**План за разработка на джъдж система за оценяване на задачи по JavaScript и Python към Сирма Академия**

**ФАЗА 1: Проучване и планиране**

**1.1 Изисквания:**

* Езици: JavaScript и Python
* Видове задачи: вход/изход от конзолата
* Формат на вход/изход
* Типове тестове: стандартни, edge cases, негативни тестове

**1.2 Анализ на съществуващи системи:**

* Преглед на: SoftUni Judge, HackerRank, DOMJudge, Codeforces
* Да се идентифицират добри практики и слаби страни.

**1.3 Технологии:**

* Backend: Node.js (JS) и Python runtime (Python задачи)
* Frontend: Angular (или React/Vue)
* База данни: MongoDB
* Sandbox: Docker/Firejail за сигурно изпълнение на код
* Скрипт за тестване: Python скрипт за сравнение вход/изход

**ФАЗА 2: MVP (Минимална работеща версия)**

**2.1 Потребителски интерфейс:**

* Форма за качване на решение
* Избор на език: JS/Python
* Избор на задача
* Submit бутон
* Минималистичен UI с фокус върху функционалност.

**2.2 Backend API:**

* **POST /submit** – качва решение
* **GET /result/:id** – връща резултат от тестване

**2.3 Judge Engine:**

* Получава код + вход + очакван изход
* Създава Docker контейнер и изпълнява кода
* Сравнява резултата с очаквания
* Връща статус: Passed / Failed / Error

**Пример Docker команда:**

docker run --rm -v $(pwd)/submission:/app python:3.10 python3 /app/solution.py<input.txt>output.txt

**ФАЗА 3: Работа с решения и тестове**

**3.1 База задачи:**

* Параметри: title, description, input, output, test cases
* Админ панел за създаване и редакция

**3.2 Формат на тестовете:**

{

"input": "5\n1 2 3 4 5\n",

"expected\_output": "15\n"

}

**ФАЗА 4: Сигурност и оптимизация**

**4.1 Sandbox:**

* Изолирано пускане на код
* Ограничения по време и памет
* Защита от безкрайни цикли и опасни операции

**4.2 Опашка за задачи:**

* Изпратените решения се поставят в Redis или RabbitMQ опашка
* Обработват се от workers (няколко judge модула)

**ФАЗА 5: Надграждане и подобрения**

**5.1 UI подобрения:**

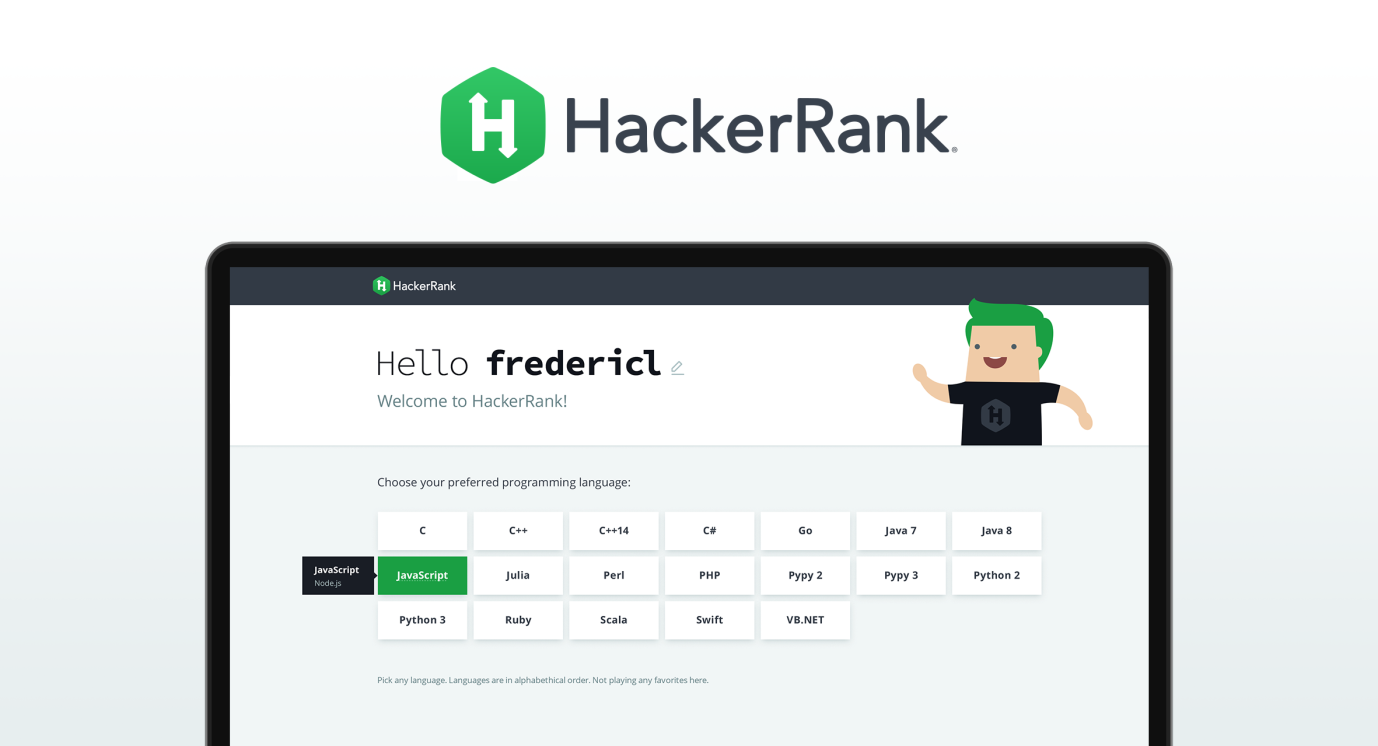
* История на решенията
* Прогрес по задачи
* Таблица с резултати
* Статистика на профила  
  *(Май събира горните 3 подточки – пускам картинка за справка*

