Part A: Web2 Backend

Task 1: Simple Backend Endpoint

Create a simple Python Django app that has the following features:

- a backend API endpoint called get_score with a simple dummy formula that could be anything e.g. result = input + 1
- a PostgreSQL database that the backend API uses to log the user ID and the score
- demonstrate that the endpoint is behaving as expected (how do we test it? how can we prove that it is working as expected?)

Part A: Web2 Backend

Task 1: Simple Backend Endpoint

index.html U | urls.py U | views.py U

logrequest > views.py

```python
from django.http import HttpResponse
from django.shortcuts import render
import datetime
from .models import Score
import json

# Create your views here.

def index(request):
    today = datetime.datetime.now().date()
    return render(request, "index.html")
def logview(request):
    if request.method == 'POST':
        print(request.body)
        username = request.POST['username']
        m_score = int(request.POST['m_score'])+1
        Score.objects.create(User_ID =username , Score=m_score)
        print(m_score)

        return HttpResponse(username+"      "+str(m_score))
```

index.html U | urls.py U | views.py U

logrequest > urls.py

```python
from django.urls import path

from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('get_score', views.logview, name='logview')
]
```

index.html U | urls.py U | views.py U | models.py U

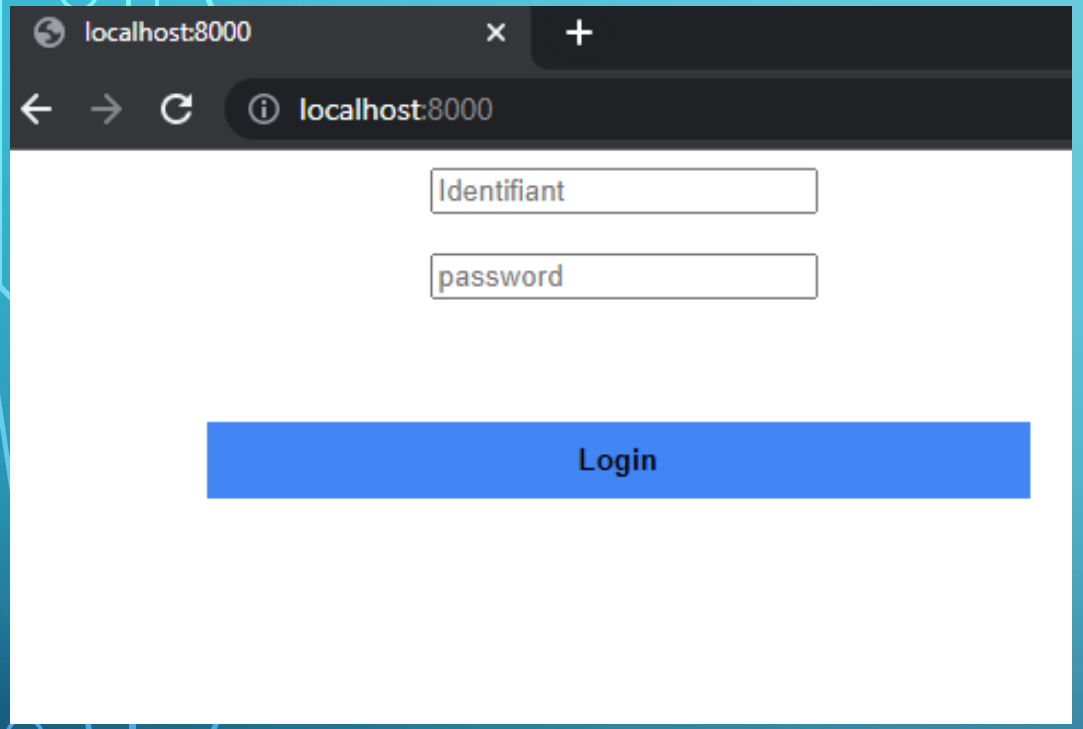
logrequest > models.py

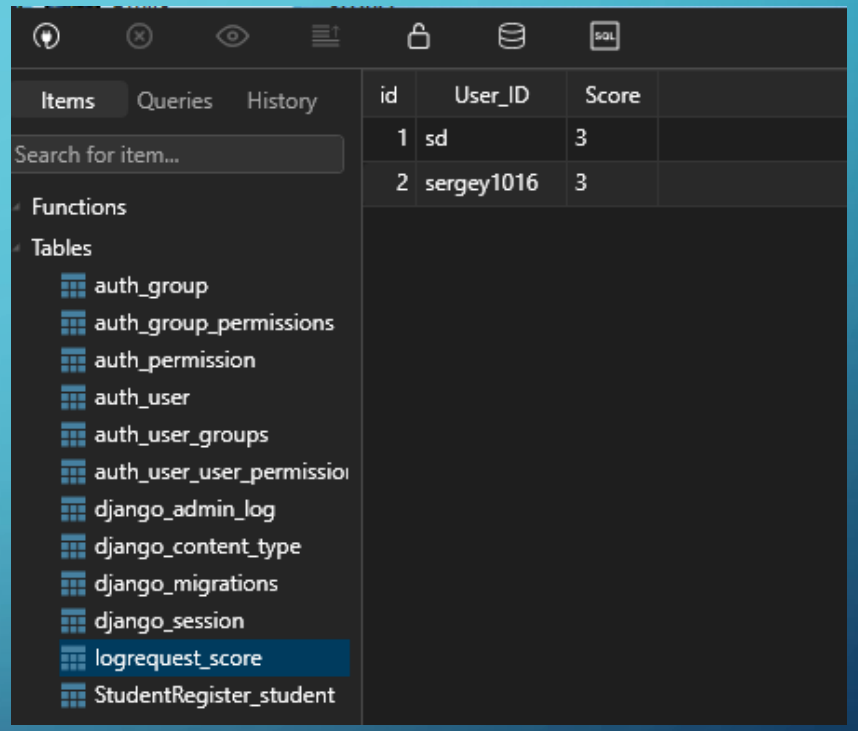
```python
from django.db import models

# Create your models here.
class Score(models.Model):
    User_ID = models.CharField(max_length=20, verbose_name='User Id')
    Score = models.CharField(max_length=20, verbose_name='Score')
```

```html
<html>
  <body>
    <form name="form" action="./get_score" method="POST">
      {% csrf_token %}
      <div>
        <input
          type="text"
          placeholder="Identifiant"
          name="username"
        />
      </div>

      <br />

      <div>
        <input
          type="text"
          placeholder="password"
          name="m_score"
        />
      </div>

      <br />

      <div>
        <button type="submit" value="Login">
          <strong>Login</strong>
        </button>
      </div>
    </form>
  </body>
</html>
```

