МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

МЕТОДИЧНІ ВКАЗІВКИ
ЩОДО ВИКОНАННЯ САМОСТІЙНОЇ РОБОТИ
З НАВЧАЛЬНОЇ ДИСЦИПЛІНИ
«ОСНОВИ АНАЛІЗУ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»
ДЛЯ СТУДЕНТІВ ДЕННОЇ ФОРМИ НАВЧАННЯ
ЗІ СПЕЦІАЛЬНОСТІ 123 – «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ»

КРЕМЕНЧУК 2020

Методичні вказівки щодо виконання самостійної роботи з навчальної дисципліни «Основи аналізу якості програмного забезпечення» для студентів денної форми навчання зі спеціальності 123 – «Комп’ютерні системи та мережі»

Укладач к. т. н., доц. О. Г. Славко

Рецензент к. т. н., доц. В. М. Сидоренко

Кафедра комп’ютерних та інформаційних систем

Затверджено методичною радою КрНУ імені Михайла Остроградського
Протокол № __ від _____ 2020 р.

Голова методичної ради _____ проф. В. В. Костін

ЗМІСТ

ВСТУП

Навчальна дисципліна є логічним продовженням навчальних дисциплін «Прикладне програмування в комп'ютерних мережах», «Інженерія програмного забезпечення», «Комп'ютерні мережі», «Технології критичної програмної інженерії», «Організація баз даних». Предметом вивчення навчальної дисципліни є теорія і практика аналізу якості програмного забезпечення на базі сучасних технологій розробки програмного забезпечення, інструментів і систем баг-трекінгу та тест-управління.

У межах навчальної дисципліни розглядаються питання аналізу, забезпечення та контролю якості програмного забезпечення. Студенти набувають практичних навичок і здатності застосовувати технології та інструментальні засоби надання інформації про якість ПЗ кінцевому замовнику, підвищення якості ПЗ, запобігання появи дефектів, тестування програмного забезпечення.

Мета і завдання навчальної дисципліни: набуття студентами загальних теоретичних і практичних знань у галузі аналізу якості програмного забезпечення (ПЗ), які базуються на сучасних методологіях проектування програмних проектів і програмного забезпечення, організації функціонування та супроводження програмних проектів, а також контролю та забезпечення якості і тестування програмного забезпечення.

Місце навчальної дисципліни у навчальному процесі: навчальна дисципліна «Основи аналізу якості програмного забезпечення» тісно пов'язана з такими дисциплінами, як «Прикладне програмування в комп'ютерних мережах», «Інженерія програмного забезпечення», «Комп'ютерні мережі», «Технології критичної програмної інженерії», «Організація баз даних» та іншими спеціальними дисциплінами навчального плану.

У результаті вивчення навчальної дисципліни студент повинен знати:

– основні моделі розробки програмного забезпечення;

– основні завдання та цілі забезпечення і контролю якості та тестування програмного забезпечення;

– рівні, типи і види тестування;

– тестові артефакти;

– життєвий цикл дефекту;

– основи вимоги до тестування та оцінювання якості програмного забезпечення, web-проектів і мобільних застосунків;

уміти:

– працювати із системами обліку дефектів (баг-трекінговими системами);

– працювати з Test Management системами;

– перевіряти функціональність, бізнес-логіку продукту, графічний інтерфейс, коректність виконання головних завдань продукту та зручність користувачів;

– працювати з техніками тест-дизайну;

– грамотно висловлювати свої думки в тексті, спілкуватися з різними типами людей (навіть тими, хто викликає неприязнь), планувати свій час.

1 ТЕМИ ТА ПОГОДИННИЙ РОЗКЛАД ЛЕКЦІЙ І САМОСТІЙНОЇ РОБОТИ

Назви змістових модулів і тем	Кількість годин				
	денна форма				
	усього	у тому числі			
		л	п	лаб	с.р.
1	2	3	4	5	6
Модуль 1					
Змістовний модуль 1 Основні поняття якості програмного забезпечення					
Тема 1 Введення до тестування ПЗ	14	2	-	-	12
Тема 2 Опис і структура дефектів. Робота з дефектами	16	2	-	2	12
Тема 3. Тестова документація і артефакти тестування	16	2	-	2	12
Тема 4. Основні види і типи тестування ПЗ	16	2	-	2	12
Тема 5. Основні підходи до тестування	16	2	-	2	12
Усього за змістовим модулем 1	78	10	-	8	60
Змістовий модуль 2 Основи тестування програмного забезпечення, web-проектів та мобільних застосунків					
Тема 6 Основні поняття SDLC: головні етапи та артефакти життєвого циклу розробки ПЗ	16	2	-	-	14
Тема 7 Основний інструментарій для аналізу якості під час роботи з ПЗ	22	2	-	6	14
Тема 8 Основи тестування ПЗ	24	4	-	6	14
Тема 9 Основи тестування web-проектів	18	2	-	2	14
Тема 10 Основи тестування мобільних застосунків	16	2	-	-	14
Усього за змістовим модулем 2	96	12	-	14	70
ІНДЗ (КР, РГ, к/р)	4	-	-	-	4
Семестровий контроль (диф. залік)	2	-	-	-	2
Усього годин	180	22	-	22	136

2 ЗАВДАННЯ ДЛЯ САМОСТІЙНОЇ РОБОТИ

Тема. Розробка тест плану

Мета: набуття практичних навичок планування робіт з розробки та впровадження автоматизованих інформаційних систем на основі поширених моделей життєвих циклів програмних продуктів.

Короткі теоретичні відомості

Тестування програмного забезпечення (Software Testing) – це перевірка відповідності між реальною і очікуваною поведінкою програми, що здійснюється на кінцевому наборі тестів, вибраних певним чином. У більш широкому змісті, тестування – це одна з технік контролю якості, що передбачає діяльність з планування робіт (Test Management), проектування тестів (Test Design), виконання тестування (Test Execution) і аналізу отриманих результатів (Test Analysis).

Верифікація (verification) програми та її компонентів – це визначення відповідності результатів поточного етапу розробки умовам, сформованим на початку цього етапу (IEEE). Тобто чи виконуються цілі, терміни, задачі з розробки проекту, визначені на початку поточної фази.

Валідація (validation) – це визначення відповідності розроблювального ПЗ очікуванням і потребам користувача, вимогам до системи.

План тестування (Test Plan) – це документ, що описує обсяг робіт з тестування, починаючи з опису об'єкта, стратегії, розкладу, критеріїв початку і закінчення тестування, до необхідного в процесі роботи устаткування, спеціальних знань, а також оцінки ризиків з варіантами їхнього рішення.