**П.С. Показаните изображения са с илюстративен характер, както и наклонения текст са за нас и с цел да се добие по – добра представа какво е искал да каже автора. След като ги обсъдим ги махаме**



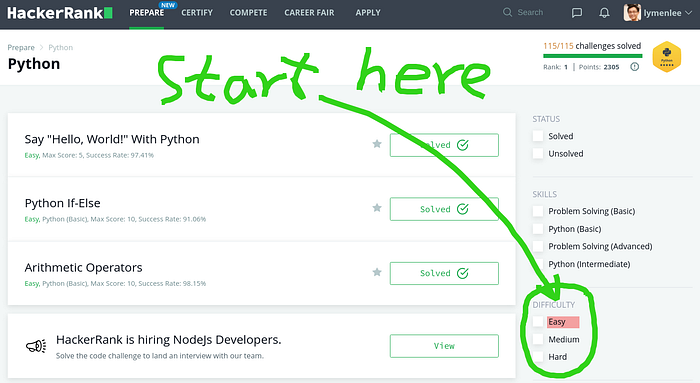
*Пълна статистика на профила, че дори и класация имат в CODE FORCES*  
*Това обаче ще го направи достъпен за колеги (в Софтуни като минус давам, че не може да достъпиш профил на приятелче ☺, не да му гледаш решеният без разрешение, а да го видиш върви ли напред, ще го бъде ли с програмирането ☺.*

