАТВЕРДЖУЮ:
завідувач кафедри комп'ютерних наук
_____ Сергій МІЧКІВСЬКИЙ
«__» _____ 20__ р.

ЗАВДАННЯ НА ПРОЄКТ ДРУГОГО РІВНЯ

ПІБ керівника команди/виконавця

Назва проекту	*Напишіть тут назву вашого проекту*
Мета проекту	*Яка мета вашого проєкту*
Опис проєкту	*Яку основну проблему вирішує проєкту, короткий опис завдань проекту*
Очікувані результати	*Що має бути створено до завершення проєкту*
Ролі та відповідальність виконавців	*Командний проєкт: вказати - які ролі в проєктній команді передбачено* *Індивідуальний проєкт:що має робити виконавець*
Критерії успішності проєкту	*За чим визначається, що проєкт буде завершено успішно*
Технологічний стек, що використовується, та його обґрунтування	
Технологія/Інструмент	*Назви інструментів / технології*
Опис	*Для чого використовується*
Причини вибору	*Обґрунтувати чому обрано саме ці технології*

Дата видачі завдання ____ 202_р.

Керівник команди/Виконавець Ім'я ПРІЗВИЩЕ

Керівник проєкту Ім'я ПРІЗВИЩЕ

ДОДАТОК Д

Тестування програмних продуктів для проєкту

Модульне тестування (Unit testing) – це метод тестування програмного забезпечення, який полягає в окремому тестуванні кожного модуля коду програми.

Модулем називають найменшу частину програми, яка може бути протестованою.

Check-list (контрольний список, що містить ряд необхідних перевірок для тестування) для тестування програмного продукту (Web-додаток) складається з:

1) тестування зручності використання;

2) функціонального тестування;

3) тестування сумісності;

4) тестування бази даних;

5) тестування безпеки;

6) тестування продуктивності.

Тестування зручності використання (юзабіліті) – це тестування доброзичливості додатку для користувача. При тестуванні зручності використання перевіряється, чи легко новому користувачеві розібратися в додатку і тестується системна навігація.

Мета тестування зручності використання: впевнитися у простоті і ефективності використання продукту при використанні стандартних практик тестування зручності використання.

Сценарії тестування зручності використання:

1) вміст web-сторінки вірний, без граматичних і орфографічних помилок;

2) всі шрифти відповідають вимогам;

3) всі тексти правильно вирівняні;

4) всі повідомлення про помилки вірні, без орфографічних і граматичних помилок, і відповідають заголовку вікна;

5) підказки існують для всіх полів;

6) всі поля правильно вирівняні;

7) між полями, колонками, рядами і повідомленнями про помилки залишено досить вільного місця;

8) всі кнопки повинні мати стандартний формат і розмір;

9) посилання на домашню сторінку має бути на кожній сторінці сайту;

10) неактивні поля мають бути сірими;

11) перевірте, що на сайті немає битих посилань і зображень;

12) підтверджуючі повідомлення повинні відображатися для всіх операцій оновлення і видалення;

13) перевірте сайт при різних роздільних здатностях екрану (640 x 480, 600x800 і т. д.);

14) перевірте, що користувач може користуватися системою без роздратування;

15) перевірте, що Tab правильно працює;

16) панель скролу повинна з'являтися лише тоді, коли вона потрібна;

17) якщо при відправці форми є повідомлення про помилку, у ньому має міститися інформація, передана користувачем;

18) заголовок повинен відображатися на кожній сторінці;

19) всі поля (текстові, випадаючі меню, радіо-кнопки і т. д.) і кнопки мають бути доступні з клавіатури, і користувач має бути в змозі користуватися сайтом, використовуючи лише клавіатуру;

20) переконайтеся, що дані у випадаючих списках не обрізуються із-за розмірів поля, і перевірте, чи зашиті дані в код або керуються адміністратором.

Функціональне тестування – тестування функціональностей і операційної поведінки продукту з метою переконатися, що вони відповідають специфікаціям. Даний вид тестування ігнорує внутрішні механізми системи або компоненту і концентрується виключно на вихідних даних, отриманих у

відповідь на призначене для користувача введення і умови виконання сценаріїв.

Мета функціонального тестування: переконатися, що продукт відповідає потрібній функціональній специфікації, згаданій у документації з розробки.

Сценарії функціонального тестування:

1) протестуйте валідацію всіх обов'язкових полів;

2) переконайтеся, що знак зірочки відображається у всіх обов'язкових полях;

3) переконайтеся, що система не відображає вікно помилки при незаповнених необов'язкових полях;

4) переконайтеся, що коди коректно валідуються і не викликають помилок в розрахунках;

5) протестуйте числові поля: вони не повинні приймати літери, у випадку введення літери повинне відображатися відповідне повідомлення про помилку;

6) протестуйте від'ємні значення в числових полях, якщо вони дозволені;

7) протестуйте, що ділення на нуль вірно обраховується;

8) протестуйте максимальну довжину кожного поля, щоб переконатися, що дані не обрізуються;