Тест дизайн (Test Design) – це етап процесу тестування ПЗ, на якому проектуються і створюються тестові випадки (тест кейси), відповідно до визначених раніше критеріїв якості та цілей тестування.

Тестовий випадок (Test Case) – це сукупність кроків, конкретних умов і параметрів, необхідних для перевірки реалізації функції або її частини, що

тестуються.

Звіт про помилку/Дефект Репорт (Bug Report) – це документ, що описує ситуацію або послідовність дій, що призвели до некоректної роботи об'єкта тестування, із вказівкою причин і очікуваного результату.

Тестове Покриття (Test Coverage) – це одна з метрик оцінювання якості тестування, що являє собою щільність покриття тестами вимог або коду, що виконується.

Деталізація Тест Кейсів (Test Case Detalization) – це рівень деталізації опису тестових кроків і необхідного результату, за якого забезпечується розумне співвідношення часу проходження до тестового покриття.

Час Проходження Тест Кейса (Test Case Pass Time) – це час від початку проходження кроків тест кейса до одержання результату тесту.

Залежно від переслідуваних цілей, види тестування можна умовно розділити на такі типи: функціональне, нефункціональне, пов'язане зі змінами.

А. *Функціональне* тестування.

Функціональні тести базуються на функціях і особливостях, а також взаємодії з іншими системами, і можуть бути на всіх рівнях тестування: компонентному або модульному (Component/Unit testing), інтеграційному (Integration testing), системному (System testing) і приймальному (Acceptance testing). Функціональні види тестування розглядають зовнішнє поводження системи. Далі названі найпоширеніші види функціональних тестів:

А.1 *Функціональне* тестування (Functional testing).

А.2 Тестування *безпеки* (Security and Access Control Testing).

А.3 Тестування *взаємодії* (Interoperability Testing):

А.3.1 Тестування *сумісності* (compatibility testing);

А.3.2 *Інтеграційне* тестування (integration testing).

Б. *Нефункціональне* тестування.

Нефункціональне тестування описує тести, необхідні для визначення характеристик програмного забезпечення, що можуть бути вимірювані різними величинами. У цілому, це тестування того, «як» система працює. Основні види

нефункціональних тестів.

Б.1 Види тестування *продуктивності*:

Б.1.1 *Навантажувальне* тестування (Performance and Load Testing).

Б.1.2 *Стресове* тестування (Stress Testing).

Б.1.3 Тестування *стабільності* або *надійності* (Stability / Reliability Testing).

Б.1.4 *Об'ємне* тестування (Volume Testing).

Б.2. Тестування *установки* (Installation testing).

Б.3 Тестування *зручності користування* (Usability Testing).

Б.4 Тестування на *відмовлення* та *відновлення* (Failover and Recovery Testing).

Б.5 *Конфігураційне* тестування (Configuration Testing).

В *Пов'язане зі змінами* тестування.

Після проведення необхідних змін, таких як виправлення помилки – бага/дефекту, ПЗ має бути знов протестоване для підтвердження того факту, що проблема була дійсно розв'язана. Нижче подано види тестування, які необхідно проводити після установки ПЗ, для підтвердження працездатності додатка або вірності здійсненого виправлення дефекту.

В.1 *Димове* тестування (Smoke Testing).

В.2 *Регресійне* тестування (Regression Testing).

В.3 *Тестування складання* (Build Verification Test).

В.4 *Санітарне* тестування або *перевірка погодженості/справності* (Sanity Testing).

Розглянемо детальніше деякі види тестування.

А.1 Функціональне тестування (Functional Testing) розглядає заздалегідь зазначену поведінку і ґрунтується на аналізі специфікацій функціональності компонента або системи в цілому.

Функціональні тести ґрунтуються на функціях, виконуваних системою, і можуть проводитися на всіх рівнях тестування (компонентному, інтеграційному, системному, приймальному). Зазвичай ці функції описуються у

вимогах, функціональних специфікаціях або у вигляді випадків використання системи (use cases).

Тестування функціональності може проводиться в двох аспектах:

- вимоги;
- бізнес-процеси.

Тестування в перспективі «вимоги» використовує специфікацію функціональних вимог до системи як підґрунтя для дизайну тестових випадків (Test Cases). У цьому випадку необхідно зробити перелік того, що буде тестуватися, а що ні, визначають пріоритетність вимог на основі ризиків (якщо це не зроблено в документі з вимогами), а на підставі цього переліку пріоритетів формуються тестові сценарії (test cases). Це дозволяє сфокусуватися під час тестування на важливішому функціоналі.

Тестування в перспективі «бізнес-процеси» використовує знання самих бізнес-процесів, що описують сценарії щоденного використання системи. У цій перспективі тестові сценарії (test scripts) зазвичай ґрунтуються на випадках використання системи (use cases).

А.2 Тестування безпеки (Security and Access Control Testing) – це перевірка безпеки системи, а також аналіз ризиків, пов’язаних із забезпеченням цілісного підходу до захисту додатка, атак хакерів, вірусів, несанкціонованого доступу до конфіденційних даних. Тестування безпеки може виконуватися як автоматизовано, так і вручну, у тому числі перевірку як позитивних, так і негативних тестових випадків. Ґрунтується на трьох основних принципах – *конфіденційність, цілісність і доступність* (confidentiality, integrity, availability)

А.3 Тестування взаємодії (Interoperability Testing) – це функціональне тестування, що перевіряє здатність додатка взаємодіяти з одним і більш компонентами або системами, що передбачає тестування сумісності (compatibility testing) та інтеграційне тестування (integration testing).

Б.1 Види тестування продуктивності (Performance testing)

Завданням *тестування продуктивності (Performance testing)* є визначення масштабованості додатка під навантаженням, під час якого

відбувається:

− вимірювання часу виконання вибраних операцій за визначених інтенсивностей їх виконання;

− визначення кількості користувачів, що одночасно працюють з додатком;

− визначення меж прийнятної продуктивності за збільшеного навантаження (за умови збільшення інтенсивності виконання цих операцій);

− дослідження продуктивності за високих, граничних, стресових навантажень.

Б.1.1 Навантажувальне тестування (Load vs Performance Testing). У англомовній термінології існує ще один вид тестування – Load Testing – тестування реакції системи на зміну навантаження (у межах припустимого). Load і Performance мають на меті таку саму мету: перевірка продуктивності (часів відгуку) за різних навантажень. Власне тому їх не розділяють.

