МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра програмних систем і технологій

Дисципліна

«ЯКІСТЬ ПРОГРАМНОГО ЗАБЕСПЕЧЕННЯ ТА ТЕСТУВАННЯ**»**

Лабораторна робота № 3

«Розробка та оформлення тестового плану»

Виконав:	Гоша Давід	**Перевірив**:	
Група	ІПЗ-33	Дата перевірки	
Форма навчання	денна	Оцінка	
Спеціальність	121		
2022			

Завдання:

1. Створити «Тестовий план»
2. Заповнити всі розділи шаблону тестового плану конкретною інформацією по продукту, що тестується згідно заданої специфікації.

Зміст:

Виконання:

1. Введення

1.1. **Мета документа:**

Мета плану тестування - визначити систематичний підхід до тестування продукту, системи або компонента, щоб переконатися, що він відповідає встановленим вимогам і стандартам якості. План тестування окреслює стратегію тестування, необхідні ресурси, графік та очікувані результати. Основними завданнями плану тестування є:

1.1.1. Визначити обсяг тестування: План тестування окреслює, що буде тестуватися, а що ні.

1.1.2. Визначити підхід до тестування: План тестування визначає методи тестування, які будуть використовуватися, наприклад, ручне тестування, автоматизоване тестування або їх поєднання.

1.1.3. Розподіл ресурсів: План тестування визначає ресурси, необхідні для тестування, включаючи людей, обладнання та програмне забезпечення.

1.1.4. Оцінити графік: План тестування містить графік проведення

тестування, включаючи дати початку і закінчення тестування, а також очікувану тривалість кожного етапу тестування.

1.1.5. Переконатися, що всі зацікавлені сторони знають про підхід до тестування: План тестування допомагає переконатися, що всі учасники проекту, включаючи команду розробників, зацікавлені сторони та керівництво, знають про підхід до тестування та очікування від нього.

1.1.6. Визначити потенційні ризики та стратегії їх зменшення: План тестування окреслює потенційні ризики та надає стратегії для їх зменшення.

1.1.7. Документування та відстеження результатів тестування: План тестування забезпечує спосіб документування та відстеження результатів тестування, що полегшує визначення областей, які потребують вдосконалення, та вимірювання успішності процесу тестування.

Таким чином, мета плану тестування - забезпечити комплексний і добре організований підхід до тестування, який гарантує, що продукт, система або компонент відповідає встановленим вимогам і стандартам якості.

1.2.Рамки документа:

a. Вступ:
   i. Мета плану тестування
   ii. Огляд системи, що тестується (SUT)
   iii. Цілі тестування
   iv. Обсяг тестування
   v. Припущення та обмеження
b. Підхід до тестування:
   i. Огляд підходу до тестування
   ii. Типи тестування, які необхідно виконати
   iii. Тестове середовище та конфігурація
   iv. Інструменти та методи тестування
c. Тестові кейси:
   i. Огляд тестових кейсів
   ii. Детальні тестові кейси та процедури тестування
   iii. Тестові дані та вхідні дані
   iv. Тестові скрипти (для автоматизованого тестування)
d. Розклад тестування:
   i. Огляд розкладу тестування
   ii. Графік тестування
   iii. Етапи та результати
e. Ресурси для тестування:

i. Огляд тестових ресурсів
ii. Вимоги до персоналу
iii. Вимоги до апаратного та програмного забезпечення

f. Ризики та їх мінімізація:
    i. Огляд ризиків та їх мінімізація
    ii. Ідентифікація ризиків
    iii. Стратегії пом'якшення наслідків
g. Результати тестування:
    i. Огляд результатів тестування
    ii. Метрики тесту
    iii. Статус та відстеження тесту
    iv. Журнали тестування та звіти
h. Висновок
    i. Зведення результатів тестування
    ii. Остаточні рекомендації
    iii. Майбутня робота
i. Додатки:
    i. Глосарій
    ii. Сторінка затвердження плану випробувань та підписів
    iii. Довідкові документи

Ця структура надає вичерпну схему для створення плану тестування, але вона може бути модифікована відповідно до конкретних потреб проекту. Головне - переконатися, що вся необхідна інформація включена до Плану тестування, а також, що він організований у чіткий і стислий спосіб.