Part A: Web2 Backend

Task 1: Simple Backend Endpoint





Task 2: Admin Panel

Create a simple admin panel where operations staff can use to manage the database:

- a non engineer should be able to view the SQL tables, and search/make queries
- even better if the staff can edit the entries too

Part A: Web2 Backend

Task 2: Admin Panel

Django CRUD Operations

A Simple Student Register for Creating, Reading, Updating
& Deleting Student Information

Admission. No.	First Name	Last Name	Date Joined	Course Name	Year Of Study	Unit Name	Grade	Actions
gfd	gfd	gfd	April 4, 0005	Bsc. Electrical & Computer Engineering	2nd Year	Economics	A	
fdsaf	fdsaf	fsda	March 31, 0003	Bsc. Telecommunications Engineering	3rd Year	Complex Analysis	B	

Add Student Info

Part A: Web2 Backend

Task 2: Admin Panel

Admission. No.	First Name	Last Name	Unit Name	Grade	Actions
gfd	gfd	gfd	Economics	A	
fdsaf	fdsaf	fsda	Complex Analysis	B	

Add Student Info

Add New Student Info

Admission Number

First Name

Last Name

Date Of Birth

mm/dd/yyyy

Date Joined

mm/dd/yyyy

Faculty

Department

Course Name

Year Of Study

Unit Name

Grade

Cancel

Save Student Info

Part A: Web2 Backend

Task 2: Admin Panel

PostgreSQL 14.5 : testdb : public.StudentRegister_student

Items Queries History

earch for item...

Functions

Tables

- auth_group
- auth_group_permissions
- auth_permission
- auth_user
- auth_user_groups
- auth_user_user_permissio
- django_admin_log
- django_content_type
- django_migrations
- django_session
- logrequest_score
- StudentRegister_student

logrequest_score | StudentRegister_student

id	Admission_Number	First_Name	Last_Name	Date_Of_Birth	Date_Joined	Faculty	Department	Course_Name	Year_Of_Study	Unit_Name	Grade
2	gfd	gfd	gfd	0004-04-04	0005-04-04	Faculty Of Engineering	Department of Electrical & Computer Engineering	Bsc. Electrical & Computer Engineering	2nd Year	Economics	A
3	fdsaf	fdsaf	fsda	0002-03-04	0003-03-31	Faculty Of Engineering	Department of Mechanical Engineering	Bsc. Telecommunications Engineering	3rd Year	Complex Analysis	B

Task 3: Database Migration

If we want to change the schema of the database, say we want to add, edit, or remove columns:

- demonstrate a process or script, or a demo video of the schema update process
- demonstrate that the endpoints and the services are not affected, and the tests are running fine

