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

Харків
ХНЕУ ім. С. Кузнеця
2024

УДК 004.415(07.034)
М54

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

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

Самостійне електронне текстове мережеве видання

Методичні рекомендації до написання дипломного проєкту
М54 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. – 77 с.

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

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

УДК 004.415(07.034)

© Харківський національний економічний
університет імені Семена Кузнеця, 2024

Вступ

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

Головне завдання дипломного проєктування – підготовка здобувача вищої освіти до виконання завдань і обов'язків, передбачених для первинних посад у певному виді економічної діяльності.

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

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

Зміст, обсяг і структура дипломного проєкту, які наведені у методичних рекомендаціях, є типовими. В окремих випадках за письмовим дозволом випускової кафедри їх може бути змінено.

Методичні рекомендації призначено для здобувачів вищої освіти, які навчаються за спеціальністю 121 "Інженерія програмного забезпечення" освітньої програми "Інженерія програмного забезпечення" першого (бакалаврського) рівня.

Освітній компонент "Дипломний проєкт" є завершальним етапом підготовки бакалаврів за освітньою програмою "Інженерія програмного забезпечення". Відповідно до освітньої програми підготовки бакалаврів з інженерії програмного забезпечення цей освітній компонент формує результати навчання та компетентності, які визначені в табл. 1.

Таблиця 1

**Результати навчання та компетентності,
які формує навчальна дисципліна**

Результати навчання	Компетентності, якими має оволодіти здобувач вищої освіти
1	2
РН09	ЗК02
	ЗК03

Закінчення табл. 1

1	2
РН09	СК01
	СК04
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

Примітка.

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 проєктуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування.

СК04. Здатність формулювати та забезпечувати вимоги щодо якості програмного забезпечення відповідно до вимог замовника, технічних завдань та стандартів.

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 проєктування, керуючись формальними методами опису вимог та моделювання.

РН12. Застосовувати на практиці ефективні підходи щодо проєктування програмного забезпечення.

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

РН15. Мотивовано вибирати 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, призначена здійснювати об'єктивний контроль за ступенем сформованості умінь у випускників

виконувати 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 вищої освіти, набуті 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 вищої освіти набувають навичок з аналізу науково-технічної, нормативної та довідкової літератури, використання державних стандартів, 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 і навичок у процесі виконання конкретних завдань дипломного проєкту;

розвиток і закріплення навичок самостійної роботи;

удосконалення вміння користуватися сучасними системами програмування, виконувати інженерні завдання з проєктування інформаційних систем та їхніх елементів, застосовуючи сучасні методології, інформаційні технології, проводити комп'ютерне моделювання, а також уміння 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 здатності до самостійної роботи в умовах ринкової економіки, сучасного виробництва, прогресу науки і техніки.

Виконуючи дипломний проєкт, здобувач вищої освіти має повною мірою використовувати знання з інформаційних технологій та комп'ютерної техніки, інтелектуальних систем і баз знань, наявні пакети, методи та засоби математичного оброблення інформації; поєднувати теоретичні знання з виробничим досвідом, набутим під час проходження практики; використовувати досягнення вітчизняної та світової науки і техніки; ураховувати техніко-економічні показники функціонування створюваних програмно-інформаційних систем і комплексів; на високому теоретичному і професійному рівні здійснювати проєктування вибраних технічних рішень; грамотно, повно і водночас лаконічно викладати свої рішення.

Під час захисту дипломного проєкту здобувач вищої освіти 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 вищої освіти. За всі розроблені в ньому проєктні рішення, а також правильність, обґрунтованість розрахунків і належне оформлення матеріалів відповідає автор.

До дипломного проєктування допускають здобувача вищої освіти, 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 вищої освіти може бути призначено тему дипломного проєкту з переліку рекомендованих тем. Також йому надають право самостійного вибору теми з урахуванням його схильностей і можливостей найбільш повно застосувати здобуті знання. Якщо тему пропонує здобувач вищої освіти, то її слід обговорити й погодити з керівником дипломного проєкту.

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

Безпосереднє керівництво дипломним проєктом здійснюють викладачі кафедри, яких призначає завідувач кафедри інформаційних систем.

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

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

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

На основі вивчення літературних джерел, які охоплюють як монографії, підручники та навчальні посібники, статті в періодичних виданнях, так і патентні матеріали, науково-технічні звіти, реферативні видання, здобувач вищої освіти 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 оформлюють аналітичний огляд (порівняльний аналіз), із якого логічно випливають вибрані методики досліджень.

Після вивчення літературних джерел здобувач вищої освіти складає попередній план виконання дипломного проєкту й обговорює його з керівником. У процесі обговорення уточнюють вихідні дані для проєктування та строки, що регламентують роботу здобувача вищої освіти. Після цього здобувач вищої освіти складає уточнений план роботи над проєктом, погоджує його з керівником і розпочинає проєктування. Під час виконання дипломного проєкту потрібно регулярно відвідувати консультації керівника, подавати йому на перевірку робочі матеріали відповідно до плану-графіка виконання етапів проєкту.

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

На засіданнях кафедри інформаційних систем регулярно заслуховують повідомлення керівників дипломних проєктів про хід виконання календарних планів. Здобувачі вищої освіти, які не дотримуються графіка виконання проєкту або значно відстають у його виконанні, запрошують для звіту на засідання кафедри.

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

Дипломний проєкт виконують із додаванням обов'язкових матеріалів (схеми, діаграми, графіки залежностей, таблиці, рисунки, лістинги програм тощо), що розробляють відповідно до даних методичних рекомендацій.

Обсяг дипломного проєкту становить 50 – 75 друкованих сторінок формату А4 (без додатків).

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

Таблиця 2

Структура дипломного проєкту

Структурні елементи дипломного проєкту	Кількість сторінок
Титульний аркуш	1
Реферат	1 – 2
Зміст	2
Перелік умовних скорочень	1
Вступ	1 – 2
Розділ 1	9 – 14
Розділ 2	14 – 19
Розділ 3	14 – 22
Розділ 4	10 – 12
Висновки	1 – 2
Список використаних джерел	2 – 3
Додатки	

Дипломний проєкт друкують на аркушах паперу формату А4. Текст має бути набрано з використанням гарнітури шрифту Times New Roman (кегль 14), із міжрядковим інтервалом 1,2.

Поля сторінок залишають такі: 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 по всій роботі та дорівнювати 1,27 см.

Вирівнювання основного тексту проводять "за шириною".

Докладні вимоги до оформлення пояснювальної записки наведено у відповідних методичних рекомендаціях [12].

Дипломний проєкт спрямовано на аналіз, моделювання, прогнозування інформаційних процесів або керування такими процесами в економіці (як приклад, можна розглядати окреме підприємство, організацію,

галузь чи географічний регіон), що зумовлює 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 і в організаціях.

Змістова (основна) частина дипломного проєкту має містити:

1) аналіз предметної області;

2) специфікацію вимог до програмного забезпечення;

3) проєктні та технічні рішення;

4) тестування програмного забезпечення.

Структуру змістової частини дипломних проєктів, якої потрібно дотримуватися, наведено в додатку Б.

4. Методичні рекомендації до розроблення структурних елементів дипломного проєкту

4.1. Загальні рекомендації щодо розроблення пояснювальної записки дипломного проєкту

Загальними вимогами до тексту дипломного проєкту 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 роботи потрібно дотримуватися єдиної термінології. Вона не має бути перевантаженою малоінформативним матеріалом, описом загальновідомих даних, виведенням формул тощо. Слід посилатися на джерела інформації. У тексті має бути наведено використаний математичний апарат і результати розрахунків, виконаних за допомогою ПК.

Текст не слід викладати від першої особи, краще використовувати безособову форму (наприклад, "обчислено", "знайдено", "доведено", "визначено").

Під час викладення матеріалу не використовують:

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 установлених правилами орфографії та нормативними документами.

Першою сторінкою дипломного проєкту є **титульний аркуш**. Він містить такі дані:

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

повну назву роботи;

підписи керівника та рецензента;

назву міста і рік складання.

Зразок титульного аркуша дипломного проєкту наведено в додатку В.

Реферат – це короткий виклад змісту дипломного проєкту, що містить основні фактичні відомості та висновки, необхідні для початкового ознайомлення з ним. Реферат виконують українською та англійською мовами.

Реферат має бути стислим, інформативним і містити відомості, які дозволяють ухвалити рішення про доцільність читання роботи.

Реферат містить:

відомості про обсяг дипломного проєкту, 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 й 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. Бажано розмістити його на одній сторінці формату А4.

Ключові слова призначено для розкриття сутності проєкту і для поширення 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 включає: вступ; назви всіх розділів, підрозділів та пунктів основної частини дипломного проєкту; висновки; перелік посилань; назви додатків і номери сторінок, які містять початок матеріалу.

Приклад змісту дипломного проєкту наведено в додатку Д.

Якщо в роботі використовують маловідомі скорочення, нові символи, позначення тощо, то в ній має бути **перелік умовних скорочень,** який подають у вигляді окремого списку, розміщеного перед вступом. Незважаючи на це, за першої появи цих елементів у тексті документа подають їх розшифрування.

Якщо в роботі спеціальні терміни, скорочення, символи, позначення повторюють менше ніж три рази, то перелік не складають, а їх розшифрування наводять у тексті під час першого згадування.

Приклад переліку умовних скорочень наведено в [12].

Вступ – віддзеркалення роботи, тому слід ретельно його опрацювати. Краще формувати вступ після виконання основного тексту.

У вступі до дипломного проєкту потрібно ідентифікувати та сформулювати проблему бізнесу, яка виникла на підприємстві: обґрунтувати актуальність теми проєкту для розв'язання цієї проблеми на основі розроблюваного модуля або системи; коротко охарактеризувати функціональність модуля або системи. Слід охарактеризувати технічну та програмну платформи розроблення програмного продукту, сформулювати мету й завдання проєкту, визначити об'єкт і предметну область проєктування. Також потрібно навести інформацію щодо засобів проєктування, які використовували в дипломному проєкті, та можливих галузей застосування отриманих результатів.

