МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

Методичні рекомендації до написання дипломного проєкту для здобувачів вищої освіти спеціальності 121 "Інженерія програмного забезпечення" освітньої програми "Інженерія програмного забезпечення" першого (бакалаврського) рівня

Укладачі: **Ушакова І. О.**
Фролов О.В.
Парфьонов Ю. Е.
Поляков А. О.

Відповідальний за випуск **Бондаренко Д. О.**

Харків
ХНЕУ ім. С. Кузнеця
2024

УДК 004.415 (07.034)
М54

Затверджено на засіданні кафедри інформаційних систем.
Протокол № 8 від 19.01.2024 р.

Самостійне електронне текстове мережеве видання

Укладачі: Ю. Е. Парфьонов
А.О. Поляков
І. О. Ушакова
О. В. Фролов

Методичні рекомендації до написання дипломного проєкту для здобувачів вищої освіти спеціальності 121 "Інженерія програмного забезпечення" освітньої програми "Інженерія програмного забезпечення" першого (бакалаврського) рівня [Електронний ресурс] / уклад. І.О. Ушакова, О. В. Фролов, Ю. Е. Парфьонов, А. О. Поляков. – Харків : ХНЕУ ім. С. Кузнеця, 2024. – 65 с.

Викладено питання організації дипломного проєктування, наведено вимоги до структури дипломного проєкту, методичні рекомендації до розроблення його структурних елементів.

Рекомендовано для здобувачів спеціальностей 121 "Інженерія програмного забезпечення" освітньої програми "Інженерія програмного забезпечення" першого (бакалаврського) рівня.

УДК 004.415 (07.034)

Вступ

Дипломне проєктування є одним із найважливіших видів самостійної роботи, яка завершує підготовку здобувачів за програмою бакалавра, а також - основою для проведення державної атестації бакалаврів.

Головним завданням дипломного проєктування є підготовка здобувача до виконання завдань та обов'язків, що передбачені для первинних посад у певному виді економічної діяльності.

Дипломний проєкт бакалавра може бути початковим етапом виконання дипломного роботи магістра.

У методичних рекомендаціях викладено загальні вимоги до організації та проведення дипломного проєктування, змісту, структури та обсягу дипломних проєктів, організації роботи над проєктом, оформлення та захисту дипломного проєкту.

Зміст, обсяг та структура дипломного проєкту, які наведені у методичних рекомендаціях, є типовими і в окремих випадках за письмовим дозволом випускової кафедри можуть бути змінені.

Методичні рекомендації призначені для здобувачів, що навчаються за спеціальностями 121 "Інженерія програмного забезпечення" освітньої програми "Інженерія програмного забезпечення" першого (бакалаврського) рівня.

Освітній компонент "Дипломний проект" є завершальним етапом підготовки бакалаврів за освітньою програмою "Інженерія програмного забезпечення". У відповідності до освітньо–професійної програми підготовки бакалаврів з інженерії програмного забезпечення дисципліна бере участь у формуванні результатів навчання та компетентностей, які визначено в табл. 1.

Таблиця 1

Результати навчання та компетентності, які формує навчальна дисципліна

Результати навчання	Компетентності, якими повинен оволодіти здобувач вищої освіти
1	2
РН09	ЗК02
	ЗК03
	СК01
	СК04

Закінчення табл. 1

1	2
РН11	СК01
	СК02
РН12	СК01
	СК02
РН14	СК04
	СК05
	СК13
РН15	СК10
	СК11
	СК13
РН16	ЗК10
	СК12
РН19	ЗК02
	СК04
РН20	ЗК02
	СК04
	СК09
РН23	ЗК03
	ЗК04
	СК10
РН24	ЗК02
	ЗК10
	СК09

де, ЗК02. Здатність застосовувати знання у практичних ситуаціях;

ЗК03. Здатність спілкуватися державною мовою як усно, так і письмово;

ЗК04. Здатність спілкуватися іноземною мовою як усно, так і письмово.

ЗК10. Здатність діяти соціально відповідально та свідомо;

СК01. Здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення;

СК02. Здатність брати участь у проектуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування;

СК04. здатність формулювати та забезпечувати вимоги щодо якості програмного забезпечення у відповідності з вимогами замовника, технічним завданням та стандартами;

СК05. Здатність дотримуватися специфікацій, стандартів, правил і рекомендацій в професійній галузі при реалізації процесів життєвого циклу.

СК09. Здатність оцінювати і враховувати економічні, соціальні, технологічні та екологічні чинники, що впливають на сферу професійної діяльності;

СК10. Здатність накопичувати, обробляти та систематизувати професійні знання щодо створення і супроводження програмного забезпечення та визнання важливості навчання протягом всього життя;

СК11. Здатність реалізовувати фази та ітерації життєвого циклу програмних систем та інформаційних технологій на основі відповідних моделей і підходів розробки програмного забезпечення;

СК12. Здатність здійснювати процес інтеграції системи, застосовувати стандарти і процедури управління змінами для підтримки цілісності, загальної функціональності і надійності програмного забезпечення;

СК13. здатність обґрунтовано обирати та освоювати інструментарій з розробки та супроводження програмного забезпечення.

РН09. Знати та вміти використовувати методи та засоби збору, формулювання та аналізу вимог до програмного забезпечення;

РН11. Вибирати вихідні дані для проектування, керуючись формальними методами опису вимог та моделювання;

РН12. Застосовувати на практиці ефективні підходи щодо проектування програмного забезпечення;

РН14. Застосовувати на практиці інструментальні програмні засоби доменного аналізу, проектування, тестування, візуалізації, вимірювань та документування програмного забезпечення;

РН15. Мотивовано обирати мови програмування та технології розробки для розв’язання завдань створення і супроводження програмного забезпечення;

РН16. Мати навички командної розробки, погодження, оформлення і випуску всіх видів програмної документації;

РН19. Знати та вміти застосовувати методи верифікації та валідації програмного забезпечення;

РН20. Знати підходи щодо оцінки та забезпечення якості програмного забезпечення;

РН23. Вміти документувати та презентувати результати розробки програмного забезпечення;

РН24. вміти проводити розрахунок економічної ефективності програмних систем.

1. Мета й завдання дипломного проєктування

Дипломне проєктування – завершальний етап підготовки бакалаврів.

Дипломний проєкт – це кваліфікаційна робота, покликана здійснювати об'єктивний контроль за ступенем сформованості вмінь у випускників вирішувати типові завдання діяльності, які належать до проєктувальної (проєктно-конструкторської) та виконавської (технологічної, технічної) виробничих функцій.

Мета дипломного проєктування – узагальнити та систематизувати знання і практичні навички здобувачів, які здобуті ними під час вивчення навчальних дисциплін гуманітарної та соціально-економічної підготовки; математичної та природничо-наукової підготовки; професійної та практичної підготовки. У процесі роботи над дипломним проєктом здобувачі набувають навички з аналізу науково-технічної, нормативної та довідкової літератури, використання державних стандартів, складання пояснювальної записки до проєкту, практичного застосування знань під час ухвалення конкретних проєктних рішень.

Завданнями дипломного проєктування є:

систематизація, закріплення та розширення теоретичних знань і практичних навичок за напрямом підготовки, застосування цих знань та навичок в процесі виконання конкретних завдань дипломного проєкту;

розвиток та закріплення навичок самостійної роботи;

удосконалення вміння користуватися сучасними системами програмування, вирішувати інженерні завдання з проєктування інформаційних систем та їхніх елементів, використовуючи сучасні методології, інформаційні технології, проводити комп’ютерне моделювання, а також вміння обробляти і систематизувати результати досліджень, використовуючи комп'ютерну техніку та відповідні інструментальні засоби;

визначення відповідності рівня підготовки випускника вимогам освітньо-кваліфікаційної характеристики бакалавра, його готовності та спроможності до самостійної роботи в умовах ринкової економіки, сучасного виробництва, прогресу науки і техніки.

Виконуючи дипломний проєкт, здобувач має повною мірою використовувати набуті знання з інформаційних технологій та комп'ютерної техніки, інтелектуальних систем і баз знань, наявні пакети,

методи та засоби математичної обробки інформації; поєднувати теоретичні знання з виробничим досвідом, отриманим під час проходження практики; використовувати досягнення вітчизняної та світової науки і техніки; враховувати техніко-економічні показники функціонування створюваних програмно-інформаційних систем і комплексів; на високому теоретичному і професійному рівні виконувати проєктування обраних технічних рішень; грамотно, повно і водночас лаконічно викладати свої рішення в пояснювальній записці.

Під час захисту дипломного проєкту здобувач має стисло передати основний зміст роботи, акцентуючи увагу на її актуальності та новизні, можливості її практичного застосування, аргументовано подати ухвалені в ній технічні рішення та обґрунтувати отримані результати.

Дипломний проєкт є самостійною роботою здобувача. За всі розроблені в ньому проєктні рішення, а також правильність, обгрунтованість розрахунків і належне оформлення його матеріалів несе відповідальність автор.

До дипломного проєктування допускається здобувач, який пройшов повний курс навчання та склав усі передбачені навчальним планом заліки й екзамени, тобто виконав усі вимоги навчального плану з напряму підготовки.

2. Організація виконання дипломного проєкту

Здобувачу може бути призначена тема дипломного проєкту з переліку рекомендованих тем. Також йому надається право самостійного вибору теми з урахуванням його схильностей і можливостей найбільш повно застосувати здобуті знання. Якщо тема пропонується здобувачем, то вона має бути обговорена й погоджена з керівником дипломного проєкту.

Для затвердження обраної теми здобувач подає заяву на ім'я завідувача кафедри інформаційних систем. Зразок заяви наведено в додатку А.

Безпосереднє керівництво дипломним проєктом покладається на викладачів кафедри, які призначаються завідувачем кафедри інформаційних систем.

Керівник дипломного проєкту видає здобувачу завдання на проєкт; допомагає здобувачу в складанні календарного плану; проводить консультації; контролює процес виконання проєкту відповідно до календарного плану; рекомендує здобувачу науково-технічну літературу і нормативно-довідкові джерела з теми проєкту; перевіряє матеріали роботи; здійснює попереднє заслуховування результатів виконання дипломного проєкту.

Дипломний проєкт здобувач виконує самостійно. Це вимагає чіткої організації його роботи з моменту вибору теми роботи й до її захисту.

На початковому етапі здобувач має попередньо ознайомитися з основними публікаціями за темою дипломного проєкту та скласти їхній список.

На основі вивчення літературних джерел, які мають охоплювати як монографії, підручники та навчальні посібники, статті у періодичних виданнях, так і патентні матеріали, науково-технічні звіти, реферативні видання, здобувач має чітко уявити собі, що зроблено в теоретичному та прикладному аспектах теми дипломного проєкту, а також докладно ознайомитися з аналогічними рішеннями у відповідній галузі. За результатами цієї роботи оформляється аналітичний огляд (порівняльний аналіз), із якого мають логічно випливати вибрані методики досліджень.

Після вивчення літературних джерел здобувач складає попередній план виконання дипломного проєкту, обговорює його з керівником. У

процесі обговорення уточнюються вихідні дані для проєктування й строки, що регламентують роботу здобувача. Після цього здобувач складає уточнений план роботи над проєктом, погоджує його з керівником та приступає до проєктування. У процесі виконання дипломного проєкту здобувач має регулярно відвідувати консультації керівника, подавати йому на перевірку робочі матеріали відповідно до плану-графіка виконання етапів проєкту.

Контроль керівника дипломного проєкту не звільняє здобувача від повної відповідальності за обґрунтованість ухвалених рішень, дотримання стандартів, термінів виконання календарного плану.

На засіданнях кафедри інформаційних систем регулярно заслуховуються повідомлення керівників дипломних проєктів про хід виконання календарних планів. Здобувачі, що не дотримуються графіка виконання проєкту або значно відстають в його виконанні, запрошуються для звіту на засідання кафедри.

3. Структура, зміст та обсяг дипломного проєкту. Загальні вимоги до дипломних проєктів

Дипломний проєкт складається з пояснювальної записки та інших обов'язкових матеріалів (схеми, діаграми, графіки залежностей, таблиці, малюнки, лістинги програм тощо), що розробляються відповідно до завдання.

Обсяг пояснювальної записки становить 50 – 70 друкованих сторінок формату А4 (без додатків).

Загальну структуру пояснювальної записки дипломного проєкту та рекомендовану кількість сторінок наведено в табл. 2.

Таблиця 2

Структура пояснювальної записки дипломного проєкту

Структурні елементи пояснювальної записки	Кількість сторінок
Титульний аркуш	1
Завдання на дипломний проєкт	2
Реферат	1 - 2
Зміст	1
Перелік умовних скорочень	1
Вступ	1 - 2
Розділ 1	9 - 12
Розділ 2	14 - 19
Розділ 3	14 - 20
Розділ 4	8 - 10
Висновки	1 - 2
Список використаних джерел	2 - 3
Додатки	

Пояснювальна записка виконується у друкований спосіб на аркушах паперу формату А4. Її текст має бути виконано з використанням гарнітури шрифту Times New Roman (кегль 14), з міжрядковим інтервалом 1,2.

Поля сторінок пояснювальної записки: 30 мм від лівого краю аркуша, 10 мм від правого краю аркуша, по 20 мм від верхнього та нижнього країв аркуша.

Абзацний відступ має бути однаковим упродовж усього тексту та дорівнювати 1,27 см.

Вирівнювання основного тексту проводиться "за шириною".

Докладні вимоги до оформлення пояснювальної записки наведено у відповідних методичних рекомендаціях [20].

Дипломний проєкт спрямований на аналіз, моделювання, прогнозування інформаційних процесів або керування такими процесами в економіці (як приклад, можна розглядати окреме підприємство, організацію, галузь чи географічний регіон). Вони передбачають вибір конкретної предметної області для аналізу й дослідження. Виконання таких проєктів передбачає аналіз предметної області на основі вивчення спеціальної літератури та ознайомлення з інформаційними процесами безпосередньо на підприємствах та в організаціях.

Змістовна частина пояснювальної записки проєкту має містити:

1. Аналіз предметної області;
2. Специфікацію вимог до програмного забезпечення;
3. Проєктні та технічні рішення;
4. Тестування програмного забезпечення.

Структуру змістовної частини дипломних проєктів, якої необхідно дотримуватися, наведено в додатку Б.

4. Методичні рекомендації до розроблення структурних елементів пояснювальної записки

4.1. Загальні рекомендації щодо розроблення пояснювальної записки дипломного проєкту

Загальними вимогами до тексту пояснювальної записки є логічна послідовність викладення матеріалу, чіткість і конкретність викладення теоретичних і практичних результатів роботи, суті постановки завдання та мети роботи, методів дослідження, ухвалених рішень, доведеність висновків і обґрунтованість рекомендацій. У тексті пояснювальної записки необхідно дотримуватися єдиної термінології. Вона не має бути перевантажена малоінформативним матеріалом, описом загальновідомих даних, виведенням формул тощо. Необхідно посилатися на джерела інформації. У тексті пояснювальної записки має бути наведений використаний математичний апарат та результати виконаних розрахунків за допомогою ПК.