Б.1.2 Стресове тестування (Stress Testing) дозволяє перевірити, наскільки додаток і система в цілому працездатні в умовах стресу і також оцінити здатність системи до регенерації, тобто до повернення до нормального стану після припинення впливу стресу. Стресом у даному контексті може бути підвищення інтенсивності виконання операцій до дуже високих значень або аварійна зміна конфігурації сервера. Також одним із завдань стресового тестування може бути оцінювання деградації продуктивності, у такий спосіб мета стресового тестування може перетинатися з цілями тестування продуктивності.

Б.1.3 Тестування стабільності або надійності (Stability / Reliability Testing) – це перевірка працездатності додатка у разі тривалого (багатогодинному) тестуванні із середнім рівнем навантаження. Час виконання операцій може мати в цьому виді тестування другорядне значення. При цьому першочергове значення має відсутність витоків пам’яті, перезапусків серверів під навантаженням та інші аспекти, що впливають саме на стабільність роботи.

Б.1.4 Об'ємне тестування (Volume Testing) – це одержання оцінки продуктивності за збільшення обсягів даних у базі дані додатка, під час якого відбувається:

− вимірювання часу виконання обраних операцій за визначених інтенсивностей їх виконання;

− може визначатися кількість користувачів, які одночасно працюють з додатком.

Б.2 Тестування установки (Installation Testing) спрямоване на перевірку успішної інсталяції та настроювання, а також відновлення або видалення ПЗ. Інсталяція відбувається автоматично вручну та за допомогою візардів.

Б.3 Тестування зручності користування (Usability Testing) – це метод тестування, спрямований на встановлення ступеня зручності використання, навчання, зрозумілості та привабливості для користувачів розроблювального продукту в контексті заданих умов.

В. Тестування, що пов'язане зі змінами

В.1 Димове тестування (Smoke Testing) спрямоване на поверхневу перевірку всіх модулів додатка щодо працездатності та наявність швидкого знаходження критичних дефектів і дефектів, що блокують. За результатами димового тестування робиться висновок про те, чи приймається, чи ні установлена версія ПЗ в тестування, експлуатацію або на постачання замовникові. Для полегшення роботи, економії часу і людських ресурсів рекомендується автоматизувати димові тести.

В.2 Регресійне тестування (Regression Testing) спрямоване на перевірку змін, зроблених у додатку або навколишньому середовищі (налагодження дефекту, злиття коду, міграція на іншу операційну систему, базу даних, веб сервер або сервер додатка), для підтвердження того факту, що функціональність, яка існувала раніше, працює, як і колись. Регресійними можуть бути тести як функціональні, так і не функціональні.

Зазвичай для регресійного тестування використовуються тест-кейси, написані на ранніх стадіях розробки і тестування. Це надає гарантію того, що

зміни в новій версії додатка не зашкодили вже наявну функціональність. Рекомендується робити автоматизацію регресійних тестів для прискорення наступного процесу тестування і виявлення дефектів на ранніх стадіях розробки програмного забезпечення.

Сам по собі термін «регресійне тестування», залежно від контексту використання може мати різний сенс.

Визначено 3 основних типи регресійного тестування:

− регресія помилок (Bug regression) – спроба довести, що виправлена помилка насправді не виправлена;

− регресія старих помилок (Old bugs regression) – спроба довести, що недавня зміна коду або даних зламала виправлення старих помилок, тобто старі помилки – баги почали знову відтворюватися;

− регресія побічного ефекту (Side effect regression) – спроба довести, що недавня зміна коду або даних зламала інші частини розроблювального додатка.

В.3 Тестування складання (Build Verification Test)

Тестування спрямоване на визначення відповідності, випущеної версії, критеріям якості для початку тестування. За своїми цілями є аналогом димового тестування, спрямованого на приймання нової версії в подальше тестування або експлуатацію. Вглибину воно може проникати далі, залежно від вимог до якості випущеної версії.

В.4 Санітарне тестування або перевірка погодженості/справності (Sanity Testing). Вузькоспеціалізоване тестування достатнє для доведення того, що конкретна функція працює згідно заявленим у специфікації вимогам і є підмножиною регресійного тестування. Використовується для визначення працездатності визначеної частини додатка після змін зроблених у ньому або у навколишньому середовищі. Зазвичай виконується вручну.

Порядок виконання самостійної роботи

1. Вибрати за допомогою викладача або самостійно ПЗ для тестування.

2. Обґрунтувати вибір видів тестування для відповідного ПЗ.

3. Розробити тест план для тестування вибраної програми (на власний вибір).

4. Заповнити документ тест-плану для відповідного ПЗ (форма тест-плану наведена у додатках).

5. Оформити звіт із самостійної роботи зі створеним тест-планом.

Зміст звіту

1. Титульна сторінка.

2. Опис виконання роботи відповідно до пп. 1–8.

Питання для самоперевірки

1. У чому полягає різниця між функціональним і не функціональним тестуванням?

2. Надайте характеристику тестуванню навантаження?

3. Що передбачає тестування, пов’язане зі змінами?

4. У чому полягає відмінність димового та санітарного тестування?

5. Які особливості тестування установки?

6. Які особливості регресійного тестування?

Література: [1–3, 5, 7–10, 11].

3 КРИТЕРІЇ ОЦІНЮВАННЯ ЯКОСТІ ВИКОНАННЯ САМОСТІЙНОЇ РОБОТИ СТУДЕНТАМИ

У 11-му семестрі студенти виконують самостійну роботу, що складається з 5 завдань. Загальна кількість балів, яку отримують студенти за виконання самостійної роботи, складає максимально 5 балів.