2. Тестування вимог

1. Арифметичні дії:
    a. Протестувати операції додавання, віднімання, множення, ділення та знаходження остачі.
    b. Переконайтеся, що операції виконуються відповідно до специфікацій, зазначених у п. 3.2.
2. Цілочисельний діапазон:
    a. Переконайтеся, що калькулятор підтримує цілі числа в діапазоні від MININT до MAXINT.
    b. Протестуйте калькулятор на цілих числах, що виходять за межі вказаного діапазону, і переконайтеся, що він не видає помилку 06.
3. Пам'ять:
    a. Переконайтеся, що калькулятор має пам'ять на одне ціле число і може відображати, скидати та додавати до неї числа.
    b. Протестуйте клавішу M+ і перевірте, чи додається число з пам'яті до числа в полі Результат.

c. Протестуйте калькулятор, коли поле Результат містить код помилки, і перевірте, чи виводить він відповідне повідомлення.
d. Переконайтеся, що кнопка MC обнуляє число в пам'яті.
e. Переконайтеся, що кнопка MR додає число з пам'яті в кінець виразу.

4. Унарний плюс/мінус:
    a. Переконайтеся, що калькулятор дозволяє користувачеві працювати з унарними операціями "плюс" та "мінус".
    b. Протестуйте калькулятор на наявність перемикача унарного плюса/мінуса і перевірте, чи відбувається реверсування або додавання введеного оператора до виразу.
5. Графічний інтерфейс:
    a. Переконайтеся, що калькулятор має графічний інтерфейс з кнопками для чисел та арифметичних операцій, пам'яті, редагування дужок, скидання, унарного перемикача плюс/мінус та текстовими полями для введення виразів і відображення результатів.
    b. Переконайтеся, що натискання клавіші Enter призводить до обчислення виразу.
    c. Переконайтеся, що натискання клавіші ESC зупиняє програму.
6. Перевірка правильності виразу:
    a. Переконайтеся, що калькулятор перевіряє правильність виразу і виводить відповідне повідомлення, якщо вираз неправильний.
7. Повідомлення про помилки:
    a. Переконайтеся, що калькулятор виводить відповідні повідомлення про помилки, як зазначено у пункті 2.2.3.

3. Стратегія тестування

3.1. Типи тестування

3.1.1. Тестування функціональності

Функціональне тестування арифметичних операцій є важливим аспектом тестування програмного забезпечення, особливо для калькулятора. Метою цього тестування є перевірка того, що програма виконує арифметичні операції правильно і повертає очікувані результати.

a. Нижче наведено ключові області, які необхідно перевірити для кожної арифметичної операції:
b. Додавання: Операція додавання повинна повертати правильну суму для чисел, кожне з яких менше або дорівнює MAXINT і більше або дорівнює MININT. Якщо сума більша за MAXINT або менша за MININT, програма повинна згенерувати помилку Error 06.
c. Віднімання: Операція віднімання повинна повертати правильну різницю для чисел, кожне з яких менше або дорівнює MAXINT і більше або дорівнює MININT. Якщо різниця більша за MAXINT або менша за MININT, програма повинна згенерувати помилку Error 06.

d. Множення: Операція множення повинна повертати правильний добуток для чисел, добуток яких менше або дорівнює MAXINT і більше або дорівнює MININT. Якщо добуток більший за MAXINT або менший за MININT, програма повинна видати помилку Error 06.
e. Ділення: Операція ділення повинна повертати правильну частку для чисел, які менші або дорівнюють MAXINT і більші або дорівнюють MININT, а дільник не дорівнює 0. Якщо частка більша за MAXINT або менша за MININT, то програма повинна згенерувати помилку Error 06. Якщо дільник дорівнює 0, програма повинна згенерувати помилку Error 09.
f. Ділення з остачею: Операція ділення з остачею повинна повертати правильну остачу для чисел, які менші або дорівнюють MAXINT і більші або дорівнюють MININT, а дільник не дорівнює 0. Якщо остача більша за MAXINT або менша за MININT, то програма повинна згенерувати помилку Error 06. Якщо дільник дорівнює 0, програма повинна згенерувати помилку Error 09.
g. Унарний плюс/мінус: Унарна операція плюс/мінус повинна повертати число відповідного знаку для чисел, які менші або дорівнюють MAXINT і більші або дорівнюють MININT. Якщо число більше MAXINT або менше MININT, програма повинна згенерувати помилку Error 06.
h. Ці тести слід виконати, щоб переконатися, що програма-калькулятор виконує всі арифметичні операції правильно і повертає очікувані результати.