Part A: Web2 Backend

Task 3: Database Migration

Project Setup Instructions

1. Git clone the repository
2. Go To Project Directory
3. Create Virtual Environment

```
virtualenv env
```

4. Active Virtual Environment

```
env\scripts\activate
```

5. Install Requirements File

```
pip install -r requirements.txt
```

6. Make Migrations

```
py manage.py makemigrations
```

7. Migrate Database

```
py manage.py migrate
```

8. Run Project

```
py manage.py runserver
```

Part B: Web3

Web3 Integration: Reading

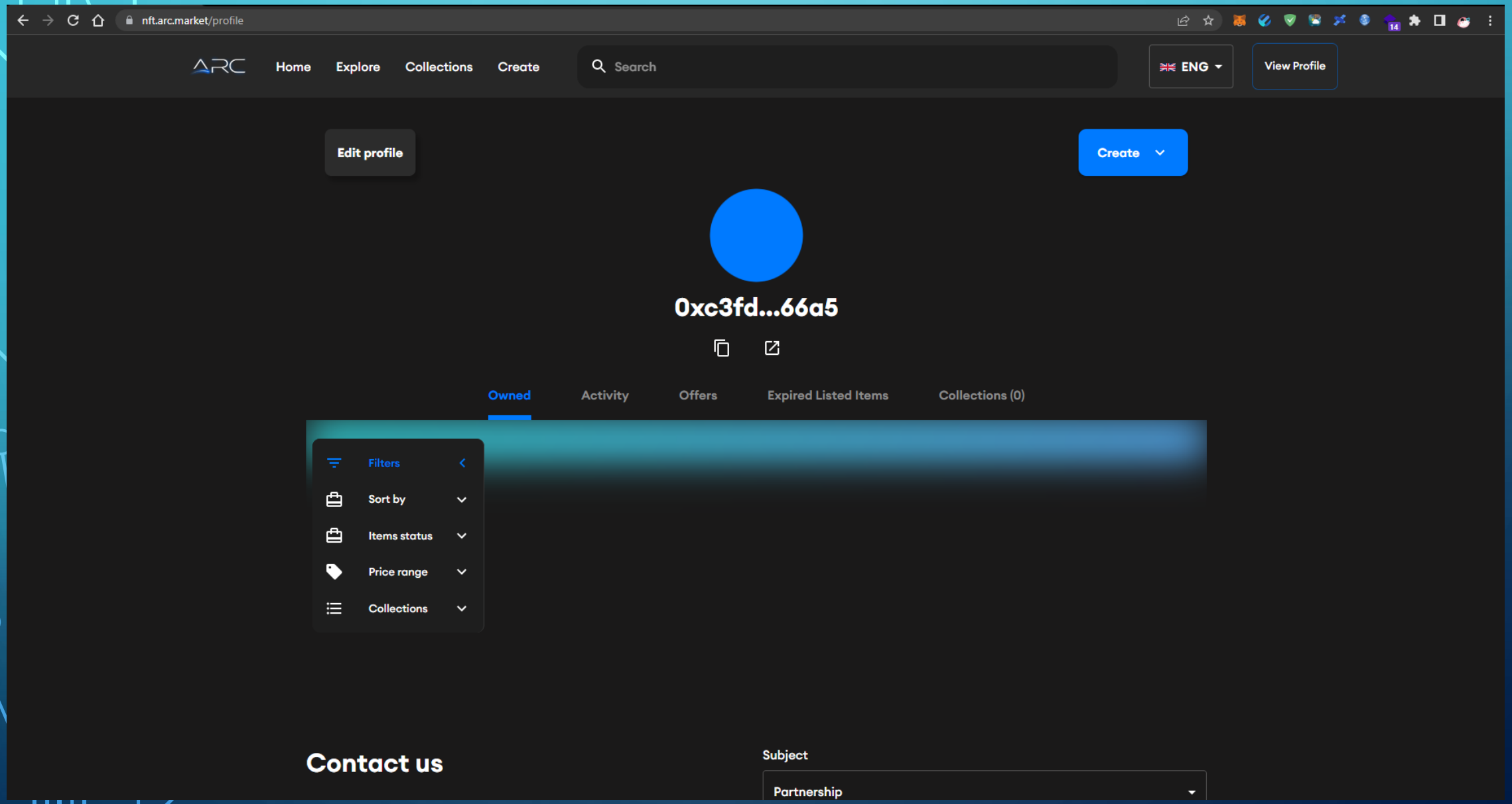
Our mobile app will allow users to interact with the web3 ecosystem, for the purpose of this challenge, you can use any EVM-compatible tools. Create functionalities and flows for the following use cases:

Web3 Integration: Reading

- create a functionality to verify that a user owns a wallet address
- allow users to connect their wallet and then display their token balances, and NFTs (even better if you can even retrieve the metadata/traits and media content of the NFTs)
- add filter functionality to only display NFTs from a list of contracts (for example, you can only choose to display Axie Infinity and Genopets NFTs)
- play to earn backend, that is, allowing the minting of tokens (when player wins) and burning of tokens (when player uses tokens to buy in-game items, for example)

