Самостійна робота №1

Тема:

Опрацювання базових понять програмної
інженерії та SDLC і порівняльної таблиці моделей Waterfall/Agile за критеріями застосування.

Виконав: Студент групи 2РП-11 Іванаш Дмитро Іванович

1. Опис проекту

Назва проєкту: Платформа для обміну ігровими предметами "SkinSwap"

Ідея проєкту: Створення безпечного вебмайданчика, де гравці популярних багатокористувацьких ігор (наприклад, CS, Dota 2, Roblox) можуть обмінюватися внутрішньоігровими предметами (скінами) без ризику бути ошуканими.
Мета проєкту: Мінімізувати випадки шахрайства під час трейдингу та надати користувачам зручний інтерфейс для оцінки вартості та перегляду предметів.

Основні користувачі:

1. Геймери: Люди, які хочуть оновити свій інвентар.
2. Колекціонери: Користувачі, що шукають рідкісні та дорогі предмети.
3. Модератори: Персонал, що вирішує спірні питання під час обмінів.

Ключова функціональність:

- Інтеграція з ігровими профілями (через API) для відображення інвентарю.
- Система "Безпечна угода" (Escrow), яка утримує предмети до підтвердження обома сторонами.
- Чат для узгодження умов обміну.
- Фільтри для пошуку предметів за ціною, рідкістю та типом.
- Рейтинг надійності користувачів.

2. Базові поняття програмної інженерії та етапи SDLC

Програмна інженерія — це **технологічна дисципліна**, що передбачає застосування системного підходу до розробки, експлуатації та підтримки програмного забезпечення.

SDLC (Життєвий цикл розробки) — це модель, що описує кроки від виникнення ідеї до випуску продукту.

Етапи розробки проєкту "SkinSwap":

1. **Аналіз вимог:** Вивчення API ігрових платформ, аналіз ринку конкурентів та визначення протоколів безпеки для запобігання крадіжкам.
2. **Проєктування:** Розробка архітектури сайту, створення макетів інтерфейсу (сторінка обміну, особистий кабінет) та проєктування бази даних користувачів.
3. **Розробка:** Написання серверної частини (наприклад, мовою **Python**) для обробки запитів на обмін та фронтенд-частини для візуального інтерфейсу.
4. **Тестування:** Перевірка системи на вразливості, симуляція спроб шахрайства, тестування швидкості завантаження інвентарів.
5. **Впровадження:** Запуск платформи на хостингу, залучення перших користувачів (бета-тестування) та запуск рекламної кампанії серед геймерів.
6. **Супровід:** Постійний моніторинг безпеки, оновлення бази предметів після виходу нових оновлень у іграх та виправлення помилок.

3. Порівняльна таблиця моделей розробки

Критерій порівняння	Waterfall (Каскадна модель)	Agile (Гнучка модель)
Гнучкість	Дуже низька; зміни практично неможливі після початку розробки.	Дуже висока; проєкт розбитий на короткі цикли.
Робота з вимогами	Вимоги жорстко зафіксовані в документації на початку.	Вимоги можуть додаватися або змінюватися в будь-який момент.
Залучення замовника	Тільки під час написання ТЗ та в кінці проєкту.	Замовник залучений постійно (перевірка результатів кожні 2 тижні).
Ризики	Високі: якщо в вимогах була помилка, її знайдуть лише в кінці.	Низькі: проблеми виявляються та вирішуються на ранніх етапах.

Швидкість внесення змін	Потребує багато часу на переписування документації.	Швидка реакція на зміни ринку чи бажання користувачів.
Сфера застосування	Проєкти з безпеки, медицини, де все має бути суворо за планом.	Стартапи, ігрові сервіси, мобільні додатки.

4. Обґрунтування вибору моделі SDLC

Для проєкту "SkinSwap" я обираю **модель Agile**.

Обґрунтування:

1. **Мінливість ринку:** Ціни на ігрові предмети та правила платформ (наприклад, політика Steam) часто змінюються. Agile дозволяє команді миттєво реагувати на ці зміни.
2. **Безпека та тестування:** Ми можемо випускати функціонал частинами. Наприклад, спочатку запустити систему перегляду інвентарю, протестувати її, а потім додавати систему обміну.
3. **Зворотний зв'язок:** Геймери — вимоглива аудиторія. Постійне отримання відгуків допоможе зробити інтерфейс максимально зручним для трейдингу.

Альтернатива (Waterfall): Ця модель була б кращою, якби ми створювали вузькоспеціалізовану закриту систему для однієї державної установи, де всі умови обміну регламентовані законом і не змінюватимуться роками. Але для динамічного ігрового ринку вона є занадто ризикованою.

5. Висновок

Для реалізації проєкту платформи для обміну ігровими предметами **"SkinSwap"** найбільш доцільним є використання **моделі Agile (гнучка розробка)**.

Обґрунтування вибору:

- **Відповідність масштабу та динаміці:** Ринок ігрових предметів та умови роботи з API (наприклад, Steam чи Epic Games) постійно змінюються. Agile дозволяє команді швидко адаптуватися до нових правил чи технічних обмежень, не зупиняючи весь процес розробки.
- **Робота зі змінами:** Оскільки "SkinSwap" — це сервіс для користувачів, нам важливо постійно збирати відгуки. Гнучка модель дозволяє вносити

правки в інтерфейс або логіку обміну безпосередньо під час розробки, що неможливо в жорстких рамках Waterfall.

- **Випуск MVP (мінімального продукту):** Завдяки ітеративному підходу ми можемо спочатку запустити базовий функціонал (наприклад, тільки перегляд інвентарю), а вже в наступних циклах (спринтах) додавати систему безпечних угод та чат. Це дозволяє проєкту швидше вийти на ринок.

Коли альтернативна модель (Waterfall) була б кращою?

Попри переваги Agile, каскадна модель (**Waterfall**) могла б стати кращим вибором лише за певних специфічних умов:

1. **Якби проєкт мав суворо обмежений і незмінний бюджет** та фіксовані терміни, встановлені зовнішнім замовником (наприклад, державний тендер).
2. **Якби всі технічні вимоги до безпеки та інтеграції були відомі на 100% заздалегідь** і не передбачали жодних змін протягом року розробки.
3. **Якби проєкт був частиною критично важливої інфраструктури**, де кожен крок має бути задокументований та затверджений ще до написання першого рядка коду.

Підсумок: Для сучасного комерційного IT-продукту у сфері геймінгу, такого як "SkinSwap", модель Agile є оптимальною, оскільки вона мінімізує ризики створення неактуального продукту та дозволяє гнучко керувати пріоритетами розробки.