Текст пояснювальної записки не слід викладати від першої особи, краще використовувати безособову форму (наприклад, "обчислюється", "знаходимо") за всім текстом у визначеному відмінку й часі.

Під час викладення матеріалу не слід використовувати:

розмовні звороти;

жаргонні слова та звороти;

різні терміни для позначення одного поняття;

іншомовні слова та терміни за наявності в українській мові рівнозначних слів і термінів;

скорочення слів і словосполучень, крім встановлених правилами орфографії та нормативними документами.

Першою сторінкою пояснювальної записки є **титульний аркуш.** Він містить такі дані:

відомості про виконавця роботи - юридичну особу (організацію) або фізичну особу;

повна назва документа;

підписи відповідальних осіб;

рік складання пояснювальної записки;

Зразок титульного аркуша дипломного проєкту наведено в додатку В.

Реферат – це короткий виклад змісту пояснювальної записки, що містить основні фактичні відомості та висновки, необхідні для початкового ознайомлення з нею. Реферат виконується українською та англійською мовами.

Реферат має бути стислим, інформативним і містити відомості, які дозволяють ухвалені рішення про доцільність читання пояснювальної записки.

Реферат має містити:

відомості про обсяг пояснювальної записки, кількість ілюстрацій, таблиць, додатків, кількість джерел згідно з переліком посилань (усі відомості наводять, охоплюючи дані додатків);

текст реферату;

перелік ключових слів.

Текст реферату має відбивати подану в пояснювальній записці інформацію в такій послідовності:

об'єкт дослідження або розроблення;

мета роботи;

методи дослідження та апаратура;

результати та їхня новизна;

основні технологічні й техніко-експлуатаційні характеристики та показники;

взаємозв'язок з іншими роботами;

рекомендації щодо використання результатів роботи;

галузь застосування;

значущість роботи та висновки;

прогнозні припущення про розвиток об'єкта дослідження або розроблення.

Реферат належить виконувати обсягом не більш, як 500 слів, і, бажано, щоб він уміщувався на одній сторінці формату А4.

Ключові слова призначені для розкриття сутності проєкту та для розповсюдження інформації про розробку. Їх розміщують після тексту реферату. Перелік ключових слів містить від 5 до 15 слів (словосполучень), надрукованих великими літерами в називному відмінку в рядок через коми.

Приклади рефератів наведено в додатку Г.

Зміст містить: вступ; назви всіх розділів, підрозділів та пунктів основної частини пояснювальної записки; висновки; перелік посилань; назви додатків і номери сторінок, які містять початок матеріалу.

Приклад змісту пояснювальної записки наведено в додатку Д.

Якщо в пояснювальній записці використовуються маловідомі скорочення, нові символи, позначення і таке інше, то в ній має бути **перелік умовних скорочень,** який подається у вигляді окремого списку, що розміщують перед вступом. Незважаючи на це, за першої появи цих елементів у тексті документу надають їхню розшифровку.

Якщо в роботі спеціальні терміни, скорочення, символи, позначення повторюються менше трьох разів, перелік не складають, а їх розшифровку наводять у тексті під час першого згадування.

Приклад переліку умовних скорочень наведено в [19].

Вступ - це віддзеркалення роботи, тому слід ретельно його опрацювати. Краще формувати вступ після виконання основного тексту пояснювальної записки.

У вступі до дипломного проєкту необхідно ідентифікувати та сформулювати проблему бізнесу, яка виникла на підприємстві, обґрунтувати актуальність теми проєкту для вирішення цієї проблеми на основі розроблюваного модуля або системи. Коротко охарактеризувати функціональність модуля або системи. Необхідно охарактеризувати технічну та програмну платформи розроблення автоматизованого модуля. Потрібно сформулювати мету й завдання проєкту, визначити об'єкт і предметну область проєктування. Також необхідно навести інформацію щодо засобів проєктування, які використовувалися в дипломному проєкті, та можливих галузей застосування результатів, отриманих у дипломному проєкті.

Висновки до роботи – це резюме за результатами всієї роботи. Ця частина має особливу важливість, оскільки тут мають бути наведені підсумкові результати роботи.

У пояснювальній записці мають міститися висновки за кожним етапом виконаного проєкту та по роботі загалом, які необхідно співвіднести з метою і завданням на дипломне проєктування.

Необхідно зазначити практичну цінність результатів роботи, дати рекомендації для подальшого вдосконалення об’єкта проєктування. Зазначаючи практичну цінність отриманих результатів, важливо окреслити ступінь їхньої готовності до використання, масштабів

використання, а також надати стислі відомості щодо впровадження результатів досліджень із зазначенням назв організацій, в яких здійснена реалізація, форм реалізації, реквізитів документів тощо. Якщо дипломний проєкт впроваджений на підприємстві, то до дипломного проєкту додається довідка або акт про впровадження.

Наводять відомості, де й коли було апробовано отримані результати (на яких конференціях, семінарах вони доповідалися), перераховуються публікації здобувача за матеріалами дипломного проєкту.

Список використаних джерел має містити відомості про літературні джерела, використані в процесі розроблення проєкту.

Список літератури – це реєстр використаних джерел за темою проєкту в найширшому значенні. Тому не слід обмежуватися лише цитованою літературою. У список варто вміщувати всі матеріали, які були прочитані, переглянуті, проаналізовані в процесі роботи над дипломним проєктом і стосуються його теми. Бажано подавати джерела якомога повніше, пам'ятаючи, що бібліографічний список до проєкту – це підсумок вивчення проблеми і передумова подальших наукових досліджень.

Список використаних джерел оформлюється згідно з ДСТУ 8302:2015 "Бібліографічне посилання. Загальні вимоги та правила складання" [9].

Список використаних джерел має містити не менше 30 джерел. Він подається мовою оригіналу, розміщується в алфавітному порядку прізвищ перших авторів або назв та нумерується в порядку їхнього зростання. Нумерація безперервна.

Роботи одного автора розташовані за алфавітом назв або в хронології їхнього написання. Алфавітний список розташований за алфавітом у такій послідовності:

література зведеного кириличного алфавіту, для джерел на мовах із кириличною графікою (українська, болгарська та ін.);

література в латинському алфавіті;

електронні ресурси в тій же послідовності що й друковані видання (спочатку кирилицею, а потім латиницею).

Список використаних джерел обов'язково має містити прізвище та ініціали автора, повну назву джерела, місто видавництва, видавництво та рік видання, кількість сторінок чи посилання на сторінки тощо.

Загальний обсяг книги в сторінках вказується, якщо посилання на неї проводиться повністю, сторінки (від ... до) відмічаються, якщо посилання відносяться до окремої частини літературного джерела.

У **додатках** вміщують матеріал, який є необхідним для повноти пояснювальної записки, але не може бути послідовно розміщений в її основній частині через великий обсяг або з інших причин.

Ілюстрації (діаграми бізнес-процесів, сценарії діалогів та ін.), таблиці, проміжні математичні докази, формули та розрахунки, текст допоміжного характеру тощо можуть бути оформлені у вигляді додатків.

4.2. Рекомендації щодо розроблення розділів основної частини пояснювальної записки дипломного проєкту

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ <НАЗВА>

Метою Розділу 1 є докладне дослідження основних аспектів та характеристик предметної області проєкту, а саме визначення змісту і моделювання предметної області, огляд літературних джерел щодо наявних рішень і аналогів, позиціонування програмного продукту.

Підрозділ 1.1. Визначення змісту предметної області

Визначення змісту предметної області забезпечує контекст і розуміння задачі або проблеми, яку необхідно вирішити. Якщо тема диплома пов’язана з бізнесом, то в цьому підрозділі наводиться коротка характеристика напрямків діяльності об'єкту управління (підприємства, організації); визначається проблема бізнесу, яку слід вирішити. Інакше предметна область визначається без прив’язки до підприємства.

Тут треба визначити:

цілі і завдання розроблення;

контекст і сферу застосування. Контекст - це межі, в яких виконуються процеси і відбуваються пов’язані з ними події, та ресурси, необхідні для їх відповідної інтерпретації; сфери застосування, зазвичай, визначаються в термінах продукту, типу замовника або галузі;

зв'язки і взаємодії з іншими предметними областями. Предметна область може бути складною та включати різні елементи, пов'язані один з одним;

ключові поняття і терміни, характерні для даної предметної області. Вона може включати безліч взаємопов'язаних понять, процесів, суб'єктів і об'єктів;

зміни і еволюції предметної області. Предметна область може бути схильна до змін і розвитку з часом. Нові технології, дослідження та вимоги можуть впливати на предметну область, призводячи до її зміни та розвитку.

Підрозділ 1.2. Моделювання предметної області

Моделювання предметної області передбачає створення системи моделей, що імітують структуру та функціонування досліджуваної предметної області і відповідає основним вимогам – бути адекватною цій області.

У підрозділі 1.2. необхідно:

визначити склад підрозділів підприємства та зв’язки між ними, розробити схему організаційної структури підприємства;

розробити схему організаційної структури підрозділу (підрозділів), пов’язаних з визначеними бізнес-процесами;

визначити склад функцій, що входять до бізнес-процесу, розробити діаграму дерева функцій;

розробити модель управління бізнес-процесом та описати його (табл. 3).

Таблиця 3

Характеристика бізнес-процесу «Назва»

Назва характеристики	Значення характеристики
Ім'я бізнес-процесу	
Основні учасники*	
Вхідна подія	
Вхідні повідомлення / документи**	
Вихідна подія	
Вихідні повідомлення / документи**	
Клієнт бізнес-процесу***	

*для кожного учасника вказати структурний підрозділ, посаду, його роль у бізнес-процесі,

** навести перелік документів

*** процес, що використовує інформацію процесу

Для моделювання предметної області використовують CASE-інструменти.

У процесі моделювання необхідно виділити транзакційну складову бізнес-процесу, яка забезпечує збір, накопичення та обробку кількісних даних про поточний стан об’єкта управління, а також аналітичну складову, яка забезпечує аналіз кількісних показників, сформованих у транзакційні складові.

Аналітична складова бізнес-процесу повинна забезпечити дослідження кількісних показників у різних розрізах та вимірах (за періодами часу, за товарами, за клієнтами, підрозділами тощо).

Проведення такого багатоаспектного аналізу забезпечить інформаційну підтримку прийняття рішень, спрямованих на вирішення виявленої проблеми.

Підрозділ 1.3. Огляд і аналіз літературних джерел щодо існуючих рішень та аналогів

У підрозділі 1.3. треба провести огляд літературних джерел щодо існуючих підходів до розроблення програмного забезпечення відповідно визначеної проблеми, цілі і завдань, які стосуються бізнес-вимог та принципів, певних технологій, платформ, бібліотек тощо. Під час огляду треба робити посилання на відповідні джерела (наприклад, у роботі [..] автори проводять систематичний огляд; автор зачіпає (висвітлює) такі проблеми (питання, факти); робота [..] автора стосується; автор у роботі [..] робить висновок (підводить підсумок, говорить, стверджує), зазначає (аналізує, характеризує, розкриває) недоліки (суперечності, сутність), описує, називає, формулює, висуває (гіпотезу, питання), висловлює припущення, зупиняється, підкреслює, стверджує, доводить тощо. за результатами огляду робиться висновок щодо використання кращих підходів до розроблення.

Треба обрати декілька найбільш популярних програмних продуктів-аналогів, призначених для реалізації функціональності предметної області та визначити їх характеристики, такі як:

фірма-розробник;
версії продукту;
операційна система;
основна функціональність;
інтерфейс користувача;
допомога користувачу;
ціна користування за місяць/рік;
наявність безкоштовної версії / плану;

тощо.

Результати проведеного аналізу навести в таблиці (табл. 4):

Таблиця 4

Характеристики програмних продуктів-аналогів

Назва програмного продукту	<Назва продукту 1>	<Назва продукту 2>	. . .
Фірма-розробник			
Версії продукту			
Операційна система			
Основна функціональність			
Інтерфейс користувача			
Допомога користувачу			
Ціна користування			
Наявність безкоштовної версії / плану			
. . .			

Для кожного з програмних продуктів навести та коротко описати екранні форми, що характеризують основну функціональність продукту.

В кінці підрозділу зробити висновок щодо можливості використання певних підходів до розроблення програмного забезпечення і досвіду ведучих фірм-розробників програмних продуктів при розробленні проектних рішень.

Підрозділ 1.4. Позиціонування програмного забезпечення (застосунку, модулю, системи)

Позиціонування - це визначення того, як ваше програмне забезпечення (застосунок, модуль, система) вписується в IT- ринок, і як ви подаєте унікальну цінність до цільової аудиторії. Позиціонування визначає, яким чим ваше програмне забезпечення відрізняється від конкурентів і чому воно потрібно вашим клієнтам, тому це один із найважливіших кроків для досягнення успіху.

Позиціонування містить короткий опис наступних характеристик:

ділові переваги,

визначення проблеми,

визначення позиції.

Пункт 1.4.1. Ділові переваги

Ділові переваги – це короткий опис переваг, що досягаються проектом:

актуальність, тобто важливість, значущість, затребуваність на сьогодні створюваного програмного забезпечення;

призначення, тобто яким чином розроблене програмне забезпечення буде задовольняти потреби основних користувачів та інших зацікавлених сторін і яким чином це буде реалізовано.

Пункт 1.4.2. Визначення проблеми

Під час визначення проблеми - дається підсумок проблеми, яка вирішується проектом. Для визначення проблеми використовуватися наступний формат (табл. 5).

Таблиця 5

Визначення проблеми

Проблема	<Опис проблеми>
торкається	<Зацікавлені сторони, яких торкається проблема>
Її наслідком є	<Який є вплив проблеми>
Успішне вирішення	<Список деяких ключових переваг від успішного вирішення>

Пункт 1.4.3. Визначення позиції

Під час визначення позиції програмного забезпечення описується на найвищому рівні унікальна позиція на ринку, яку має намір воно заповнити. Визначення позиції описується у наступному форматі (табл. 6)

Таблиця 6

Визначення позиції

Для	<цільовий замовник>
Який	<визначення потреб і можливостей>
<Назва продукту>	- це <категорія програмного забезпечення>
Який	<визначення ключової переваги (причини, яка спонукає придбати / користуватисьпрограмним забезпеченням>
На відміну від	<основна конкурентоспроможна альтернатива>
Наш продукт	<визначення основної відмінності>

РОЗДІЛ 2. СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Метою розділу 2 є розроблення і детальна специфікація вимог до програмного забезпечення (застосунку, модуля, системи тощо), що розробляється. Цей розділ містить глосарій, розроблення варіантів використання, специфікації функціональних та нефункціональних вимог відповідно до методології RUP. Якщо до специфікації вимог буде використаний інший підхід, то структура розділу за погодженням з керівником може бути іншою.