Шкала оцінювання: національна та ECTS

<table>
<tr><td rowspan="2">Сума балів за всі види навчальної діяльності</td><td rowspan="2">Оцінка ECTS</td><td colspan="2">Оцінка за національною шкалою</td></tr>
<tr><td>Для іспиту, курсового проекту (роботи), практики</td><td>Для заліку</td></tr>
<tr><td>90–100</td><td>A</td><td>Відмінно</td><td rowspan="5">Зараховано</td></tr>
<tr><td>82–89</td><td>B</td><td rowspan="2">Добре</td></tr>
<tr><td>74–81</td><td>C</td></tr>
<tr><td>64–73</td><td>D</td><td rowspan="2">Задовільно</td></tr>
<tr><td>60–63</td><td>E</td></tr>
<tr><td>35–59</td><td>FX</td><td>Незадовільно з можливістю повторного складання</td><td>Не зараховано з можливістю повторного складання</td></tr>
<tr><td>0–34</td><td>F</td><td>Незадовільно з обов’язковим повторним вивченням навчальної дисципліни</td><td>Не зараховано з обов’язковим повторним вивченням навчальної дисципліни</td></tr>
</table>

СПИСОК ЛІТЕРАТУРИ

1. Майерс Г., Баджетт Т., Сандлер К. Искусство тестирования программ. М.: «Диалектика», 2012. 272 с.
2. Криспин Л., Грегори Дж. Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд. М.: «Вильямс», 2010. 464 с.
3. Канер К., Фолк Дж., Нгуен Е. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений. К.: ДиаСофт, 2001. 544 с.
4. Калбертсон Р., Браун К., Кобб Г. Быстрое тестирование. М.: «Вильямс», 2002. 374 с.
5. Синицын С. В., Налютин Н. Ю. Верификация программного обеспечения. М.: БИНОМ, 2008. 368 с.
6. Бейзер Б. Тестирование чёрного ящика. Технологии функционального тестирования программного обеспечения и систем. СПб.: Питер, 2004. 320 с.
7. Кон М. Scrum: гибкая разработка ПО. М.: «Вильямс», 2011. 576 с.
8. Мартин Р. С., Ньюкирк Дж. В., Косс Р. С. Быстрая разработка программ. Принципы, примеры, практика. М. : «Вильямс», 2004. 752 с.
9. Савин Р. Тестирование Дот Ком или Пособие по жестокому обращению с багами в интернет-стартапах. М.: Дело, 2007. 312 с.
10. Котляров В. П., Коликова Т. В. Основы тестирования программного обеспечения. М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2006. 285 с.
11. Портал специалистов по тестированию и обеспечению качества ПО [Электронный ресурс]. – Режим доступа: http://software-testing.ru/.
12. Технические статьи [Электронный ресурс]. – Режим доступа: – http://training.qatestlab.com/front-page/blog/technical-articles/.
13. Куликов С.С. Тестирование программного обеспечения. Базовый курс. Минск: Четыре четверти, 2017. 312 с.

ДОДАТКИ

Форма тест-плану

Назва проекту

(Project Name)

План випробувань і тестування

(Test Plan)

Версія 1.0

(Version 1.0)

Історія змін (Revision History)

Дата (Date dd/mmm/yy)	Версія (Version x.x)	Детальний опис (Description details)	ПІБ Автора (Author name)

Зміст (Table of Contents)

Тестовий план (Test Plan)

1 Вступ (Introduction)

1.1 Мета (Purpose)

- Виявлення наявних проектів і програмних компонентів, які необхідно перевірити.
- Перелік вимог для проведення випробування.
- Рекомендації щодо опису стратегії тестування.
- Визначення необхідних ресурсів і забезпечення оцінювання випробувань.
- Перелік тестових елементів проекту.

1.2 Довідкова інформація (Background)

Опис елементів тестування (компоненти, додатки, системи тощо). Інформація щодо основних функцій і можливостей, архітектури, короткої історії проекту. Цей розділ містить від 3 до 5 пунктів.

1.3 Галузь застосування (Scope)

Описують:

- типи, наприклад, групи, інтеграцію, систему, етапи тестування (функціональність і продуктивність);
- перелік особливостей функцій, що будуть протестовані;
- перелік пропозицій, що можуть вплинути на проектування, розробку тестування;
- перелік усіх ризиків й непередбачуваних обставин, що можуть вплинути на розробку тестування;
- перелік обмежень, які можуть вплинути на проектування, розробку тестування.

1.4 Визначення проекту (Project Identification)

Примітка: видаляти або додавати елементи таблиці можна за необхідності.

Документ і версія / дата) (Document and version / date)	Створено або Доступно (Created or Available)	Поступило або Відзив (Received or Reviewed)	Автор або ресурс (Author or Resource)	Приміт ка (Notes)
Специфікація вимог (Requirements Specification)	☐ Так ☐ Ні	☐ Так ☐ Ні		
Функціональна специфікація (Functional Specification)	☐ Так ☐ Ні	☐ Так ☐ Ні		
Звіти використання - випадок (Use-Case Reports)	☐ Так ☐ Ні	☐ Так ☐ Ні		
План проекту (Project Plan)	☐ Так ☐ Ні	☐ Так ☐ Ні		
Специфікація дизайну (Design Specifications)	☐ Так ☐ Ні	☐ Так ☐ Ні		
Прототип (Prototype)	☐ Так ☐ Ні	☐ Так ☐ Ні		
Керівництво користувача (User's Manuals)	☐ Так ☐ Ні	☐ Так ☐ Ні		
Бізнес модель (Business Model or Flow)	☐ Так ☐ Ні	☐ Так ☐ Ні		
Модель даних (Data Model or Flow)	☐ Так ☐ Ні	☐ Так ☐ Ні		
Бізнес-функції (Business Functions and Rules)	☐ Так ☐ Ні	☐ Так ☐ Ні		
Оцінка ризиків (Project or Business Risk Assessment)	☐ Так ☐ Ні	☐ Так ☐ Ні		

2 Вимоги до тестування (Requirements for Test)

Перелік використання функціональних вимог із зазначенням пунктів, цілей для нефункціональних вимог тестування. Цей перелік відображає, що має бути перевірено.

3 Стратегія тестування (Test Strategy)

У цьому розділі подають рекомендований підхід до тестування. У попередньому розділі описують вимоги до випробувань, що мають бути протестовані, у цьому розділі – як це має бути зроблено.

Для кожного випробування необхідно забезпечити опис випробування із зазначенням необхідності його проведення.

Якщо тип тесту не буде реалізований і виконаний, необхідно обґрунтовано вказати причину його невиконання. Наприклад, «Цей тест не буде виконуватися, тому що він непридатний для випробування системи» тощо.

Основними вимогами для випробування стратегії є використовувані методи та критерії тестування.

Тестування має бути виконане тільки з використанням відомих, контролюючих БД в захищених середовищах.

3.1 Типи тестування (Testing Types)

3.1.1 Дані і БД Інтеграційне тестування (Data and Database Integrity Testing)

БД і БД процесів мають бути перевірені як підсистеми в межах проекту. Ці підсистеми мають бути перевірені як інтерфейс з даними без тестування Інтерфейсу Користувача.

Додаткові дослідження в СУБД мають бути виконані із зазначенням інструментів і методів, які можуть існувати для підтримки тестування.