9) протестуйте спливаюче повідомлення («Це поле обмежене 500 знаками»), яке повинне відображатися, якщо введені дані перевищують дозволений розмір поля;

10) переконайтеся, що підтверджуюче повідомлення відображається для операцій оновлення і видалення;

11) переконайтеся, що значення вартості відображуються в потрібній валюті;

12) протестуйте всі поля введення на спецсимволи;

13) протестуйте функціональність тайм-ауту;

14) протестуйте функціональність сортування;

15) протестуйте функціональність доступних кнопок;

16) протестуйте умови використання і питання, що часто ставляться: вони мають бути виразними і доступними користувачеві;

17) протестуйте, що при відмові функціональності користувач перенаправляється на спеціальну сторінку помилки;

18) протестуйте, що всі завантажені документи правильно відкриваються;

19) протестуйте, що користувач може скачати завантажені файли;

20) протестуйте поштову функціональність системи;

21) протестуйте, що Java Script вірно працює в різних браузерах (IE, Firefox, Chrome, Safari, Opera);

22) поглянете, що буде, якщо користувач видалить куки (cookies-файли), знаходячись на сайті;

23) поглянете, що буде, якщо користувач видалить куки після відвідування сайту;

24) протестуйте всі дані у випадаючих списках: вони мають бути розташовані в хронологічному порядку.

Тестування сумісності перевіряє сумісність додатку з іншими елементами системи, в якій воно працює, – наприклад, браузерами, операційними системами або залізом.

Мета тестування сумісності: оцінка того, наскільки добре ПЗ працює в певному браузері, під певною ОС, з іншим ПЗ або залізом.

Сценарії тестування сумісності:

1) протестуйте сайт в різних браузерах (IE, Firefox, Chrome, Safari, Opera) і переконайтеся, що сайт правильно відображається;

2) переконайтеся, що використовувана версія HTML сумісна з відповідними версіями браузерів;

3) переконайтеся, що малюнки коректно відображаються в різних браузерах;

4) переконайтеся, що шрифти вірно відображуються в різних браузерах;

5) переконайтеся, що Java Script код працює в різних браузерах;

6) перевірте анімовані GIF в різних браузерах.

Інструмент для тестування сумісності Spoon.net: Spoon.net надає доступ до тисяч додатків (браузерів), що не вимагають установки. Цей інструмент допомагає тестувати додаток в різних браузерах на одній машині.

Тестування бази даних – перевіряються бекенд-записи, введені через web-додаток або десктоп-додаток. Дані, які відображаються у додатку, повинні збігатися з даними, що зберігаються в базі даних. Що має знати тестувальник при тестування БД:

1) тестувальник має розуміти функціональні вимоги, бізнес-логіку, основний сценарій додатку і дизайн бази даних;

2) тестувальник має знатися на таблицях, тригерах, процедурах зберігання, способах відображення і покажчиках, використовуваних для додатку;

3) тестувальник має розуміти логіку тригерів, процедур зберігання, способів відображення і покажчиків;

4) тестувальник має розуміти, які таблиці зачіпаються, коли операції вставки, оновлення і видалення виконуються в додатку.

Сценарії тестування бази даних:

1) перевірте назву бази даних: вона повинна збігатися із специфікацією;

2) перевірте таблиці, колонки, типи колонок і значення за умовчанням: все це має збігатися із специфікацією;

3) перевірте, чи дозволяє колонка значення null;

4) перевірте первинний і зовнішній ключ кожної таблиці;

5) перевірте процедури зберігання;

6) протестуйте, чи встановлена процедура зберігання;

7) перевірте назву процедури зберігання;

8) перевірте назви параметрів, їх типи і кількість;

9) перевірте, обов’язкові параметри чи ні;

10) перевірте процедуру зберігання, видаливши деякі параметри;

11) перевірте базу даних, якщо на виході нуль – записи з нулем мають бути задіяні;

12) перевірте процедуру зберігання, задавши прості SQL-запити;

13) переконайтеся, що процедура повертає значення;

14) перевірте процедуру введенням тестових даних;

15) перевірте поведінку кожного прапора у таблиці;

16) переконайтеся, що дані правильно зберігаються в базі даних після кожного введення;

17) перевірте дані під час кожної операції оновлення, видалення і вставки;

18) перевірте довжину кожного поля: довжина на бекенді і фронтенді мають збігатися;

19) перевірте назви баз даних QA, UAT і проду, імена мають бути унікальними;

20) перевірте зашифровані дані в БД;

21) перевірте розмір БД і час відгуку на кожен запит;

22) перевірте дані, що відображаються на фронтенді, і переконайтеся, що вони збігаються з бекендом;

23) перевірте цілісність даних, вводячи невалідні значення в БД;

24) перевірте тригери.

Тестування безпеки націлено на пошук недоліків і пропусків з точки зору безпеки додатку.