3.1.2. Тестування функціональності: Клавіші інтерфейсу користувача

a. Тестування клавіш "1", "2", "3", "4", "5", "6", "7", "8", "9", "0"
    i. Сценарій тестування: Тестувальник вводить послідовність чисел у поле виразу, використовуючи клавіші "1", "2", "3", "4", "5", "6", "7", "8", "9", "0".
    ii. Очікуваний результат: Введені числа повинні виводитись у полі виразу у тому порядку, у якому вони були введені.
b. Тестування клавіш "/", "*", "-", "+", "mod"
    i. Сценарій тестування: Тестувальник вводить вираз з арифметичними операціями в поле виразу з використанням клавіш "/", "*", "-", "+", "mod".
    ii. Очікуваний результат: Введені арифметичні операції повинні бути виведені у полі виразу у тому порядку, в якому вони були введені.
c. Тестування клавіш "(" та ")"
    i. Сценарій тестування: Тестувальник вводить вираз з круглими дужками в поле виразу з допомогою клавіш "(" і ")".

ii. Очікуваний результат: Введені дужки повинні бути відображені в полі виразу в тому порядку, в якому вони були введені.

d. Тестування клавіші "Reset"
   i. Сценарій тестування: Тестувальник вводить вираз у поле виразу і натискає клавішу "Reset".
   ii. Очікуваний результат: Поле виразу має бути очищене, і вираз більше не повинен відображатися.

e. Тестування клавіші "Стерети"
   i. Сценарій тестування: Тестувальник вводить вираз у поле виразу, а потім натискає клавішу "Стерти", щоб видалити останній введений символ.
   ii. Очікуваний результат: Останній введений символ має бути видалено, а в полі виразу має відобразитися оновлений вираз.

f. Тестування клавіші "="
   i. Сценарій тестування: Тестувальник вводить вираз у поле виразу і натискає клавішу "=", щоб почати обчислення.
   ii. Очікуваний результат: Калькулятор повинен обчислити вираз і вивести результат у текстовому вікні.

g. Тестування клавіш "MR", "M+" та "MC"
   i. Сценарій тестування: Тестувальник вводить число в пам'ять за допомогою клавіші "M+", потім отримує число з пам'яті за допомогою клавіші "MR" і додає його до виразу в полі виразу. Потім тестувальник обнуляє пам'ять за допомогою клавіші "MC".
   ii. Очікуваний результат: Калькулятор повинен коректно додати число з пам'яті до виразу у полі виразу, а потім обнулити пам'ять.

h. Тестування клавіші "+/-"
   i. Сценарій тестування: Тестувальник вводить число в поле виразу, а потім натискає клавішу "+/-" для перемикання між унарним плюсом і унарним мінусом.
   ii. Очікуваний результат: Калькулятор повинен коректно змінити знак введеного числа у полі виразу.

3.1.3. Стрес -тестування

Стрес тестування - це тип тестування, який має на меті оцінити продуктивнсть системи або додатку в умовах великих і тривалих робочих навантажень. Воно проводиться для визначення меж системи, точок зламу та потенційних вузьких місць. Метою стрес-тестування є перевірка стабільності та надійності системи в умовах, наближених до реальних сценаріїв.

Для стрес-тестування калькулятора можна виконати наступні кроки:

a. Навантажувальне тестування: У цьому тесті система піддається зростаючому навантаженню, щоб спостерігати за її продуктивністю і стабільністю. Тест спрямований на оцінку здатності системи обробляти велику кількість одночасних користувачів, великі обсяги даних і складні обчислення.
b. Тестування продуктивності: Продуктивність калькулятора оцінюється шляхом вимірювання часу відгуку, швидкості обробки та використання ресурсів за різних умов навантаження. Цей тест допомагає виявити будь-яке погіршення продуктивності, яке може статися з часом.
c. Стрес-тестування: Систему піддають екстремальним умовам, щоб виявити її слабкі місця. Тест передбачає перевантаження системи великим обсягом запитів і даних, що призводить до деградації програми. Цей тест проводиться для визначення стійкості системи в несприятливих умовах.
d. Тестування масштабованості: У цьому тесті оцінюється здатність системи справлятися зі збільшенням навантаження та складності. Тест передбачає поступове збільшення навантаження на систему та вимірювання її продуктивності, щоб визначити, наскільки добре система може масштабуватися.
e. Тестування надійності: У цьому тесті оцінюється здатність системи працювати стабільно і без помилок при великому і тривалому робочому навантаженні. Тест проводиться для визначення надійності та стабільності системи в реальних умовах.
f. За допомогою стрес-тестування можна оцінити та покращити продуктивність, стабільність та надійність калькулятора. Це призведе до створення більш надійного та надійного калькулятора, який зможе виконувати складні обчислення з легкістю і точністю.

3.1.4. Тестування установки

Тестування інсталяції - це тип тестування програмного забезпечення, який має на меті переконатися, що програмний продукт правильно встановлений на цільовій системі і відповідає заданим вимогам. Наступні кроки описують процес тестування встановлення програми-калькулятора:

a. Підготовка: Переконайтеся, що цільова система відповідає мінімальним вимогам до апаратного та програмного забезпечення для програми-калькулятора. Також переконайтеся, що інсталяційний пакет і всі необхідні ліцензії доступні.

b. Встановлення програми: Виконайте процес інсталяції відповідно до інструкцій, наведених у посібнику з інсталяції. Переконайтеся, що інсталятор створив необхідні файли, каталоги та записи в реєстрі, а також встановив усі необхідні залежності.

c. Перевірка встановлення: Перевірте правильність встановлення програми, запустивши калькулятор і переконавшись, що він запускається без помилок. Також перевірте, чи створені всі необхідні ярлики та асоціації файлів.
d. Тестування функціональності: Переконайтеся, що калькулятор виконує всі функції, описані в технічному завданні. Протестуйте всі кнопки та елементи керування, виконайте прості та складні обчислення і переконайтеся, що калькулятор повертає правильні результати.
e. Тестування користувацького інтерфейсу: Протестуйте користувацький інтерфейс, щоб переконатися, що він зручний і простий у навігації. Переконайтеся, що всі кнопки та елементи керування працюють належним чином і що інтерфейс забезпечує чіткий і точний зворотний зв'язок з користувачем.
f. Видалення програми: Протестуйте процес видалення, видаливши калькулятор з цільової системи. Переконайтеся, що деінсталятор видаляє всі файли, каталоги та записи реєстру, які були створені під час інсталяції.
g. Фінальна перевірка: Переконайтеся, що цільова система перебуває у тому ж стані, що і до інсталяції, і що у ній не залишилося жодних файлів або записів у реєстрі.

3.2. Інструменти

Інструменти - це програмні додатки або утиліти, які використовуються для підтримки розробки, тестування та супроводу програмних проектів. Вони використовуються для автоматизації різних завдань і роблять процес розробки більш ефективним, точним і економічно вигідним.

a. Деякі з найпоширеніших інструментів для тестування програмного забезпечення включають
b. Інструменти управління тестуванням: Ці інструменти використовуються для управління тестовими кейсами, відстеження помилок і планування діяльності з тестування. Приклади включають JIRA, TestRail та HP Quality Center.
c. Інструменти автоматизації тестування: Ці інструменти використовуються для автоматизації повторюваних завдань тестування, зменшення втручання людини і підвищення ефективності тестування. Приклади включають Selenium, Appium і TestComplete.
d. Інструменти тестування продуктивності: Ці інструменти використовуються для оцінки продуктивності програмних додатків за різних умов навантаження. Приклади включають Apache JMeter, LoadRunner і Gatling.
e. Інструменти тестування безпеки: Ці інструменти використовуються

для виявлення вразливостей у програмних додатках і запобігання порушенням безпеки. Приклади включають OWASP ZAP, Nessus і Qualys.

f. Інструменти налагодження: Ці інструменти використовуються для виявлення та виправлення помилок у програмному коді. Приклади включають GDB, Visual Studio Debugger та WinDbg.
g. Інструменти якості коду: Ці інструменти використовуються для покращення якості програмного коду, аналізуючи його на наявність синтаксичних помилок, порушень стандартів кодування та проблем з продуктивністю. Приклади включають SonarQube, CodeClimate і RuboCop.
h. На завершення, існують різні інструменти для підтримки процесу розробки та тестування програмного забезпечення. Вибір інструментів залежить від конкретних вимог проекту та цілей, які намагаються досягти.

