# Тест-план
для тестування програмного продукту
для побудови графіків

Кодова назва проекту: PLOTSCRIPT
Код документу: PLOTSCRIPT-00-TESTPLAN-000
Версія: 2017-12-06-000

# Зміст

# 1 Вступ

## 1.1 Призначення документа

PLOTSCRIPT-00-TESTPLAN-000 — мастер-план тестування, розроблений для визначення деталей проведення тестових робіт для проекту під назвою tinyplot. Даний план складений у відповідності до вимог та принципів, описаних у відповідному технічному завданні (PLOTSCRIPT-00-PRD-000).

Метою плану тестування є опис процесу тестування програми tinyplot. Розроблений план тестування визначає загальну стратегію тестування та особливості виконання процесів, пов'язаних з тестуванням.

## 1.2 Терміни

У тексті документу використовуються спеціалізовані терміни. Для запобігання їх неправильної інтерпретації наведені визначення (табл. 1).

| Термін | Значення |
|---|---|
| Проект | Проект PLOTSCRIPT. |
| Тестування | Процес, направлений на виявлення дефектів і помилок в програмному продукті шляхом пошуку невідповідностей між очікуваним і отриманим результатами. |
| Функціональне тестування | Тестування функцій програми на відповідність вимогам. |
| Смоук-тестування | Мінімальний набір тестів для виявлення явних помилок. Зазвичай виконується самим програмістом. Продовження тестування програми, що не пройшла такий тест, не має сенсу. |
| Тестування продуктивності | Тестування, яке проводиться для визначення швидкості роботи системи або її частини при заданому навантаженні. |
| Стрес-тестування | Оцінка надійності системи в умовах перевищення границі нормального функціонування. |
| Тестування навантаження | Збір показників, визначення продуктивності і часу відгуку програмно-технічної системи або пристрою у відповідь на зовнішній запит. |
| Об'ємне тестування | Оцінка продуктивності при збільшенні обсягів даних у базі даних програми. |

Табл. 1: Перелік термінів, використаних у даному документі

### 1.3 Мета тестування

Метою тестування Проекту є визначення ступеня його відповідності функціональним вимогам, перевірка правильності роботи на різних операційних системах, визначення швидкості роботи системи при навантаженні, а також виявлення недоліків і вразливостей Проекту.

## 2 Предмет тестування

Предметом тестування є програмний продукт tinyplot (версія `0.1.0dev0`), розроблений у рамках проекту з кодовою назвою «PLOTSCRIPT».

Оскільки процес тестування має на меті перевірку відповідності результату розробки вимогам, даний тест-план, а також процес тестування, що він описує, спирається на відповідні нормативні документи (табл. 2).

| Тип документу | Ідентифікатор |
|---|---|
| Стандарти кодування | `GUIDE-CODE-16b51262` |
| Стандарти розробки | `GUIDE-CONTRIB-1c10f94e` |
| Технічне завдання | `PLOTSCRIPT-00-PRD-000` |

Табл. 2: Перелік документів, на які спирається даний план тестування

Перед тим, як почати тестування, необхідно впевнитись, що у відповідного персоналу встановлена сучасна версія продукту. Сучасну версію продукту можна знайти за посиланням `https://gitlab.com/vklokun/plotscript`, яке містить необхідні інструкції для встановлення.

## 3 Функціонал, що тестується

Під час проведення тестування цільового програмного забезпечення необхідно протестувати такі процеси програмного продукту:

1. Розповсюдження.
2. Встановлення.
3. Робота.

### 3.1 Розповсюдження

Розповсюдження програми — це процес, що описує появу ресурсів, що необхідні для встановлення цільового програмного продукту, на апаратному забезпеченні кінцевого користувача.

### 3.2 Встановлення

Встановлення програми — це процес перетворення початкових (вихідних) ресурсів встановлення програми у робочий програмний продукт.

### 3.3 Робота

Робота — це процес безпосереднього виконання програмним продуктом функцій, закладених в нього. Згідно з PLOTSCRIPT-00-PRD-000, суть роботи програми полягає в перетворенні текстових даних, що описують графік, у візуальні.

## 4 Функціонал, що не тестується

### 4.1 Графічний інтерфейс

Графічний інтерфейс — це набір об'єктів, що інтерактивно взаємодіють з користувачем візуальним способом: за допомогою піктограм, рисунків, кнопок тощо.

Процес тестування не передбачує тестування графічного інтерфейсу користувача, оскільки відповідно до PLOTSCRIPT-00-PRD-000, графічний інтерфейс не є важливою складовою Проекту, а слугує лише додатковою функцією, що надається сторонньою бібліотекою.