Висновки до роботи – це резюме за результатами всієї роботи. Ця частина є особливо важливою, оскільки тут наводять підсумкові результати роботи.

У роботі мають міститися висновки за кожним етапом виконаного проєкту та за роботою загалом, які спід співвіднести з метою і завданням на дипломне проєктування.

Потрібно зазначити практичну цінність результатів роботи, дати рекомендації для подальшого вдосконалення об'єкта проєктування. Зазначаючи практичну цінність отриманих результатів, слід окреслити ступінь їхньої готовності до використання, масштабів використання, а також подати стислі відомості щодо впровадження результатів дослідження з указанням назв організацій, у яких здійснено реалізацію, форм реалізації, реквізитів документів тощо. Якщо дипломний проєкт упроваджено на підприємстві, то до дипломного проєкту додають довідку або акт про впровадження.

У висновках наводять відомості про те, де й коли було апробовано отримані результати (на яких конференціях, семінарах зроблено доповіді), перераховують публікації здобувача вищої освіти за матеріалами дипломного проєкту.

Список використаних джерел містить 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, прочитані, переглянуті, проаналізовані в процесі роботи над дипломним проєктом, які 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 оформлюють згідно з ДСТУ 8302:2015 "Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання" [9].

Список використаних джерел має містити не менше ніж 30 джерел. Його подають мовою оригіналу, розміщують в алфавітному порядку прізвищ перших авторів або назв і нумерують в порядку їхнього зростання. Нумерація безперервна.

Роботи одного автора розташовують за алфавітом назв або в хронології їх написання. Алфавітний список подають за алфавітом у такій послідовності:

література українською мовою та мовами з кириличною графікою (болгарською та ін.);

література мовами з латинською графікою;

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

Список використаних джерел обов'язково має містити прізвище та ініціали автора, повну назву джерела, назву міста видавництва, назву видавництва та рік видання, кількість сторінок чи посилання на сторінки тощо. Загальний обсяг книги у сторінках указують, якщо посилання на неї дають повністю; сторінки (від ... до) відмічають, якщо посилання належать до окремої частини літературного джерела.

У **додатках** уміщують матеріал, який є необхідним для повноти дипломного проєкту, але не може бути послідовно розміщеним у її основній частині через великий обсяг або з інших причин.

Ілюстрації (діаграми бізнес-процесів, сценарії діалогів тощо), таблиці, проміжні математичні докази, формули та розрахунки, текст допоміжного характеру теж можна оформити у вигляді додатків.

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

У цьому підрозділі методичних рекомендацій подано зміст усіх розділів основної частини дипломного проєкту згідно з його структурою (див. додаток Б).

1. Аналіз предметної області <назва>

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

1.1. Визначення змісту предметної області

Визначення змісту предметної області забезпечує контекст і розуміння задачі або проблеми, яку потрібно розв'язати. Якщо тема диплома пов'язана з бізнесом, то в цьому підрозділі наводять коротку характеристику напрямів діяльності об'єкта управління (підприємства, організації); визначаються проблему бізнесу, яку слід розв'язати. В іншому разі предметну область визначають безвідносно до підприємства.

Тут треба визначити:

цілі та завдання розроблення;

контекст і сферу застосування. Контекст – це межі, у яких виконуються процеси і відбуваються пов'язані з ними події, та ресурси, необхідні для їх відповідної інтерпретації. Сферу застосування, зазвичай, визначають у 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 понять і терміни, характерні для цієї предметної області. Вона може включати безліч взаємопов'язаних понять, процесів, суб'єктів і об'єктів;

зміни й еволюцію предметної області. Предметна область може бути схильною до змін і розвитку з часом. Нові технології, дослідження та вимоги можуть впливати на предметну область, приводячи до її зміни та розвитку.

1.2. Моделювання предметної області

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

У підрозділі 1.2 потрібно:

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

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

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

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

Для моделювання предметної області використовують CASE-інструменти.

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

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 указати структурний підрозділ, посаду, його роль у бізнес-процесі

** навести перелік документів

*** процес, що використовує інформацію процесу

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

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

1.3. Огляд і аналіз літературних джерел щодо наявних рішень і аналогів

У підрозділі 1.3 треба провести огляд літературних джерел щодо наявних підходів до розроблення програмного забезпечення відповідно до визначеної проблеми, мети і завдань, які стосуються бізнес-вимог та принципів, певних технологій, платформ, бібліотек тощо. Під час огляду слід 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 (підбиває підсумок, стверджує), зазначає (аналізує, характеризує, розкриває) недоліки (суперечності, сутність), описує, називає, формулює, висуває (гіпотезу, питання), висловлює припущення, зупиняється, підкреслює, стверджує, доводить тощо). За результатами

огляду роблять висновок щодо використання кращих підходів до розроблення.

Слід вибрати декілька найбільш популярних програмних продуктів-аналогів, призначених для реалізації функціональності предметної області та визначити їхні 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 потрібно подати 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 слід навести та коротко описати екранні форми, що характеризують основну функціональність продукту.

У кінці підрозділу зробити висновок щодо можливості використання певних підходів до розроблення програмного забезпечення і досвіду провідних фірм-розробників програмних продуктів під час розроблення проєктних рішень.

1.4. Позиціонування програмного забезпечення (застосунку, модуля, системи)

Позиціонування – це визначення того, як ваше програмне забезпечення (застосунок, модуль, система) "вписується" в IT-ринок, і як ви подаєте його унікальну цінність цільовій аудиторії. Позиціонування визначає, чим ваш програмний продукт відрізняється від конкурентів і чому він потрібен вашим клієнтам. Тому це один із найважливіших кроків для досягнення успіху.

Позиціонування містить короткий опис таких характеристик:

ділові переваги;

визначення проблеми;

визначення позиції створюваного продукту.

1.4.1. Ділові переваги

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

актуальність, тобто важливість, значимість, затребуваність на сьогодні створюваного продукту;

призначення, тобто яким чином розроблюваний продукт буде задовольняти потреби основних користувачів та інших зацікавлених сторін і яким чином це буде реалізовано.

1.4.2. Визначення проблеми

Під час визначення проблеми дають підсумок проблеми, яку розв'язує проєкт. Для визначення проблеми використовують такий формат (табл. 5).

Таблиця 5

Визначення проблеми

Проблема	<Опис проблеми>
Стосується	<Зацікавлені сторони, яких стосується проблема>
Її наслідком є	<Який є вплив проблеми>
Успішне розв'язання	<Список деяких ключових переваг від успішного розв'язання>

1.4.3. Визначення позиції

Під час визначення позиції створюваного продукту описують на найвищому (стратегічному) рівні унікальну позицію на ринку, яку має намір заповнити програмний продукт. Визначення позиції описують у такому форматі (табл. 6).

Таблиця 6

Визначення позиції програмного продукту

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

2. Специфікація вимог до програмного забезпечення

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

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 розгорнених моделей предметної області, які на стадії реалізації інформаційної системи стають основою об'єктної моделі (для об'єктно орієнтованих застосувань) і моделі даних (для генерації схеми бази даних).

Глосарій потрібно подати у вигляді табл. 7.

Таблиця 7

Глосарій

Термін	Опис терміна
1. Основні поняття і категорії предметної області та проєкту	
2. Користувачі системи	
3. Вхідні та вихідні документи	

2.2. Розроблення варіантів використання

Розроблення варіантів використання містить:

- діаграму варіантів використання;
- специфікацію варіантів використання.

2.2.1. Діаграма варіантів використання

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

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

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

Таблиця 8

Список акторів

Актор	Короткий опис

Таблиця 9

Список варіантів використання

Варіант використання	Короткий опис варіанта використання	Актор	Основний / допоміжний

Актор – дійова особа, зовнішня щодо проєктованої системи сутність, яка взаємодіє із системою і використовує її функціональні можливості для досягнення певної мети або виконання особистих завдань. Зазвичай актором буде користувач системи. Окрім користувача, як актора можна розглядати іншу програмну систему, апаратний пристрій, час. Пошук акторів системи зазвичай зводиться до аналізу ролей різних користувачів. Вибір акторів залежить від їхніх 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.

Наступним кроком є декомпозиція основних варіантів використання для більш докладного їх опису. Розглядають додаткові варіанти використання, що деталізують основну мету (наприклад, для основного варіанта використання "Формування замовлення" виділяють додаткові варіанти "Створити замовлення", "Розрахувати знижку" тощо). На діаграмі варіантів використання відносини між основними варіантами використання та додатковими визначають 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 створюють діаграму варіантів використання у середовищі певного CASE-засобу.

2.2.2. Специфікація варіантів використання

Кожен варіант використання повинен мати опис. У дипломному проєкті 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 варіанта використання>
Тригер	<Опис події, яка запускає виконання варіанта використання>
Пост-умови	<Набір станів, які гарантує система незалежно від успіху виконання варіанта 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 формі дієслова в інфінітиві завершеного виду або віддієслівного іменника, яка відображає мету.

Контекст – стисло одним абзацом описуються, що має робити варіант і який кінцевий результат від нього очікують.

Дійові особи – це:

основні (первинні) актори;

допоміжні (вторинні) актори.

Передумова – описує стан, у якому система має перебувати до початку виконання варіанта. Тут формулюють умови, за яких цей варіант використання може бути ініційований. Це може бути інший варіант використання, який обов'язково слід виконати, щоб можна було виконати цей варіант. Також в передумові необхідно вказувати, в якій частині системи знаходиться користувач, які дії вже виконав.

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

Пост-умова – це стан, у якому система має перебувати після закінчення виконання варіанта; це умови, що гарантується акторам, не залежно від успіху виконання даного варіанта. Наприклад, у разі невдалої транзакції всі дані, що були в системі до її початку, зберігаються незмінними.

Події, описувані передумовами або пост-умовами, мають бути станами, які користувач може спостерігати.