4. Ресурси

4.1. Співробітники

Для тестування працівників у проекті можна виконати наступні кроки:

a. Створіть план тестування: Визначте обсяг тестування працівників та цілі, яких потрібно досягти.
b. Визначте ролі співробітників: Визначте різні ролі співробітників, які необхідно протестувати, наприклад, менеджерів, керівників і працівників, які працюють на лінії.
c. Підготуйте дані для тестування: Підготуйте тестові дані, які будуть використовуватися в процесі тестування, включаючи імена, адреси та контактну інформацію співробітників.
d. Протестуйте процес реєстрації працівників: Протестуйте процес додавання нових співробітників до системи, включаючи введення їхньої особистої інформації, призначення ролі та налаштування їхніх облікових даних для входу в систему.
e. Протестуйте процес входу працівника в систему: Протестуйте процес входу в систему як працівник, включаючи введення правильного імені користувача та пароля, а також перевірку правильності відображення інформації.
f. Протестуйте процес оновлення профілю працівника: Протестуйте процес оновлення особистої інформації працівника, наприклад, його контактної інформації або ролі, і перевірте, чи відображаються зміни в системі.
g. Протестуйте процес пошуку співробітників: Протестуйте процес пошуку працівника в системі, включаючи пошук за іменем, посадою або іншими критеріями.
h. Протестуйте процес формування звітів про співробітників:

Протестуйте процес формування звітів про співробітників, включаючи їхню історію роботи, результати діяльності та іншу важливу інформацію.

i. Протестуйте процес оцінки ефективності роботи співробітників: Протестуйте процес оцінювання результатів роботи працівника, включаючи встановлення цілей, відстеження прогресу та надання зворотного зв'язку.
j. Протестуйте процес навчання співробітників: Протестуйте процес навчання співробітників, включаючи планування навчальних сесій, відстеження відвідуваності та оцінку ефективності навчання.
k. Оцініть результати: Проаналізуйте результати тестування співробітників і внесіть необхідні зміни в систему.

4.2. Системи

План системного тестування проекту зазвичай включає наступні кроки:

a. Визначення обсягу: Визначте системи та компоненти, які необхідно протестувати, цілі тестування та очікувані результати.
b. Визначення тестового середовища: Визначте вимоги до апаратного та програмного забезпечення, тестові дані та будь-які інші ресурси, необхідні для тестування.
c. Сплануйте тестування: Визначте підхід до тестування, графік і необхідні ресурси, включаючи тестові кейси, тестові скрипти і тестові дані.
d. Виконайте тести: Виконайте тести відповідно до плану, записуючи результати і документуючи будь-які виявлені проблеми.
e. Оцініть результати: Проаналізуйте результати тестів, щоб визначити, чи відповідають системи вимогам і чи потрібно вирішити якісь проблеми.
f. Повідомляйте про проблеми та відстежуйте їх: Документуйте будь-які проблеми, виявлені під час тестування, визначайте їх пріоритетність і призначайте відповідні команди для вирішення.
g. Повторне тестування: Повторне тестування систем після вирішення проблем, щоб підтвердити, що вони були виправлені і що системи продовжують відповідати вимогам.
h. Підписати: Отримання схвалення від зацікавлених сторін, що свідчить про те, що системи були протестовані і готові до випуску.
i. Примітка: Конкретні деталі плану тестування системи будуть відрізнятися залежно від проекту та систем, що тестуються.

Висновок:

Насамкінець, важливо ретельно протестувати будь-який проект, щоб переконатися, що він відповідає необхідним специфікаціям і функціонує належним чином. Процес тестування передбачає оцінку різних аспектів проекту, включаючи функціональність системи, користувацький інтерфейс і продуктивність за різних умов. План тестування повинен бути детальним і чітко описувати процедури тестування, очікувані результати та критерії успіху. Дотримуючись комплексного плану тестування, ми можемо виявити будь-які проблеми або слабкі місця в проекті та внести необхідні зміни, щоб забезпечити його високу якість і оптимальну роботу.