## 5 Стратегія тестування

Для тестування Проекту визначається чітка стратегія, яка поєднує автоматичний та ручний підхід у тестуванні. Під час тестування використовуються сучасні технології, зокрема Continuous Integration (CI) та Continuous Delivery (CD).

Для використання технологій CI та CD використовуються можливості програмної платформи GitLab. Крім того, використовуються такі утиліти для автоматичного тестування:

— модуль `unittest` стандартної бібліотеки мови програмування Python.

— платформа Travis-CI.

Успішна реалізація розробленої стратегії виконання потребує відповідного забезпечення. Ці потреби описані у розділах 6 та 8.

### 5.1 Функціональне тестування

Мета функціонального тестування полягає у виявленні функціональних помилок, невідповідностей до технічного завдання і очікувань користувача шляхом реалізації стандартних, а також нетривіальних тестових сценаріїв.

Розроблена стратегія передбачає виконання функціонального тестування в контексті розробки та перевірки тестових випадків (англ. *test cases*). Цей процес організується за допомогою модуля `unittest`. Засобами цього модуля організують автоматичне тестування на розроблених тестових випадках та повідомлення про результати їх перевірки.

Запуск перевірок на тестових випадках здійснюється вручну тестувальниками, а також автоматично щоразу при внесенні змін до захищених гілок розробки (наприклад, `master`).

Основними компонентами, що підлягають функціональному тестуванню, є ті компоненти програмного продукту, що відповідають за зчитування та обробку даних вхідних файлів. Крім того, тестуванню підлягають ті частини продукту, які відповідають за встановлення та обробку параметрів, які впливають на результат обробки вхідних даних — параметри, що впливають на презентацію виведених даних.

## 5.2 Нефункціональне тестування

Нефункціональне тестування описує тести, що необхідні для визначення чітких числових значень тих характеристик програмного забезпечення, які можуть бути виміряні.

### 5.2.1 Тестування продуктивності

Тестування продуктивності — це різновид тестування, що визначає кількісні значення ресурсів, що використовує система, а також ефективність їх використання. Для виконання тестування продуктивності рекомендується використовувати спеціальні інструменти, що надаються відповідними середовищами. Наприклад, під управлінням UNIX-подібних систем слід використовувати утиліту `time`. При перевірці уніфікованих програмних компонентів варто надавати перевагу спеціальним вбудованим функціям, які зазвичай описуються у довідкових матеріалах до відповідних мов програмування.

Для вимірювання загальних метрик ефективності роботи програми використовується *загальне тестування продуктивності*. Його суть у контексті Проекту полягає у вимірі часу, витраченого на виконання таких основних операцій як завантаження вихідного файлу з даними, їх обробка та побудова графіка за наданими даними.

Для оцінки продуктивності при збільшенні обсягів даних у вхідних файлах програми використовується *об'ємне тестування*. При виконанні об'ємного тестування вимірюється час, витрачений на успішне виконання операції. Оскільки у технічному завданні (`PLOTSCRIPT-00-PRD-000`) встановлені вимоги до зміни витрат часу при збільшенні вхідних даних, такий метод є прекрасним способом перевірки відповідності поставленим вимогам.

### 5.2.2 Тестування встановлення

Тестування встановлення використовується для перевірки правильності, цілісності та завершеності встановлення програмного продукту.

У контексті Проекту тестування встановлення полягає в автоматичному виконанні спроб встановлення на різних тестових конфігураціях та звітуванні про результат. Для виконання такого тестування використовуються можливості, надані платформою GitLab, а саме використання так званих «Runners» — автоматичних засобів виконання встановлення та перевірки роботи.

## 5.3 Тестування змін

Тестування змін полягає у перевірці роботи системи після внесення змін до неї. Метою проведення такого тестування є забезпечення правильності та коректності змін, внесених в систему, а також визначення та попередження появ недоліків.

### 5.3.1 Димове тестування

Димове тестування (англ. *smoke testing*) — це короткий цикл тестів, що виконується для підтвердження мінімальної робочої здатності програми.

Мета димового тестування полягає у попередженні найбільш серйозних дефектів, що приводять систему у критичний стан та виводять з ладу навіть базовий функціонал.

Димове тестування у контексті даного Проекту полягає в запуску програми на спеціальному тестовому наборі даних. Тестування вважається успішним, якщо програма запускається без помилок та виводить будь-яке зображення без візуальних артефактів.

# 6 Вимоги до середовища

# 7 Обов'язки

# 8 Вимоги до людських ресурсів

# 9 Результати тестування

# 10 Розклад та етапи тестування