### ****Концепція проекту****

#### ****Назва проекту та належність****

**Проект: Madevinc Pool AI (2025)  
Це програмний продукт, який реалізує децентралізовану систему майнінгу з використанням технологій штучного інтелекту та блокчейну Solana. Проект розроблений для автоматизації процесу розподілу обчислювальних ресурсів між учасниками пулу (воркерами), забезпечення ефективного розподілу нагород за майнінг та взаємодії користувачів через Telegram-бота.**

### ****Структура файлів та їх призначення****

1. **Cargo.toml**
   * **Служить для опису залежностей проекту та його метаданих (назва, версія, автор).**
   * **Містить бібліотеки, такі як actix-web, tokio, solana-sdk, teloxide, tch (для машинного навчання) тощо.**
2. **src/main.rs**
   * **Головний файл програми, який ініціалізує сервер, створює об'єкти пулу, воркерів, малого світу (SmallWorld) та Telegram-бота.**
   * **Відповідає за запуск HTTPS-сервера та обробку задач воркерів.**
3. **src/state.rs**
   * **Містить глобальний стан додатка (AppState), який зберігає дані про модель нейронної мережі, системні ресурси, RAID-статус, активних воркерів, токенізатор та бота.**
   * **Служить для централізованого управління даними всередині програми.**
4. **src/workers.rs**
   * **Реалізує логіку робочих (воркерів), які виконують задачі майнінгу та обробки даних.**
   * **Кожен воркер має унікальний ID (workername.solana\_address), параметри використання CPU/пам'яті, майнінгову потужність та механізм отримання нагород.**
5. **src/burstraid.rs**
   * **Реалізує механізм розподілу сідів (seeds) між воркерами для підтримки роботи RAID-масиву з гарячим підключенням і відключенням (seeds).**
   * **Включає функціональність перерозподілу ресурсів у разі відмови одного з воркерів.**
   * **Монтує рейд у папку /data, де зберігаються всі ШІ моделі та тимчасові дані у динамічному масиві.**
6. **src/smallworld.rs**
   * **Реалізує модель "малого світу" для представлення взаємодії між вузлами (пулами та воркерами).**
   * **Використовується для візуалізації мережі та аналізу зв'язків між учасниками.**
   * **Принцип роботи схожий на нейронну мережу у мозку людини, з динамікою коннектів та гнучкістю між якістю доступу до воркерів та розміщенням даних на сідах (seeds).**
7. **src/tgbot.rs**
   * **Реалізує Telegram-бота для взаємодії з користувачами.**
   * **Підтримує команди для отримання статусу пулу, додавання воркерів, налаштування гаманця Solana та інших операцій через аплікацію воркера (майнера) у чаті Telegram.**
8. **src/tokenizer.rs**
   * **Реалізує механізм розподілу токенів між активними воркерами на основі їх майнінгової потужності та зверненням до даних у сідах (seeds).**
   * **Використовується для автоматичного розрахунку нагород та відправки комісій на адресу пулу.**
9. **src/pool/\*.rs**
   * **Файли, які реалізують інтерфейс користувача для взаємодії з пулом (наприклад, сторінки /header, /home, /login).**
   * **Служать для відображення інформації про пул та його статус.**
10. **src/bridges.rs**
    * **Має зеркало доменів пулів, приєднаних до poolchain. Якщо адреса комісії пулів однакова, Poolchain функція з'єднує такі самі екземпляри пулів для міграції процесів завдань до інших брендів воркерів у найближчих вільних малонавантажених smallworlds нетворках пулів.**
11. **src/lmrouter.rs**
    * **Моделі ШІ, які будуть знайдені у /pool, можуть бути обрані для формування роботи з чатом або pool\_cok.rs.**
12. **src/pool\_cok.rs**
    * **Інтелектуальна розетка для передачі потоку інтелектуальної потужності в інші інфраструктури з урахуванням токенайзеру.**
13. **src/loadbalancer.rs**
    * **Управління балансуванням навантаження на пул та його нетворк.**
14. **src/lib.rs**
    * **Стандартний файл для експорту модулів.**
15. **src/model.rs**
    * **Логіка роботи моделі з управлінням smallworld та burst нетворками, розподілення задач воркерам, робота з lmrouter для взаємодії з іншими доступними моделями у папці /data.**
16. **Файли src/pool/\***
    * **src/pool/header.rs — заголовок сторінки.**
    * **src/pool/home.rs — сторінка з інформацією про проект.**
    * **src/pool/login.rs — логін через Telegram кнопку.**
    * **src/pool/mod.rs — модуль для організації підсистем пулу.**
    * **src/pool/playground.rs — чат для користування обраною однією з доступних моделей з /data.**
    * **src/pool/sidebar.rs — меню.**
    * **src/pool/settings.rs — налаштування воркера клієнта.**