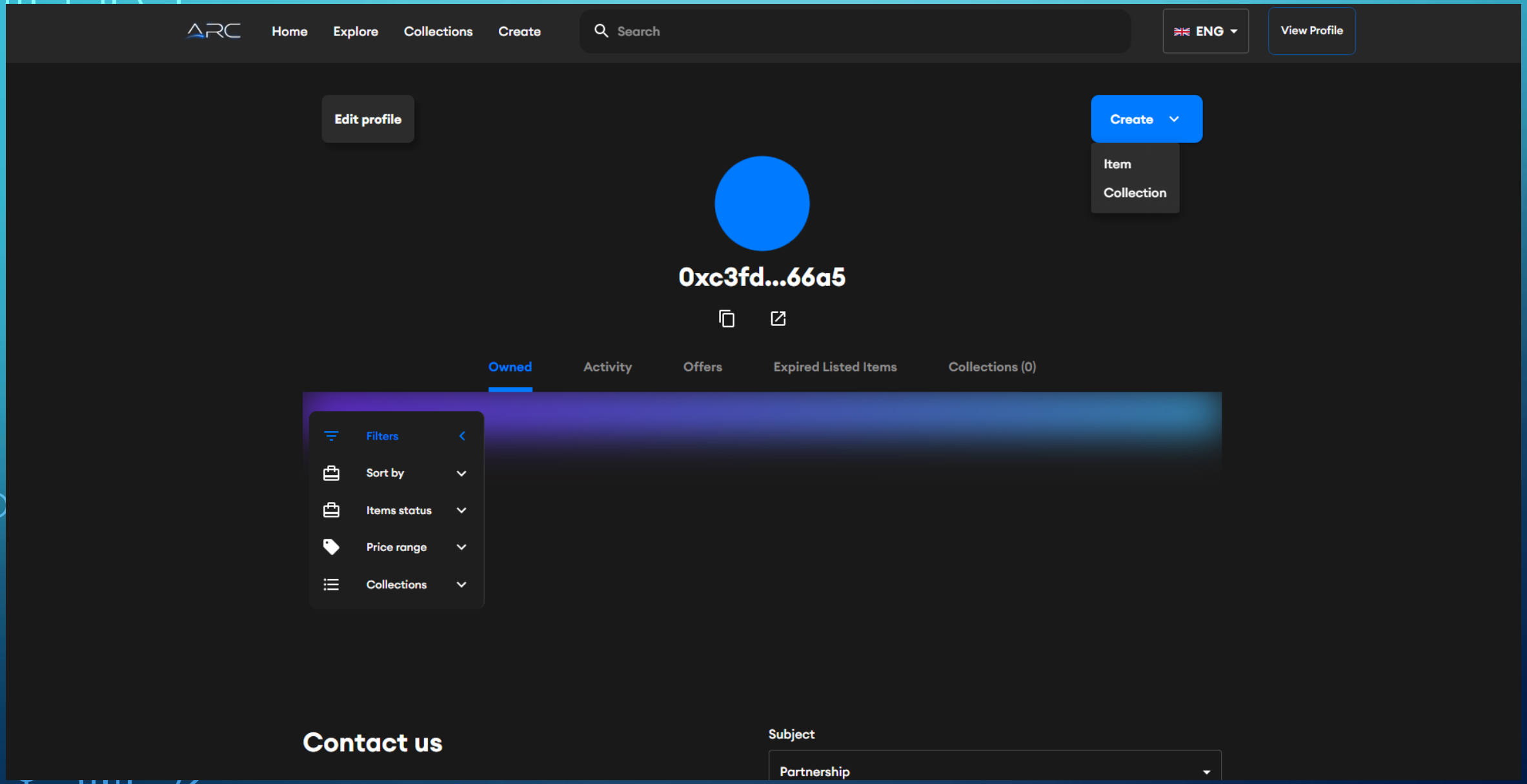
Part B: Web3

Web3 Integration: Reading



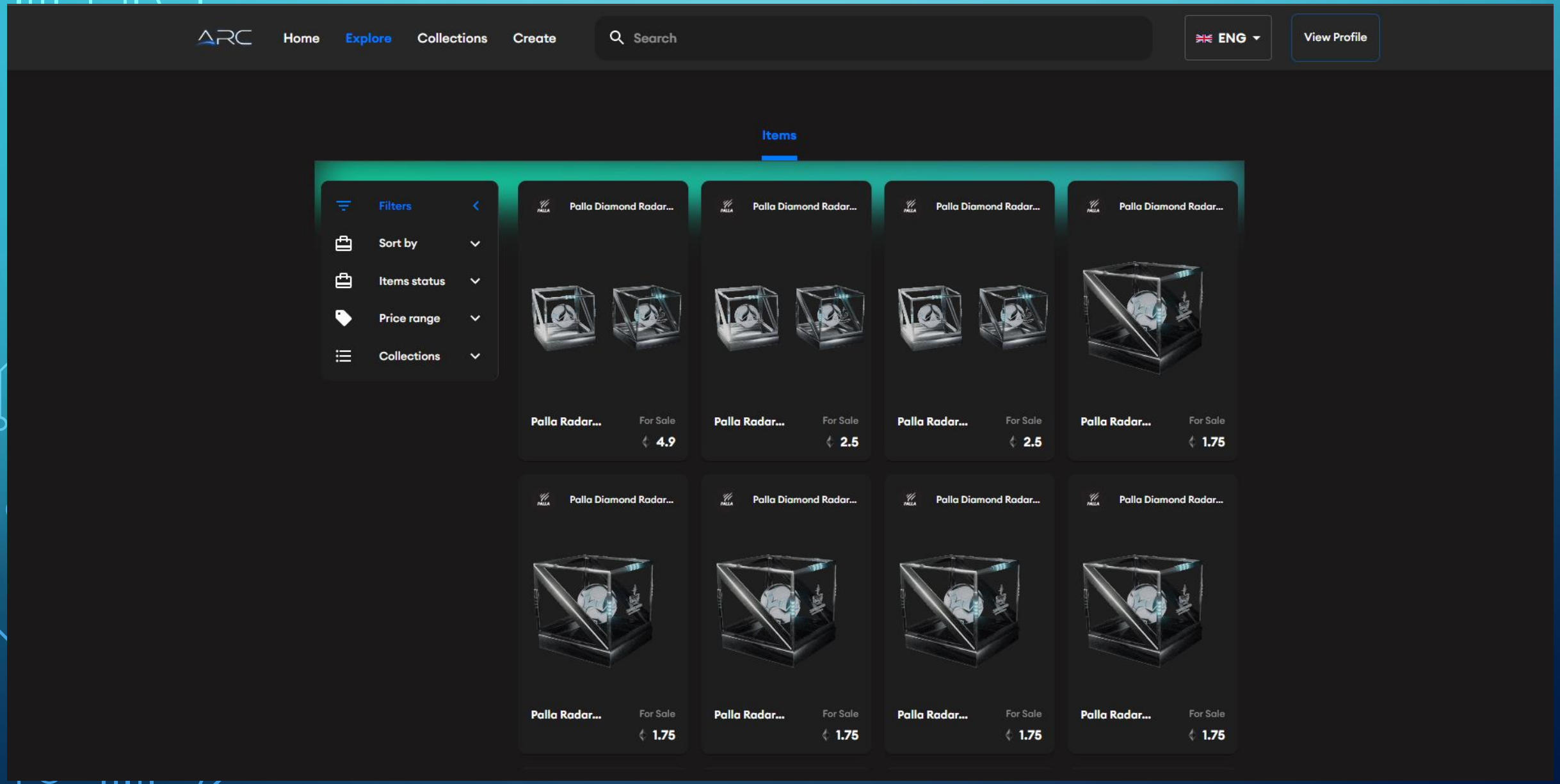
Part B: Web3

Web3 Integration: Reading



Part B: Web3

Web3 Integration: Reading



Smart Contracts

- create an ERC20 token smart contract, call it your name or a unique id such as $JOHN123
- create a staking smart contract, that is able to accept deposits of tokens and NFTs
- create a marketplace smart contract, that is able to allow users to trade and swap tokens and NFTs
- create a bridge contract
- deploy the smart contracts, write tests, and demonstrate that they work on testnet or mainnet

