

A decorative graphic on the left side of the slide, consisting of a network of white lines and small circles on a dark blue background, resembling a circuit board or a neural network.

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. $\text{result} = \text{input} + 1$
- 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 X
logrequest > views.py
1  from django.http import HttpResponse
2  from django.shortcuts import render
3  import datetime
4  from .models import Score
5  import json
6
7  # Create your views here.
8
9  def index(request):
10     today = datetime.datetime.now().date()
11     return render(request, "index.html")
12  def logview(request):
13     if request.method == 'POST':
14         print(request.body)
15         username = request.POST['username']
16         m_score = int(request.POST['m_score'])+1
17         Score.objects.create(User_ID=username , Score=m_score)
18         print(m_score)
19
20     return HttpResponse(username+" "+str(m_score))
```

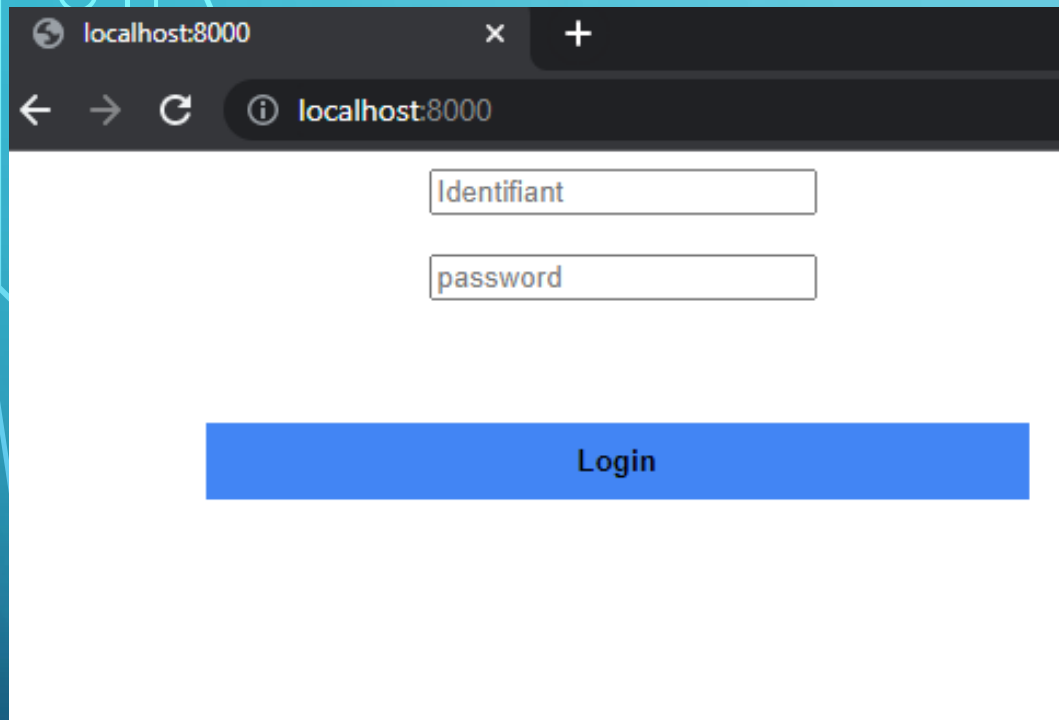
```
index.html U  urls.py U  views.py U  models.py U X
logrequest > models.py
1  from django.db import models
2
3  # Create your models here.
4  class Score(models.Model):
5     User_ID = models.CharField(max_length=20, verbose_name='User Id')
6     Score = models.CharField(max_length=20, verbose_name='Score')
7
8
```