Основний потік – перераховують 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 позначається номером кроку основного потоку з додавання букви латинського алфавіту (a, b, c ...). Далі йде послідовність дій системи та користувача.

2.3. Специфікація функціональних вимог

Для специфікації функціональних вимог, визначених у попередньому пункті, треба визначити їхні атрибути:

пріоритет – заповнюється аналітиком, показує пріоритет реалізації вимоги для клієнта. Використовують під час управління проєктом, визначає пріоритет розроблення вимоги. Можливі значення: обов'язкове, рекомендоване, опційне (за вибором);

складність – заповнюється менеджером проєкту, показує рівень трудовитрат, пов'язаних із реалізацією вимоги. Використовують під час управління проєктом, вливає на черговість розроблення. Складність виконання вимоги може виражатися у вигляді трудомісткості та вказувати кількість людино-днів, потрібних для його реалізації, або у вигляді значень шкали: висока, середня, низька. Висока складність – це ймовірність того, що вимога є дуже дорогою щодо ресурсів або грошей. Її слід виконати 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). Використовують, щоб упорядкувати виконання проєкту за пріоритетами варіантів використання. Треба заздалегідь установити, які варіанти

використання зачіпають основну частину архітектурних рішень. Ці варіанти використання реалізують першими;

контакт – заповнюється аналітиком, ідентифікує зацікавлену особу, яка може надати необхідну інформацію про вимогу (ім'я контакту). Використовують для гарантії того, що розробники можуть отримати інформацію, необхідну їм для реалізації вимоги. Замість атрибуту "Контакту" часто використовують атрибут "Виконавець".

Якщо в проєкті контактом виступає одна людина, то цю колонку можна опустити, але треба ідентифікувати контакт один раз в тексті цього пункту. Специфікацію функціональних вимог наводять в таблиці (табл. 12).

Таблиця 12

Специфікація функціональних вимог

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

2.4. Специфікація нефункціональних вимог

Нефункціональні вимоги можна поділити на такі групи:

1. Застосовність:

час, необхідний для навчання звичайних і досвідчених користувачів;

вимірний час відгуку для типових завдань;

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

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

2. Надійність:

доступність – визначає % доступного часу, час використання, час, що витрачається на обслуговування, порушення режиму роботи і т. ін.;

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

середнє напрацювання до ремонту – як довго системі дозволяють працювати до того, коли потрібно буде провести 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, слід 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.

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

5. Проєктні обмеження.

Ці вимоги мають містити усі проєктні обмеження до створюваної системи. Проєктні обмеження – це рішення, сформульовані як обов'язкові і яких потрібно суворо дотримуватися. Прикладами можуть бути мови програмування, вимоги до технології програмування, обов'язкове використання інструментальних засобів розроблення, архітектурні та конструктивні обмеження купованих компонентів, бібліотек класів і т. ін.

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 тощо.

7. Куповані компоненти.

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

8. Інтерфейси.

Визначають інтерфейси, які підтримує застосунок. Він має містити адекватну специфіку, протоколи, порти і логічні адреси тощо, щоб програмне забезпечення можна було розробити і перевірити на відповідність вимогам інтерфейсів.

8.1. Інтерфейси користувача.

Описують інтерфейси користувача, які повинні бути реалізовані програмним забезпеченням.

8.2. Апаратні інтерфейси.

Визначають усі апаратні інтерфейси, які підтримує програмне забезпечення, зокрема логічну структуру, фізичні адреси, очікувану поведінку тощо.

8.3. Програмні інтерфейси.

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

8.4. Комунікаційні інтерфейси.

Описуються усі комунікаційні інтерфейси до інших систем або пристроїв типу локальних мереж, віддалених послідовних пристроїв тощо.

9. Вимоги до ліцензування.

Визначають усі вимоги обов'язкового ліцензування або інші вимоги обмеження використання, які має виконувати програмне забезпечення.

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

Описують усі необхідні юридичні застереження, гарантії, оголошення про авторське право, право спадкоємства, торгівельні марки або емблеми для програмного забезпечення.

11. Уживані стандарти.

Указують посилання на всі вживані стандарти і на конкретні розділи таких стандартів, які належать до описуваної системи. Наприклад, це можуть бути правові та регулюючі стандарти, стандарти якості, промислові стандарти щодо застосовності, здатності до взаємодії, інтернаціоналізації, відповідності операційній системі тощо.

У специфікації нефункціональних вимог указують лише ті вимоги, які є суттєвими для проєкту.

Специфікацію нефункціональних вимог з їхніми атрибутами слід навести в таблиці (табл. 13).

Таблиця 13

Специфікація нефункціональних вимог

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

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 виконувати користувачі за допомогою цієї програми і які вимоги до інтерфейсу можуть виникнути в користувачів.

Загальні принципи проєктування інтерфейсу користувача:

1. Програма має допомагати виконувати завдання.

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

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

Для реалізації цього принципу потрібно:

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

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

надати досвідченим користувачам можливість вимкнути виведення інформаційних повідомлень;

добре продумувати зміст повідомлень, що виводяться користувачеві.

Широковідомими є евристичні правила авторитетного американського фахівця в галузі проєктування інтерфейсів Якоба Нільсена [32]:

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. Упізнання, а не згадування.
7. Гнучкість і ефективність використання.
8. Естетичний і мінімалістський дизайн.
9. Допомога користувачам у розпізнаванні й виправленні помилок.
10. Довідка й документація.

Проєктування форм

Форми – це "будівельні блоки" інтерфейсу користувача.

Щоб створити добре спроєктовану форму, слід 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 уведення даних, доцільно:

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

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

1. Дотримуйтеся стандартних угод щодо розташування пунктів меню.
2. Групуйте пункти меню в логічному порядку й за змістом.
3. Для групування 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. По можливості використовуйте клавіатурні еквіваленти команд і "гарячі" клавіші.

Далі наведено деякі рекомендації з проєктування вебінтерфейсу користувача:

1. Мінімізуйте зусилля, які потрібно зробити користувачеві для ухвалення рішень щодо навігації.

2. По 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 поєднуйте їх у розділи. Спростіть 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 візуального погляду.

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

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

Інтерфейс користувача (UI – від англ. user interface) забезпечує передавання 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 використовують Figma, PhotoShop, Marvel, Pencil Project тощо.

Вайрфрейм (від 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-нахрест для зображень. Тому вайрфрейми зазвичай називають даними низької точності (lo-fi). Схему доповнюють текстовим документом із описом логіки роботи продукту і взаємодії з ним користувача.

Мокап (від англ. mockup – макет) – це середньо- або високодеталізоване статичне подання 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 змогу оцінити візуальну сторону проєкту.

3. Проєктні та технічні рішення

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 [15] визначає архітектуру як "фундаментальні поняття або властивості системи в її середовищі, утілені в її елементах, зв'язках і в принципах її дизайну та еволюції".

Головна ідея архітектури полягає в тому, щоб знизити складність сприйняття системи внаслідок розмежування повноважень і 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 ухвалені на ранніх етапах проєктування рішення про високорівневий дизайн системи, а також використовувати компоненти цього дизайну і шаблони проєктування повторно в інших проєктах.

Архітектурою є набір елементів, які мають певну форму (властивості й обмеження, що накладаються на елементи), і їх обґрунтування. Обґрунтування фіксує мотиви вибору певного архітектурного стилю, елементів і обмежень. Обґрунтування є необхідним на етапі створення архітектури і корисним надалі, тому що архітектура має набір властивостей, що дозволяють їй задовольняти вимоги, і незнання цих вимог може призвести до змін, які порушують архітектуру. Але до архітектури входять властивості, а не вимоги. Архітектуру будують, щоб найкраще відповідати вимогам до системи, що створюють, згідно з принципом "форма відповідає функції". Отже, проєктування архітектури ПЗ – це процес виконуваний після етапу аналізу і формулювання вимог.

Завдання етапу проєктування архітектури ПЗ – перетворення вимог до системи у вимоги до ПЗ і побудова на їхній основі архітектури системи. Побудову архітектури системи здійснюють шляхом визначення цілей системи, її вхідних і вихідних даних, декомпозиції системи на підсистеми, компоненти або модулі та розроблення її загальної структури. Проєктування архітектури системи можна виконувати різними методами (стандартизованим, об'єктно-орієнтованим, компонентним тощо), кожний із 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 відображено в термінах опису архітектури на вибраній мові опису архітектури (Architecture Description Language (ADL)). До таких мов належать [15]: ArchiMate, Architecture Analysis & Design Language, C4 model (software), Darwin (ADL), EAST-ADL, Wright (ADL). Ці мови переважно 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 розподілено на три етапи:

1) визначення важливих для архітектури вимог;

2) архітектурне проєктування (Architectural Design);

3) архітектурна документація.

У цьому підрозділі дипломного проєкту здобувач вищої освіти має виділити прийняті архітектурні рішення, які потрібно подати у вигляді таблиці (табл. 14).

Таблиця 14

Прийняті архітектурні рішення

Архітектурне рішення	Обґрунтування рішення	Посилання на вимогу

На основі аналізу архітектурних вимог слід визначити основні принципи та рішення, які закладають в основу майбутньої програмної системи. Це, насамперед, еталонна архітектура, патерни розгортання, архітектурні патерни (стилі), архітектурні тактики, внутрішні та зовнішні компоненти. Кожне рішення має бути обґрунтовано з указанням вимог, що задовольняються, у вигляді ідентифікаторів вимог, зазначених у розділі 2 (див. табл. 12 і 13).

У посиланнях на вимоги додають ті з функціональних вимог, що визначили 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 наявні атрибути якості, тобто ті нефункціональні вимоги, які можуть вплинути на архітектуру: розширюваність системи, її доступність, здатність модифікуватись, швидкодія, захищеність і т. ін. Також у посиланнях потрібно відобразити архітектурні обмеження з групи проєктних обмежень, сформованих у підрозділі 2.4.