Підрозділ 2.1. Глосарій

Глосарій – це словник основних використовуваних термінів проєкту. Цей документ є найпершим результатом концептуального аналізу предметної області. Глосарій можна розглядати як документ, що засвідчує спільне розуміння основної термінології Замовником і Розробником.

Крім того, глосарій є відправною точкою для побудови більш розгорнутих моделей предметної області, які на стадії реалізації інформаційної системи лягають в основу об'єктної моделі (для об'єктно-орієнтованих застосувань) і моделі даних (для генерації схеми бази даних).

Глосарій необхідно подати у вигляді табл. 7.

Таблиця 7

Глосарій

Термін	Опис терміну
1. Основні поняття та категорії предметної області та проекту	
2. Користувачі системи	
3. Вхідні та вихідні документи	

Підрозділ 2.2. Розроблення варіантів використання

Розроблення варіантів використання містить:

- діаграму варіантів використання;
- специфікацію варіантів використання.

Пункт 2.2.1. Діаграма варіантів використання

Діаграма варіантів використання відображає функціональність, яка буде реалізована в програмному продукті. Варіант використання можна розглядати як функцію, що реалізується системою. Однак будь-яка функція повинна мати цінність і давати можливість отримати кінцевий результат для кінцевого користувача продукту або послуги. Тому при специфікації варіанта використання виділяють серед усієї функціональності системи лише ту функціональність, яка:

- корисна конкретному кінцевому користувачеві;
- дозволяє отримувати користувачеві конкретні закінчені результати.

Перш ніж приступити власне до специфікації вимог у формі варіанта використання, в RUP звичайно складають реєстр (список) акторів (actors) (табл. 8) і варіантів використання (Use Case) (табл. 9).

Актор – дійова особа, зовнішня стосовно проєктованої системі сутність, яка взаємодіє з системою і використовує її функціональні можливості для досягнення певної мети або вирішення приватних завдань. Актором звичайно буде користувач системи. Окрім користувача, як актор може розглядатися інша програмна система, апаратний пристрій, час. Пошук акторів системи зазвичай зводиться до аналізу ролей різних користувачів. Вибір акторів залежить від їх функціональних обов'язків, розмежування доступу, способів використання інформаційної системи.

Варіант використання – це опис послідовності дій, які може здійснювати система у відповідь на зовнішні впливи акторів. Мета варіанта використання – визначити закінчений аспект або фрагмент поведінки системи, яка має власну поведінку, без розкриття її внутрішньої структури.

Таблиця 8

Список акторів

Актор	Короткий опис

Таблиця 9

Список варіантів використання

Варіант використання	Короткий опис варіанта використання	Актор	Основний/допоміжний

Наступним кроком є декомпозиція основних варіантів використання для більш докладного їх опису. Розглядаються додаткові варіанти використання, що деталізують основну мету (наприклад для основного варіанту використання «Формування замовлення» виділяють додаткові варіанти «Створити замовлення», «Розрахувати знижку» тощо). На діаграмі варіантів використання відносини між основними варіантами використання та додатковими визначаються відносинами узагальнення, включення та розширення. Результати декомпозиції слід подати в табл. 10.

Таблиця 10

Декомпозиція варіантів використання

Основний варіант використання	Деталізований варіант використання	Тип зв'язку

Далі створюється діаграма варіантів використання у середовищі певного CASE-засобу.

Пункт 2.2.2. Специфікація варіантів використання

Кожен варіант використання повинен мати опис. У дипломному проекті слід навести опис варіантів використання, що реалізують основну функціональність (зазвичай крім ведення довідників) у вигляді таблиці (табл. 11).

Таблиця 11

Варіант використання <Назва>

Варіант використання	<ID><Назва варіанта використання, у формі дієслова з пояснювальними словами>
Контекст використання	<Опис набора функцій, об’єднаних контекстом>
Дійові особи	<Перелік дійових осіб>
Передумови	<Опис дій, які треба виконати перед виконанням варіанту використання>
Тригер	<Опис події, яка запускає виконання варіанту використання>

Закінчення табл. 11

Пост-умови	<Набір станів, які гарантує система незалежно від успіху виконання варіанту використання>
Основний потік	1. Актор <дія> 2. Система <дія> 3. Актор <дія> . . .
Розширення	4а. Назва точки розширення для певного сценарію 4а.1. Система <дія> 4а.2. Система <дія> . . .

Назва – коротка фраза в вигляді дієслова в неозначеній формі завершеного виду або віддієслівного іменника, яка відображає мету.

Контекст - стисло одним абзацом описується, що повинен робити варіант і який кінцевий результат від нього очікується.

Дійові особи це:

Основні (первинні) актори.

Допоміжні (вторинні) актори.

Передумова – варіант використання, який має бути обов'язково виконаний, щоб можна було виконати даний варіант. Передумову описує стан, у якому система повинна перебувати до початку виконання варіанту.

Тригер – подія предметної області, що викликає використання варіанта використання. Іноді тригер передує першому кроку варіанту використання, а іноді сам є першим кроком.

Постумова – варіант використання, який обов'язково має бути виконаний після виконання даного варіанту; це стан, у якому система повинна перебувати після закінчення виконання варіанту; це те, що гарантується акторам-учасникам, незалежно від успіху виконання даного варіанту. Наприклад, у разі невдалої транзакції всі дані, що були в системі до її початку, зберігаються незмінними.

Події, що описуються передумовами або постумовами, мають бути станами, які користувач може спостерігати.

Основний потік - перераховуються пронумеровані кроки типових дій починаючи від тригера аж до досягнення гарантії успіху як певний сценарій взаємодії системи і користувача.

Розширення - винятки з кроків основного потоку, які обробляються

за своїми алгоритмами згідно з бізнес-логікою процесу. Розширення починається з назви точки розширення, яка починається номером кроку основного потоку з додавання букви латинського алфавіту (a, b, c), далі йде послідовність дій системи та користувача.

Підрозділ 2.3. Специфікація функціональних вимог

Для специфікації функціональних вимог, визначених в попередньому пункті, треба визначити їх атрибути:

пріоритет – заповнюється аналітиком, показує пріоритет реалізації вимоги для клієнта. Використовується при управлінні проектом і визначає пріоритет розроблення вимоги. Можливі значення: обов'язкове, рекомендоване, опційне (за вибором);

трудність – заповнюється менеджером проекту, показує рівень трудовитрат, пов'язаних з реалізацією вимоги. Використовується при управлінні проектом і впливає на черговість розроблення. Трудність виконання вимоги може виражатися у вигляді трудомісткості і вказувати кількість людино-днів, потрібних для його реалізації або у вигляді значень шкали: висока, середня, низька. Висока трудність – це вірогідність того, що вимога є дуже дорогою в термінах ресурсів або грошей. Вона має бути виконана спочатку або від неї відмовляються;

вплив на архітектуру – заповнюється архітектором. Показує, чи будуть варіанти використання зачіпати основну частину архітектури програмного забезпечення. Атрибут може приймати значення: так, якщо зачіпає основну частину архітектурних рішень, або ні – якщо не зачіпає. Використовується, щоб упорядкувати виконання проекту за пріоритетами варіантів використання. Необхідно заздалегідь встановити, які варіанти використання зачіпають основну частину архітектурних рішень. Ці варіанти використання реалізуються першими.

контакт – заповнюється аналітиком, ідентифікує зацікавлену особу, яка може надати необхідну інформацію про вимогу (ім’я контакту). Використовується для гарантії, що розробники можуть отримати інформацію, необхідну їм для реалізації вимоги. Замість Контакту часто використовується атрибут Виконавець.

Якщо в проекті в якості контакту виступає одна людина – цю колонку можна опустити, але треба ідентифікувати контакт один раз в тексті цього пункту. Специфікація функціональних вимог наводиться в таблиці (табл. 12)

Таблиця 12

Специфікація функціональних вимог

Ідентифікатор вимоги	Назва вимоги (варіанту використання)	Атрибути вимог			
		Пріоритет	Трудність	Вплив на архітектуру	Контакт / Виконавець
UC1					
...					

Підрозділ 2.4. Специфікація нефункціональних вимог

Нефункціональні вимоги можна поділити на наступні групи:

1. Застосовність:

час, необхідний для навчання звичайних і досвідчених користувачів;

вимірний час відгуку для типових завдань;

основні вимоги застосовності нової системи відносно інших систем, які знають користувачі;

вимоги по відповідності загальним стандартам застосовності, наприклад, стандартам інтерфейсу користувача IBM або стандартам графічного інтерфейсу користувача Microsoft для Windows.

2. Надійність:

Доступність – визначає % доступного часу (xx.xx %), час використання, час, що витрачається на обслуговування, порушення режиму роботи і т. д.

середній час безвідмовної роботи – зазвичай визначається в годинах, але може також визначатися в днях, місяцях або роках;

середнє напрацювання до ремонту – як довго системі дозволяють працювати до того, коли повинне бути проведене її обслуговування;

точність – визначає розрядність (роздільну здатність) і точність (за деяким відомим стандартом), які потрібні у вихідних даних системи;

максимальна норма помилок або дефектів – зазвичай виражається в термінах кількості помилок на тисячу рядків коду або кількості помилок у функціональній одиниці.

3. Робочі характеристики.

Тут треба виділити конкретні характеристики продуктивності, швидкодії, місткості, використання ресурсів системи. Де можливо, потрібно зробити посилання на пов'язані варіанти використання за ім’ям.

швидкодія для транзакції (середнє значення, максимальне);

продуктивність (наприклад, число транзакцій за секунду);

місткість (наприклад, число замовників або транзакцій, яке може розміщувати система);

режими зниженої продуктивності (що є прийнятним режимом роботи, коли система стала деякою мірою гіршою);

використання ресурсів: пам'яті, дискового простору, комунікацій тощо.

4. Експлуатаційна придатність

В цю групу включають всі вимоги, які розширюють експлуатаційну придатність або надійність формованої системи, включаючи стандарти кодування, угоди про імена, бібліотеки класів і утиліти підтримки.

5. Проектні обмеження.

Ці вимоги повинні містити всі проектні обмеження до створюваної системи. Проектні обмеження становлять рішення, які були сформульовані як обов'язкові і повинні твердо дотримуватися. Прикладами можуть бути мови програмування, вимоги до технології програмування, обов'язкове використання інструментальних засобів розробки, архітектурні і конструктивні обмеження купованих компонентів, бібліотек класів і т. п.

6. Вимоги до документації, призначеної для користувача і до системи допомоги.

Описують вимоги, якщо вони є, до інтерактивної документації користувача, до системи довідки, до попереджувальних повідомлень і т. п.

7. Куповані компоненти.

Описують всі куповані компоненти, які має використовувати система, всі вживані ліцензії або обмеження щодо використання і всі відомості про сумісність і / або здібність до взаємодії або про стандарти інтерфейсу.

8. Інтерфейси.

Визначають інтерфейси, які повинні бути підтримані застосувнком. Він має містити адекватну специфіку, протоколи, порти і логічні адреси і т. п. так, щоб програмне забезпечення могло бути розроблене і перевірене на відповідність вимогам інтерфейсів.

8.1. Інтерфейси користувача.

Описуються інтерфейси користувача, які повинні бути реалізовані програмним забезпеченням.

8.2. Апаратні інтерфейси.

Визначаються всі апаратні інтерфейси, які повинні бути підтримані програмним забезпеченням, включаючи логічну структуру, фізичні адреси, очікувану поведінку тощо.

8.3. Програмні інтерфейси.

Описуються програмні інтерфейси з іншими компонентами програмної системи. Це можуть бути куповані компоненти, багато разів використовувані компоненти з іншої прикладної програми або компоненти, що розробляються для підсистем поза контекстом цих специфікацій вимог до програмного забезпечення, але з яким ця прикладна програма повинна взаємодіяти.

8.4. Комунікаційні інтерфейси.

Описуються всі комунікаційні інтерфейси до інших систем або пристроїв типу локальних мереж, видалених послідовних пристроїв тощо.

9. Вимоги до ліцензування.

Визначаються всі вимоги обов'язкового ліцензування або інші вимоги обмеження використання, які повинні виконуватися програмним забезпеченням.

10. Застереження щодо питань, пов'язаних з авторськими правами.

Описуються всі необхідні юридичні застереження, гарантії, оголошення про авторське право, право спадкоємства, торговельні марки або емблеми для програмного забезпечення.

11. Вживані стандарти

Вказуються посилання на всі вживані стандарти і на конкретні розділи таких стандартів, які відносяться до описаної системи. Наприклад, це можуть бути правові і регулюючі стандарти, стандарти якості, промислові стандарти щодо застосовності, здібності до взаємодії, інтернаціоналізації, відповідності операційній системі тощо.

У специфікації нефункціоналтних вимог вказують лише ті вимоги, які є суттєвими для проекту.

Специфікацію нефункціональних вимог з їх атрибутами треба навести в таблиці (табл. 13)

Таблиця 13

Специфікація нефункціональних вимог

Ідентифікатор вимоги	Назва вимоги	Атрибути вимог		
		Пріоритет	Трудність	Контакт
1. Застосовність				
SUPP1				
2. Надійність				
. . .				

Підрозділ 2.5. Проєктування інтерфейсу користувача

Інтерфейс користувача є своєрідним комунікаційним каналом, за яким здійснюється взаємодія користувача й програми. Щоб створити ефективний інтерфейс, потрібно розуміти, які завдання будуть вирішувати користувачі за допомогою цієї програми і які вимоги до інтерфейсу можуть виникнути у користувачів.

Загальні принципи проєктування інтерфейсу користувача:

1. Програма має допомагати виконувати завдання.

Це означає, що інтерфейс має бути легким для освоєння, а не перешкодою, яку користувач має подолати, щоб почати роботу.

2. Під час роботи із програмою користувач не має відчувати дискомфорт.

Для реалізації цього принципу необхідно:

забезпечити перевірку результатів якомога більшої кількості "некоректних" дій користувача, але не робити її повсюдно;

вказувати користувачеві, що саме йому робити, і виводити інформаційні повідомлення в ситуаціях, коли це дійсно необхідно;

надати досвідченим користувачам можливість вимкнення виведення інформаційних повідомлень;

добре продумувати зміст повідомлень, що виводяться користувачеві.

Широко відомі евристичні правила авторитетного американського фахівця в галузі проєктування інтерфейсів Якоба Нільсена [38]:

1. Видимість стану системи.

2. Відповідність між системою й реальним світом.
3. Управління користувачами та свобода їхніх дій.
4. Несуперечність і стандарти.
5. Запобігання виникненню помилок.
6. Впізнання, а не згадування.
7. Гнучкість і ефективність використання
8. Естетичний і мінімалістський дизайн
9. Допомога користувачам розпізнавати й виправляти помилки.
10. Довідка й документація.

Проєктування форм

Форми – це "будівельні блоки" інтерфейсу користувача.

Щоб створити добре спроєктовану форму, необхідно усвідомити її призначення, спосіб і час використання, а також її зв'язки з іншими елементами програми.

Особливий вид форм – це форми, призначені для введення даних. Під час розроблення основну увагу варто приділити швидкості їхнього використання.

