**Опис проекту: SmartLance**

**Мета проекту**

Створення платформи для управління завданнями з інтеграцією блокчейну, яка забезпечує прозорість, безпеку та децентралізовану обробку даних.

**Основні функції**

**1. Користувачі та профілі**

* **Реєстрація користувачів:**
  + Логін через криптогаманець (наприклад, Metamask).
  + Зберігання базової інформації (наприклад, ім'я, біографія) у блокчейні.
* **Авторизація:**
  + Користувачі авторизуються через свої криптогаманці.
  + Бекенд забезпечує перевірку токену авторизації або адреси, переданої з фронтенду.
* **Профілі:**
  + Зчитування детальної інформації про користувача з бази даних.
  + Зберігання метаданих (наприклад, аватарів, рейтингу) у бекенді.

**2. Завдання (Tasks)**

* **Створення завдання:**
  + Замовник створює завдання через фронтенд.
  + Дані зберігаються як у базі даних, так і в смарт-контракті.
* **Прийняття завдання:**
  + Виконавець приймає завдання через фронтенд.
  + Контракт блокує кошти замовника до завершення завдання.
* **Підтвердження виконання:**
  + Замовник підтверджує виконання завдання, після чого кошти перераховуються виконавцю.

**3. Арбітраж (Disputes)**

* **Ініціація:**
  + У разі конфлікту сторони можуть ініціювати арбітраж через смарт-контракт.
* **Рішення:**
  + Арбітр ухвалює рішення щодо передачі коштів одній із сторін.

**4. Обробка подій із контрактів**

* Бекенд підписується на події смарт-контрактів (наприклад, TaskCreated, TaskCompleted).
* Події обробляються Web3j і синхронізуються з базою даних.

**Приклад обробки подій**

web3j.ethLogFlowable(filter).subscribe(log -> {

EventValues eventValues = Contract.staticExtractEventParameters(event, log);

String taskId = eventValues.getIndexedValues().get(0).getValue().toString();

String description = eventValues.getNonIndexedValues().get(0).getValue().toString();

// Оновлення бази даних

taskRepository.save(new Task(taskId, description));

});

**5. Бекенд API**

* REST API для зчитування та зберігання даних.
* Основні ендпоінти:
  + /users для управління профілями.
  + /tasks для роботи із завданнями.
  + /arbitration для роботи з арбітражами.

**6. Інтеграція з блокчейном**

* Всі транзакції на блокчейні виконуються через фронтенд (наприклад, Web3.js).
* Бекенд використовує Web3j для:
  + Зчитування подій.
  + Синхронізації даних між базою та контрактами.

**7. Архітектура**

* **Фронтенд:** Виклик смарт-контрактів через Web3.js.
* **Бекенд:**
  + Зберігання метаданих і синхронізація.
  + Обробка подій контрактів.
  + Авторизація через токени чи адреси користувача.
* **Блокчейн:**
  + Смарт-контракти для управління завданнями, профілями, рейтингами та арбітражем.

**Технології**

* **Бекенд:** Java, Spring Boot, Web3j.
* **База даних:** H2 (для тестування).
* **Блокчейн:** Solidity, Ethereum.
* **Фронтенд:** Web3.js, Metamask (планується).

**Наступні кроки**

1. Налаштувати обробку подій контрактів у бекенді.
2. Реалізувати базову авторизацію користувачів через адреси гаманців.
3. Оптимізувати API для зчитування даних із бази.
4. Підготувати середовище для інтеграції з фронтендом.