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

3.2. Проєктування програмного забезпечення

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 тощо.

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

вибір основної мови програмування, або (за потреби) декількох мов програмування;

вибір основного фреймворку та допоміжних бібліотек;

вибір системи управління базами даних;

вибір програмного оточення та інших засобів, що використовуються в проєкті.

Для кожного технічного рішення потрібно зазначити його призначення й альтернативні варіанти, що розглядали. Рекомендовано подати результати вибору у вигляді таблиці (табл. 15).

Таблиця 15

Вибір засобів реалізації проєкту

Назва засобу	Призначення	Альтернативні варіанти, що розглядали	Обґрунтування вибору

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

Програмна система означає програмне забезпечення, що розробляють. 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 потрібно 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 подають в одному з додатків до дипломного проєкту.

Далі наводимо методичні рекомендації щодо проєктування програмної системи.

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

Для подання статичної структури моделі програмної системи призначено UML-діаграму 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 позначає безліч об'єктів, які мають однакову структуру, поводження і відносини з об'єктами інших класів. Клас може не мати екземплярів або об'єктів. У цьому разі його називають абстрактним класом. Графічно клас зображують у вигляді прямокутника, що додатково може бути розділеним горизонтальними лініями на секції. У цих секціях можна вказувати ім'я класу, атрибути й операції.

Обов'язковим елементом позначення класу є його ім'я. Ім'я класу має бути унікальним у межах пакету, який описують деякою сукупністю діаграм класів. Ім'я вказують в першій верхній секції прямокутника. Рекомендовано як імена класів використовувати іменники, записані без пробілів. Важливо 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", а також ті, що мають безпосередній стосунок до предметної області й функціонального призначення проєктованої системи.

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

Квантор видимості може приймати одне з трьох можливих значень:

1. Загальнодоступний (public). Атрибут із цією зоною видимості доступний із будь-якого іншого класу пакету, у якому визначено діаграма.

2. Захищений (protected). Атрибут із цією зоною видимості недоступний для всіх класів, за винятком підкласів цього класу.

3. Закритий (private). Атрибут із цією зоною видимості недоступний для всіх інших класів без винятку.

Ім'я атрибута становить рядок тексту, що використовують як ідентифікатор відповідного атрибута, тому воно 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 вислів, семантика якого визначається мовою специфікації відповідної моделі. У нотації UML тип атрибута іноді визначають залежно від мови програмування, яку передбачають використовувати для реалізації цієї моделі. У найпростішому випадку тип атрибута вказують рядком тексту, що має осмислене значення в межах пакету або моделі, до яких належить розглянутий клас.

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

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

Крім внутрішнього устрою або структури класів, на відповідній діаграмі вказують різні відношення між класами. Водночас сукупність типів таких відношень зафіксовано в мові UML і визначено 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 використовують, щоб показати потік управління в системі та кроки в процесі виконання варіанта використання. Застосовуючи діаграми діяльності, можна моделювати послідовні та паралельні дії. Діаграма діяльності фокусується на умовах і послідовності виконання деякого потоку.

Діаграма діяльності зображує потік управління від стартової точки до кінцевої точки, показує різні шляхи ухвалення рішень під час виконання діяльності. ЇЇ використовують для моделювання робочого процесу із урахуванням умов, обмежень, послідовних та одночасних дій.

Щоб розробити діаграму діяльності, потрібно визначити:

1. Початковий стан і кінцевий стан.

2. Проміжні дії, необхідні для досягнення кінцевого стану з початкового стану.

3. Умови або обмеження, які спонукають систему змінити потік управління.

3.2.3. Файлова структура проєкту програмної системи (застосунку)

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

Структуру наводять 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 наводять структуру окремо для клієнтської та серверної його частин. Починати рекомендовано зі структури папок проєкту (частини проєкту), яку можливо зобразити у вигляді "дерева", далі, викладати вміст папок у вигляді переліків.

Приклад опису вмісту папки:

Папка config містить такі файли:

.editorconfig – файл налаштування робочої ide, на якій виконували проєкт;

.eslintignore – файл конфігурації винятків для плагіна es-lint;

.eslintrc – файл конфігурації правил плагіна es-lint;

.gitignore – файл конфігурації винятків для системи контролю версій;

.prettierrc – файл конфігурації правил плагіна prettier;

package-lock.json – технічна інформація про встановлені модулі проєкту;

package.json – перелік залежностей проєкту та допоміжних скриптів.

3.3. Проєктування моделі даних

Початковими даними для проєктування моделі даних є:

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

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

опис технології оброблення інформації в межах цієї предметної області конкретного підрозділу з зазначенням вхідної та вихідної інформації, виконуваних завдань, функцій і регламенту роботи виконавців оброблення інформації, періодичності виконання функцій із оброблення інформації;

аналіз наявних засобів автоматизації оброблення інформації в межах цієї предметної області (апаратне та програмне забезпечення, система управління базами даних – СУБД);

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

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

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.

База даних (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.

Аналітичну частину БД використовують у процесі виконання оперативного аналізу інформації та розроблення моделей для систем підтримки ухвалення 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 та редагування даних і накопичення їх, переважно, за хронологією.

У процесі розроблення транзакційної частини БД можна використати структурне чи об'єктно орієнтоване моделювання, застосовуючи

відповідні CASE-інструменти: ErWin Data Modeler, IBM Rational Software тощо. Під час побудови сховища даних слід обґрунтувати вибір моделі зберігання (кубічна модель MOLAP або ROLAP: "зірка" чи "сніжинка").

Якщо тема дипломного проєкту пов'язана зі створенням складних транзакційних баз даних для інформаційної системи, рекомендовано у цей підрозділ додати проєктування концептуальної, логічної та фізичної моделей даних. В інших випадках можна обмежитися проєктуванням концептуальної та логічної моделей даних.

3.3.1. Концептуальне інфологічне проєктування

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

Словник даних. На основі аналізу вхідних та вихідних документів будують модель відображення множини реквізитів вихідних і вхідних документів на множину елементів даних, що підлягають збереженню в базі даних; потім зібрану інформації оформлюють у вигляді, зручному для проєктування. Для цього складають словник даних (табл. 16).

Таблиця 16

Словник даних

Назва елемента	Ідентифікатор	Тип і довжина	Призначення елемента

Елементи даних словника наводять в алфавітному порядку за графою "Назва елемента" з метою подальшого вилучення омонімів та дублювальних елементів. Якщо словник уміщує багато елементів, його виносять у додаток. У полі "Призначення елемента" слід указати: елемент збереження є фактичним чи обчислюваним.

Під час проєктування глобальної інфологічної моделі даних потрібно здійснити виявлення еквівалентних сутностей та їхнє злиття, виявлення категорій і синтез узагальнювальних сутностей, виявлення й усунення дублювання атрибутів і зв'язків. Будують 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

Ім'я атрибута або агрегату	Межі / допустимі значення	Структура (формат)	Умова	Значення за замовчуванням

3.3.2. Проєктування логічної (фізичної) моделі даних

Проєктування логічної моделі даних містить: графічне подання логічної моделі у вигляді ERD (у 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); погодженість; актуальність; безпека; відновлюваність; логічна та фізична незалежність; ефективність.

У разі використання реляційних СУБД наводять логічну модель у вигляді ER-діаграми, що включають сутності (таблиці), атрибути (колонки / поля) та відношення (ключі). Якщо застосовують ORM-технологію, то ER-діаграму можна замінити на діаграму класів, на якій представити тільки ті класи, що пов'язані з моделлю даних. У разі використання NoSQL СУБД (наприклад, MongoDB) наводять схему даних для колекцій і зв'язки (relation) між колекціями, якщо такі є.

3.4. Захист інформації

Підрозділ захисту інформації містить розроблення елементів політики безпеки інформаційної (програмної) системи. Політика безпеки інформаційної системи має включати такі пункти:

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

Закону України "Про інформацію", у якому визначено поняття конфіденційної інформації [30];

Закону України "Про захист персональних даних", у який визначає, що таке персональні дані і регламентує правила їх обробки [29];

Закону України "Про банки і банківську діяльність", у якому регламентовано правила використання банківської інформації [28].

Перелік конфіденційної інформації, яка циркулює в ІС, рекомендовано оформити у вигляді таблиць (табл. 18 і 19). Щодо кожного виду інформації слід указати бізнес-процеси, у яких її використовують, і в колонці "Обґрунтування" зазначити, чому її варто відносити до конфіденційної інформації.

Таблиця 18

Перелік конфіденційної інформації, яка циркулює в системі

Назва інформації	Назва бізнес-процесу, у якому її використовують	Обґрунтування

Таблиця 19

Перелік критично важливих ресурсів інформаційної системи

Назва ресурсу	Назва бізнес-процесу, у якому його використовують	Обґрунтування

2. Перелік осіб, які мають доступ до конфіденційної інформації, яка циркулює в інформаційній системі. У цьому пункті слід визначити права

доступу до ресурсів бази даних щодо кожного користувача, способи і види доступу до ресурсів мережі інформаційної системи, а також указати доступність ресурсу з інтернету. Як доступні ресурси в локальній мережі та в мережі інтернет потрібно вказати необхідні порти і протоколи, що вимагають для правильного функціонування ресурсу.

Перелік осіб, які обробляють конфіденційну інформацію, слід подати у вигляді таблиці (табл. 20). Для зменшення кількості рядків можна групувати інформацію про тих, хто має однакові права доступу.

Таблиця 20

Перелік користувачів, які мають доступ до конфіденційної інформації та критично важливих ресурсів

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

3. Потрібно здійснити аналіз інформаційної системи з погляду криптоаналітика, виявити слабкі компоненти системи і сформувати перелік можливих загроз безпеці. Слід класифікувати загрози безпеці щодо основних послуг безпеки, таких як конфіденційність, цілісність, доступність, причетність, спостережуваність, автентичність та ін. Перелік загроз подати у вигляді таблиці (табл. 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

4. У таблиці 21 щодо кожної загрози слід указати мету загрози і можливе місце виникнення / проникнення загрози. Метою загрози можуть бути певні ресурси системи, роботу яких може бути порушено у разі її настання. Це виконують для можливості оцінювання завданого збитку, а також для правильного вибору механізмів захисту. Під місцем виникнення / проникнення розуміють ресурс системи або бізнес-процес, у 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.