Щоб прискорити процес введення даних, доцільно:

1. За можливості використовувати для додавання й редагування даних одну й ту ж саму форму.
2. Призначати клавіатурні сполучення для команд.
3. Не змушувати користувача "перестрибувати" з однієї частини форми в іншу.
4. Не ставити процес уведення даних у залежність від вмісту окремих елементів управління форми.
5. Використовувати засоби зворотного зв'язку з користувачем.

Ще одна важлива частина розроблення форм – створення змістовних і ефективних меню. От деякі важливі рекомендації:

1. Дотримуйтеся стандартних угод щодо розташування пунктів меню.
2. Групуйте пункти меню в логічному порядку й за змістом.
3. Для угруповання пунктів у меню, що розкриваються, використовуйте розділові лінії.
4. Уникайте надлишкових меню.
5. Уникайте пунктів меню верхнього рівня, які не мають меню, що розкриваються.

6. Не забувайте використовувати символ <…> для позначення пунктів меню, що активізують діалогові вікна.

7. За можливості використовуйте клавіатурні еквіваленти команд і "гарячі" клавіші.

Далі наведені деякі рекомендації із проєктування Web-інтерфейсу користувача:

1. Мінімізуйте зусилля, які необхідно зробити користувачеві для прийняття рішень щодо навігації.

2. За можливості використовуйте один екран. Це сприяє тому, щоб користувачі могли виконувати якомога більшу частину завдань, не використовуючи зайву навігацію.

3. Під час створення декількох сторінок поєднайте їх у розділи. Спростить переміщення між розділами за допомогою основного елемента управління навігацією та додаткового елемента управління для переміщення між розділами.

4. Переконайтеся, що ключові елементи, такі як меню, заголовки та інша інформація, що відображається на всіх сторінках, оформлені одноманітно з візуальної точки зору.

5. Завжди думайте про майбутнє розширення. Якщо в майбутньому очікується внесення в програмну систему нових функцій, спочатку необхідно вирішити, як буде розширюватися навігація для ефективного додавання нових екранів.

В цьому підрозділі необхідно розробити графічний інтерфейс користувача застосунку у вигляді вайрфрейму або мокапу, які використовуються для отримання схвалення зацікавлених осіб, щодо пропонованої концепції.

Інтерфейс користувача (UI - англ. User interface) забезпечує передачу інформації між користувачем-людиною і програмно-апаратними компонентами комп'ютерної системи.

Метою проєктування інтерфейсу є отримання ранньої реакції користувачів на пропоновану концепцію системи. Як інструментальні засоби використовуються Figma, PhotoShop, Marvel, Pencil Project тощо.

Вайрфрейм (англ. wireframe, каркас) – структурна схема розташування елементів інтерфейсу з прикладом контенту (текстом, ілюстраціями, таблицями) і виділенням акцентів, яка частково відображає роботу з продуктом. Вайрфрейм - це образ дизайну низької точності. Він має чітко показувати:

основні групи контенту (Що?);

структуру інформації (Де?);

опис і базову візуалізацію взаємодії між інтерфейсом і користувачем (Як?).

Вайрфрейм не містить графічного оформлення (візуального дизайну). Елементи інтерфейсу подають у спрощеному вигляді, наприклад, використовуючи "наповнювачі" - прямокутники, пересічені лініями хрест навхрест для зображень. Тому вайрфрейми зазвичай називають даними низької точності (lo-fi). Схема доповнюється текстовим документом із описом логіки роботи продукту і взаємодії користувача з ним.

Мокап (англ. mockup, макет) – це середньо- або високодеталізоване статичне уявлення дизайну. Передає структуру інформації, візуалізує зміст і демонструє основні функціональні можливості у вигляді статичних зображень. Дозволяє зрозуміти, як буде виглядати кінцевий продукт.

Мокап - неклікабельний, але більш візуально оформлений макет (практично вже дизайн). Він має:

показувати інформаційну структуру;

візуалізувати контент;

демонструвати базову функціональність у статиці;

дати можливість оцінити візуальну сторону проєкту.

РОЗДІЛ 3. ПРОЄКТНІ ТА ТЕХНІЧНІ РІШЕННЯ

Підрозділ 3.1. Обґрунтування архітектури програмної системи

Обґрунтоване створення архітектури системи – це проєктування на найвищому рівні. Логічна архітектура описує систему в термінах її принципової організації у вигляді компонентів, модулів, пакетів, програмних класів і підсистем.

Стандарт ISO/IEC/IEEE 42010:2022 [22] визначає архітектуру як «фундаментальні поняття або властивості системи в її середовищі, втілені в її елементах, зв'язках і в принципах її дизайну та еволюції».

Головна ідея архітектури полягає у тому, щоб знизити складність сприйняття системи внаслідок розмежування повноважень та створення чіткої структури. Архітектура та дизайн програмного забезпечення дозволяють створити чітку структуру, за якою зручно працювати

програмістам. Від її якості залежить, наскільки просто проходитиме обслуговування ПЗ, його зміни, доповнення та підтримка.

Терміном "Архітектура" також називають документування архітектури програмного забезпечення. Документування архітектури ПЗ спрощує процес комунікації між зацікавленими особами, дозволяє зафіксувати прийняті на ранніх етапах проєктування рішення про високорівневий дизайн системи і дозволяє використовувати компоненти цього дизайну і шаблони проєктування повторно в інших проєктах.

Таким чином, архітектурою є набір елементів що мають певну форму (властивості і обмеження що накладаються на елементи), і їх обґрунтування. Обґрунтування фіксує мотиви вибору певного архітектурного стилю, елементів і обмежень. Обґрунтування необхідне на етапі створення архітектури, і корисне надалі. Тому що архітектура має набір властивостей, що дозволяють їй задовольняти вимоги, і незнання цих вимог може призвести до змін що порушують архітектуру, але до архітектури входять властивості, а не вимоги. Архітектура повинна будуватись щоб краще відповідати вимогам до системи що створюється, згідно принципу "форма відповідає функції". Отже, проєктування архітектури ПЗ — це процес, що виконується після етапу аналізу і формулювання вимог.

Задача етапу проєктування архітектури ПЗ — перетворення вимог до системи у вимоги до ПЗ і побудова на їхній основі архітектури системи. Побудова архітектури системи здійснюється шляхом визначення цілей системи, її вхідних і вихідних даних, декомпозиції системи на підсистеми, компоненти або модулі та розроблення її загальної структури. Проєктування архітектури системи може проводитися різними методами (стандартизованим, об’єктно-орієнтованим, компонентним і ін.), кожний з яких пропонує свій шлях побудови архітектури, а саме, визначення концептуальної, об’єктної й інших моделей за допомогою відповідних конструктивних елементів (блок-схем, графів, діаграм тощо).

Етап архітектурного проєктування інформаційних систем може бути відображений в термінах опису архітектури на обраній мові опису архітектури (Architecture Description Language (ADL)). До таких мов відносять [60] - ArchiMate, Architecture Analysis & Design Language, C4 model (software), Darwin (ADL), EAST-ADL, Wright (ADL). Ці мови у своєї більшості базуються на мові UML.

Якщо мовою опису архітектури є UML, то проєктування архітектури відображається шляхом опису процесу об’єктно-орієнтованої декомпозиції системи до рівня переліку підсистем та їх зв’язків, опису її логічної структури у вигляді пакетів компонентів (діаграма компонентів).

Отже, обґрунтування архітектури у спрощеному вигляді може бути поділено на три етапи:

1) визначення важливих для архітектури вимог;
2) архітектурне проєктування (Architectural Design);
3) архітектурна документація.

У цьому підрозділі пояснювальної записки здобувачем мають бути виділені прийняті архітектурні рішення, які необхідно представити у вигляді таблиці – див. табл. 14.

На основі аналізу архітектурних вимог необхідно визначити основні принципи та рішення, які закладаються в основу майбутнього застосунка. Це, насамперед, еталонна архітектура, патерни розгортання, архітектурні патерни (стилі), архітектурні тактики, внутрішні та зовнішні компоненти. Кожне рішення повинно бути обгрунтоване з вказанням вимог, що задовольняються, у вигляді ідентифікаторів вимог, які були призначені в розділі 2 (табл. 12, 13).

У посилання на вимоги додаються ті з функціональних вимог, що були визначені на попередньому етапі, як такі, що мають вплив на архітектуру програмної системи (наприклад, наявність потоків даних, необхідність перетворення даних, отримання даних із зовнішніх джерел та інші. Також у посиланнях на вимоги можуть бути присутні атрибути якості - ті нефункціональні вимоги, які можуть вплинути на архітектуру - розширюваність системи, її доступність, здатність модифікувати, швидкодія, захищеність і т. д. На кінець у посиланнях повинні бути відображені архітектурні обмеження з групи проектних обмежень, що були сформовані у підрозділі 2.4.

Таблиця 14

Прийняті архітектурні рішення

Архітектурне рішення	Обґрунтування рішення	Посилання на вимогу

Закінчується цей підрозділ спроєктованими архітектурними відображеннями (Architectural Views) на обраній мові архітектурного проєктування (наприклад, UML діаграмами пакетів (компонентів)) з описом призначення структурних одиниць.

Підрозділ 3.2. Проєктування програмного забезпечення

Пункт 3.2.1. Вибір засобів реалізації проєкту

Вибір засобів реалізації проєкту передбачає обґрунтування мови (мов) програмування, програмного оточення проєктованої системи (насамперед, операційні системи, на базі яких буде працювати програмне забезпечення), системи управління базами даних, фреймворків та бібліотек, засобів кешування даних, брокерів повідомлень і т. д.

У пункті 3.2.1. необхідно обґрунтувати прийняті технічні рішення (стек технологій) для реалізації проєкту, а саме:

вибір основної мови програмування, або якщо необхідно, то декількох мов програмування;

вибір основного фреймворку та допоміжних бібліотек;

вибір системи управління базами даних;

програмне оточення та інших засобів, що використовуються в проєкті.

Для кожного технічного рішення повинні бути зазначені його призначення та альтернативні варіанти, що розглядалися. Рекомендується представити результати вибору у вигляді таблиці (табл. 15).

Таблиця 15

Вибір засобів реалізації проєкту

Назва засобу	Призначення	Альтернативні варіанти, що розглядалися	Обґрунтування вибору

Пункт 3.2.2. UML-діаграма класів (UML-діаграма діяльності)

Програмна система означає програмне забезпечення, що розробляється. Це може бути крупна колекція з багатьох компонентів програмного забезпечення, одна програма або частина програми.

У цьому пункті пояснювальної записки має бути:

1. Для об’єктно-орієнтованих програмних систем - UML-діаграма класів (Class Diagram), що реалізують основну бізнес-логіку програмної системи, та її короткий опис, у якому щонайменше необхідно навести призначення кожного класу. Для інших програмних систем - UML-діаграма діяльності (Activity Diagram), яка відбиває основну бізнес-логіку програмної системи, та її короткий опис.

2. Посилання на лістинг програми, де міститься вихідний код, який відповідає UML-діаграмам, наведеним у пункті 1. Сам лістинг програми має знаходитися в одному з додатків до пояснювальної записки.

Далі знаходяться методичні рекомендації щодо проєктування програмної системи.

Рекомендації з розроблення UML-діаграми класів (Class Diagram), що реалізують основну бізнес-логіку програмної системи.

Для уявлення статичної структури моделі програмної системи призначена UML-діаграма класів. Вона може відбивати, зокрема, різні взаємозв'язки між окремими сутностями предметної області, такими як об'єкти й підсистеми, а також описує їхню внутрішню структуру й типи відносин.

Клас у мові UML слугує для позначення безлічі об'єктів, які мають однакову структуру, поводження і відносини з об'єктами інших класів. Клас може не мати екземплярів або об'єктів. У цьому випадку він називається абстрактним класом. Графічно клас зображується у вигляді прямокутника, що додатково може бути розділений горизонтальними лініями на секції. У цих секціях можуть вказуватися ім'я класу, атрибути й операції.

Обов'язковим елементом позначення класу є його ім'я. Ім'я класу має бути унікальним у межах пакета, що описується деякою сукупністю діаграм класів. Воно вказується в першій верхній секції прямокутника. Рекомендується як імена класів використовувати іменники, записані без пробілів. Необхідно пам'ятати, що саме імена класів утворять словник предметної області. Прикладами імен класів можуть бути такі іменники, як "Співробітник", "Компанія", "Керівник", "Клієнт", "Продавець",

"Менеджер", "Офіс" та ті, що мають безпосереднє відношення до предметної області й функціонального призначення проєктованої системи.

У другій зверху секції прямокутника класу записуються його атрибути. Кожному атрибуту класу відповідає окремий рядок тексту, що складається із квантора видимості атрибута, імені атрибута, його кратності, типу значень атрибута й, можливо, його вихідного значення.

Квантор видимості може приймати одне із трьох можливих значень:

1. Загальнодоступний (public). Атрибут із цією областю видимості доступний із будь-якого іншого класу пакета, у якому визначена діаграма.
2. Захищений (protected). Атрибут із цією областю видимості недоступний для всіх класів, за винятком підкласів цього класу.
3. Закритий (private). Атрибут із цією областю видимості недоступний для всіх інших класів без винятку.

Ім'я атрибута становить рядок тексту, що використовується як ідентифікатор відповідного атрибута й тому має бути унікальним в межах цього класу. Ім'я атрибута є єдиним обов'язковим елементом синтаксичного позначення атрибута.

Кратність атрибута характеризується загальною кількістю конкретних атрибутів певного типу, що входять до складу окремого класу.

Тип атрибута становить вираз, семантика якого визначається мовою специфікації відповідної моделі. У нотації UML тип атрибута іноді визначається залежно від мови програмування, яку передбачається використовувати для реалізації цієї моделі. У найпростішому випадку тип атрибута вказується рядком тексту, що має осмислене значення в межах пакета або моделі, до яких ставиться розглянутий клас.

У третій зверху секції прямокутника записуються операції класу. Операція становить деякий сервіс, що надає будь-який екземпляр класу на певну вимогу. Сукупність операцій характеризує функціональний аспект поводження класу. Кожній операції класу відповідає окремий рядок, що складається із квантора видимості операції, імені операції, вираження типу, що повертається операцією.

Ім'я операції становить рядок тексту, що використовується як ідентифікатор відповідної операції й тому має бути унікальним у межах цього класу.

Крім внутрішнього устрою або структури класів на відповідній діаграмі вказуються різні відношення між класами. Водночас, сукупність типів таких відношень фіксована в мові UML і визначена семантикою їхніх типів.

Базовими відношеннями в мові UML є:

1. Відношення залежності.
2. Відношення асоціації.
3. Відношення узагальнення.
4. Відношення реалізації.

Рекомендації з розроблення UML-діаграми діяльності (Activity Diagram), яка відбиває основну бізнес-логіку програмної системи.