Сценарії тестування безпеки:

1) переконайтеся, що сторінки, що містять важливі дані (пароль, номер кредитної картки, відповіді на секретні питання і т. п.) відкриваються через HTTPS (SSL);

2) переконайтеся, що важлива інформація (пароль, номер кредитної картки) відображається у зашифрованому вигляді;

3) переконайтеся, що правила створення паролів упроваджені на всіх сторінках авторизації (реєстрація, сторінка «забули пароль», зміна паролю);

4) переконайтеся, що якщо пароль змінений, користувач не може зайти під старим паролем;

5) переконайтеся, що повідомлення про помилки не містять жодної секретної інформації;

6) переконайтеся, що якщо користувач вийшов з системи або сесія завершена, він не може користуватися сайтом;

7) перевірте доступ до закритих і відкритих сторінок сайту безпосередньо без авторизації;

8) переконайтеся, що опція «Перегляд початкового коду» відключена і її не бачить користувач;

9) переконайтеся, що обліковий запис користувача блокується, якщо він кілька разів ввів пароль невірно;

10) переконайтеся, що пароль не зберігається в кукі;

11) переконайтеся, що якщо яка-небудь функціональність не працює, система не відображає інформацію про додаток, сервер або базу даних; замість цього відображається відповідне повідомлення про помилку;

12) перевірте сайт на SQL-ін’єкції;

13) перевірте права користувачів і їх ролі (наприклад, кандидат не має бути здатний дістати доступ до сторінки адміністратора);

14) переконайтеся, що важливі операції пишуться в логи і інформацію можна відстежити;

15) переконайтеся, що значення сесій відображаються в адресному рядку у зашифрованому вигляді;

16) переконайтеся, що кукі зберігаються в зашифрованому вигляді;

17) перевірте додаток на стійкість до брутфорс-атак (брутфорс атака (brute force) – метод підбору паролю, шляхом почергового перебору можливих комбінацій. Зазвичай брутфорс виробляється масово з використанням стандартних комбінацій логінів (admin, administrator) і паролів (осмислені словосполуки, словники)).

Тестування продуктивності – проводиться для оцінки відповідності системи або компоненту специфічним вимогам до продуктивності. Сценарії тестування продуктивності:

1) визначення продуктивності, стабільності і масштабованості додатку під різним навантаженням;

2) визначення, чи може актуальна архітектура підтримувати додаток при пікових навантаженнях;

3) визначення, яка конфігурація призводить до найкращих показників продуктивності;

4) визначення, чи не змінився час відгуку в нової версії додатку;

5) визначення пляшкового горла додатку і інфраструктури;

6) оцінка продукту і заліза з метою упевнитися, що вони витримають прогнозовані обсяги навантаження.

Неможливо провести тестування продуктивності вручну з ряду причин:

1) знадобиться велика кількість ресурсів;

2) неможливо одночасно здійснювати ряд дій;

3) відсутній відповідний спосіб відстежування поведінки системи;

4) складність виконання завдань, що повторюються.

Щоб впоратися з вище переліченими проблемами, використовуються спеціальні інструменти тестування продуктивності:

1) Apache Jmeter;

2) Load Runner;

3) Borland Silk Performer;

4) Rational Performance Tester;

5) WAPT;

6) NEO LOAD.

Шаблони тестування

1. Шаблон тест-кейсу (Functional Testing)

ID тест-кейсу	**TC-01**
Назва	*Перевірка авторизації користувача з валідними даними*
Опис	*Перевірити, що користувач може увійти в систему з правильним логіном та паролем*
Вхідні дані	*Логін: user@example.com, Пароль: correct_password*
Кроки виконання	*1. Відкрити форму входу 2. Ввести логін і пароль 3. Натиснути кнопку "Увійти"*
Очікуваний результат	*Користувач успішно входить у систему, відкривається головна сторінка*
Фактичний результат	*(заповнює студент після тестування)*
Статус	*Passed / Failed*

2. Таблиця тестування юніт-тестів (Unit Testing)

Модуль	**Метод**	**Вхідні дані**	**Очікуваний результат**	**Реальний результат**	**Тест пройдено**
AuthService	*login()*	*логін: user, пароль: pass*	*true*	*true*	*true*

3. Шаблон баг-репорту (Defect Report)

ID багу	**BUG-001**
Назва	*Система не приймає правильний пароль*
Середовище	*Windows 10, Chrome v124*
Кроки відтворення	*1. Ввести правильний логін і пароль 2. Натиснути "Увійти"*
Очікуваний результат	*Успішна авторизація*
Фактичний результат	*Виводиться повідомлення "Невірний пароль"*
Скріншот / Лог	*додати*
Пріоритет / Статус	*Високий/ Виправлено*

4. Чекліст нефункціонального тестування

Перевірка	**Так / Ні**	**Коментар**
Чи відображається інтерфейс коректно на мобільному?	*Так*	*Виглядає добре на iPhone 15*
Час відгуку при натисканні кнопки "Зберегти" < 1 сек?	*Ні*	*~2.3 сек*

5. Загальна таблиця покриття тестами

Компонент системи	**Вид тестування**	**Покриття (кількість кейсів)**	**Пройдено**	**Невдачі**
Реєстрація	*Функціональне*	*5*	*4*	*1*
Авторизація	*Юніт-тести*	*3*	*3*	*0*
База даних	*Інтеграційне*	*2*	*2*	*0*