5. Щодо кожного механізму безпеки слід визначити, якими засобами його можна реалізувати і на перекриття яких загроз спрямувати. Під засобами захисту розуміються конкретні програмні продукти або спеціалізовані сервісні функції ОС, Web-серверів, баз даних. Перелік механізмів безпеки і засобів захисту подають у вигляді таблиці (табл. 22).

Таблиця 22

Перелік механізмів безпеки, що перекривають загрози безпеці ІС

Механізм безпеки	Засіб захисту	Загроза безпеці

6. Потрібно дати детальний опис засобів захисту використовуваних у програмному продукті, що розробляють. Тут слід зазначити використовувані криптографічні методи захисту, методи автентичності користувачів, спосіб зберігання паролів у системі, засоби автентичності під час підключення до бази даних, використання захищених протоколів SSL і т. ін.

7. Для розрахованих на багато користувачів систем, слід, розглянути можливість реалізації послуги безпеки спостережуваності у формі логування основних дій користувача під час роботи з базою даних і основних подій системи, таких як підключення до бази даних, вхід та вихід користувача тощо.

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

Цей підрозділ присвячено процесам CI/CD, які стосуються програмного забезпечення, що було спроєктовано та розробляється. Зміст

і структуру цього підрозділу може бути скореговано залежно від конкретного проєкта за погодженням із керівником дипломного проєкта.

Метод неперервної інтеграції (continues 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. Мета використання неперервної інтеграції – забезпечити погоджений і автоматизований спосіб створення і тестування додатка.

Неперервна доставка (Continue Delivery, CD) автоматизує процес упровадження програмного продукту і внесення змін у код, у підготовлену серверну інфраструктуру. Розробники створюють програмне забезпечення з використанням середовищ, відповідних для певних етапів роботи. Ключова особливість CD – 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. ін.).

Інструменти CI/CD. Ідея неперервної інтеграції базується на використанні інструментів, які підтримують цей процес. Одним із них є система керування вихідним кодом (SCM). ЇЇ застосовують для відстеження змін у вихідному коді, що допомагає командам розробників об'єднувати зміни, створені різними розробниками (системи Bitbucket, Github, GitLab).

Інші інструменти – це системи, що підтримують створення, тестування й упровадження в режимі неперервної інтеграції (системи Gitlab CI, Jenkins, CircleCI, Travis та інші).

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

GitHub Flow;

GitFlow;

Forking Workflow;

GitLab Flow;

Trunk Based Development.

У межах цих стратегій слід прийняти, які гілки будуть використовувати, для яких цілей, як часто будуть проводити злиття коду, коли будуть формувати реліз, які додаткові концепції можуть ще використовувати. Для цього складають перелік типів гілок (табл. 23).

Таблиця 23

Словник даних

№ з/п	Найменування / шаблон гілки	Призначення	Частота інтеграції

Інфраструктура.

У роботі потрібно зробити обґрунтування вибору типу інфраструктури – локальної (On-Premise) або хмарної (Cloud) – за основними факторами, які максимально впливають на вибір рішення (табл. 24).

Таблиця 24

Основні фактори локальної та 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 залежно від того, скільки спожито 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 конфіденційності компанії/організації із суворо 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. Якщо станеться непередбачуване і виникне час простою, є ймовірність вчасно не отримати доступ до цих даних	Виявити компоненти інфраструктури, і канали передачі, сервіси обробленння та зберігання даних; навести відповідні вимоги щодо захисту даних у цих компонентах (конфіденційність, цілісність, доступність, спостережуваність); навести механізми безпеки, що забезпечують ці вимоги

Закінчення табл. 24

1	2	3	4
Реалізація	Компанія несе відповідальність за підтримку рішення та всіх пов'язаних із ним процесів	Ресурси розміщуються на території постачальника послуг – приватної, публічної або гібридної хмари	Описати, які ресурси розміщуються (віртуальні машини з різними операційними системами, їхні характеристики), які сервери або кластери використовують (Docker Swarm, Kubernetes)
Нормативні вимоги	Для урядових компаній, а також для підприємств, галузеві та нормативні документи яких суворо регламентуються, дуже важливо, щоб зберігання даних чітко відповідало вимогам закону	Слід виявити належну обачність і переконатися, що сторонній постачальник цих послуг 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 і клієнтом – єдиного стандарту обслуговування та послуг, у тому числі вимог доступності	Проаналізувати нормативні вимоги до вашого типу компанії/організації щодо 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 ін.

У дипломному проєкті слід розглянути, як будують інфраструктуру для кожного середовища (environment). Показати, яка частина інфраструктури буде створюватися та підтримуватися командою DevOps, тобто формуватиме платформу, яка частина буде налаштовуватися відповідно до розгортання сервісів програмного забезпечення, на яку частину має вплив команда розробників і тестувальників. Платформа може надаватись або Cloud-провайдером, або розгортатися на локальній інфраструктурі. Слід обґрунтувати одну або декілька інфраструктур, що будуть використовувати 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 обладнанням займається провайдер. Він установлює на нього програмне забезпечення 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; 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 розроблення та тестування, систем управління базами даних. Провайдер контролює не лише сервери, системи зберігання даних і 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. Для локальної інфраструктури розгортання різних кластерів як платформ – 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). Програми та сервіси розробляє й обслуговує провайдер, розміщує їх у хмарі та пропонує кінцевому користувачеві через браузер або додаток 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), офісний пакет документів для роботи (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 (табл. 25).

Таблиця 25

Характеристика інфраструктур

Властивість інфраструктури	Призначення	Тип інфраструктури	Середовище, у якому буде застосовано

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

Локальне середовище розробника – це робоча станція розробника на якій безпосередньо працює розробник і на якій установлено всі необхідні компоненти для підтримки процесу розроблення. Це можуть бути утілити, IDE, плагіни для IDE, сервери, набір бібліотек, текстові редактори та інші інструментальні засоби розроблення.

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

Під виробничим середовищем розуміють 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 слід описати всі сервіси, що будуть розгортатися на базі вибраної інфраструктури, та їх характеристики (налаштування). Опис середовищ наведено в табл. 26.

Таблиця 26

Характеристика середовищ

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 проєкт має свій набір технологій, який може вплинути на процес. Конвеєр можна подати у вигляді діаграми (рис. 1).

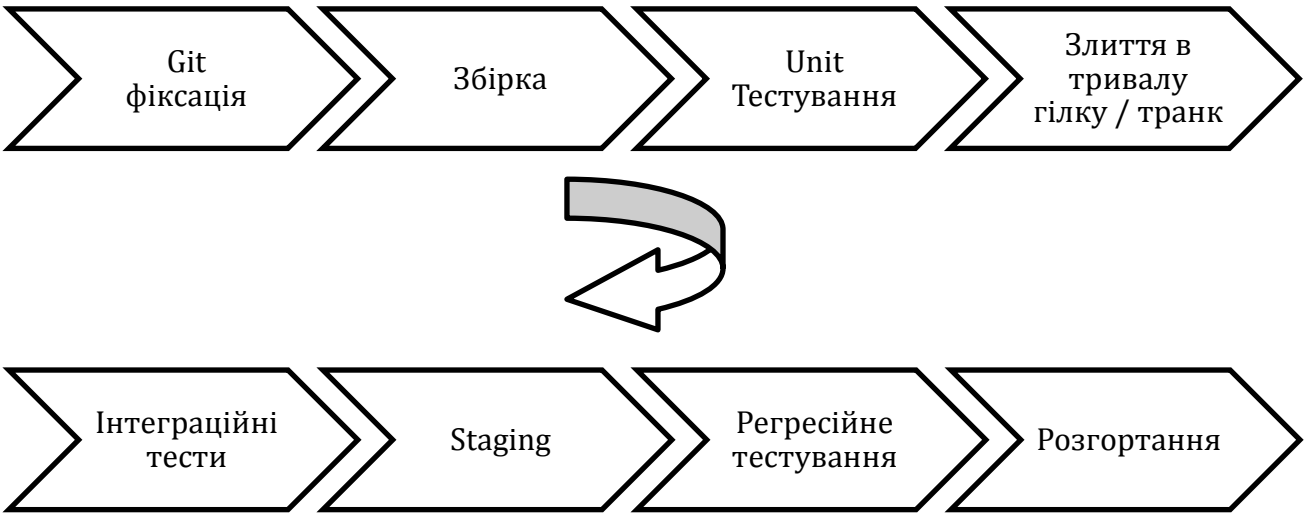

Рис. 1. **Приклад етапів конвеєра DevOps**

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

У дипломному проєкті треба зробити опис етапів конвеєра, які ви вважаєте необхідними для розгортання вашого додатку. Слід описати скрипти, налаштування відповідних інструментів, надати інструкцію із запуску конвеєра DevOps. Усі ці налаштування також можна зберігати у відповідному Git-репозиторій. Для побудови конвеєра DevOps можна використовувати такі системи: Jenkins, GitLab CI, GitHub Actions. Відповідні скрипти потрібно подати у додатках до дипломного проєкту.

4. Тестування програмного забезпечення

Метою розділу 4 є створення документації, необхідної для тестування програмного забезпечення й аналізу результатів тестування, що включає чеклисти, тест-кейси, тестові набори даних, а також звіт про дефекти і звіт про результати тестування. З урахуванням теми дипломного проєкту і за погодженням із керівником у цьому розділі можна подати результати інших видів тестування.