Діаграми діяльності використовуються, щоб показати потік управління в системі та кроки, що виконуються в процесі виконання варіанта використання. Використовуючи діаграми діяльності, можна моделювати послідовні та паралельні дії. Діаграма діяльності фокусується на умовах та послідовності виконання деякого потоку.

Діаграма діяльності зображує потік управління від стартової точки до кінцевої точки, показує різні шляхи прийняття рішень, які існують під час виконання діяльності. Вона використовується для моделювання робочого процесу з урахуванням умов, обмежень, послідовних та одночасних дій.

Щоб розробити діаграму діяльності, необхідно визначити:

1. Початковий стан та кінцевий стан.
2. Проміжні дії, необхідні для досягнення кінцевого стану із початкового стану.
3. Умови або обмеження, які спонукають систему змінити потік управління.

Пункт 3.2.3. Файлова структура проєкту програмного продукту (застосунку)

В цьому пункті необхідно викласти вміст файлової структури розробленого програмного продукту, тобто описати призначення папок і файлів, що утворюють вихідний код проєкту.

Структура наводиться у вигляді переліків, де кожен пункт визначає окрему структурну одиницю і надає опис призначення та вмісту цієї одиниці. Якщо програмний продукт має розподілену архітектуру (наприклад, клієнт-серверну), то наводиться структура окремо клієнтської та серверної його частин. Починати рекомендується зі

структури папок проєкту (частини проєкту), яку можливо зобразити у вигляді «дерева». І, далі, викладати вміст папок у вигляді переліків.

Приклад опису вмісту папки:

Папка config містить наступні файли:

- .editorconfig — файл налаштування робочої ide на якій виконувався проект;

- .eslintignore — файл конфігурації винятків для плагіну es-lint;

- .eslintrc — файл конфігурації правил плагіну es-lint;

- .gitignore — файл конфігурації винятків для системи контролю версій;

- .prettierrc — файл конфігурації правил плагіну prettier;

- package-lock.json — технічна інформація про встановлені модулі проекту;

- package.json — перелік залежностей проекту та допоміжних скриптів.

Підрозділ 3.3. Проєктування моделі даних

Початковими даними для проєктування моделі даних є:

опис функціональної схеми підприємства (за наявності), для якого виконується проєкт;

виявлення місця та функцій конкретного підрозділу в структурі усієї організації (підприємства);

опис технології обробки інформації у рамках цієї предметної області конкретного підрозділу з зазначенням вхідної та вихідної інформації, вирішуваних завдань, функцій і регламенту роботи виконавців обробки інформації, періодичності виконання функцій із обробки інформації;

аналіз наявних засобів автоматизації обробки інформації у рамках цієї предметної області (апаратне та програмне забезпечення, СКБД);

обґрунтування необхідності та можливості розроблення модуля автоматизації обробки інформації і бази даних цього модуля як складового елементу загальної інфраструктури даних організації;

аналіз і опис завдань, що автоматизуються в модулі (підсистемі), що розробляється.

Логічна модель (logical data model) даних визначає: набір підтримуваних типів структур даних; набір допустимих операцій над підтримуваними структурами даних; набір загальних правил цілісності

даних, що явно або неявно визначають коректний стан бази даних або їхні зміни;

БД складається з двох частин: транзакційної та аналітичної. Характерними рисами транзакційної частини є:

1) реляційна структура (переважно);

2) можливість накопичення значних обсягів фактичних даних;

3) можливість виконання операцій додавання, видалення, редагування записів;

4) відсутні агреговані (обчислені) дані;

5) використовується для виконання різноманітних операцій обліку;

6) є основою для розроблення аналітичної частини БД.

Аналітична частина БД використовується в процесі виконання оперативного аналізу інформації і розроблення моделей для систем підтримки прийняття рішень. Характерними рисами аналітичної частини БД є:

1) багатовимірна структура (підтримка моделей MOLAP, ROLAP, HOLAP);

2) зберігання агрегованих даних;

3) відсутність можливості виконання операцій видалення і редагування даних і накопичення їх, переважно, за хронологією.

У процесі розроблення транзакційної частини БД можна використати структурне або об'єктно-орієнтоване моделювання, застосовуючи відповідні CASE-інструменти: ErWin Data Modeler, IBM Rational Software тощо. Під час побудови сховища даних необхідно обґрунтувати вибір моделі зберігання (кубічна модель MOLAP або ROLAP: "зірка" або "сніжинка").

Якщо тема дипломного проєкту, пов’язана зі створенням складних транзакційних баз даних для інформаційної системи, рекомендується в цей підрозділ долучати проєктування концептуальної, логічної та фізичної моделей даних. В інших випадках можна обмежитися проєктуванням концептуальної і логічної моделей даних.

Пункт 3.3.1. Концептуальне інфологічне проєктування

У цьому пункті виконується побудова моделі даних, незалежної від СКБД, яка охоплює створення словника даних і глобальної інфологічної моделі даних.

Словник даних. На основі аналізу вхідних та вихідних документів будується модель відображення множини реквізитів вихідних і вхідних

документів на множину елементів даних, що підлягають збереженню у базі даних, потім виконується приведення зібраної інформації до вигляду, зручного для проєктування. Для цього складають словник даних (табл. 16).

Таблиця 16

Словник даних

№ за/п	Найменування елемента	Ідентифікатор	Тип і довжина	Призначення елемента

Елементи даних словника наводяться в алфавітному порядку за графою "Найменування елемента" з метою подальшого вилучення омонімів та дублювальних елементів. Якщо словник вміщує багато елементів, його виносять у додаток. У полі "Призначення елемента" необхідно вказати: елемент збереження є фактичним чи обчислюваним.

Під час проєктування глобальної інфологічної моделі даних необхідно здійснити виявлення еквівалентних сутностей та їхнє злиття, виявлення категорій і синтез узагальнювальних сутностей, виявлення й усунення дублювання атрибутів і зв'язків. Будується графічне подання глобальної моделі у вигляді ERD (нотація IDEF1X), діаграми класів або інших нотацій.

Для всіх сутностей розроблюваної системи слід навести специфікації обмежень цілісності та операційних правил, а саме:

1) обмеження атрибутів сутностей (табл. 17);
2) обмеження кортежів;
3) обмеження унікальності;
4) динамічні обмеження;
5) інші обмеження;
6) операційні правила;
7) правила посилальної цілісності.

Таблиця 17

Обмеження атрибутів сутностей

№ за/п	Ім'я атрибута або агрегату	Межі / допустимі значення	Структура (формат)	Умова	Значення за замовчуванням

Пункт 3.3.2. Проєктування логічної (фізичної) моделі даних

Проєктування логічної моделі даних містить: графічне подання логічної моделі у вигляді ERD (в нотації IDEF1X), діаграми класів тощо.

Обґрунтування властивостей моделі бази даних містить: опис засобів, інструментів, методів, які застосовувалися для забезпечення таких властивостей бази даних: функціональна повнота; мінімальна надмірність; цілісність бази даних (таблична, посилальна цілісність, забезпечувана ключами і тригерами тощо); узгодженість; актуальність; безпека; відновлюваність; логічна та фізична незалежність; ефективність.

Для випадку використання реляційних СУБД наводиться логічна модель у вигляді ER діаграми, що включають сутності (таблиці), атрибути (колонки / поля) та відношення (ключі). Якщо використовується ORM технологія, то ER діаграми замінює діаграма класів, яка представляє класи, що відображають модель даних. При використанні NoSQL СУБД (наприклад, MongoDB) наводиться схема даних для колекцій та зв’язки (relation) між колекціями, якщо такі існують.

Підрозділ 3.4. Захист інформації

Підрозділ захисту інформації має на увазі розробку елементів політики безпеки на інформаційну систему і програмний продукт, що розробляється. Політика безпеки інформаційної системи повинна в себе включати наступні пункти:

Перелік ресурсів і циркулюючої інформації (даних), які є критичними (конфіденційними) для цій інформаційній системі. Кожен ресурс або інформація повинні супроводжуватися коротким обґрунтуванням, чому їх варто відносити до конфіденційних даних. Тут слід виділити інформацію, яка потрапляє під дію наступних законів:

закону України «Про інформацію», в якому дається поняття конфіденційної інформації [40];

закону України «Про захист персональних даних», в який визначає, що таке персональні дані і регламентує правила їх обробки [40];

закону України «Про банки і банківську діяльність», в якому регламентується правила використання банківської інформації [41].

Перелік конфіденційної інформації, циркулюючої в ІС, рекомендується оформити у вигляді таблиць (див. табл. 18 - 19). Відносно кожного виду інформації вказати бізнес процеси, в яких вона

використовується, і в колонці «обґрунтування» вказати, чому її варто відносити до конфіденційної інформації.

Таблиця 18

Перелік конфіденційної інформації циркулюючої в системі

Назви інформації	Назва бізнес процесу в якому використовується	Обґрунтування

Таблиця 19

Перелік критично важливих ресурсів інформаційної системи

Назва ресурсу	Назва бізнес процесу, в якому використовується	Обґрунтування

1. Перелік осіб, що мають доступ до конфіденційної інформації, циркулюючої в інформаційній системі. У цьому пункті слід розписати права доступу до ресурсів бази даних відносно кожного користувача, способи і види доступу до ресурсів мережі інформаційної системи, а також вказати доступність ресурсу з Інтернету. У якості доступності ресурсів в мережі та Інтернеті потрібно вказати необхідні порти і протоколи, що вимагаються для правильного функціонування ресурсу.

Перелік осіб, які обробляють конфіденційну інформацію, представити у вигляді таблиці (див. табл. 20). Для зменшення кількості рядків, можна групувати інформацію, що має однакові права доступу.

Таблиця 20

Перелік користувачів, що мають доступ до конфіденційної інформації і критично важливих ресурсів

Користувач ІС	Конфіденційна інформація або ресурс	Права доступу					
		Читання	Запис	Оновлення	Видалення	Архівація	Відновлення
Адміністратор	Паролі до бази даних	+	+	+	-	+	+

2. Здійснити аналіз інформаційної системи з точки зору криптоаналітика. Виявити слабкі компоненти системи і сформувати перелік можливих загроз безпеки. Слід класифікувати загрози безпеки відносно основних послуг безпеки, таких як: конфіденційність, цілісність, доступність, причетність, спостережуваність, автентичність та інш. Перелік загроз представити у вигляді таблиці 21.

Таблиця 21

Перелік загроз безпеки інформаційної системи

Назва загрози	Де виникає (бізнес процес, ресурс)	На який ресурс або інформацію спрямована	Яка послуга безпеки порушується

3. У таблиці 19 відносно кожної загрози слід вказати мету загрози і можливе місце виникнення / проникнення загрози. Метою загрози можуть бути певні ресурси системи, робота яких може бути порушена у випадки її проходження. Це проводиться для можливості оцінки завданого збитку, а також для правильного вибору механізмів захисту. Під місцем виникнення / проникнення розуміється ресурс системи або бізнес процес, в якому може виникнути загроза або через який може проникнути загроза. Виявлення таких тонких місць в ІС дозволить обґрунтовано застосовувати механізми безпеки для відповідних ресурсів системи.

4. Визначити відносно кожного механізму безпеки, якими засобами він може бути реалізований і на перекриття яких загроз спрямований. Під засобами захисту розуміються конкретні програмні продукти або спеціалізовані сервісні функції ОС, Web серверів, баз даних. Перелік механізмів безпеки і засобів захисту представити у вигляді таблиці 22.

Таблиця 22

Перелік механізмів безпеки, що перекривають загрози безпеки ІС

Механізм безпеки	Засіб захисту	Загроза безпеки

5. Дати детальний опис засобів захисту використовуваних в програмному продукті, що розробляється. Тут слід зазначити використовувані криптографічні методи захисту, методи автентичності користувачів, спосіб зберігання паролів у системі, засоби автентичності при підключенні до бази даних, використання захищених протоколів SSL і так далі.

6. Для розрахованих на багато користувачів систем, слідує, розглянути можливість реалізації послугу безпеки спостережуваності у вигляді логування основних дій користувача при роботі з базою даних і основних подій системи, таких як підключення до бази даних, вхід та вихід користувача і т.п.

Підрозділ 3.5. Процес неперервної інтеграції CI та неперервної доставки програмного CD забезпечення або його компонентів

Цей підрозділ присвячений процесам CI/CD, які стосуються програмного забезпечення, що було спроектоване та розробляється. Зміст та структура цього підрозділу може бути скоригований в залежності від конкретного проєкта за погодженням з керівником дипломного проєкта.

Метод неперервної інтеграції (continuous integration, CI) вимагає від розробника періодичної фіксації змін коду в репозиторії версій GIT. Рекомендовано публікувати правки не рідше одного разу в день. Підхід допомагає легше і швидше виявляти помилки та інші проблеми з якістю розробки, своєчасно їх усунути, без затримки в основних процесах. Ціль використання неперервної інтеграції – забезпечити погоджений і автоматизований спосіб створення і тестування програмного продукту.

Неперервна доставка (Continue Delivery автоматизує процес впровадження програмного продукту і внесення змін в код, у підготовлену серверну інфраструктуру. Розробники створюють програмне забезпечення із використанням середовища, відповідних для певних етапів роботи. Ключова особливість ПО - автоматизація процесу

доставки змінених для всіх використовуваних середовищ і реалізація необхідних додаткових механізмів (наприклад: відправлення запиту на сервер, виконання SQL-запитів, налаштування повідомлень або перезапуск, і т.п.).

Інструменти CI / CD. Ідея неперервної інтеграції базується на використанні підтримуючих цей процес інструментів. Одним із них – система керування вихідним кодом (SCM). ПО застосовується для відстеження змін в вихідному коді, допомагає командам розробників об'єднувати зміни, створенні різними розробниками (системи – Bitbucket, Github, GitLab).

Інші інструменти – системи, підтримуючі створення, тестування і впровадження в режимі неперервної інтеграції (системи – Gitlab CI, Jenkins, CircleCI, Travis та інші).

Процес розробки починається з вибору стратегії розгалуження в Git. Стратегія розгалуження – важливий робочий інструмент під час розробки програмного забезпечення. Існують декілька основних стратегій:

GitHub Flow;

GitFlow;

Forking Workflow;

GitLab Flow;

Trunk Based Development.

У рамках цих стратегій необхідно прийняти які гілки будуть використовуватися, для яких цілей, як часто будуть проводитись злиття коду, коли буде формуватися реліз, які додаткові концепції можуть ще використовуватись. Для цього складають перелік типів гілок (табл. 23).

Таблиця 23

Словник даних

№ п/п	Найменування / шаблон гілки	Призначення	Частота інтеграції

Інфраструктура.

В роботі необхідно зробити обґрунтування вибору типу інфраструктури локальна (On-Premise) або хмарна (Cloud) за основними факторами які максимально впливають на вибір рішення (див. таб. 24).