```
index.html U  urls.py U X  views.py U
logrequest > urls.py
1  from django.urls import path
2
3  from . import views
4
5  urlpatterns = [
6     path('', views.index, name='index'),
7     path('get_score', views.logview, name='logview')
8 ]
```

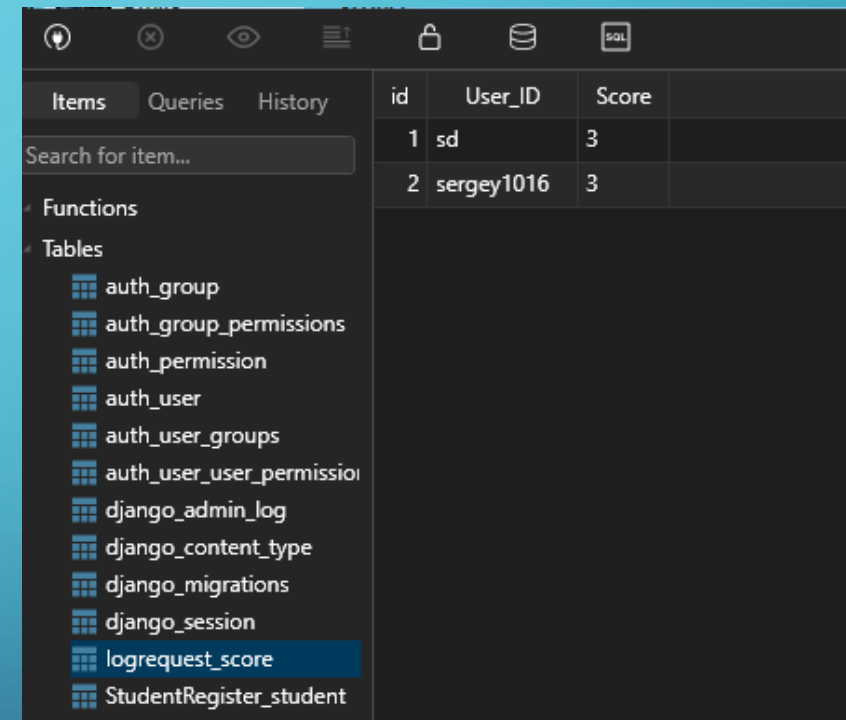
```
<html>
<body>
  <form name="form" action="./get_score" method="POST">
    {% csrf_token %}
    <div>
      <input
        type="text"
        placeholder="Identifiant"
        name="username"
      />
    </div>
    <br />
    <div>
      <input
        type="text"
        placeholder="password"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



A screenshot of a web browser window showing a login form. The address bar displays 'localhost:8000'. The form consists of two text input fields, one labeled 'Identifiant' and the other 'password', stacked vertically. Below these fields is a prominent blue button with the text 'Login' in white.



A screenshot of a database management interface. On the left, a sidebar lists various database tables, including 'auth_group', 'auth_group_permissions', 'auth_permission', 'auth_user', 'auth_user_groups', 'auth_user_user_permissions', 'django_admin_log', 'django_content_type', 'django_migrations', 'django_session', 'logrequest_score' (which is highlighted), and 'StudentRegister_student'. On the right, a table displays query results with three columns: 'id', 'User_ID', and 'Score'. The table contains two rows of data.

id	User_ID	Score
1	sd	3
2	sergey1016	3

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

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
gfd	gfd	gfd
fdsaf	fdsaf	fsda

Add New Student Info

Admission Number

First Name

Last Name

Date Of Birth

Date Joined

Faculty

Department

Course Name

Year Of Study

Unit Name

Grade

Cancel

Save Student Info

Unit Name	Grade	Actions
Economics	A	<div><div></div><div></div><div></div></div>
Complex Analysis	B	<div><div></div><div></div><div></div></div>

Add Student Info

Part A: Web2 Backend

Task 2: Admin Panel

PostgreSQL 14.5 : testdb : public.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