4.1. Створення чеклистів

У цьому підрозділі треба скласти чеклисти для димового тестування та тестування критичного шляху. Чеклист – це список, який містить необхідні перевірки (тест-кейси) під час тестування програмного забезпечення. Призначення чеклиста:

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

4.1.1. Чеклист для димового тестування

Димове тестування (Smoke testing) спрямовано на перевірку найголовнішої, найважливішої, 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 чеклиста для димового тестування треба визначити:

для чого призначене ваше програмне забезпечення;

які найбільш очевидні прості кроки потрібно виконати, щоб у нього потрапити;

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

Чеклист для димового тестування створюють в таблиці (табл. 27).

Таблиця 27

Чеклист для димового тестування

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

Список тестів має бути мінімальним – до 10 – 15, лінійним, без ієрархічного впорядкування.

Оточення перевірки. Зазвичай у кожен чеклист додають кілька стовпців, призначених для тестування на окремій платформі. У назві цих стовпців указують назву пристрою, браузера і його версії тощо.

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

Таблиця 28

Статуси тесту

Статус	Коментар
Passed / Пройдений	Перевірку пройдено успішно, багів не знайдено
Failed / Невдалий	Знайдено один або більше багів
Blocked / Заблокований	Неможливо перевірити, тому що один із багів блокує поточну перевірку
Not run / Не виконаний	Ще не перевірено
Skipped / Пропущений	Не буде перевірки з будь-якої причини. Наприклад, поточний функціонал ще не реалізовано.
In Progress / Виконується	Поточний пункт, над яким працює тестувальник

На основі створеного чеклиста треба виконати перевірку створеного програмного забезпечення. У процесі проходження тестів змінюйте статус тесту відповідно до табл. 12. У разі виявлення дефектів робіть примітку до відповідного пункту з посиланням на сторінку з багом. Усі помилки, виявлені під час цього етапу тестування треба виправити!

4.1.2. Чеклист для тестування критичного шляху

Тестування критичного шляху направлено на дослідження функціональності, використовуваної типовими користувачами в типовій повсякденній діяльності. Більшість користувачів найчастіше використовують якусь підмножину функцій програми. Саме ці функції й 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 у димовому тестуванні, але однаково досить високе (зазвичай, приблизно 70 – 90 %, залежно від сутності проєкту).

Щоб створити чеклист 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 слід виконати на кожному кроці.

Чеклист для тестування критичного шляху створюють у таблиці (табл. 29).

Таблиця 29

Чеклист для тестування критичного шляху

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

4.2. Створення тест-кейсів

Для тестування критичного шляху треба розробити тест-кейси відповідно до перевірок, визначених у чеклисті. Опис тест-кейсів для тестування критичного шляху подають у таблиці за шаблоном (табл. 30).

Таблиця 30

Шаблон для створення тест-кейсів

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

Вимоги до змісту тест-кейсу.

Ідентифікатор тест-кейсу – унікальний, осмислений (якщо дозволяє ПЗ).

Пов'язана з тест-кейсом вимога – вказує на основну вимогу для перевірки якої створено тест-кейс. Наявність цього поля поліпшує таку властивість тест-кейсу, як відстеження.

Назва (сутність) тест-кейсу покликана спростити швидке розуміння основної ідеї тест-кейсу без звернення до його інших атрибутів. Це поле є найбільш інформативним під час перегляді списку тест-кейсів. Назва завжди має бути в кожного тест-кейсу! Ні за яких умов категорично не допускають наявність тест-кейсів без назв!

Пріоритет показує важливість тест-кейсу. Може бути виражений буквами (A, B, C, D, E), цифрами (1, 2, 3, 4, 5), словами ("украй високий", "високий", "середній", "низький", "украй низький") або іншим зручним способом. Пріоритет може корелювати з:

важливістю вимоги;

потенційною важливістю дефекту, на пошук якого спрямовано тест-кейс.

Опис тест-кейсу містить вхідні дані, необхідні для виконання тест-кейсу. Дозволяють описати все те, що потрібно підготувати до початку виконання тест-кейсу, наприклад, стан бази даних, стан файлової системи та її об'єктів тощо. Все, що описують в цьому полі, готують без використання тестової програми. Якщо тут виникають проблеми, не можна писати звіт про дефект у програмному забезпеченні.

Кроки тест-кейсу описують послідовність дій, які слід реалізувати в процесі виконання тест-кейсу. Їх нумерують.

Очікувані результати за кожним кроком тест-кейсу описують реакцію програми на дії, описані в кроках тест-кейсу. Номер кроку відповідає номеру результату.

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 та граничних умов.

Клас еквівалентності – це набір даних, що обробляється однаковим способом й приводить до однакового результату.

Граничні умови – це місця (значення), у яких один клас еквівалентності переходить в інший.

Ознаки еквівалентності тест-кейсів:

тест-кейси спрямовано на пошук однієї й тієї самої помилки;

якщо один із тест-кейсів виявляє помилку, інші її теж, імовірно, виявлять;

якщо один із тест-кейсів не виявляє помилку, інші її теж, імовірно, не виявлять;

тест-кейси використовують схожі набори вхідних даних;

для виконання тест-кейсів здійснюють одні й ті самі операції;

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

усі тест-кейси приводять до спрацьовування одного й того самого блоку оброблення помилок (error handling block);

жоден із тест-кейсів не призводить до спрацьовування блоку оброблення помилок (error handling block).

Під час створення тестових наборів даних треба навести класи еквівалентності:

за довжиною поля: для неприпустимої та припустимої довжини;

за символами: для неприпустимих і припустимих символів.

На основі виділених класів еквівалентності визначають набори даних для позитивного і негативного тестування. Результати подають у 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 із неприпустимою довжиною / недопустимими символами (для надійності).

4.4. Звіт про дефекти

За результатами тестування критичного шляху програмного забезпечення на основі чеклиста, тест-кейсів і наборів даних створюють звіт про дефекти. Звіт про дефекти – це документ, який містить інформацію

про очікуваний і фактичний результат тестування, описує і визначає пріоритет виявленого дефекту, а також сприяє його усуненню. Коли виявляють дефект, обов'язково потрібно скопіювати скріншот із дефектом і вставити його або посилання на нього в додатки звіту (табл. 32).

Таблиця 32

Звіт про дефекти

Ідентифікатор	Короткий опис	Докладний опис	Кроки відтворення	Відтворюваність	Важливість	Термінновість	Симптом	Можливість обійти	Коментар	Додатки

Дефект – це розбіжність очікуваного і фактичного результату.

Очікуваний результат – це поведінка системи, описана у вимогах.

Фактичний результат – це поведінка системи, яку спостерігають у процесі тестування.

Вимоги до змісту звіту про дефект.

Ідентифікатор дефекту: унікальний, осмислений (якщо дозволяє ПЗ).

Короткий опис у лаконічній формі дає вичерпну відповідь на питання "Що сталося?" "Де це сталося"? "За яких умов це сталося?". Наприклад, "немає логотипу 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 слід виконати для відтворення дефекту. Інформація в звіті про дефект є вкрай важливою. Саме вона дозволяє розробнику швидко відтворити й 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 два-три рази перш ніж писати, що дефект відтворюється завжди. Довести, що дефект існує – завдання тестувальника. Тому відразу ж, як виявили дефект, зробіть копію

екрана. Навіть якщо вам самому більше не вдасться відтворити цей дефект, можливо за отриманою картинкою колеги зрозуміють, у чому справа.

Важливість показує ступінь шкоди, яку завдає проєкту існування дефекту. Типові значення важливості:

критична (critical). Це найстрашніші дефекти, які призводять до краху застосунку або операційної системи, серйозних пошкоджень бази даних, падіння вебсерверу або серверу застосунків;

висока (major). Серйозні дефекти: утрата даних користувача, падіння значної частини функціональності програми, падіння браузера або іншого клієнта тощо;

середня (medium). Дефекти, що зачіпають невеликий набір функцій програми. Зазвичай, 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. Це різні "косметичні" дефекти, помилки тощо.

Терміновість показує, як швидко дефект слід усунути. Типові значення:

найвища (ASAP, as soon as possible). Присвоюють дефекту, наявність якого унеможливлює подальшу роботу над проєктом або передачу замовнику поточної версії проєкту;

висока (high). Присвоюється дефекту, який потрібно виправити найближчим часом;

звичайна (normal). Присвоюють дефекту, який слід виправляти в порядку загальної черги;

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

Симптом дозволяє класифікувати дефекти за їхнім типовим проявом. Типові значення симптомів:

косметичний дефект (cosmetic flaw);

пошкодження / утрата 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 із 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 екрана, які призводять до збою, файлів тощо).

4.5. Звіт про результати тестування

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

2. Опис процесу тестування. Опис завдань, виконаних за звітний період. Тут описують середовище, у якому було проведено тестування (ОС, програмні середовища тощо), указують, яким чином здійснювались певні види тестування (вручну чи автоматизовано), зазначають, чи проводили повторне тестування і який його результат;

3. Статистику за дефектами. Статистику подають у формі таблиці, у якій подано дані щодо виявлених за весь час проєкту дефектів (із 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
1	2	3	4	5	6
Знайдено					
Виправлено					

Закінчення табл. 33

1	2	3	4	5	6
Перевірено					
Відкрито заново					
Відхилено					

4. Рекомендації. Подають обґрунтовані висновки щодо отриманих результатів і, за потреби, рекомендації щодо подальшої стратегії з поліпшення якості програмного забезпечення.

5. Порядок подання до захисту та захист дипломного проєкту

5.1. Подання дипломного проєкту до захисту

Завершений та оформлений в повному обсязі дипломний проєкт направляють на попередній захист. Процедура попереднього захисту наведена у підрозділі 5.2.

Після попереднього захисту та усунення недоліків у дипломному проєкті керівник проєкту здійснює нормоконтроль. Нормоконтроль – це процедура перевірки відповідності вимогам щодо оформлення дипломних робіт. У разі непогодження з оформленням нормоконтролер має право повернути роботу здобувачу вищої освіти на подальше доопрацювання та перенести термін його захисту.