Таблиця 24

Основні фактори локальної або хмарної інфраструктури

Фактор	Локальне середовище (on-premise)	Хмарне середовище (cloud)	Що необхідно виконати
1	2	3	4
Витрати	Несуть відповідальність за поточні витрати на серверне обладнання, енергоспоживання та простір.	Потрібно платити лише за необхідні ресурси, без будь-яких витрат на технічне обслуговування та утримання, а ціна коригується в залежності від того, скільки споживається потужностей	Необхідно виконати розрахунок витрат на підтримку інфраструктури та середнє добове, місячне та квартальне споживання ресурсів для кожного оточення: DEV, QA, PROD
Контроль та безпека	Бізнес зберігає всі свої дані та повністю контролює, що з ними відбувається. Через особливі вимоги до конфіденційність компанії/організації із строго регульованих галузей часто не наважуються перейти в хмару.	Багато компаній і постачальників переймаються проблемою володіння даними; дані та ключі шифрування знаходяться у вашого стороннього постачальника і якщо станеться непередбачуване і виникне час простою, є можливість вчасно не отримати доступ до цих даних	Виявити компоненти інфраструктури таки як канали передачі, сервіси обробки та зберігання даних; навести відповідні вимоги щодо захисту даних у цих компонентам (конфіденційність, цілісність, доступність, спостережуваність); навести механізми безпеки що забезпечують ці вимоги
Реалізація	Компанія несе відповідальність за підтримку рішення та всіх пов'язаних із ним процесів	Ресурси розміщуються на території постачальника послуг чи-то приватна, публічна або гібридна хмара	Описати які ресурси розміщуються (віртуальні машини з різними операційними системами їх характеристики), які сервери або кластери використовуються (Docker Swarm, Kubernetes)

Закінчення табл. 24

1	2	3	4
Нормативні вимоги	Для урядових компаній, а також для низки підприємств, галузеві та нормативні документи яких суворо регламентуються, дуже важливо, щоб зберігання даних чітко відповідало вимогам закону.	Повинні виявити належну обачність та переконатися, що їхній сторонній постачальник відповідає всім різноманітним нормативним вимогам у своїй галузі; Конфіденційні дані мають бути захищені, а конфіденційність клієнтів, партнерів та співробітників має бути забезпечена; Угода про рівень послуг (Service-level agreement) має важливе значення для забезпечення розуміння сторонами, постачальником та клієнтом, єдиного стандарту обслуговування та послуг, у тому числі вимог доступності.	Проаналізувати нормативні вимоги що накладаються на ваш тип компанії/організації відносно правил зберігання даних, для цього треба керуватися національними та міжнародними законами і правовими актами такими як закон України “Про захист персональних даних”, Загальний регламент про захист даних (GDPR) та інші.

В проєкті слід розглянути як будується інфраструктуру для кожного середовища (environment). Показати яка частина інфраструктури буде створюватись та підтримуватись командою DevOps тобто буде формувати платформу, а яка частина буде налаштовуватись у відповідності до розгортання сервісів програмного забезпечення на яку має вплив команда розробників та тестувальників. Платформа може надаватись або Cloud провайдером, або розгортатись на локальній інфраструктурі. Слід обґрунтувати одну або декілька інфраструктур, що будуть використовуватись для відповідного середовища.

IaaS (Infrastructure-as-a-Service). За цієї моделі споживач отримує інформаційно-технологічні ресурси – віртуальні сервери з певною обчислювальною потужністю та обсягами пам'яті. Усім "залізом" займається провайдер. Він встановлює на нього ПЗ для створення віртуальних машин, але не займається встановленням і підтримкою ПЗ користувача. Провайдер контролює тільки фізичну та віртуальну інфраструктуру. Приклади IaaS: IBM Softlayer, Hetzner Cloud, Microsoft Azure, Amazon EC2, GigaCloud, GCP, та для локальної інфраструктури

Huper-V Windows Server, KVM Ubuntu Server, Cent OS Server, Debian та інші.

PaaS (Platform-as-a-Service) – у цьому разі хмарний провайдер надає доступ до операційних систем, засобів розробки та тестування, систем управління базами даних. Провайдер контролює не лише сервери, системи зберігання даних та обчислювальні потужності, а й також пропонує користувачеві на вибір певні платформи та засоби управління ними. Приклади PaaS: Google App Engine, IBM Bluemix, Microsoft Azure, VMWare Cloud Foundry та для локальної інфраструктури розгортання різних кластерів в якості платформи це: кластерні бази даних, кластерні вебсервери.

Containers-as-a-Service (CaaS) – віртуалізація обчислювальних потужностей, ресурсів та інструментів в одному контейнері. Також послуга дає змогу керувати компонентами за допомогою віртуального середовища. Приклади CaaS: Docker, Podman, Containerd, LXC, Docker Swarm, Kubernetes.

SaaS (Software-as-a-Service) – це коли програми та сервіси розробляє та обслуговує провайдер, розміщує їх у хмарі та пропонує кінцевому користувачеві через браузер або застосунок на його ПК. Клієнт лише вносить абонплату (або користується сервісом безоплатно), оновленням і технічною підтримкою програм займається провайдер. Приклади SaaS: зберігання файлів (S3, Dropbox, Google Disk, One Drive), офисный пакет документов для работы (Google Doc, Microsoft Office 365).

Під час вибору інфраструктури треба розглядати такі властивості як вартість володіння та експлуатації, продуктивність, масштабованість, доступність, надійність, скальованість, безпека, управління та моніторинг.

Таблиця 25

Характеристика інфраструктур

№ п/п	Властивість інфраструктури	Призначення	Тип інфраструктури	Середовище в якому буде застосовуватись

Наступним кроком слід розглянути види середовищ та надати їх опис, таких як локальне середовище розробника (local environment або work environment), середовище розробки (development environment, DEV ENV), середовище тестування (QA environment) та виробниче середовище (production environment).

Локальне середовище розробника – це робоча станція розробника на якої безпосереднє працює розробник і на якому встановлено усі необхідні компоненти для підтримки процесу розробки. Це можуть бути утілити, IDE, plugins для IDE, сервери, набір бібліотек, текстові редактори, та інші інструментальні засоби розробки.

Середовище контролю якості (QA Environment), також відоме як тестове середовище, – це тестова установка, що складається з мікропрограми, програмного забезпечення, обладнання та відповідної мережевої конфігурації. Всі тести виконуються на цьому тестовому стенді, що містить всю інфраструктуру, необхідну для проведення тестів.

Під виробничим середовищем розуміється місце, де програмне забезпечення або продукти були запущені в експлуатацію для використання передбачуваними користувачами. Коли щось потрапляє у виробниче середовище, усі помилки мають бути вже виправлені, а продукт або оновлення мають працювати ідеально.

Для кожного середовища необхідно описати усі сервіси що будуть розгортатись на базі обраної інфраструктури та їх характеристики (налаштування). Опис середовищ наведено у табл. 26.

Таблиця 26

Характеристика середовищ

№ п/п	Середовище	Сервіс	Кінцеві точки	Призначення

Конвеєр DevOps.

Конвеєр DevOps – це набір автоматизованих процесів та інструментів, який дає змогу розробникам і фахівцям з експлуатації злагоджено працювати над створенням і розгортанням коду у виробничому середовищі. Хоча конвеєр DevOps може відрізнятися залежно від організації, він зазвичай містить автоматизацію збірки/безперервну інтеграцію, автоматичне тестування, перевірку і звітність. Він також може містити одні або кілька воріт із ручним

керуванням, які вимагають втручання людини, перш ніж код зможе продовжити роботу.

Кожен проект має свій набір технологій, який може вплинути на процес. Конвеєр можна представити у вигляді діаграми рис. 1.

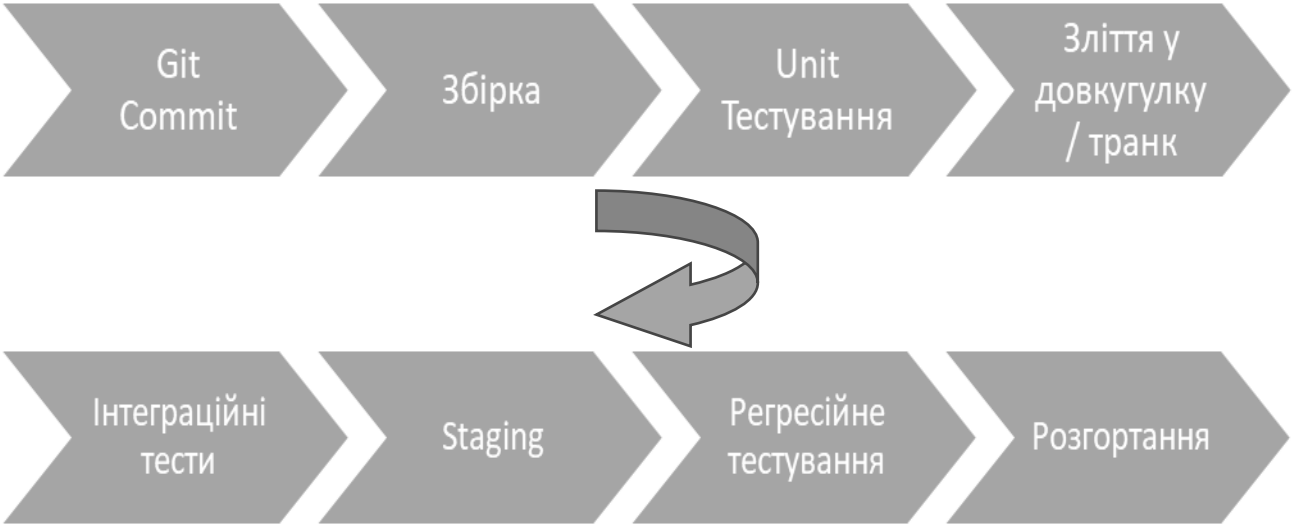

Рис. 1. Приклад етапів конвеєра DevOps.

Хоча кожен конвеєр унікальний, більшість проєктів використовують схожі фундаментальні компоненти. Кожен крок оцінюється на успіх, перш ніж перейти до наступного етапу конвеєра. У разі збою конвеєр зупиняється і розробнику надається зворотний зв'язок.

В проекті треба зробити опис кожного етапу конвеєра які ви вважаєте необхідними для розгортання програмного продукту. Описати скрипти, налаштування відповідних інструментів, надати інструкцію по запуску конвеєра DevOps. Всі ці налаштування також можуть зберігатись і відповідному Git репозиторій. Для побудови конвеєру DevOps можуть використовуватись такі системи як: Jenkns, GitLab CI, GitHub Actions. Відповідні скрипти повинні бути додані до додатків дипломного проекту.

РОЗДІЛ 4. ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Метою розділу 4 є створення документації, необхідної для тестування програмного забезпечення і аналізу результатів тестування, що включає чек-листи, тест-кейси, тестові набори даних, а також звіт про дефекти і звіт про результати тестування. З урахуванням теми дипломного проєкту і за узгодженням з керівником в цьому розділі можуть бути подані результати інших видів тестування.

Підрозділ 4.1. Створення чек-листів

В цьому підрозділі треба скласти чек-листи для димового тестування та тестування критичного шляху. Чек-лист - це список, який містить необхідні перевірки (тест-кейси) під час тестування програмного забезпечення. Призначення чек-листа:

1. Не пропустити необхідні тести.
2. Розподілити завдання за рівнями кваліфікації.
3. Зберігати звітність та результати тестування.

Пункт 4.1.1. Чек-лист для димового тестування

Димове тестування (Smoke test) направлено на перевірку найголовнішої, найважливішої, самої ключової функціональності, непрацездатність якої робить безглуздою саму ідею використання програмного забезпечення. Не слід проводити більш глибокий тест, поки не будуть виконані smoke-тести на 100%!

Для створення чек-листа для димового тестування треба визначити:

для чого призначений ваше програмне забезпечення;

які найбільш очевидні прості кроки необхідно виконати, щоб в нього потрапити;

які мінімальні важливі кроки і в якій послідовності треба виконати, щоб досягти мети.

Чек-лист для димового тестування створюється в таблиці (табл. 27).

Таблиця 27

Чек-лист для димового тестування

Номер	Опис	<Оточення>	Статус	Коментар
1				
2				
. . .				

Список тестів повинен бути мінімальним – до 10-15, та бути лінійним, без ієрархічного впорядкування.

Оточення перевірки. Як правило в кожен чек-лист додають кілька стовпців, призначених для тестування на окремій платформі. В назві цих стовпців вказують назву пристрою, браузера і його версії тощо.

Під час тестування в чек-листі зазначається статус навпроти кожного тестового пункту. Приклади статусів наведені в табл. 28.

Таблиця 28

Статуси тесту

№	Сатус	Коментар
1	2	3
1	Passed / Пройдений	Перевірка пройдена успішно, багів, не знайдено
2	Failed/ Невдалий	Знайдений один або більше багів
3	Blocked/ Заблокований	Неможливо перевірити, тому що один з багів блокує поточну перевірку
4	Not run/ Не виконаний	Ще не перевірено
5	Skipped / Пропущений	Не буде перевірки з будь-якої причини. Наприклад, поточний функціонал ще не реалізований.
6	In Progress / Виконується	Поточний пункт, над яким працює тестувальник

На основі створеного чек-листа треба виконати перевірку створеного програмного забезпечення. В процесі проходження тестів змінюйте статус тесту у відповідності до табл. 12. При виявленні дефектів вставте примітку до відповідного пункту з посиланням на сторінку з багом. Усі помилки знайдені під час цього етапу тестування треба виправити!

Пункт 4.1.2. Чек-лист для тестування критичного шляху

Тестування критичного шляху направлено на дослідження функціональності, використовуваної типовими користувачами в типовій повсякденній діяльності. Існує більшість користувачів, які найчастіше використовують якусь підмножину функцій програми. Саме ці функції і потрібно перевірити, після димового тестування. Якщо з якихось причин програма не виконує ці функції або виконує їх некоректно, дуже багато користувачів не зможуть досягти безлічі своїх цілей. Граничне значення успішного проходження тесту критичного шляху нижче, ніж в димовому тестуванні але однаково досить високе (зазвичай, близько 70-90% - в залежності від суті проекту).

Для створення чек-листа для тестування критичного шляху треба визначити основні кроки в роботі з програмним забезпеченням, які здійснюють більшість користувачів для досягнення основної мети його використання і далі розписати послідовність дій, які треба виконати на кожному кроці.

Чек-лист для тестування критичного шляху створюється в таблиці (табл. 29).

Таблиця 29

Чек-лист для тестування критичного шляху

Номер	Опис	Статус	Коментар
1			
1.1			
1.2			
. . .			
2			
2.1			
. . .			

Підрозділ 4.2. Створення тест-кейсів

Для тестування критичного шляху треба розробити тест-кейси відповідно перевірок, визначених у чек-листі. Опис тест-кейсів для тестування критичного шляху подати в таблиці за шаблоном (табл. 30).

Таблиця 30

Шаблон для створення тест-кейсів

№	№ вимоги	Назва	Пріоритет	Опис тест-кейсу	Очікуваний результат	Коментар

Вимоги до змісту тест-кейса.

Ідентифікатор.

Унікальний.