Мета випробування (Test Objective)	Забезпечення методів доступу до БД і функціональних процесів без пошкодження даних
Технічний прийом (Technique)	Виклик кожного методу доступу до БД і функціональних процесів, відбір дійсних та недійсних даних або запитів. Перевірка БД для забезпечення: цілісності та несуперечних даних, а також усіх запитів, що пов'язані з роботою БД
Критерії завершення (Completion Criteria)	Виконання всіх методів доступу до БД і функціональних процесів без пошкодження даних
Спеціальні рекомендації (Special Considerations)	Тестування середовища СУБД може вимагати розвитку або драйверів для вводу або зміни даних безпосередньо в базах даних. Процеси мають бути викликані вручну. Малий або мінімальний розмір БД (обмежена кількість записів) має бути використаний для підвищення видимості будь-якої неприйнятої події

3.1.2 Функціональне тестування (Function Testing)

Функція тестування має зосередитися на будь-яких вимогах для випробувань, які можуть бути простежені, безпосередньо використовуватимуть випадки або бізнес-функції та бізнес-правила. Мета цих тестів – перевірка правильності прийняття даних, обробки і пошуку, а також належного виконання бізнес-правил. Цей тип тестування заснований на методі тестування «чорної скрині», тобто перевірки додатка і його внутрішніх процесів, взаємодіючи з додатком через графічний інтерфейс користувача (GUI), й аналізі продукції або результатів. Зазначений нижче план тестування рекомендується для кожного додатка.

Мета випробування (Test Objective)	Забезпечити належне тестування функціональності, в тому числі навігації, введення даних, обробки і пошуку.
Технічний прийом (Technique)	Виконати перевірку кожного випадку потоку, або функції, з використанням дійсних і недійсних даних, щоб перевірити: - очікувані результати, що виникають за достовірними даними; - відповідні помилки або попередження, що виникають у разі введення неправильних даних; - правильність застосування кожного бізнес-правила
Критерії завершення (Completion Criteria)	- усі заплановані випробування було проведено; - усі виявлені дефекти були розглянуті
Спеціальні рекомендації (Special Considerations)	Визначити або описати ті пункти або питання, які впливають на здійснення та виконання основної функції

3.1.3 Бізнес-цикл тестування (Business Cycle Testing)

Бізнес-цикл тестування має здійснюватися протягом визначеного часу проекту, Наприклад, один рік, а також операції та заходи, які відбуватимуться в цей період, мають бути виконані. Це передбачає всі щоденні, щотижневі та щомісячні цикли і, події, дати з урахуванням регістру тощо.

Мета випробування (Test Objective)	Забезпечити належне тестування і фонові функціональні процеси відповідно до необхідної бізнес-моделей та графіків.
Технічний прийом (Technique)	Тестування буде імітувати кілька циклів ділової активності, а саме: – тести, що використовуються для тестування цільової функції, яка може бути змінена або розширена, для збільшення числа. Кожна функція виконується для імітації різних користувачів протягом певного періоду часу; – уесь час або тестування функцій буде виконуватися з використанням дійсних і недійсних дат або періодів часу; – усі функції, які відбуваються згідно за графіком періодично будуть виконані і розпочаті у відповідний час; – тестування передбачає використання дійсних і недійсних даних для перевірки: – використовуються очікувані результати, що виникають за достовірних даних; – використовуються відповідні помилки або попередження, що відображаються у разі введу неправильних даних; – правильно застосовувати бізнес-правила.
Критерії завершення (Completion Criteria)	Усі заплановані випробування були проведені. Усі виявлені дефекти були розглянуті
Спеціальні рекомендації (Special Considerations)	Система дат і подій може вимагати спеціальної підтримки діяльності бізнес-моделі, необхідної для визначення відповідних вимог до випробувань та процедур

3.1.4 Тестування інтерфейсу користувача (User Interface Testing)

Тестування інтерфейсу користувача (UI) – це перевірка взаємодії користувача з програмним забезпеченням. Мета тестування інтерфейсу користувача полягає в зручності надання користувачеві, з відповідним рівнем доступу, навігації цільових функцій. Окрім того, тестування інтерфейсу користувача гарантує відповідність об’єктів функцій UI очікуваним і/або корпоративним або галузевим стандартам.

Мета випробування (Test Objective)	Перевірте: - навігацію через механізм тестування програми з правильним відображенням бізнес-функцій та вимог, у тому числі вікна до вікна, поля для поля, і використання методів доступу (закладка клавіш, рух миші, поєднання клавіш); - вікно об’єктів і характеристик, таких як меню, розмір, положення, стан, має відповідати нормам
Технічний прийом (Technique)	Створення та редагування випробувань для кожного вікна, щоб перевірити правильність навігації та станів об’єкта для кожного вікна програми і об’єктів
Критерії завершення (Completion Criteria)	Кожне вікно послідовно успішно перевірене у тестовій версії або протягом прийнятного рівня
Спеціальні рекомендації (Special Considerations)	Не всі властивості для користувача об’єктів і третього учасника можуть бути доступні

3.1.5 Тестування продуктивності (Performance Profiling)

Тестування продуктивності передбачає вимірювання оцінки часу відгуку, швидкості транзакції та інших термінових вимог. Метою тестування продуктивності є перевірка

виконання вимог, що були досягнуті. Тестування продуктивності реалізується і виконується згідно з профілем, а також розглядається залежно від робочого навантаження або апаратної конфігурації.

Примітка: Операції в таблиці нижче, відносяться до «логічних операцій». Ці операції визначаються як конкретні випадки використання.

Мета випробування (Test Objective)	Перевірка продуктивності поведінки для призначених операцій або бізнес-функцій за дотримання таких умов: нормальний очікуваний обсяг; очікується гірший випадок навантаження
Технічний прийом (Technique)	Використання випробувань і тестування функцій. Зміна файлів даних зі збільшенням числа транзакцій або скриптів для збільшення кількості ітерацій кожної транзакції. Сценарії мають бути запущені на одній машині (кращий випадок для порівняння одного користувача, однієї транзакції) і повторюватися з декількома клієнтами – віртуальними або фактичними (див. спеціальні рекомендації нижче)
Критерії завершення (Completion Criteria)	Одна операція або один користувач: успішне завершення тестових скриптів без будь-яких збоїв і в межах очікування або в межах виділеного часу на транзакцію. Кілька угод або кілька користувачів: успішне завершення тестових скриптів без будь-яких збоїв і в межах прийнятного розподілу часу
Спеціальні рекомендації (Special Considerations)	Охоплююче тестування включає наявність фону навантаження на сервер. Є кілька методів, які можуть бути використані для виконання цього, в тому числі: - «драйв операцій» безпосередньо на сервері, зазвичай у вигляді мови структурованих запитів (SQL) звернень; - створення «віртуальних» користувачів навантаження для імітації чисельних клієнтів, зазвичай кілька сотень. Віддалений термінал емуляції – інструменти використовуються для досягнення цього навантаження. Цей метод може також бути використаний для завантаження мережі з трафіком; - використання декількох фізичних клієнтів, кожний з яких виконує сценарії випробувань на місці навантаження на систему. Тестування має проводитися на виділеному комп'ютері або в означений час. Це дозволяє повністю контролювати і точніше проводити вимірювання. Бази даних, що використовуються для тестування продуктивності, мають бути або фактичного розміру або однакового масштабу.