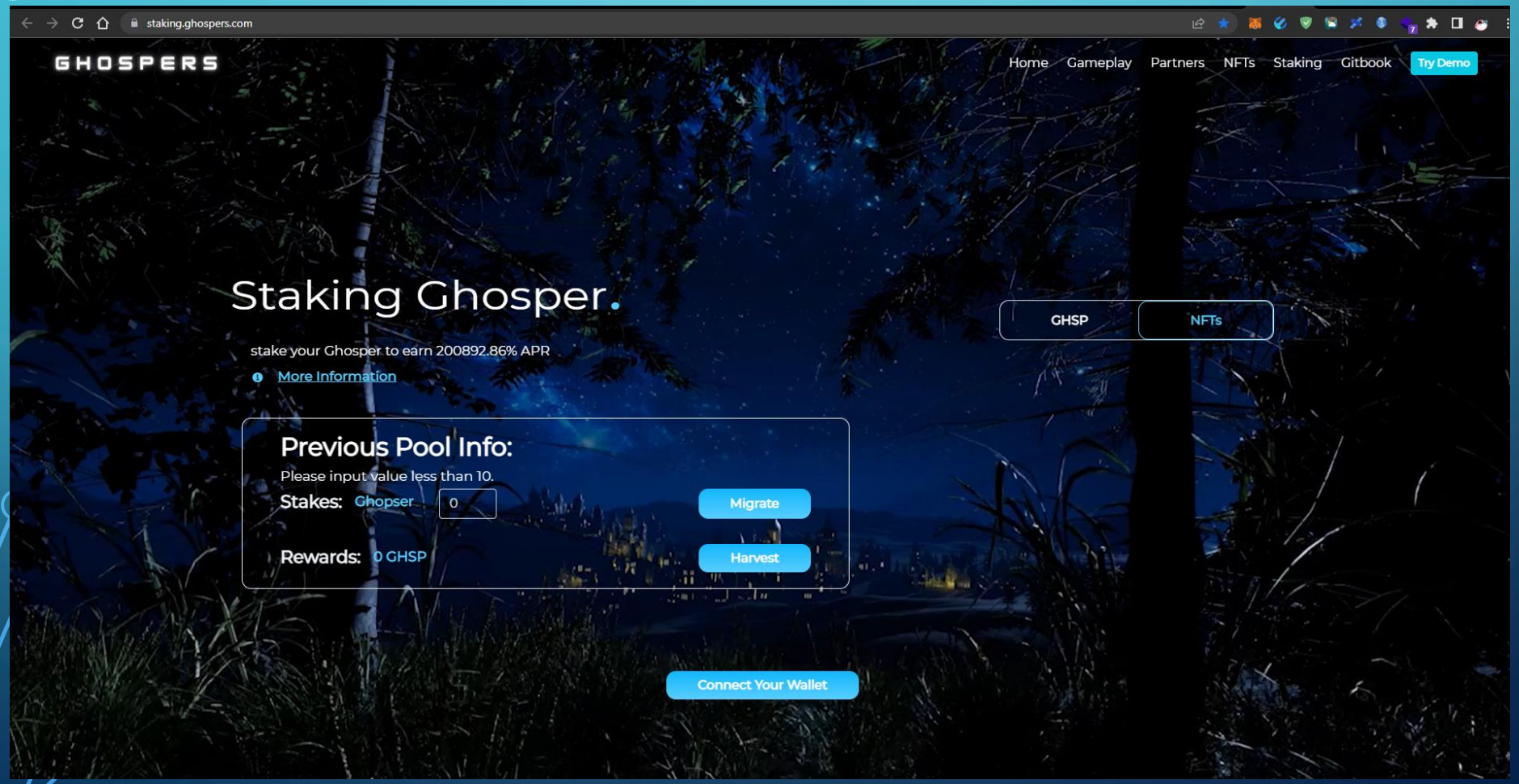
Smart Contracts

- create an ERC20 token smart contract, call it your name or a unique id such as $JOHN123