Осмислений (якщо дозволяє ПЗ).

Пов'язана з тест-кейсом вимога.

Вказує на основну вимогу для перевірки якої створений тест-кейс.

Наявність цього поля покращує таку властивість тест-кейса як відстеження.

Назва (суть) тест-кейса

Покликана спростити швидке розуміння основної ідеї тест-кейса без звернення до його інших атрибутів.

Це поле є найбільш інформативним при перегляді списку тест-кейсів.

Назва завжди має бути у кожного тест-кейса! Ні за яких умов категорично не допускається наявність тест-кейсів без назв!

Пріоритет.

Показує важливість тест-кейса.

Може бути виражений буквами (A, B, C, D, E), цифрами (1, 2, 3, 4, 5), словами («вкрай високий», «високий», «середній», «низький», «вкрай низький») або іншим зручним способом.

Може корелювати з:

важливістю вимоги;

потенційної важливістю дефекту, на пошук якого спрямований тест-кейс.

Опис тест-кейсу містить:

вхідні дані, необхідні для виконання тест-кейсу.

Дозволяють описати все те, що повинно бути підготовлено до початку виконання тест-кейса, наприклад, стан бази даних, стан файлової системи і її об'єктів і тощо.

Все, що описується в цьому полі, готується без використання тестової програми: якщо тут виникають проблеми, не можна писати звіт про дефект в програмному забезпеченні;

кроки тест-кейса.

Кроки тест-кейса описують послідовність дій, які необхідно реалізувати в процесі виконання тест-кейса, і нумеруються.

Очікувані результати.

Очікувані результати по кожному кроку тест-кейса описують реакцію програми на дії, описані кроках тест-кейса. Номер кроку відповідає номеру результату.

Підрозділ 4.3. Створення наборів вхідних даних для тестування

Для тест кейсів, що перевіряють роботу програмного забезпечення з наборами вхідних даних треба створити тестовий набір для введення даних на основі класів еквівалентності і граничних умов.

Клас еквівалентності - набір даних, що обробляється однаковим способом й призводить до однакового результату.

Граничні умови - місця (значення), в яких один клас еквівалентності переходить в інший.

Ознаки еквівалентності тест-кейсів:

тест-кейси спрямовані на пошук однієї і тієї ж помилки;

якщо один з тест-кейсів виявляє помилку, інші її теж, швидше за все, виявляють;

якщо один з тест-кейсів не виявляє помилку, інші її теж, швидше за все, не виявляють;

тест-кейси використовують схожі набори вхідних даних;

для виконання тест-кейсів здійснюються одні і ті ж операції;

тест-кейси генерують однакові вихідні дані або призводять програмне забезпечення в один і той же стан;

всі тест-кейси призводять до спрацьовування одного і того ж блоку обробки помилок («error handling block»);

жоден з тест-кейсів не призводить до спрацьовування блоку обробки помилок («error handling block»).

Під час створення тестових наборів даних треба навести класи еквівалентності:

за довжиною поля: для неприпустимої довжини та припустимої довжини;

за символами: для неприпустимих та припустимих символів.

На основі виділених класів еквівалентності визначити набори даних для позитивного і негативного тестування. Результати подати в таблиці (табл. 31).

Таблиця 31

Набори даних для тестування

	Позитивні тест-кейси		Негативні тест-кейси			
Значення						
Пояснення						

Для позитивних тест кейсів це:

рядок мінімальної припустимої довжини;

рядок максимальної припустимої довжини.

Для негативних тест кейсів це:

рядок неприпустимої довжини за нижньою межею;

рядок неприпустимої довжини за верхньою межею;

рядок припустимої довжини з недопустимими символами

можна додати рядок з неприпустимою довжиною / недопустимими символами для надійності.

Підрозділ 4.4. Звіт про дефекти

За результатами тестування критичного шляху програмного забезпечення на основі чек-листа, тест-кейсів і наборів даних створюється звіт про дефекти. Звіт про дефекти - документ, в якому наводиться інформація про очікуваний і фактичний результат тестування і який описує і визначає пріоритет виявленого дефекту, а також сприяє його усуненню. Коли виявляється дефект, обов'язково необхідно скопіювати скриншот з дефектом и вставити його або посилання на нього в додатки звіту (табл. 32).

Таблиця 32

Звіт про дефекти

Іденти-фікатор	Короткий опис	Докладний опис	Кроки відтворення	Відтворюваність	Важли-вість	Терміновість	Симптом	Можливість обійти	Комен-тар	Додатки

Дефект - розбіжність очікуваного і фактичного результату.

Очікуваний результат - поведінка системи, описана в вимогах.

Фактичний результат - поведінка системи, яка спостерігається в процесі тестування.

Вимоги до змісту звіту про дефект.

Ідентифікатор.

- Унікальний.
- Осмислений (якщо дозволяє ПЗ).

Короткий опис.

В лаконічній формі дає вичерпну відповідь на питання «Що сталося?» «Де це сталося»? «За яких умов це сталося?»

Наприклад, «відсутній логотип на сторінці привітання, якщо користувач є адміністратором».

Докладний опис.

Дає в розгорнутому вигляді необхідну інформацію про дефект, а також (обов'язково!)

- опис фактичного результату,
- опис очікуваного результату,
- посилання на вимогу (якщо це можливо).

Кроки відтворення.

Описують дії, які необхідно виконати для відтворення дефекту. Дана інформація в звіті про дефект є вкрай важливою. Саме вона дозволяє розробнику швидко відтворити і усунути проблему.

Відтворюваність.

Показує, чи відтворюється дефект завжди («always») або лише іноді («sometimes»). Дефекти, що відтворюються завжди, набагато простіше діагностувати і виправляти.

Рекомендація: відтворіть свої кроки хоча б 2-3 рази перш, ніж писати, що дефект відтворюється завжди. Довести, що дефект існує – завдання тестувальника! Тому відразу ж, як виявили дефект, зробіть копію екрану. Навіть якщо вам самому більше не вдасться відтворити цей дефект, можливо, по отриманій картинці колеги зрозуміють, в чому справа.

Важливість.

Показує ступінь шкоди, яка завдається проекту існуванням дефекту. Типові значення:

Критична (critical). Це найстрашніші дефекти, які призводять до краху застосунку або операційної системи, серйозних пошкодженнях бази даних, падіння веб-серверу або сервера застосунків.

Висока (major). Серйозні дефекти, такі як: втрата даних користувача, падіння значної частини функціональності програми, падіння браузера або іншого клієнта тощо.

Середня (medium). Дефекти, що зачіпають невеликий набір функцій програми. Як правило, такі дефекти можна «обійти», тобто виконати потрібні дії іншим способом, який не призводить до виникнення дефекту.

Низька (minor). Дефекти, які не заважають безпосередньо роботі з програмним забезпеченням. Як правило, сюди відносяться всілякі косметичні дефекти, помилки тощо.

Терміновість.

Показує, як швидко дефект повинен бути усунений. Типові значення:

Найвища (ASAP, as soon as possible). Присвоюється дефекту, наявність якого унеможливлює подальшу роботу над проектом або передачу замовнику поточної версії проекту.

Висока (high). Присвоюється дефекту, який потрібно виправити в самий найближчий час.

Звичайна (normal). Присвоюється дефекту, який слід виправляти в порядку загальної черги.

Низька (low). Присвоюється дефекту, яким слід займатися в останню чергу (коли і якщо на нього залишиться час).

Симптом.

Дозволяє класифікувати дефекти за їх типовою проявою. Типові значення симптомів:

косметичний дефект (cosmetic flaw);

пошкодження / втрата даних (data corruption / loss);

проблема в документації (documentation issue);

некоректна операція (incorrect operation);

проблема інсталяції (installation problem);

помилка локалізації (localization issue);

нереалізована функціональність (missing feature);

проблема масштабованості (scalability);

низька продуктивність (slow performance);

крах системи (system crash);

несподівана поведінка (unexpected behavior);

недружня поведінка (unfriendly behavior);

розбіжність з вимогами (variance from specs);

пропозиція щодо поліпшення (enhancement).

Можливість обійти.

Показує, чи існує альтернативна послідовність дій, виконання яких дозволило б користувачеві досягти поставленої мети (в обхід виникнення дефекту).

Коментар.

Може містити будь-які корисні для розуміння і виправлення дефекту дані.

Додатки.

Містить список прикріплених до звіту про дефект додатків (копій екрану, які призводять до збою, файлів тощо).

Підрозділ 4.5. Звіт про результати тестування

В цьому підрозділі треба узагальнити результати тестування у формі звіту, який містить:

1. Короткий опис. У гранично стислій формі відбиває основні досягнення, проблеми, висновки та рекомендації за результатами тестування. Наприклад, за звітний період успішно пройшло 100 % тест-

кейсів димового тестування и XX % тест-кейсів тестування критичного шляху, XX % тестів високої важливості реалізовано коректно тощо;

2. Опис процесу тестування. Опис завдань, які були виконані за звітний період. Тут описується середовище, в якому було проведено тестування (ОС, програмні середовища тощо), яким чином здійснювалися певні види тестування (вручну чи автоматизовано), чи здійснювалося повторне тестування і який його результат;

3. Статистика за дефектами. Статистика подається в формі таблиці, в якій представлені дані щодо виявлених за весь час проекту дефектів (з класифікацією за стадіями життєвого циклу і важливістю) (табл. 33). Цей розділ містить також графік, що відображає кількість знайдених дефектів (загальну і за важливістю) за періодами часу;

Таблиця 33

Статистика за дефектами

Статус	Загальна кількість	Кількість за важливістю			
		Низька	Середня	Висока	Критична
Знайдено					
Виправлено					
Перевірено					
Відкрито заново					
Відхилено					

4. Рекомендації. Подаються обґрунтовані висновки щодо отриманих результатів і за необхідності рекомендації щодо подальшої стратегії з поліпшення якості програмного забезпечення.

5. Порядок подання до захисту та захист дипломного проєкту

5.1. **Попередній захист дипломного проєкту**

Із метою виявлення готовності здобувача до захисту виконується попередній захист дипломного проєкту. Попередній захист складається з двох частин:

доповідь із презентацією за повністю виконаним проєктом;

демонстрація роботи програмного продукту.

Мета попереднього захисту – перевірка готовності здобувача до захисту відповідно до вимог випускової кафедри, оцінювання обсягу поданого проєкту і якості його виконання й оформлення. Незалежно від ступеня готовності проєкту, здобувач має з’явитися на попередній захист.

Для проведення попереднього захисту випускова кафедра визначає склад комісій та складає графік попереднього захисту.

На попередній захист подаються:

оформлене і своєчасно затверджене завдання;

повністю оформлена, але не переплетена пояснювальна записка з підписами здобувача, керівника і консультантів на завданні;

готовий програмний продукт і відеоролик його роботи;

план доповіді та презентація, погоджені з керівником.

На підставі доповіді здобувача, його відповідей на питання, результатів перевірки пояснювальної записки, презентації комісія визначає рекомендації здобувачу щодо:

доповіді;

відповідей на запитання;

змісту пояснювальної записки;

оформлення пояснювальної записки;

презентації;

демонстрації програмного продукту.

Допуск до захисту можливий у разі позитивного оцінювання за обома видами попереднього захисту (пояснювальна записка; програмна частина). Здобувачі, що не пройшли попередній захист, не допускаються до захисту.

5.2. **Подання дипломного проєкту до захисту**

Після попереднього захисту та усунення недоліків закінчений дипломний проєкт подається керівнику проєкту. Він остаточно перевіряє відповідність виконаної роботи завданню та відповідним вимогам, складає подання (письмовий відгук), в якому дає характеристику роботи здобувача.

Цілком оформлений дипломний проєкт, що підписаний його керівником, проходить нормоконтроль. Після нормоконтролю дипломний проєкт підписує завідувач кафедри, Дипломний проєкт направляється на рецензування.

5.3. **Захист дипломного проєкту**

Не пізніше ніж за добу до захисту здобувач подає дипломний проєкт секретарю державної екзаменаційної комісії (ДЕК). Обов'язковим є роздатковий матеріал щодо виконаної роботи для кожного члена ДЕК, який містить роздруківку слайдів презентації.

У ДЕК можуть бути подані інші матеріали, які характеризують наукову та практичну цінність виконаного дипломного проєкту, а саме:

друковані статті за темою роботи;

документи, які характеризують практичну цінність розробки здобувача;

документи, що вказують на практичне застосування роботи (підписані офіційними особами);

макети, зразки виробів тощо.

Захист дипломних проєктів проводиться на засіданні ДЕК.

Захист одного дипломного проєкту, переважно, не має перевищувати 30 хвилин. Для доповіді щодо проєкту здобувачу надається не більше 10 хвилин.

Захист комплексного дипломного проєкту, здебільшого, планується і проводиться на одному засіданні ДЕК, причому здобувачу, який захищається першим, доручається доповісти як про загальну частину роботи, так і про індивідуальну частину зі збільшенням (за необхідності) часу на доповідь. Усі здобувачі, які виконували комплексну роботу, мають бути повною мірою обізнані із загальною частиною роботи і готові до запитань членів комісії не тільки з індивідуальної, а й із загальної

частини роботи.

Доповідь здобувача має складатися з трьох основних частин, а саме: вступу, основної частини та висновків.

У вступі необхідно зазначити актуальність теми проєкту, дати загальний аналіз стану проблеми і сформулювати основні завдання, з вирішенням яких було пов'язано виконання проєкту.

В основній частині доповіді у стислій формі необхідно навести звіт про зміст виконаних розробок, показати ефективність ухвалених технічних рішень, навести короткий звіт з отриманих результатів.

У заключній частині доповіді необхідно зробити загальні висновки і дати рекомендації щодо можливої сфери застосування об'єкта проєктування, перелічити публікації за темою роботи, навести відомості про впровадження.

Доповідь має супроводжуватися посиланнями на електронну презентацію, яка демонструється здобувачем. Презентація має містити таки слайди:

титульний слайд із вихідними даними щодо дипломного проєкту;

зміст презентації (з посиланнями на відповідні слайди);

актуальність теми та мета дипломного проєкту;

модель організаційної структури підприємства, підрозділу підприємства;

модель управління бізнес-процесом;

UML-діаграма варіантів використання;

вайрфрейм або мокап проєкту інтерфейсу користувача;

математична (логічна) постановка;

заповнені форми вихідних та вхідних документів, діаграми, карти;

логічна та фізична моделі бази даних;

UML-діаграма класів (Class Diagram), що реалізують основну бізнес-логіку програмної системи, або UML-діаграма діяльності (Activity Diagram), яка відбиває основну бізнес-логіку програмної системи;

UML-діаграма станів (State Diagram), у яких можуть знаходитися елементи графічного інтерфейсу користувача;

результати тестування програмного забезпечення та тестування його безпеки;

використані інструментальні засоби та технології;

висновки за результатами дипломного проєкту;

апробація результатів дипломного проєкту;

заключний слайд.

Під час захисту може додатково використовуватися демонстраційний матеріал у вигляді відеоролика.