3.1.6 Завантажувальне тестування (Load Testing)

Тестування навантаження – це тестування продуктивності для вимірювання та оцінювання ефективності поведінки і здатності продовжувати нормально функціонувати в цих різних робочих навантаженнях. Метою навантажувального тестування є визначення і переконання того, що система функціонує належним чином після передбачуваного максимального обсягу. Окрім того, тестування навантаження оцінює характеристики, такі як час відгуку, швидкість транзакції, час тощо.

Примітка. Операції нижче, відносяться до «логічних». Ці операції визначаються як специфічні функції, які кінцевий користувач системи повинен виконувати за допомогою програми

Мета випробування (Test Objective)	Перевірка роботи в часі для призначених операцій або бізнес-кейсів за різних умов навантаження
Технічний прийом (Technique)	Використовуйте тести, розроблені для функцій або для тестування бізнес-циклів. Змінити файли даних для збільшення числа операцій, щоб збільшити кількість звернень
Критерії завершення (Completion Criteria)	Декілька транзакцій або декількох користувачів. Успішне завершення випробувань без будь-яких збоїв і в межах прийнятного розподілу часу
Спеціальні рекомендації (Special Considerations)	Завантажувальне тестування має проводитися на виділеному комп'ютері або у відведений час. Це дозволяє повністю контролювати і точно вимірювати. Бази даних, що використовуються для завантажувального тестування, мають бути або фактичного розміру або однакового масштабу

3.1.7 Стресове тестування (Stress Testing)

Реалізується і виконується для пошуку помилок через дефіцит ресурсів або конкуренції за ресурси. Недостатність пам'яті або місця на диску може виявити дефекти у випробуванні, які не є очевидними за нормальних умов. Інші дефекти можуть виникнути унаслідок конкуренції за спільні ресурси, такі як бази даних чи пропускна здатність мережі. Стресове тестування може бути використано для виявлення максимального робочого навантаження.

Мета випробування (Test Objective)	Тестування за таких умов стресу: - мала або взагалі відсутня пам'ять, що доступна на сервері (RAM і DASD); - фактична або фізична максимальна кількість клієнтів, підключених або змодельованих; - велика кілька користувачів, що виконують однакові операції, або одночасні транзакції
Технічний прийом (Technique)	Використовуйте тести, розроблені для профілювання продуктивності або навантажувального тестування. Для перевірки обмеженості ресурсів, тести мають бути запущені на одній машині, і обсяг оперативної пам'яті і DASD на сервері має бути зменшений або обмежений. Для інших стрес-тестів, необхідно достатньо користувачів
Критерії завершення (Completion Criteria)	Усі заплановані тести виконуються у зазначених межах системи
Спеціальні рекомендації (Special Considerations)	Стрес мережі може вимагати мережевих інструментів для завантаження мережі з повідомленнями або пакетів. DASD, використовувані в системі, мають бути тимчасово зменшено або обмежений вільний простір для БД. Синхронізація одночасних клієнтів, які звертаються до такого самого запису або до даних розрахунків

3.1.8 Навантажувальне тестування (Volume Testing)

Навантажувальне тестування суб'єктів проводиться для великих обсягів даних, для визначення меж, які викликають провал програмного забезпечення. Навантажувальне тестування також визначає безперервне максимальне навантаження в зазначений період. Наприклад, необхідно протестувати обробку набору записів БД для створення звіту. Навантажувальне тестування використовуватиме велику БД випробувань для впевненості, що програмне забезпечення поводилося нормально і створено правильний звіт.

Мета випробування (Test Objective)	тест успішно працює в таких сценаріях: - максимальна (фактична або фізична) кількість клієнтів, підключених, або змодельованих, виконують бізнес-функції протягом тривалого часу; - максимальний розмір бази даних було досягнуто (фактичний або збільшений) і одночасно виконуються декілька запитів або транзакцій операцій
Технічний прийом (Technique)	Використовуйте тести, розроблені для профілювання продуктивності або навантажувального тестування
Критерії завершення (Completion Criteria)	Усі заплановані тести виконуються і зазначені межі системи
Спеціальні рекомендації (Special Considerations)	Який період часу необхідно розглядати для тестування

3.1.9 Тестування безпеки і контролю доступу (Security and Access Control Testing)

Тестування безпеки і контролю доступу має зосередити увагу на:

- застосуванні рівня безпеки, у тому числі доступ до даних або бізнес-функцій,
- рівнях системи безпеки, в тому числі за умови віддаленого доступу до системи.

Застосування рівня безпеки гарантує, що, ґрунтуючись на бажаному рівні безпеки, актори в нотації UML обмежені специфічними функціями або використаннями, або обмежені

в даних, які їм необхідні. Наприклад, кожен може вводити дані і створювати нові облікові записи, але тільки менеджери можуть видаляти їх. Якщо є безпека на рівні даних, тестування гарантує, що «користувач 1» може побачити всю інформацію про клієнта, у тому числі фінансові дані, проте, «користувач 2» бачить лише демографічні дані для одного клієнта.

Система безпеки на рівні гарантує, що тільки ті користувачі отримують доступ до системи для забезпечення доступу до додатків і тільки через відповідні шлюзи.