I uploaded USDT.sol

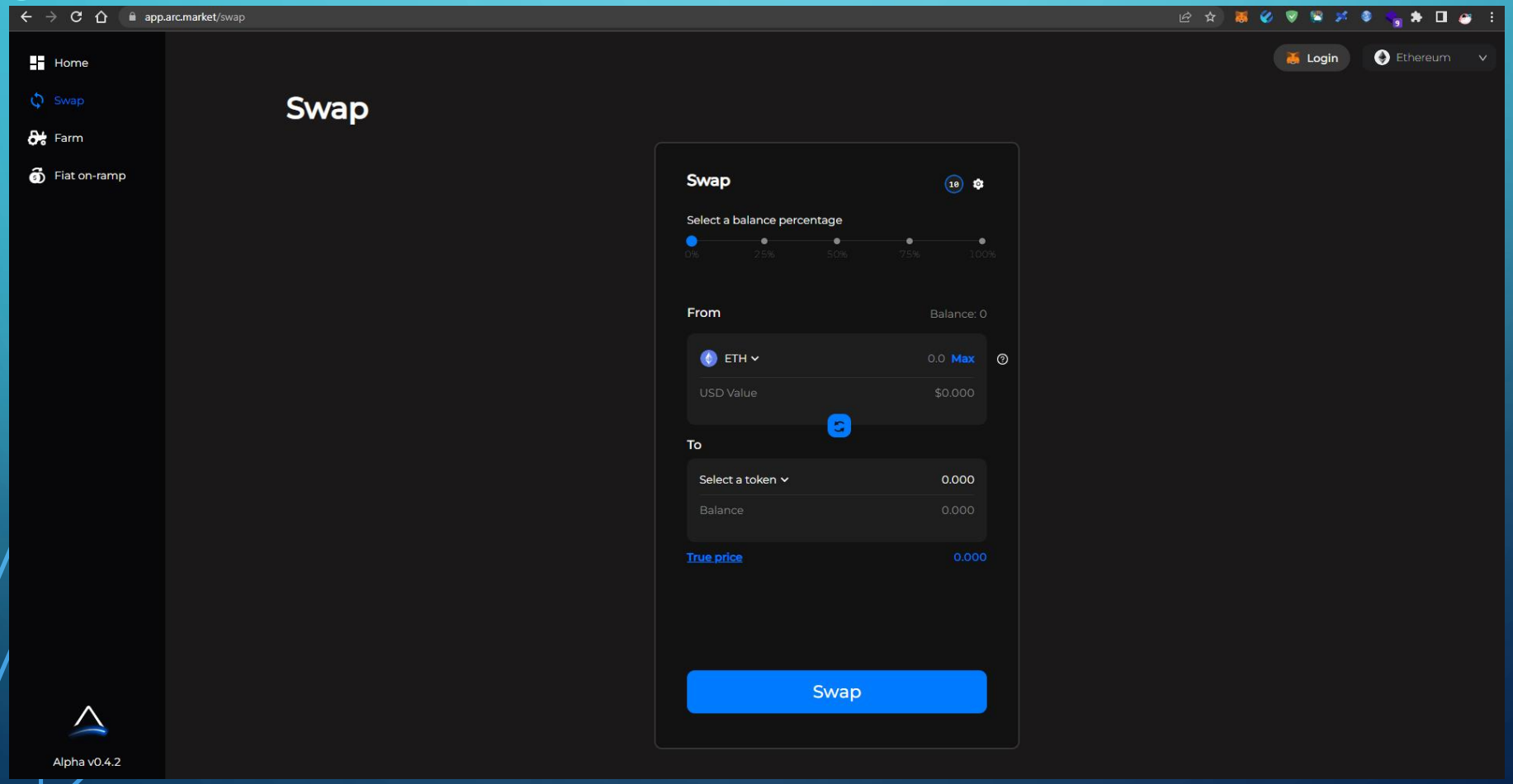
Smart Contracts

- create a staking smart contract, that is able to accept deposits of tokens and NFTs (https://staking.ghospers.com/)



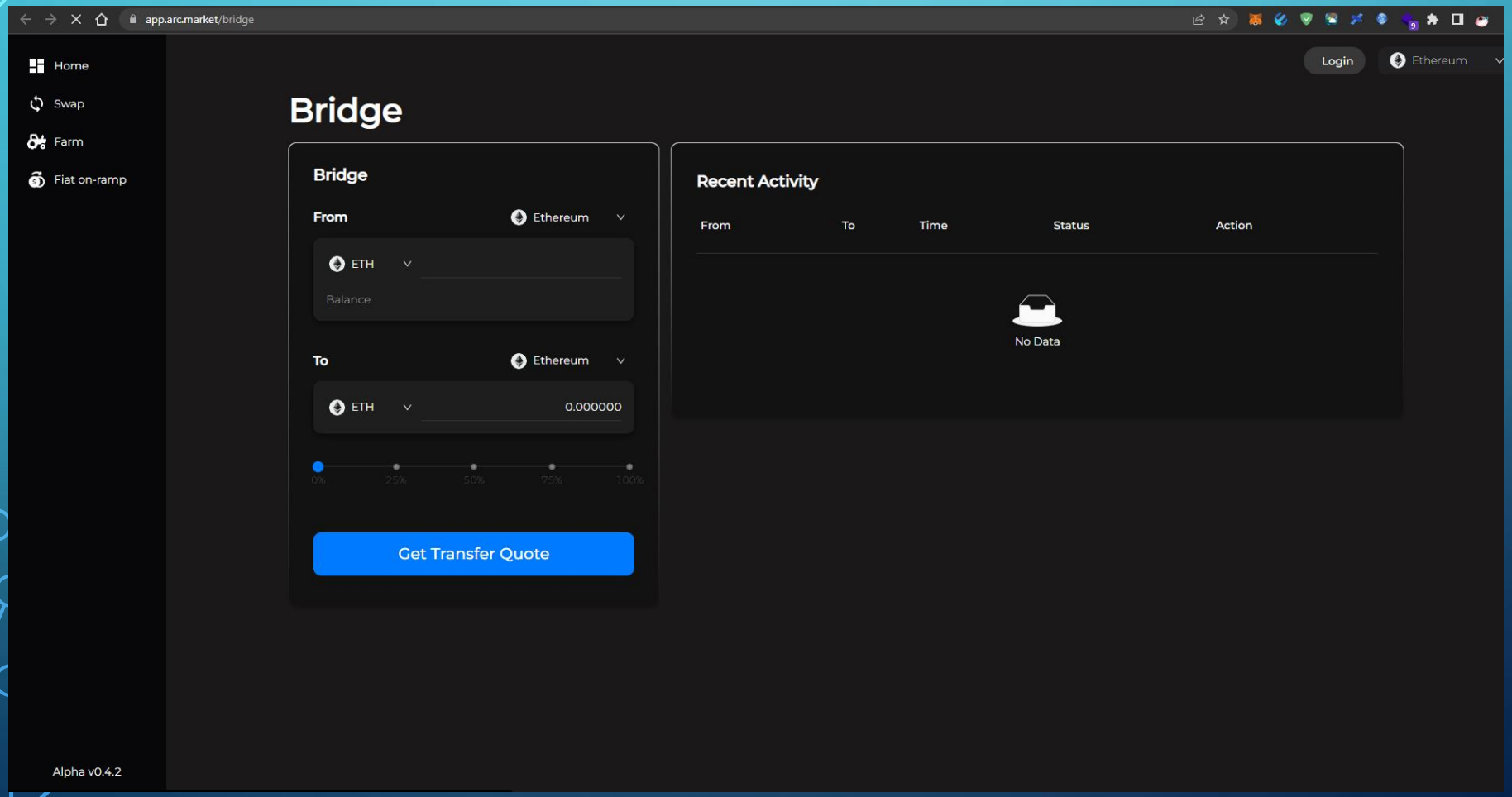
Smart Contracts

- create a marketplace smart contract, that is able to allow users to trade and swap tokens and NFTs (https://app.arc.market/swap)



Smart Contracts

- create a bridge contract (https://app.arc.market/bridge)