Після доповіді здобувач стисло відповідає на запитання членів ЕК. Далі зачитується рецензія. Здобувачу надається можливість відповісти на зауваження рецензента.

Після закінчення захисту всіх заявлених здобувачів комісія проводить закрите обговорення кожного захисту й оцінює його відповідно до критеріїв оцінювання. Водночас приймається до уваги рівень виконаної роботи та розробленого програмного продукту, якість оформлення пояснювальної записки, рівень наукової, практичної та теоретичної підготовки здобувача, ритмічність роботи над проєктом, наявність публікацій, виступів на конференціях тощо.

Результати захисту дипломного проєкту доводяться до відома здобувачів після завершення роботи ДЕК.

Рекомендована література

Основна

1. Бородкіна І.Л. Інженерія програмного забезпечення : навч. посіб. / І.Л. Бородкіна, Г.О. Бородкін. - Київ. : ЦУЛ, 2019. - 204 с.

2. Козак О.Л. Опорний конспект лекцій з курсу «Аналіз вимог до програмного забезпечення» для студентів напрямку підготовки «Програмна інженерія» / О.Л. Козак. – Тернопіль, 2021. – 56 с.

3. Кучеров Д.П. Інженерія програмного забезпечення : навчальний посібник / Д.П. Кучеров, Є.Б. Артамонов. - Київ. : НАУ, 2017. - 386 с.

4. Проектування інформаційних систем: Загальні питання теорії проектування ІС (конспект лекцій) [Електронний ресурс]: навч. посіб. для студ. спеціальності 122 «Комп'ютерні науки» / КПІ ім. Ігоря Сікорського; уклад.: О. С. Коваленко, Л. М. Добровська. – Електронні текстові дані (1 файл: 2,02 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2020. – 192 с.

5. Ушакова І. О. Лабораторний практикум з системного аналізу та проектування інформаційних систем [Електронний ресурс] : навчальний посібник / І. О. Ушакова, І. Б. Медведєва. – Харків : ХНЕУ ім. С. Кузнеця, 2022. – 251 с.

6. Бази даних : лабораторний практикум для студентів галузі знань 12 "Інформаційні технології" першого (бакалаврського) рівня [Електронний ресурс] / укл. В. В. Федько, В. П. Бурдаєв; Харківський національний економічний університет ім. С. Кузнеця. - Х. : ХНЕУ ім. С. Кузнеця, 2019. - 229 с.

7. Якість програмного забезпечення та тестування: базовий курс. Навчальний посібник / За ред. Крепич С.Я., Співак І.Я. / для бакалаврів галузі знань 12 «Інформаційні технології» спеціальності 121 «Інженерія програмного забезпечення». – Тернопіль: ФОП Паляниця В.А., 2020. – 478 с.

Додаткова

8. Дейт К. Дж. Введення в системи баз даних / К. Дж. Дейт. – 6-е вид. – Київ : Діалектика, 2020 - 1328 с.

9. ДСТУ 8302:2015 "Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання" - Київ : Держстандарт України, 2016. — 16 с.

10. ДСТУ ISO 9000:2015. Системи управління якістю. Основні положення та словник термінів: чинний з 01.07.2016 – Київ. : УкрНДНЦ, 2016. – 49 с.

11. ДСТУ ISO/IEC 25000:2016. Інженерія систем і програмних засобів. Вимоги до якості систем і програмних засобів та її оцінювання (SQuaRE). Настанова до SQuaRE: чинний з 01.01.2018. – Київ. : УкрНДНЦ.

12. ДСТУ ISO/IEC 25010:2016. Інженерія систем і програмних засобів. Вимоги до якості систем і програмних засобів та її оцінювання (SQuaRE). Моделі якості системи та програмних засобів: чинний з 01.01.2018. – Київ. : УкрНДНЦ.

13. ДСТУ ISO/IEC 25020:2016. Інженерія систем і програмних засобів. Вимоги до якості систем і програмних засобів та її оцінювання (SQuaRE). Рамкова модель і настанова щодо вимірювання: чинний з 01.01.2018. – Київ. : УкрНДНЦ.

14. ДСТУ ISO/IEC 25022:2019 (ISO/IEC 25022:2016, IDT) Інженерія систем і програмних засобів. Вимоги до якості систем програмних засобів та їхнього оцінювання (SQuaRE). Вимірювання якості під час застосування. : чинний з 01.01.2020. – Київ. : УкрНДНЦ.

15. ДСТУ ISO/IEC 25023:2019. Інженерія систем і програмних засобів. Вимоги до якості систем програмних засобів та їхнього оцінювання (SQuaRE). Вимірювання якості систем та програмних продуктів: чинний з 01.11.2019. – Київ. : УкрНДНЦ.

16. ДСТУ ISO/IEC 25040:2016 (ISO/IEC 25040:2011, IDT) Інженерія систем і програмних засобів. Вимоги до якості систем і програмних засобів та її оцінювання (SQuaRE). Процес оцінювання: чинний з 01.01.2018. – Київ. : УкрНДНЦ.

17. ДСТУ ISO/IEC/IEEE 15939:2018. Інженерія систем і програмних засобів. Процес вимірювання: чинний з 01.01.2019. – Київ. : УкрНДНЦ.

18. Життєвий цикл програмного забезпечення : навчальний посібник / Є.В. Левус, Т.А. Марусенкова, О.О. Нитребич. - Львів. : Видавництво Львівської політехніки, 2017. – 208 с.

19. Методичні рекомендації до оформлення звітів, курсових проєктів та дипломних робіт (проєктів) для студентів спеціальності 121 "Інженерія програмного забезпечення", 122 "Комп'ютерні науки", 126 "Інформаційні системи і технології": [Електронний ресурс] / уклад. І.О.Ушакова, Г.О. Плеханова, О.М. Беседовський. – Х. : ХНЕУ ім. С. Кузнеця, 2021. – 48 с.

20. Ушакова І. О. Проектування інформаційних систем : Практикум / І. О. Ушакова. – Харків : Вид. ХНЕУ ім. С. Кузнеця, 2015. - 250 с.

21. Developing Information Systems: Practical guidance for IT professional / P. Thompson, D. Paul, A. Paul et al. – London : BCS Learning & Development Limited, 2014. – 206 p.

22. ISO/IEC/IEEE: Systems and Software Engineering – Architecture Description. ISO/IEC/IEEE 42010:2022. - [Electronic resource]. – Access mode - http://www.iso-architecture.org/42010/index.html.

23. Lampathaki F. Business process modelling. Business process reengineering // F.Lampathaki, S. Koussouris, J. Psarras. – Zografou : NTUA, 2013. – 89 p.

24. Standard glossary of terms used in Software Testing. Version 3.1. All Terms [Electronic resource]. – ISTQB, 2024. – 82 p. – Access mode - https://astqb.org/assets/documents/Glossary-of-Software-Testing-Terms-v3.pdf

25. Wiegers K. Software Requirements Essentials: Core Practices for Successful Business Analysis; 1st Edition / K. Wiegers, C. Hokanson. – Addison-Wesley Professional, 2023.– 208 p.

Інформаційні ресурси

26. Інформаційний портал CRM [Електронний ресурс]. — Режим доступу : https://www.crm.com.ua

27. Державна служба статистики України [Електронний ресурс]. — Режим доступу : https://www.ukrstat.gov.ua/.

28. Спільноста розробників dou.ua [Електронний ресурс]. – Режим доступу : https://dou.ua/.

29. Gartner, Inc. [Electronic resource]. — Access mode : http://www.gartner.com/.

30. IBM [Електронний ресурс]. Режим доступу : — http://www.ibm.com/ua-en.

31. Microsoft [Електронний ресурс]. — Режим доступу : http: //www.microsoft.com/.

32. Agile alliance [Electronic resource]. – Access mode : http://www.agilealliance.org.

33. ITC Online [Electronic resource]. — Access mode : http://itc.ua/.

34. Object Management Group [Electronic resource]. – Access mode : http://www.omg.org.

35. SCRUM [Electronic resource]. – Access mode : http://www.scrum.org.

36. UML [Electronic resource]. – Access mode : http://www.uml.org/.

37. Try QA [Electronic resource]. – Access mode : http://tryqa.com/.

38. 10 usability heuristics for user interface design. - [Electronic resource]. – Access mode - https://www.nngroup.com/articles/ten-usability-heuristics/.

39. Про інформацію : Закон України від 02.10.1992 № 2657-XII: станом на 21 березня 2023 р. - — Режим доступу : https://zakon.rada.gov.ua/laws/show/2657-12#Text (дата звернення: 20.01.2024).

40. Про захист персональних даних: Закон України від 01.06.2010 р. № 2297-VI: станом на 27 жовт. 2022 р. — Режим доступу : https://zakon.rada.gov.ua/laws/show/2297-17#Text (дата звернення: 20.01.2024).

41. Про банки і банківську діяльність: Закон України від 07.12.2000 р. № 2121-III: станом на 1 січня 2024 р. — Режим доступу : https://zakon.rada.gov.ua/laws/show/2121-14#Text (дата звернення: 20.01.2024).

42. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). – Режим доступу: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679 (дата звернення: 12.02.2024).

Додатки

Додаток А

Зразок заяви студента на затвердження теми дипломного проєкту

Завідувачу кафедри
інформаційних систем
доценту Бондаренко Д.О.
студента 4 курсу <номер> групи
<П. І. Б студента>

ЗАЯВА

Прошу затвердити мені тему дипломного проєкту <Назва теми>.

<Дата> <Підпис студента> <П. І. Б студента>

Керівник
дипломного проєкту
<Посада
наук. ступінь, вчене звання> <Підпис керівника> <П. І. Б. керівника>

Додаток Б

Структура змістовної частини дипломного проєкту практичного характеру

№ з/п	Назва структурного елементу
1.	АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ <НАЗВА ПРЕДМЕТНОЇ ОБЛАСТІ>
1.1.	Визначення змісту предметної області
1.2.	Моделювання предметної області
1.3.	Огляд і аналіз літературних джерел щодо існуючих рішень та аналогів
1.4.	Позиціонування створюваного програмного забезпечення (застосунку, модулю, системи)
1.4.1.	Ділові переваги
1.4.2.	Визначення проблеми
1.4.3.	Визначення позиції
2.	СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
2.1.	Глосарій
2.2.	Розроблення варіантів використання
2.2.1.	Діаграма варіантів використання
2.2.2.	Специфікація варіантів використання
2.3.	Специфікація функціональних вимог
2.4.	Специфікація нефункціональних вимог
2.5.	Проєктування інтерфейсу користувача
3.	ПРОЄКТНІ ТА ТЕХНІЧНІ РІШЕННЯ
3.1.	Обґрунтування архітектури програмної системи
3.2.	Проєктування програмного забезпечення
3.2.1.	Вибір засобів реалізації проєкту
3.2.2.	UML-діаграма класів (UML-діаграма діяльності)
3.2.3.	Файлова структура проєкту програмного продукту (застосунку)
3.3.	Проєктування моделі даних
3.3.1.	Концептуальне інфологічне проєктування
3.3.2.	Проєктування логічної (фізичної) моделі даних
3.4.	Захист інформації
3.5.	Процес неперервної інтеграції CI та неперервної доставки програмного CD забезпечення або його компонентів
4.	ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
4.1.	Створення чек-листів
4.1.1.	Чек-лист для димового тестування.
4.1.2.	Чек-лист для тестування критичного шляху
4.2.	Створення тест-кейсів
4.3.	Створення наборів вхідних даних для тестування
4.4.	Звіт про дефекти
4.5.	Звіт про результати тестування

Додаток В

Зразок титульного аркуша дипломного проєкту

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ СЕМЕНА КУЗНЕЦЯ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ

Рівень вищої освіти	Перший (бакалаврський)
Спеціальність	Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення
Група	<Шифр групи>

ДИПЛОМНИЙ ПРОЄКТ

на тему: «Назва теми»

Виконав: студент(ка) <Ім’я ПРІЗВИЩЕ студента>

Керівник: <науковий ступінь>, <вчене звання> <Ім’я ПРІЗВИЩЕ>

Рецензент: <посада> <Ім’я ПРІЗВИЩЕ>

Харків – 202_ рік

Додаток Г

Приклади рефератів

РЕФЕРАТ

Пояснювальна записка до дипломного проєкту: 90 с., 30 рис., 20 табл., 12 додатків, 55 джерел.

Об'єктами проєктування є функціональні елементи, архітектура, інформаційне і програмне забезпечення модуля моніторингу складських поставок ТОВ "СІГМА".

Мета проєктування – створення модуля "Моніторинг складських поставок".

Метод проєктування – використання програмних систем ARIS Toolset, IBM Rational, Microsoft Visual Studio.

Створений модуль дозволяє підвищити достовірність обліку товарів на складі, обґрунтованість та оперативність прийняття рішень під час управління поставками товарів і їхніми запасами на складі.

Результати розробки можуть бути впроваджені на торгових підприємствах.

СИСТЕМА УПРАВЛІННЯ ЛАНЦЮГАМИ ПОСТАВОК, ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОЄКТУВАННЯ, CASE-ДІАГРАМИ, БАЗА ДАНИХ, МОНІТОРИНГ СКЛАДСЬКИХ ПОСТАВОК, WEB-СЛУЖБА

ABSTRACT

The bachelor’s thesis report: 90 pages, 30 figures, 20 tables, 12 appendices, 55 sources.

The objects of designing are functional elements, architecture and software of a module which provides monitoring of warehouse supplies.

The purpose of designing is to create "The monitoring of warehouse supplies" software module for Sigma Ltd.

The method of designing is using the ARIS Toolset, IBM Rational, Microsoft Visual Studio software systems.

The advantage of the developed module is the increasing of reliability of goods accounting, validity and timeliness of decision-making.

The obtained results can be applied at commercial enterprises.

SUPPLY CHAIN MANAGEMENT SYSTEM, OBJECT ORIENTED DESIGN, CASE DIAGRAMS, DATABASE, MONITORING OF WAREHOUSE SUPPLIES, WEB SERVICE

Додаток Д

Зразок оформлення змісту

ЗМІСТ

ЗМІСТ

НАВЧАЛЬНЕ ВИДАННЯ

Методичні рекомендації до виконання кваліфікаційних робіт

для здобувачів вищої освіти спеціальності 121 "Інженерія програмного забезпечення" освітньої програми "Інженерія програмного забезпечення" першого (бакалаврського) рівня

Самостійне електронне текстове мережеве видання

Укладачі:**Ушакова** Ірина Олексіївна
Фролов Олег Васильович
Парфьонов Юрій Едуардович
Поляков Андрій Олександрович

Відповідальний за видання Д. О. Бондаренко

Редактор В. Ю. Степаненко

Коректор

План 2024 р. Поз. № 82 ЕВ. Обсяг 80 с.

Видавець і виготовлювач — ХНЕУ ім. С. Кузнеця, 61166, м. Харків, просп. Науки, 9-А

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру **Дк № 4853 від 20.02.2015 р.**