Мета випробування (Test Objective)	Застосування рівня безпеки. Переконайтеся, що актор може отримати доступ тільки до тих функцій або даних, до яких їх тип користувача має дозвіл. Система безпеки на рівні: переконайтеся, що тільки ті суб'єкти, що мають доступ до системи і додатків мають доступ до них
Технічний прийом (Technique)	Застосування рівня безпеки: Визначення та перелік кожного типу користувачів з відповідним дозволом до відповідних функцій. Створювати тести для кожного типу користувача і перевірки кожного рішення зі створенням специфічної операції для кожного типу користувача. Для тих же користувачів змінити тип користувача для повторних випробувань. У кожному разі необхідно перевірити ці додаткові функції або дані, які доступні або заборонені
Критерії завершення (Completion Criteria)	Запустити у попередній заявці функціональний тест для кожного відомого типу акторів, відповідної функції або даних. І порівняти результати з очікуваними
Спеціальні рекомендації (Special Considerations)	Доступ до системи має бути переглянутим або обговорюватися з відповідними системними адміністраторами мережі. Це тестування не може вимагатися як функція адміністрації мережі або системи

3.1.10 Тестування відмовостійкості та відновлення (Failover and Recovery Testing)

Цей вид тестування гарантує працездатність ПЗ, мережі або збереження даних.

Для тих систем, що працюють відмовостійкість означає збереження даних, створення резерву без втрати даних або операцій

Тестування відновлення передбачає випробування, у яких програма або система підлагає впливу екстремальних умов або штучно створених умов, причин відмови, наприклад, пристрої введення/виведення, збої або недійсні покажчики бази та ключі.

Мета випробування (Test Objective)	Переконайтеся, що процеси відновлення (ручний або автоматичний) правильно відновлюють БД, додатки і систему. Такі типи умов мають бути включені в тестування: - відключення живлення клієнта; - відключення живлення сервера; - переривання зв'язку через мережу серверів; - перерви зв'язку, а також утрата потужності на DASD або контролерах; - неповні цикли (дані фільтра процесів перериваються, дані процесу синхронізації перервана); - неприпустимий покажчик БД або ключів; - недійсний або пошкоджений елемент даних у БД

Технічний прийом (Technique)	Тести створюються для функцій і бізнес-циклів. Як тільки необхідна початкова тестова точка досягнута, такі дії повинні бути виконані, або змодельовані, індивідуально: - відключення живлення клієнта: потужність ПК зменшується; - відключення живлення сервера: імітувати або ініціювати виключення процедур сервера; - переривання через мережу серверів: імітувати або ініціювати втрати зв'язку з мережею (фізично відключити проводи зв'язку або відключення живлення мережі серверів або маршрутизаторів. Переривання зв'язку, або втрата потужності на DASD і DASD контролерах: імітувати або фізично ліквідувати зв'язок з одним або кількома контролерами DASD або пристроїв. Після цих штучно створених умов досягаються додаткові операції. Тестування вимагає, щоб деякі поля БД, покажчики і ключі були пошкоджені вручну і безпосередньо в БД (за допомогою інструментів для БД). Додаткові операції тестуються із застосуванням функцій і бізнес-циклів тестування і повних циклів
Критерії завершення (Completion Criteria)	Усі випадки тестування, що були зроблені вище, містять додатки, БД і системи, які мають по завершенні процедури відновлення, повернення до відомого стану, крім того містять дані, пошкоджені поля, покажчики або ключі, а також повідомлення про те, процеси або операції, які не були завершені через перерву
Спеціальні рекомендації (Special Considerations)	Процедура відключення кабелів не бажана. Може знадобитися альтернативний метод – діагностичний інструмент програмного забезпечення. Ресурси системи (чи комп'ютерних операцій), бази даних і мережі групи не потрібні. Ці тести мають бути запущені після години роботи або на ізольованій машині

3.1.11 Тестування конфігурації (Configuration Testing)

Конфігурація тестування перевіряє через тест роботу ПЗ на різних програмних та апаратних конфігураціях. У більшості середовищ виробництва, особливо апаратні специфікації для клієнтських робочих станцій, мережні з'єднання і сервера БД змінюються. Клієнтські робочі станції можуть мати різне ПЗ (додатки, драйвери тощо). ПЗ завантажується у будь-який момент часу і може активно використовувати різну комбінацію ресурсів).

Мета випробування (Test Objective)	Тестування проводиться належним чином на необхідному апаратному та програмному забезпеченні
Технічний прийом (Technique)	Використовувати функцію тестових сценаріїв. Відкриття та закриття різного відповідного ПЗ, що використовується як частина тестування або до початку випробування. Виконання зазначеної операції для імітації взаємодії актора з цільовим ПЗ. Повторення процесу за мінімальної конфігурації апаратного і ПЗ. Для кожної комбінації виконання тестів, усі операції мають бути успішно завершені без збоїв
Критерії завершення (Completion Criteria)	Для кожної комбінації цільових і нецільових тестів, усі операції успішно завершені без збоїв
Спеціальні рекомендації (Special Considerations)	Які нецільові програмні засоби, доступні та є на робочому столі? Які програми зазвичай використовуються? У яких програмах, що працюють, відкрито великі таблиці, наприклад Excel або 100-сторінковий документ Word? Необхідно задокументувати в межах цього тесту системи, NetWare, мережні сервери, бази даних тощо

3.1.12 Тестування інсталяції (Installation Testing)

Тестування інсталяції має дві мети.
1. ПЗ (нова установка, оновлення, або вибіркової установка) може бути встановлено в різних умовах за нормальних і ненормальних умов. Аномальні умови включають в себе недостатньо

місця на диску, відсутність привілеїв для створення каталогів тощо.
2. Перевірка після установки того, що ПЗ працює правильно. Це зазвичай означає функціональне тестування.

Мета випробування (Test Objective)	Правильна установка ПЗ на кожне машину з необхідною конфігурацією за дотриманням таких умов: - нові установки, нові машини, ніколи не встановлювалося ПЗ з означеного проекту; - оновлення, на ПК раніше встановлено ПЗ такої ж версії; - оновлення, на ПК раніше встановлено стару версію ПЗ
Технічний прийом (Technique)	Ручна установка або автоматична з набором скриптів. Для встановлення ПЗ – перевірка стану цільової машині і наявності ПЗ і ПЗ, що ставиться. Було воно встановлено, чи ні, якщо ПЗ встановлено було, то необхідно дізнатися, якої версії. Використання зумовило суб-набір скриптів функціонального тесту
Критерії завершення (Completion Criteria)	ПЗ виконано успішно без збоїв
Спеціальні рекомендації (Special Considerations)	Операції мають бути обрані для впевненості того, що тест був успішно виконаний і є достатня кількість основних компонентів ПЗ

3.2 Інструменти (Tools)

Такі інструменти будуть використані для цього проекту:
Примітка. Видаляти або додавати елементи таблиці можна за необхідністю

	Інструмент (Tool)	Постачальник (Vendor/In-house)	Версія (Version)
Управління тестами (Test Management)			
Відслідковування дефектів (Defect Tracking)			
ASQ інструмент для функціонального тестування (ASQ Tool for functional testing)			
ASQ інструмент для тестування продуктивності (ASQ Tool for performance testing)			
Тестовий монітор покриття або Profiler (Test Coverage Monitor or Profiler)			
Управління проектами (Project Management)			
СУБД інструменти (DBMS tools)			

4 Ресурси (Resources)

У цьому розділі подані ресурси, що рекомендуються для виконання проекту, їх основні обов'язки, знання і вміння.