**Ці файли відносяться до HTTPS-інтерфейсу з пулом з автоформуванням під дисплей пристрою або веб-аплікації і чату Telegram-боту.**

### ****Принцип роботи пулу****

1. **Ініціалізація пулу**
   * **Під час запуску програми створюється об'єкт AppState, який зберігає загальні дані про пул, включаючи список воркерів, RAID-статус та конфігурацію нейронної мережі.**
2. **Додавання воркерів**
   * **Користувач може налаштувати свого воркера через Telegram-бота або інтерфейс пулу.**
   * **Кожен воркер ініціалізується з унікальним ID (workername.solana\_addr), початковою майнінговою потужністю та розміром сідів не менше ніж 10% від вільного місця на диску, 25% RAM, 25% потужності CPU. Клієнт також може виставити своє значення вище (диск по кількості seeds 100 MB, RAM у %, CPU у % та кількості ядер).**
3. **Обробка задач**
   * **Воркери отримують задачі через канал (mpsc::channel) і обробляють їх залежно від доступних ресурсів (CPU, RAM, пам'ять диску seeds).**
   * **Воркер, якщо вільний, має запитувати пул про завдання, відправляючи статистику по ресурсам та стан зв'язку з нетворком (пінг). У разі успішної обробки завдання він отримує нагороду:**
     + **Якщо робота з burst — показує затрачені токени та пише заробіток у Solana з маркером SEED.**
     + **Якщо обчислення на CPU виконане — з маркером YAY!.**
4. **Розподіл нагород**
   * **Токенізатор розраховує нагороди для активних воркерів на основі їх майнінгової потужності.**
   * **Комісія (0.01%) автоматично відправляється на адресу пулу (POOL\_COMMISSION\_ADDRESS).**
5. **Перерозподіл ресурсів**
   * **У разі відмови одного з воркерів система автоматично перерозподіляє його сіди між іншими активними воркерами.**
6. **Моніторинг**
   * **Користувачі можуть отримувати інформацію про статус пулу, активність воркерів та розподіл нагород через аплікацію Telegram-бота або веб-інтерфейс пулу.**

### ****Принцип роботи бота для взаємодії з пулом****

1. **Ініціалізація бота**
   * **Бот створюється за допомогою бібліотеки teloxide і ініціалізується з токеном Telegram API.**
   * **Об'єкт бота зберігається в глобальному стані (AppState) для подальшої взаємодії.**
2. **Обробка команд**
   * **Бот слухає повідомлення від користувачів і обробляє команди, такі як:**
     + **/status — показує поточний статус пулу та домен.**
     + **/worker\_name — встановлює ім'я воркера.**
     + **/wallet\_set — налаштовує Solana-гаманець.**
     + **/wallet\_show — показує поточний гаманець.**
     + **/mine — запускає аплікацію воркера у Telegram-боті.**
3. **Сповіщення про помилки**
   * **Якщо воркер офлайн або виникає помилка при обробці задачі, бот надсилає сповіщення клієнту воркера.**
4. **Взаємодія з користувачем**
   * **Бот надає зручний інтерфейс з аплікацією воркера, дозволяючи користувачам легко налаштовувати, перевіряти статус і моніторити процес майнінгу.**

### ****Загальна схема роботи системи****

1. **Користувач заходить і логінюється у пул через веб-інтерфейс або Telegram-бота.**
2. **Воркери підключаються до пулу і починають обробляти задачі.**
3. **Система автоматично розподіляє нагороди та перерозподіляє ресурси у разі необхідності.**
4. **Користувачі отримують актуальну інформацію про стан пулу та воркера через аплікацію бота або веб-інтерфейс.**