* Дефиниране на програмен език при вход

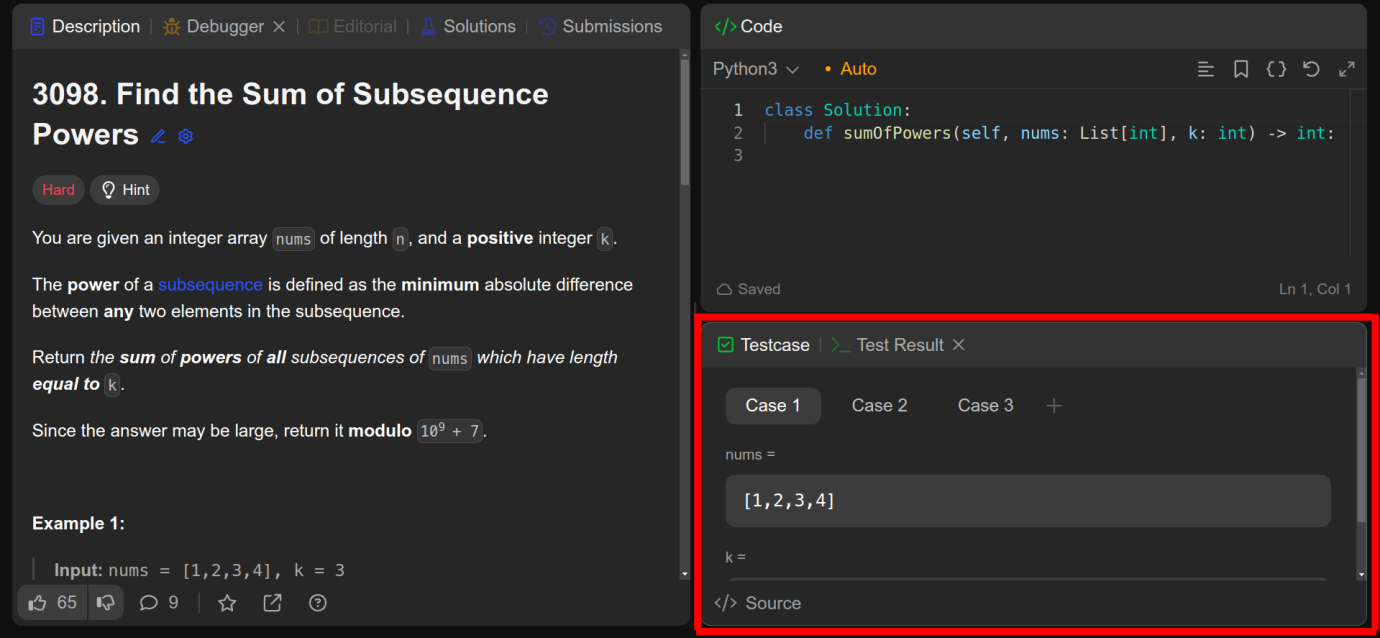


*Яко е да влизаме в езика предварително, а не да се ровиш като кокошка къде си, къде ти е папката и задачата. После да забравиш да си смениш езика при пускане на кода (като в Basics курса) - гръмнеш и седиш и чакаш да зареди. Знам, че ще го правим само за JS и Python, но исках да го отбележа.*

* Материал по сложност



*Харесва ми идеята за нивата на сложност, както и да вади в списък само решени или нерешени задачи*

* Визуализация на условията по време на решение  
    
  

*Изображение: LEET CODE*

*Вместо има само списък със задачи, ще е хубаво, когато се кликне на задачата да се отваря условие, вместо да продължава да стои списък със задачи*

*Хич не ми харесва мястото за писане на код в alpha judge , много ми е дървено. Не знам дали защото съм новак и нямам опит, ама не е моето.*

**5.2 Нови възможности:**

* Custom тестове от потребителя
* Time/Memory limits
* Интеграции с LMS, Discord или Slack

**5.3 Подобрения спрямо съществуващи Judge systems**

* По – ясна обратна връзка при грешки
* Да се показва тестовия случай при грешка  
  *(При гръмналия вход/изход, ако всичко гърми – по def = 2/3 да се виждат.)*
* AI tips
* User friendly интерфейс
* Фокус върху образованието

**5.4 Интеграция на AI:**

* Предлагане на специални случаи, които да се тестват
* Генериране на edge cases
* Подсказки при повтарящи се грешки   
  *(Например при 5 неуспешни опита може да даде подсказка)*
* Анализ на допуснатите грешки
* Генериране на сходни задачи при проблем с материала  
  (*При даден брой неуспешни опити и последвал успешен такъв да генерира смислово подобна задача за затвърждаване на знанията)*
* Обяснение на условия  
  *(Ако има задачи със \* (визирам като нашите последните задачи в упражненията) AI да даде разяснение (реформулиране на условието), tip)*
* Анализ на кода   
  *( Решението оптимално добро ли е или би могло да се подобри)*

**Екип и роли**

* **Team Lead** – архитектура и организация
* **Backend Developer** – API, Judge Engine
* **Frontend Developer** – UI
* **DevOps** – Docker, CI/CD, хостинг
* **QA** – Тестване и сигурност

**Примерен график**

| **Фаза** | **Времетраене** |
| --- | --- |
| Проучване | 1 седмица |
| MVP | 2 седмици |
| Тестове и сигурност | 1 седмица |
| UI и функционалности | 1 седмица |
| Обратна връзка и подобрения | 1 седмица |