Part C: Infrastructure and Devops

Part C

Deployment to cloud

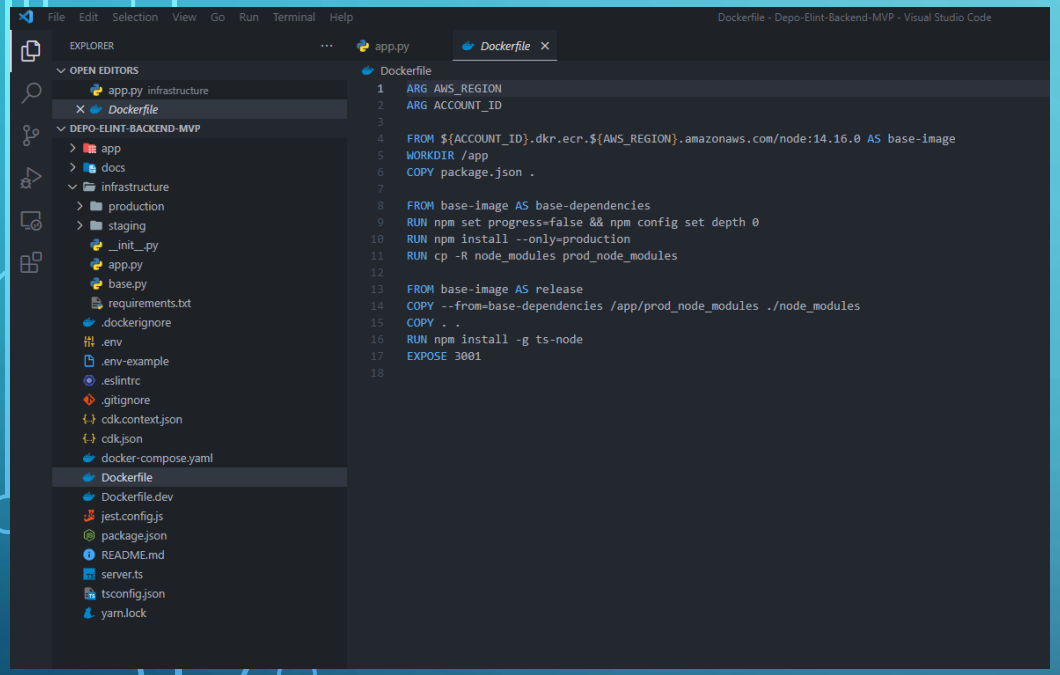
Demonstrate the tasks above can be deployed end to end on cloud (preferably on AWS)

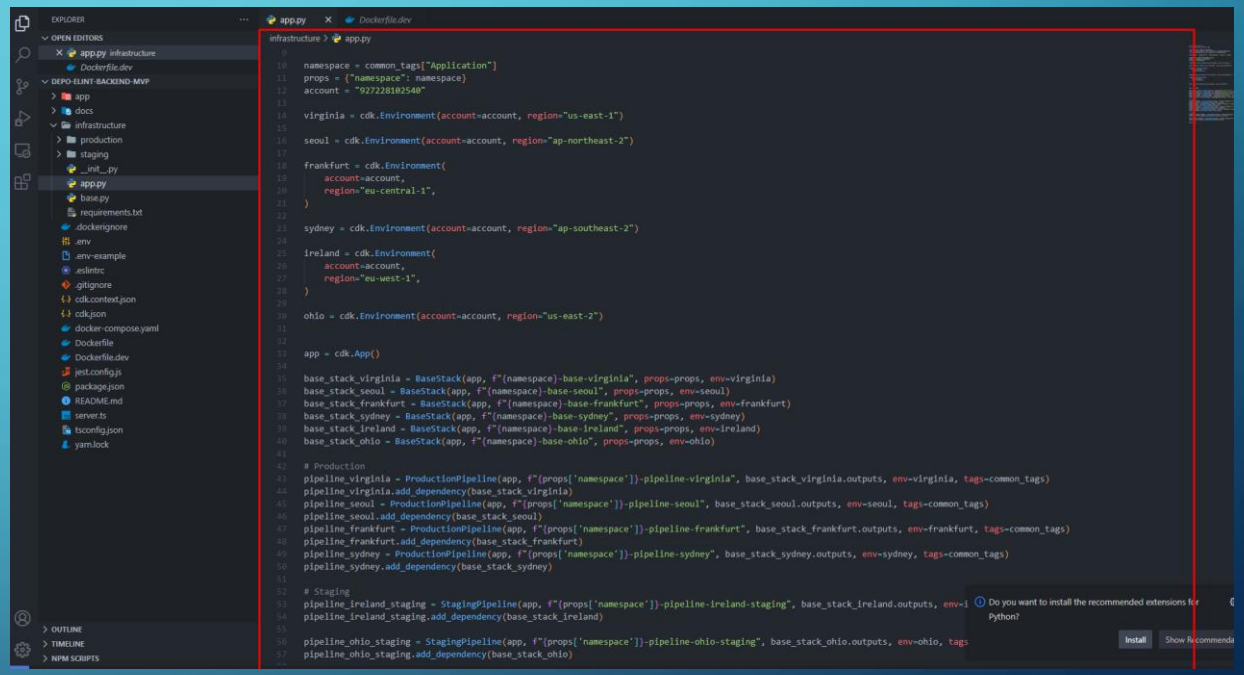
- dockerise the app
- set up CI/CD pipelines or automated unit testing
- host the app on serverless hosting on AWS e.g. container / cluster hosting services
- connect to a database that is hosted on the cloud

Part C

Deployment to cloud

I uploaded Depo-Elint-Backend-MVP project.





Part C

Bonus Task: Data Pipeline

We are also creating data pipelines, for storing audio recordings, and also other wearable device data e.g. Fitbit, Oura Ring, Apple Healthkit

- create a data processing pipeline (ETL pipeline) where data stored in a cloud DataLake (e.g. s3 bucket) is processed by an ETL pipeline (e.g. PySpark) and finally ingested into a Data Warehouse (e.g. Amazon Redshift)
- even better if the entire pipeline is deployed to the cloud, and has a serverless, autoscaling architecture

Part C

Bonus Task: Data Pipeline

IntrusionINTZ
Matthew

Dashboard
Projects
Insights
Organization Settings
Plan

CI behind your firewall just got easier

A more scalable, container friendly self-hosted runner is now in open preview

Notifications
Status OPERATIONAL
Docs

Dashboard: All Pipelines > Project: BrowserRenderer > Branch: feature/SH-3106/Create-CircleCI-file-for-Merge-Request > Workflow: pr-merge

pr-merge Success

Insights | Rerun | ...