На наступному етапі дипломний проєкт в електронному варіанті подають на кафедру інформаційних систем для перевірки на унікальність в термін не пізніше, ніж за два тижні до призначеної дати захисту. Проєкт подають єдиним файлом. Назва файлу має бути: **"Плагіат_ПІБ_спеціальність"**.

Порядок перевірки дипломних проєктів на унікальність та її відсоток визначають протоколом засідання кафедри. Результати перевірки є задовільними, якщо значення оцінок подібності нижче критичних значень метрик, отриманих під час перевірки.

У разі, якщо результати перевірки не є задовільними, дипломний проєкт потребує доопрацювання з повторним проходженням нормоконтролю та перевірки на унікальність.

У разі успішної перевірки на унікальність керівник повідомляє про готовність здобувача вищої освіти до захисту на засіданні кафедри інформаційних систем, де затверджуються списки допущених до захисту проєкту перед ЕК. До захисту не допускають здобувачів вищої освіти, які не виконали навчальний план освітньої програми і на момент подання дипломного проєкту до захисту мають академічну заборгованість, або не пройшли процедуру попереднього захисту.

Якщо здобувач вищої освіти є допущеним до захисту, то далі його дипломний проєкт направляють на рецензування. Рецензентами можуть бути професіонали-практики, експерти галузі, науково-педагогічні працівники з відповідної галузі, які не працюють на кафедрі інформаційних систем. У рецензії зазначають позитивні сторони роботи та її недоліки і роблять висновок щодо присвоєння здобувачу вищої освіти кваліфікації "бакалавр з інженерії програмного забезпечення" за спеціальністю 121 "Інженерія програмного забезпечення" із зазначенням рекомендованої оцінки (з вказанням балів за 100-бальною шкалою).

Після отримання рецензії керівник проєкта остаточно перевіряє відповідність виконаної дипломної роботи вимогам, складає подання (письмовий відгук), в якому дає характеристику роботи здобувача вищої освіти, і підписує проєкт. Після цього завідувач кафедри інформаційних систем засвідчує своїм підписом рішення кафедри щодо дипломного проєкту до ЕК для публічного захисту.

Здобувач вищої освіти подає до захисту такі документи:

1) надрукований або електронний варіант дипломного проєкту з обов'язковими додатками (назва файлу має бути: **"Диплом_ПІБ_спеціальність"**);

2) звіт з перевірки проєкту на унікальність (видає секретар ЕК);

3) ілюстративний матеріал (презентація) (назва файлу має бути: **"Презентація_ПІБ_ спеціальність"**);

4) заяву про дотримання професійної етики (назва файлу має бути: **"Заява_ПІБ_ спеціальність"**);

5) файл з інформацією, яка потрібна для розміщення дипломного проєкту у репозитарії ХНЕУ ім. С. Кузнеця (назва файлу має бути: **"Репозитарій_ПІБ_ спеціальність"**);

6) електронні версії своїх публікацій, грамот і сертифікатів;

7) рецензію на дипломний проєкт.

Перед захистом здобувач вищої освіти зобов'язаний ознайомитися з відгуком і рецензією, проаналізувати їх та підготувати відповіді на зауваження.

Здобувачі вищої освіти, які не захистили дипломний проєкт, не мають права на присвоєння кваліфікації "бакалавр з інженерії програмного забезпечення" за спеціальністю 121 "Інженерія програмного забезпечення".

У разі невиконання роботи вчасно за поважної причини (хвороба, виробнича необхідність тощо) дозволяють захист роботи в інші строки, але в межах строку дії ЕК.

5.2. Попередній захист дипломного проєкту

Із метою перевірки готовності здобувача вищої освіти до захисту проводять попередній захист дипломного проєкту, який складається з двох частин:

доповіді з презентацією за повністю виконаним проєктом;

демонстрації роботи програмного продукту.

Мета попереднього захисту – перевірка готовності здобувача вищої освіти до захисту відповідно до вимог випускової кафедри, оцінювання обсягу поданого проєкту, відповідності його структури та змісту даним методичним рекомендаціям, а також якості його виконання й оформлення. Незалежно від ступеня готовності проєкту, здобувач вищої освіти має з'явитися на попередній захист.

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

На попередній захист подають:

повністю оформлений дипломний проєкт;

готовий програмний продукт і відеоролик його роботи;

звіт перевірки на унікальність;

план доповіді та презентацію, погоджені з керівником.

На підставі доповіді здобувача вищої освіти, його відповідей на питання, результатів перевірки дипломного проєкту і презентації комісія визначає рекомендації щодо:

доповіді;

відповідей на запитання;

структури та змісту дипломного проєкту;

оформлення дипломного проєкту;

презентації;

демонстрації програмного продукту.

Допуск до захисту можливий у разі позитивного оцінювання за обома видами попереднього захисту (дипломний проєкт і програмна частина). Здобувачі вищої освіти, які не пройшли попередній захист, не допускають до захисту.

5.3. Захист дипломного проєкту

Не пізніше ніж за три робочі дні до захисту здобувач вищої освіти подає дипломний проєкт та презентацію секретареві екзаменаційної комісії (ЕК).

До ЕК можна подати 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 вищої освіти;

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

макети, зразки виробів тощо.

Захист дипломних проєктів проводять на засіданні ЕК.

Захист одного дипломного проєкту, зазвичай, не має перевищувати 15 хв. Для доповіді щодо проєкту здобувачу вищої освіти надають до 7 хв.

Захист комплексного дипломного проєкту планують і проводять на одному засіданні ЕК, причому здобувачу вищої освіти, який доповідає першим, доручають охарактеризувати як загальну, так й індивідуальну частини роботи, зі збільшенням (за потреби) часу на доповідь. Усі здобувачі вищої освіти, які виконували комплексну роботу, мають бути повною мірою обізнані з її загальною частиною і готові до запитань членів комісії не тільки за індивідуальною, а й за загальною частинами роботи.

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

У вступі потрібно 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 в стислій формі слід навести звіт про зміст виконаних розробок, обґрунтувати ухвалені технічні рішення, подати короткий звіт з отриманих результатів.

У завершальній частині доповіді потрібно зробити загальні висновки і дати рекомендації щодо можливої сфери застосування об'єкта проєктування, зазначити публікації за темою роботи, навести відомості про впровадження (якщо такі є).

Доповідь супроводжують презентацією, яку здобувач вищої освіти демонструє. Презентація має містити такі 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 (у випадку, якщо програмний продукт пов'язаний з конкретним підприємством (організацією));

модель бізнес-процесу;

UML-діаграма варіантів використання;

вайрфрейм або мокап проєкту інтерфейсу користувача;

заповнені форми вихідних і вхідних документів, діаграми, карти;

архітектура програмного продукту;

використані інструментальні засоби та технології;

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 відображає основну бізнес-логіку програмної системи;

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

результати тестування програмного забезпечення та тестування його безпеки;

висновки за результатами дипломного проєкту;

апробація результатів дипломного проєкту (за наявності);

завершальний слайд.

Під час захисту також додатково використовують матеріал у вигляді відеоролика з демонстрацією роботи розробленого програмного забезпечення.

Після доповіді здобувач вищої освіти стисло відповідає на запитання членів ЕК. Далі зачитують рецензію на дипломний проєкт і здобувачу вищої освіти надають можливість відповісти на зауваження рецензента.

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

Результати захисту дипломного проєкту повідомляють здобувачам вищої освіти після закінчення роботи ЕК.

5.4. Критерії оцінювання дипломного проєкту

Відповідно до "Порядку оцінювання результатів навчання здобувачів вищої освіти за накопичувальною бально-рейтинговою системою в ХНЕУ ім. Семена Кузнеця" оцінка результатів захисту дипломних робіт (проєктів) здійснюється за 100-бальною системою оцінювання результатів навчання, прийнятою в університеті, й відображається у протоколах роботи ЕК.

Дипломний проєкт бакалавра є підсумковою індивідуальною письмовою роботою здобувача вищої освіти, яка дає змогу отримати комплексне уявлення про рівень засвоєння теоретичних знань та практичної підготовки, здатність до самостійної роботи за обраною спеціальністю.

У дипломному проєкті здобувач вищої освіти повинен продемонструвати рівень сформованості компетентностей з певної спеціальності, володіння навичками наукового дослідження, здатність мислити, аналізувати, узагальнювати й робити висновки.

Кількість отриманих балів за дипломний проєкт формується ЕК на основі оцінки рецензента, відгука наукового керівника та захисту дипломної роботи.

Система оцінювання дипломного проєкту базується на таких складових, що підлягають оцінюванню: зміст та оформлення проєкту; розроблене здобувачем вищої освіти програмне забезпечення; захист дипломного проєкту перед ЕК.

Критерії оцінювання змісту проєкту враховують такі складові:

актуальність тематики і практичне значення проєкту;

відповідність змісту дипломного проєкту його темі, чіткість і повнота постановки завдань проєктування;

об'єктивність та актуальність висвітлення сучасного стану предметної області з творчим використанням сучасних джерел інформації та повнота її дослідження;

обґрунтованість вибору методів та засобів вирішення поставленого завдання;

рівень обґрунтування прийнятих проєктних і програмних рішень;

застосування сучасних технологій та мов програмування;

наочність та якість ілюстративного матеріалу;

наявність/відсутність дублювання, описового матеріалу, стереотипних рішень, що не впливають на сутність отриманих результатів.

Критерії оцінювання оформлення пояснювальної записки:

відповідність оформлення чинним стандартам та методичним рекомендаціям;

органічний зв'язок текстового матеріалу з графічним;

загальна та професійна грамотність, лаконізм і логічна послідовність викладення матеріалу.

Критерії оцінювання розробленого програмного забезпечення:

працездатність програмного продукту;

відповідність програмного продукту розробленій специфікації функціональних та нефункціональних вимог;

зручність та рівень дизайну інтерфейсу;

можливості впровадження програмного засобу.

