## Загальні рекомендації до виконання тестового завдання

Тестове завдання покликане продемонструвати ваш підхід до вирішення прикладних задач розробника, які охоплюють як технічні навички, так і загальну культуру програмування. Оцінювання базується не лише на правильності кінцевого результату, але й на якості реалізації. Описані вимоги стосуються коду написаного вами, а також є вхідними параметрами для результуючого коду згенерованого АІ

Очікується, що рішення:

* буде працюючим і функціональним;
* матиме зрозумілу та логічну архітектуру;
* демонструватиме хороший стиль програмування;
* буде легко перевірити, протестувати й масштабувати.

## Основні критерії оцінки

### 1. Читабельність коду (Code Readability)

Читабельність — це фундамент якісного коду. У тестовому завданні очікується, що код:

* має чітку структуру, з логічним розділенням на частини (класи, методи, функції);
* містить осмислені назви змінних, методів і класів (без абревіатур або неочевидних скорочень);
* має оформлення згідно зі стандартами (indentation, spacing);
* супроводжується за потреби коментарями — не надмірними, але такими, що пояснюють нетривіальні моменти.

Мета: навіть розробник, який бачить код уперше, повинен швидко зрозуміти, що відбувається.

### 2. Простота коду (Code Simplicity)

Ми очікуємо, що кандидат уникає зайвої складності. Це означає:

* мінімальне використання "магічних чисел" і складних конструкцій без потреби;
* перевага простих, прямолінійних рішень над складними, якщо вони вирішують задачу однаково ефективно;
* уникнення передчасної оптимізації;
* дотримання принципу KISS (Keep It Simple, Stupid).

Мета: код має бути зрозумілим і підтримуваним, а не вражати складністю.

### 3. Декомпозиція задачі (Task Decomposition)

Оцінюється здатність розбивати задачу на логічні підзадачі та структурувати код відповідно. Хороша декомпозиція включає:

* поділ логіки на окремі функції/класи з чіткою відповідальністю (SRP — Single Responsibility Principle);
* уникнення дублювання коду (DRY — Don't Repeat Yourself);
* гнучкість у масштабуванні та внесенні змін;
* можливість покриття ключових частин коду юніт-тестами (опціонально, але вітається).

Мета: продемонструвати навички архітектурного мислення та організації коду.