Duration / Finished	Branch	Commit	Author & Message
8m 13s / 2d ago	feature/SH-3106/Create-CircleCI-file-for-Merge-Request	eb793b0	PASS test commit

remove_existing_images 25s
jira_comment_failed_m... 8m 8s
build_and_push_imag... 6m 49s
deploy_images_to_eks 16s

Did you know? CircleCI teams that commit 4x as often fix failed builds 2x faster. Learn more

20 pipelines/day — 70 min to recovery
5 pipelines/day — 142 min

Part C

Bonus Task: Infrastructure optimisation

Implement infrastructure features to the whole backend stack

- cybersecurity measures
- or some kind of monitoring tool / alert / disaster recovery method
- proxy servers, firewalls, load-balancers etc

Bonus Task: Infrastructure optimisation

cybersecurity measures

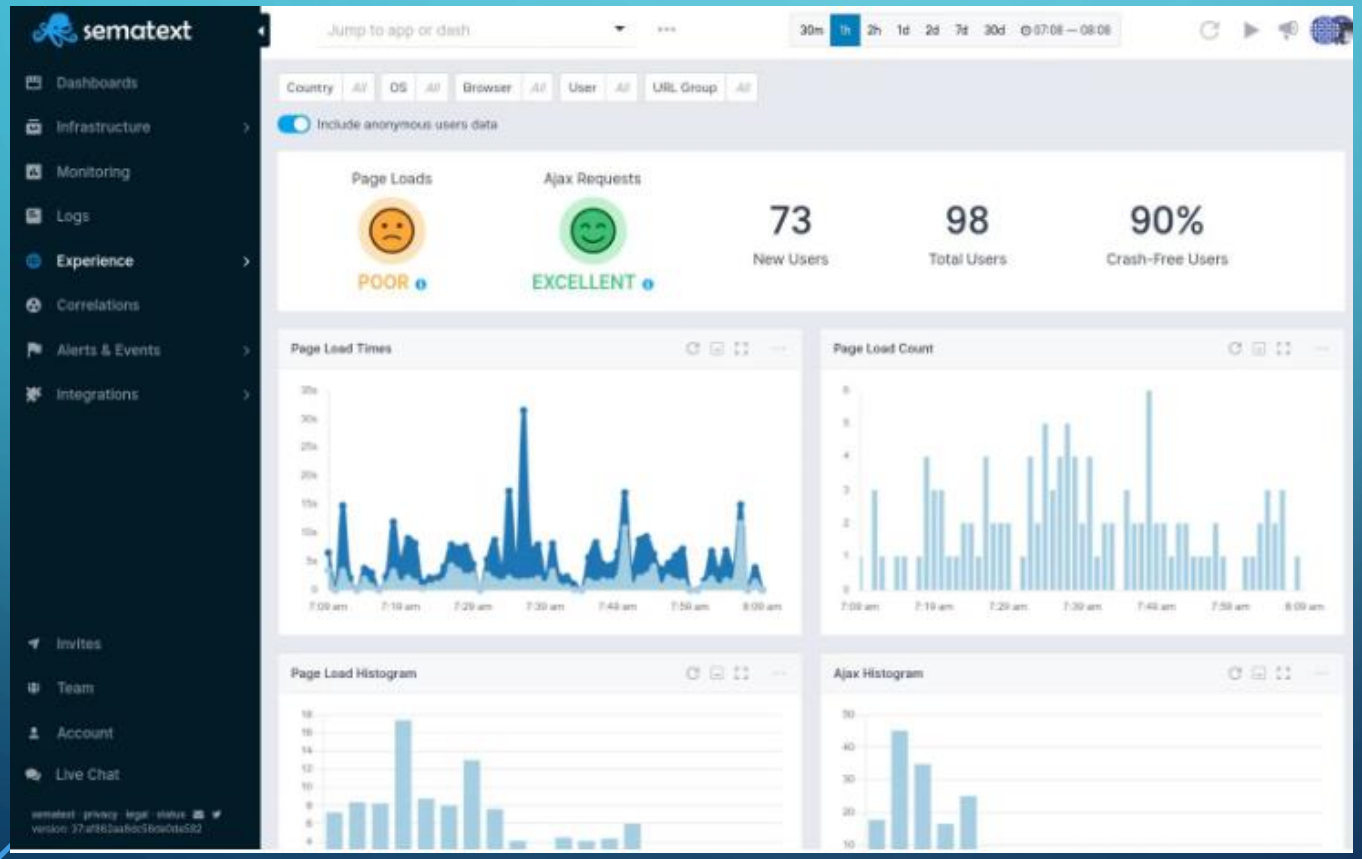
- Use strong passwords. Strong passwords are vital to good online security.
- Control access to data and systems.
- Put up a firewall.
- Use security software.
- Update programs and systems regularly.
- Monitor for intrusion.
- Raise awareness.

Part C

Bonus Task: Infrastructure optimisation

some kind of monitoring tool / alert / disaster recovery method

Sematext [freemium: free trial available]



Part C

Bonus Task: Open-ended!

If you have specific strengths in certain areas, feel free to demonstrate it. We are a company that incorporates games, data, and web3. There of opportunities to expand upon your work!

Part C

Bonus Task: Open-ended!

GHOSPERS

Home Gameplay Partners NFTs Staking Gitbook Try Demo

Welcome to Ghospers!

We're Changing the Way the World Thinks About Gaming

Immerse yourself in a fantasy world that has never existed before. Ghostalia is the world where the magic blows through the wind. From different elements to different Ghospers, there is nothing you won't find! A game for those who can't get enough of magic, strategy, and skill-based gameplay!

Explore the P2E world by fighting your way through tournaments, customizing your Ghospers, and becoming the best mage you