Критерії оцінювання захисту дипломного проєкту:

якість і повнота доповіді під час захисту проєкту: якість презентаційного матеріалу, відповідність доповіді темі і меті проєкту; володіння матеріалом, послідовність, логіка, грамотність викладення матеріалу; здатність аргументовано обґрунтувати прийняті в проєкті рішення, коротко пояснити призначення і роботу розробленого програмного забезпечення, робити висновки тощо;

правильність і повнота відповідей на питання ЕК: уміння сформулювати аргументовану відповідь на питання, відповідати на нестандартні

(проблемні) питання, обґрунтувати власну позицію у проблемних ситуаціях.

У процесі оцінювання дипломного проєкту враховують також відгук керівника проєкту та оцінку рецензента.

Оцінку **"відмінно"** (90 – 100 балів) здобувачу вищої освіти отримує, якщо він виконав дипломний проєкт у повному обсязі, з дотриманням всіх вимог, а під час захисту показав: грамотне, логічне викладення доповіді, правильні та повні відповіді на питання (у тому числі – нестандартні); глибоке і повне опанування змісту навчального матеріалу; вміння пов'язувати теорію з практикою, обґрунтовувати свої судження, робити висновки; володіння різносторонніми навичками, прийомами і компетентностями. Дипломний проєкт повністю відповідає вимогам до його змісту та оформлення і розкриває всі положення. Розроблене програмне забезпечення відповідає специфікації вимог і є повнофункціональним; використано сучасні засоби розроблення. Програмний продукт впроваджено на підприємстві (в організації).

Оцінку **"добре"** (74 – 89 балів) виставляють здобувачу вищої освіти у разі, коли він виконав дипломний проєкт у повному обсязі, з дотриманням вимог, а під час захисту демонструє тверде знання матеріалу проєкту, грамотно і суттєво викладає його, не допускає значних неточностей у відповідях на запитання, правильно застосовує теоретичні положення у процесі вирішення практичних завдань, володіє необхідними навичками і прийомами їх виконання. Пояснювальна записка достатньою мірою відповідає вимогам і розкриває ключові положення проєкту. Розроблене програмне забезпечення відповідає специфікації вимог, виконує основні функції; використано сучасні засоби розроблення.

Оцінку **"задовільно"** (60 – 73 балів) отримує здобувач вищої освіти, який виконав дипломний проєкт за завданням, але припустився неточностей під час виконання; у процесі захисту виявив знання основного матеріалу в обсязі, необхідному для професійної діяльності; засвоїв і набув практичних навичок у галузі, в основному справляється з виконанням практичних завдань, але допускає порушення логічної послідовності у викладенні матеріалу, помилки у відповідях на запитання, відчуває труднощі під час відповідей на видозмінені запитання. Пояснювальна записка переважно відповідає вимогам і розкриває більшість положень дипломного проєкту. Розроблене програмне забезпечення виконує

більшість необхідних функцій або його реалізація виконана у спрощеному вигляді (наприклад, сайт-візитівка).

Оцінку **"незадовільно"** (до 60 балів) виставляють тоді, коли дипломний проєкт не відповідає вимогам, а під час захисту здобувач вищої освіти показав безсистемні знання, відсутність вміння виокремлювати головне від другорядного, припустився помилок у визначенні понять; викладення матеріалу виконує хаотично і невпевнено, демонструє не можливість застосовування знань під час вирішення практичних завдань. Пояснювальна записка не відповідає вимогам, недостатньо розкриває положення проєкту. Розроблене програмне забезпечення не відповідає специфікації вимог і темі дипломного проєкту, або відсутнє взагалі.

Рекомендована література

Основна

1. 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 імені Семена Кузнеця. – Харків : ХНЕУ ім. С. Кузнеця, 2019. – 229 с. – Режим доступу : http://repository.hneu.edu.ua/handle/123456789/21526.

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

3. 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.

4. Кучеров Д. П. Інженерія програмного забезпечення : навч. посіб. / Д. П. Кучеров, Є. Б. Артамонов. – Київ : НАУ, 2017. – 386 с.

5. Проектування інформаційних систем: Загальні питання теорії проєктування 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.

6. Ушакова І. О. Лабораторний практикум з системного аналізу та проєктування інформаційних систем [Електронний ресурс] : навч. посіб. / І. О. Ушакова, І. Б. Медведєва. – Харків : ХНЕУ ім. С. Кузнеця, 2022. – 251 с. – Режим доступу : http://repository.hneu.edu.ua/handle/123456789/27815.

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

Додаткова

8. Дейт К. Дж. Введення в системи баз даних / К. Дж. Дейт. – 6-те 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 р. – Київ : УкрНДНЦ, 2016. – 49 с.

11. Левус Є. В. Життєвий цикл програмного забезпечення : навч. посіб. / Є. В. Левус, Т. А. Марусенкова, О. О. Нитребич. – Львів : Вид-во Львівської політехніки, 2017. – 208 с.

12. 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. – Харків : ХНЕУ ім. С. Кузнеця, 2021. – 48 с.

13. Ушакова І. О. Проєктування інформаційних систем : практикум / І. О. Ушакова. – Харків : Вид. ХНЕУ ім. С. Кузнеця, 2015. – 250 с.

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

15. 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.

16. ISO/IEC/IEEE 15939:2017: Systems and software engineering – Measurement process [Electronic resources]. – Access mode : https://www.iso.org/obp/ui/en/#iso:std:iso-iec-ieee:15939:ed-1:v1:en.

17. ISO/IEC 25000:2014: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE [Electronic resources]. – Access mode : https://www.iso.org/obp/ui/#iso:std:iso-iec:25000:ed-2:v1:en.

18. ISO/IEC 25010:2011: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models [Electronic resources]. – Access mode : https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-2:v1:en.

19. ISO/IEC 25020:2019: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Quality measurement framework [Electronic resources]. – Access mode : https://www.iso.org/obp/ui/#iso:std:iso-iec:25020:ed-2:v1:en.

20. ISO/IEC 25022:2016: Systems and software engineering – Systems and software quality requirements and evaluation (SQuaRE) – Measurement of quality in use [Electronic resources]. – Access mode : https://www.iso.org/obp/ui/#iso:std:iso-iec:25022:ed-1:v1:en.

21. ISO/IEC 25023:2016: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) Measurement of system and software product quality [Electronic resources]. – Access mode : https://www.iso.org/obp/ui/#iso:std:iso-iec:25023:ed-1:v1:en.

22. ISO/IEC 25040:2011: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Evaluation process [Electronic resources]. – Access mode : https://www.iso.org/obp/ui/#iso:std:iso-iec:25040:ed-1:v1:en.

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 / K. Wiegers, C. Hokanson. – Addison-Wesley Professional, 2023.– 208 p.

Інформаційні ресурси

26. Державна служба статистики України. – Режим доступу : https://www.ukrstat.gov.ua/.

27. Інформаційний портал CRM. – Режим доступу : https://www.crm.com.ua.

28. Про банки і банківську діяльність: Закон України № 2121-III від 07.12.2000 р. : за станом на 01 липня 2024 р. – Режим доступу : https://zakon.rada.gov.ua/laws/show/2121-14#Text.

29. Про захист персональних даних : Закон України № 2297-VI від 01.06.2010 р. : за станом на 27 жовтня 2022 року [Електронний ресурс]. – Режим доступу : https://zakon.rada.gov.ua/laws/show/2297-17#Text.

30. Про інформацію : Закон України № 2657-XII від 02.10.1992 р. : за станом на 21 березня 2023 року [Електронний ресурс]. – Режим доступу : https://zakon.rada.gov.ua/laws/show/2657-12#Text.

31. Спільнота розробників dou.ua. – Режим доступу : https://dou.ua/.

32. 10 usability heuristics for user interface design [Electronic resource]. – Access mode : https://www.nngroup.com/articles/ten-usability-heuristics/.

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

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

35. IBM [Electronic resource]. – Access mode : http://www.ibm.com/ua-en.

36. ITC Online [Electronic resource]. – Access mode : http://itc.ua/.

37. Microsoft [Electronic resource]. – Access mode : http://www.microsoft.com/.

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

39. 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) [Electronic resource]. – Access mode : https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679.

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

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

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

Додатки

Додаток А

Зразок заяви на затвердження теми дипломного проєкту

Завідувачу кафедри
інформаційних систем
доценту Бондаренку Д. О.
студента 4-го курсу <номер> групи
<ПІБ студента>

ЗАЯВА

Прошу затвердити мені тему дипломного проєкту <Назва теми>.

<Дата> <ПІБ> <Підпис студента>

Керівник
дипломного проєкту

<Посада

наук. ступінь, учене звання> <Підпис керівника> <ПІБ керівника>

Додаток Б

Структура змістової 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 наявних рішень і аналогів
1.4.	Позиціонування створюваного програмного забезпечення (застосунку, модуля, 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 і неперервної доставки програмного CD забезпечення або його компонентів*
4.	*ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ*
4.1.	*Створення чеклистів*
4.1.1.	*Чеклист для димового тестування.*
4.1.2.	*Чеклист 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(-ла): студент(ка) <Ім'я ПРІЗВИЩЕ студента>

Керівник: <науковий ступінь>, <учене звання> <Ім'я ПРІЗВИЩЕ>

Рецензент: <посада> <Ім'я ПРІЗВИЩЕ>

Харків – 202_

Додаток Г

Приклади рефератів

РЕФЕРАТ

Пояснювальна записка до дипломного проєкту: 90 с., 30 рисунків, 20 таблиць, 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 і оперативність ухвалення рішень під час управління поставками товарів і їхніми запасами на складі.

Результати розроблення можуть бути впроваджені на торговельних підприємствах.

СИСТЕМА УПРАЛІННЯ ЛАНЦЮГАМИ ПОСТАВОК, ОБ'ЄКТНООРІЄНТОВАНЕ 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 uses 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 написання дипломного проєкту

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 *Н. Г. Войчук*

Коректор *В. О. Дмитрієва*

План 2024 р. Поз. № 82 ЕВ. Обсяг 77 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.