4.1 Ролі (Roles)

Ця таблиця показує, кадрові забезпечення для проекту.
Примітка. Видаляти або додавати елементи таблиці можна за необхідності.

Людський ресурс (Human Resources)		
Працівник (Worker)	Рекомендований мінімальний обсяг осіб (повний робочий день). (Minimum Resources Recommended (number of full-time roles allocated))	Конкретні обов'язки або Коментарі (Specific Responsibilities or Comments)

Тест-менеджер, менеджер з тестування (Test Manager, Test Project Manager)		Забезпечує управління наглядом. Обов'язки: - технічна підтримка; - придбання відповідних ресурсів; - забезпечення управлінської звітності
Конструктор тестів (Test Designer)		Визначення пріоритетів і реалізація тестів. Обов'язки: - створення плану тестування; - генерація тестових моделей; - оцінювання ефективності тестових зусиль.
Тестувальник (Tester)		Виконання тестів. Обов'язки: - виконання тестів; - журнал результатів; - відновлення в журналі реєстрації після помилок, - документ зміни.
Тестовий системний адміністратор (Test System Administrator)		Забезпечує тестове середовище і управління активами. Обов'язки: - адміністрування тестової системи управління; - установлення і управління доступом до тест-системи.
Адміністратор бази даних (Database Administrator, Database Manager)		Забезпечує управління і підтримку тестових даних (бази даних), навколишнє середовище та активи Обов'язки: - адміністрування тестових даних (БД)
Дизайнер (Designer)		Виявляє і визначає операції, атрибути та асоціації тестів. Обов'язки: - виявляє і визначає тестові класи; - ідентифікує і визначає тестові пакети
Виконавець (Implementer)		Реалізує модульні тести і тестові класи Обов'язки: - створює тестові класи і пакети; - виконує тестові моделі

4.2 Система (System)

Нижче в таблиці подані системні ресурси для тестування проекту.

Рекомендується, щоб системи моделювалися з урахуванням виробничого середовища, обмеженого доступу та розмірів баз даних, якщо це доцільно у відповідних випадках.

Примітка. Видаляти або додавати елементи таблиці можна за необхідності.

Системні ресурси (System Resources)	
Ресурси (Resource)	Назва (Name) / Тип (Type)
Сервер БД (Database Server)	
- мережі та підмережі (Network or Subnet)	підлягає подальшому уточненню (TBD)
- ім'я сервера (Server Name)	підлягає подальшому уточненню (TBD)
- ім'я БД (Database Name)	підлягає подальшому уточненню (TBD)
Клієнт випробувань ПК (Client Test PC's)	
- містить спеціальні вимоги до конфігурації (Include special configuration requirements)	підлягає подальшому уточненню (TBD)
Тестування репозиторія (Test Repository)	
- мережі та підмережі Network or Subnet	підлягає подальшому уточненню (TBD)
- ім'я серверу Server Name	підлягає подальшому уточненню (TBD)
Тест розвитку ПК (Test Development PC's)	підлягає подальшому уточненню (TBD)

5 Етапи проекту (Project Milestones)

Тестування відповідного проекту має містити цільове випробування.

Цільове завдання (Milestone Task)	Обсяг робіт (Effort)	Дата початку (Start Date)	Дата закінчення (End Date)
План випробувань (Plan Test)			
Тест-дизайн (Design Test)			
Реалізація випробувань (Implement Test)			
Виконання тесту (Execute Test)			
Оцінювання випробувань (Evaluate Test)			

6 Кінцевий продукт (Deliverables)

У цьому розділі подано список різних документів, інструментів і звітів, які будуть створені, представлені та доставлені.

6.1 Тестова модель (Test Model)

У цьому розділі визначаються звіти, які будуть створені та поширені з тестування моделі. Ці артефакти у тестовій моделі мають бути створені або вказують посилання на інструменти ASQ (American Society Quality – Американська спільнота якості).

6.2 Тестовий журнал (Test Logs)

Описують методи та інструменти, що використовувалися для запису і звітів тестових результатів і статусу тестування.

6.3 Звіти з дефектів (Defect Reports)

У цьому розділі визначають методи та інструменти, що використовують для запису, відстежування і повідомлення про тестові інциденти та їх статус.

7 Додаток А Завдання проекту (Appendix A Project Tasks)

Нижче наведено завдання тесту:

1. План тестування містить:
- визначення вимог тестування,
- оцінювання ризику,
- розробку стратегії тестування,
- визначення тест-ресурсів,
- створення графіка,
- створення плану тестування.

2. Дизайн випробувань передбачає:
- підготовку аналізу робочого навантаження,
- визначення та опис тестів,
- визначення та структуру тестових процедур,
- огляд та оцінювання тестового покриття.

3. Упровадження випробувань передбачає:
- запис або сценарії програмних випробувань,
- визначення тестових функцій у розробці та запровадження моделі,
- установлення зовнішніх наборів даних.

4. Виконання тесту:
- виконання процедури випробувань,
- оцінювання виконання випробувань,
- перевірка результатів,
- дослідження несподіваних результатів,
- журнал дефектів.

5. Оцінювання випробувань:
- оцінювання випробувань разі покриття,
- оцінювання покриття коду,
- аналіз дефектів,
- визначення критеріїв завершення випробувань і критеріїв успіху, що були досягнуті.

Методичні вказівки щодо виконання самостійної роботи з навчальної дисципліни «Основи аналізу якості програмного забезпечення» для студентів денної форми навчання зі спеціальності 123 – «Комп'ютерна інженерія»

Укладач к. т. н., доц. О. Г. Славко

Відповідальний за випуск зав. кафедри КІС М. І. Гученко

Підп. до др. _____. Формат 60×84 1/16. Папір тип. Друк ризографія.
Ум. друк. арк. ____. Наклад _____ прим. Зам. №_____. Безкоштовно.

Редакційно-видавничий відділ
Кременчуцького національного університету
імені Михайла Остроградського
вул. Першотравнева, 20, м. Кременчук, 39600