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
```

A decorative graphic on the left side of the slide, consisting of white lines and circles on a dark blue background, resembling a circuit board or a stylized tree structure.

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:

- 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)

← → ↻ 🏠 🔒 nft.arc.market/profile

ARC

HomeExploreCollectionsCreate

🔍 Search

🇬🇧 ENG

View Profile

Edit profile

Create

0xc3fd...66a5

Owned

Activity

Offers

Expired Listed Items

Collections (0)

☰ Filters

☰ Sort by

☰ Items status

🏷️ Price range

☰ Collections

Contact us

Subject

Partnership

ARC

HomeExploreCollectionsCreate

Search

ENG

View Profile

Edit profile

Create

ItemCollection

0xc3fd...66a5

OwnedActivityOffersExpired Listed ItemsCollections (0)

Filters

Sort by

Items status

Price range

Collections

Contact us

Subject

Partnership


Items

▼

▼


▼

Palla Radar... For Sale  **4.9**

Palla Radar... For Sale  **2.5**

Palla Radar... For Sale  **2.5**

Palla Radar... For Sale **1.75**

Palla Radar... For Sale  **1.75**

Palla Radar... For Sale **1.75**

Palla Radar... For Sale **1.75**

Palla Radar... For Sale  **1.75**

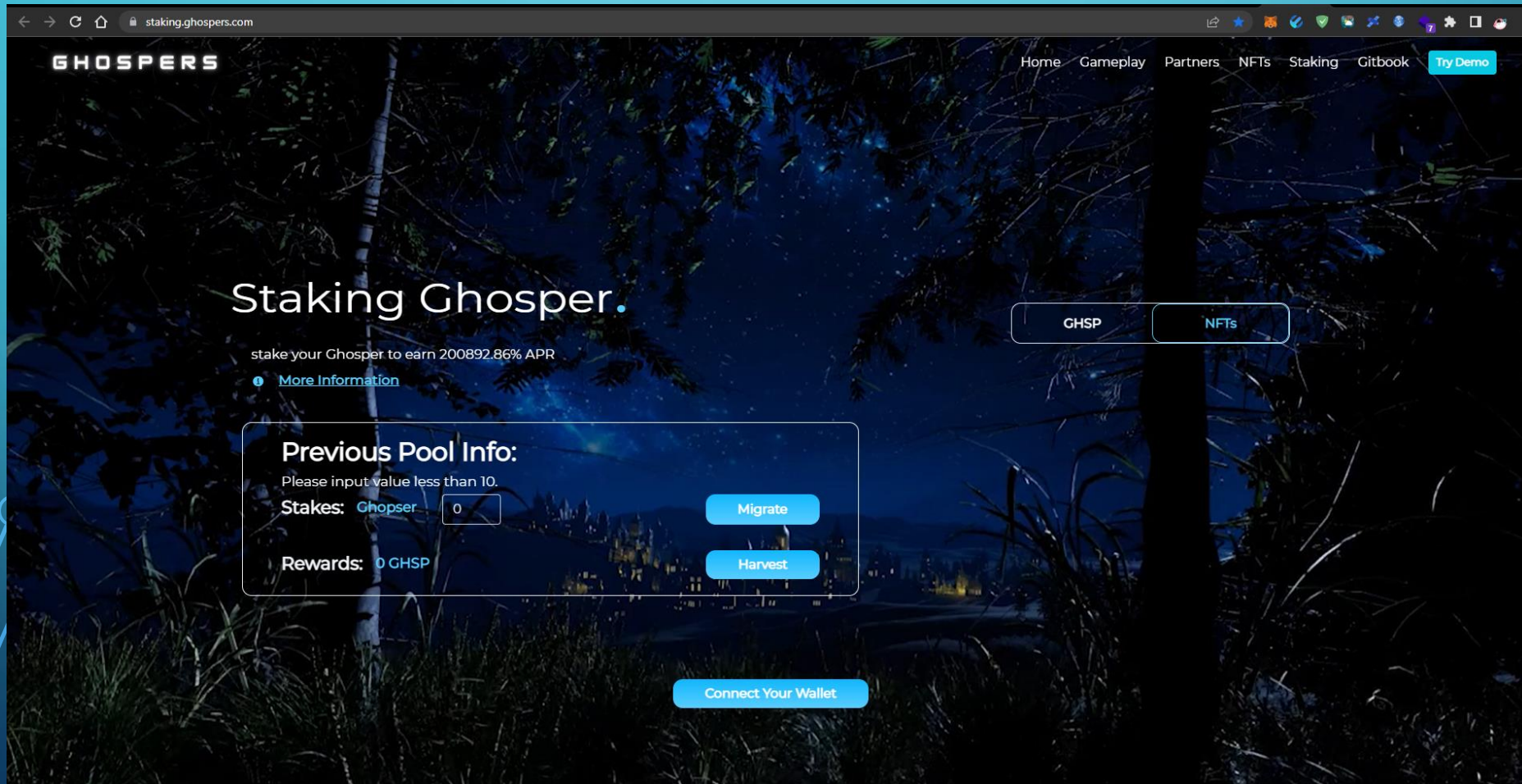
Smart Contracts

- 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

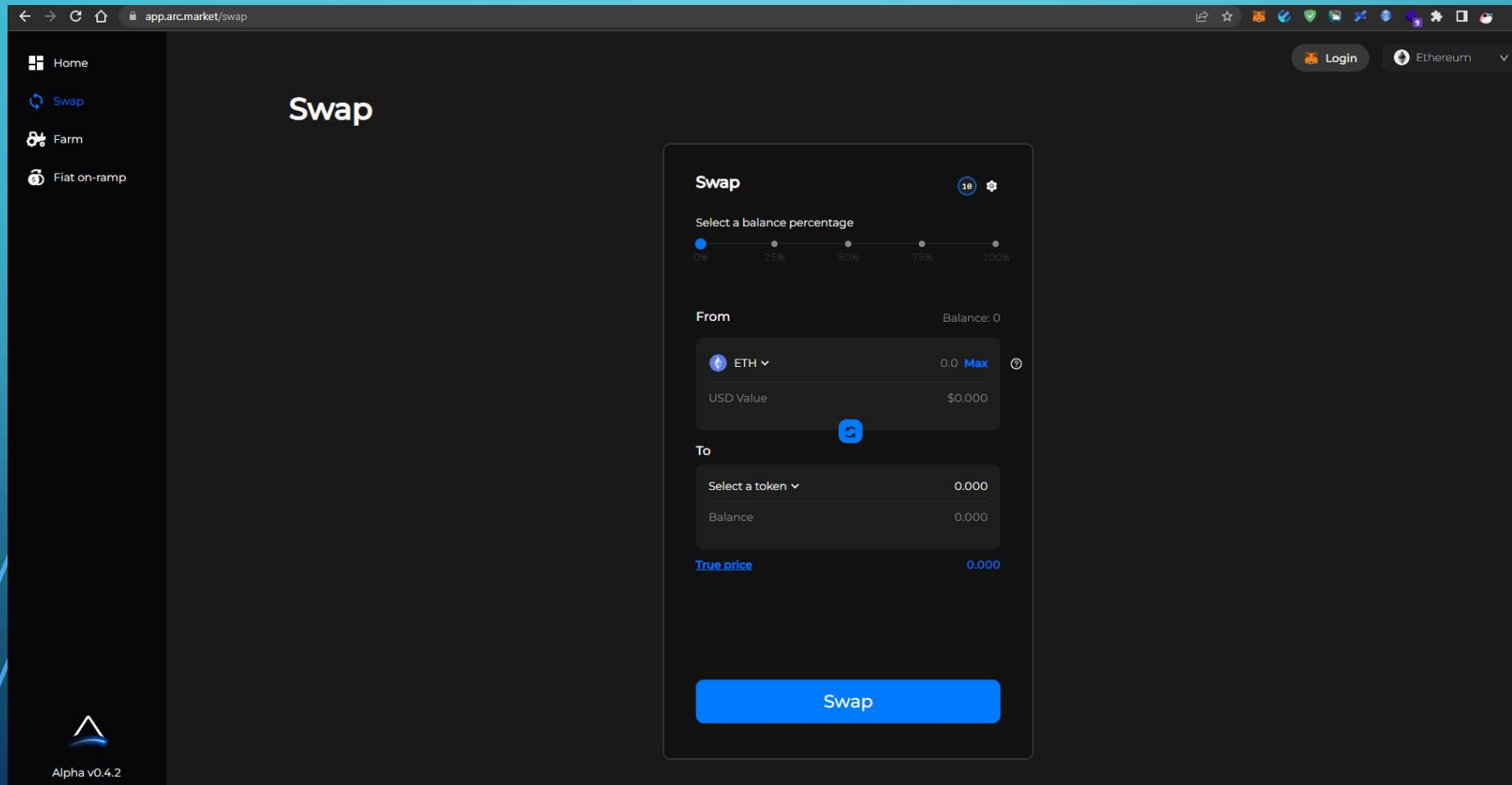
- create an ERC20 token smart contract, call it your name or a unique id such as \$JOHN123

I uploaded USDT.sol

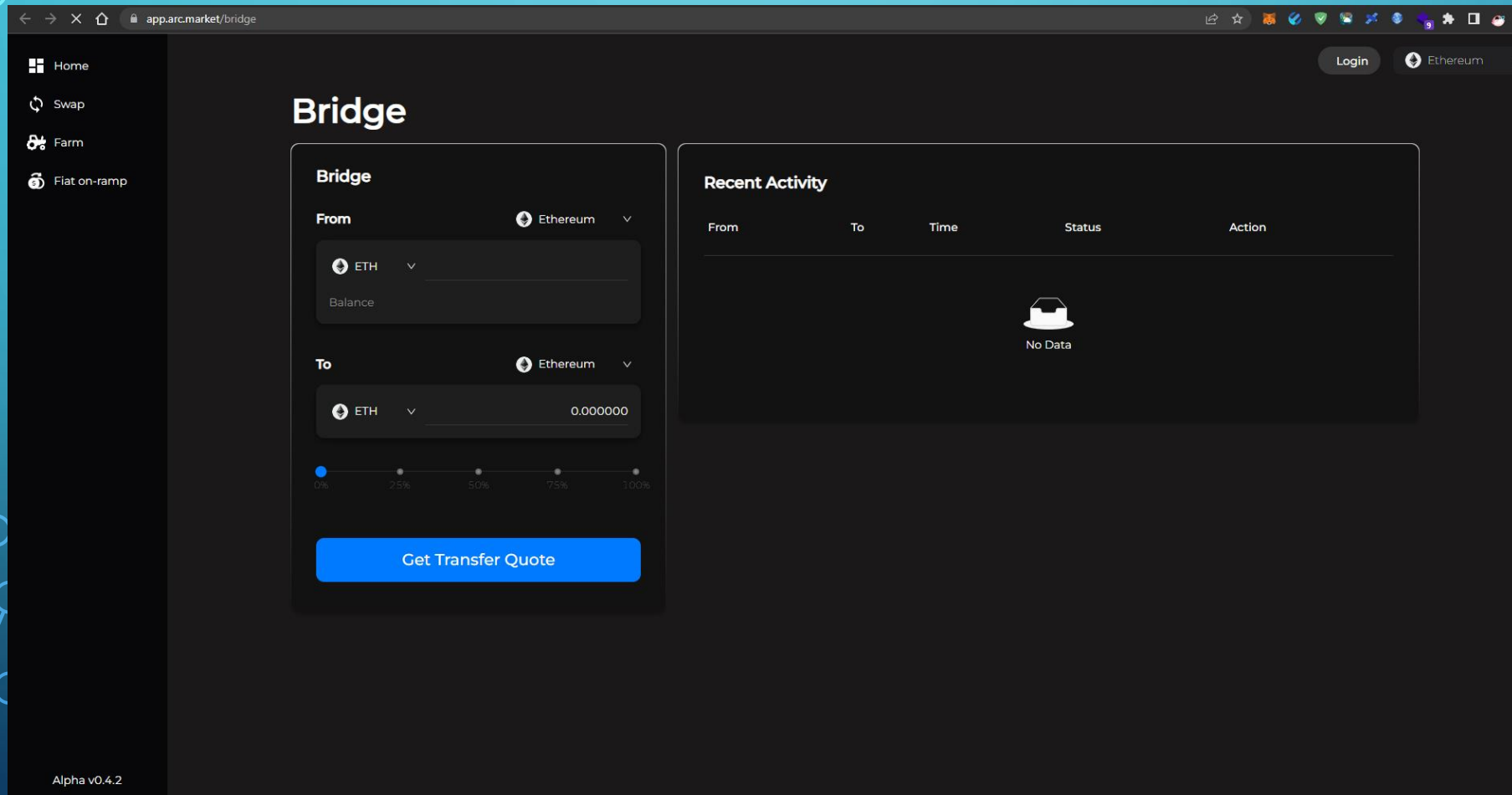
- create a staking smart contract, that is able to accept deposits of tokens and NFTs (<https://staking.ghospers.com/>)



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



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





Part C: Infrastructure and Devops

Deployment to cloud

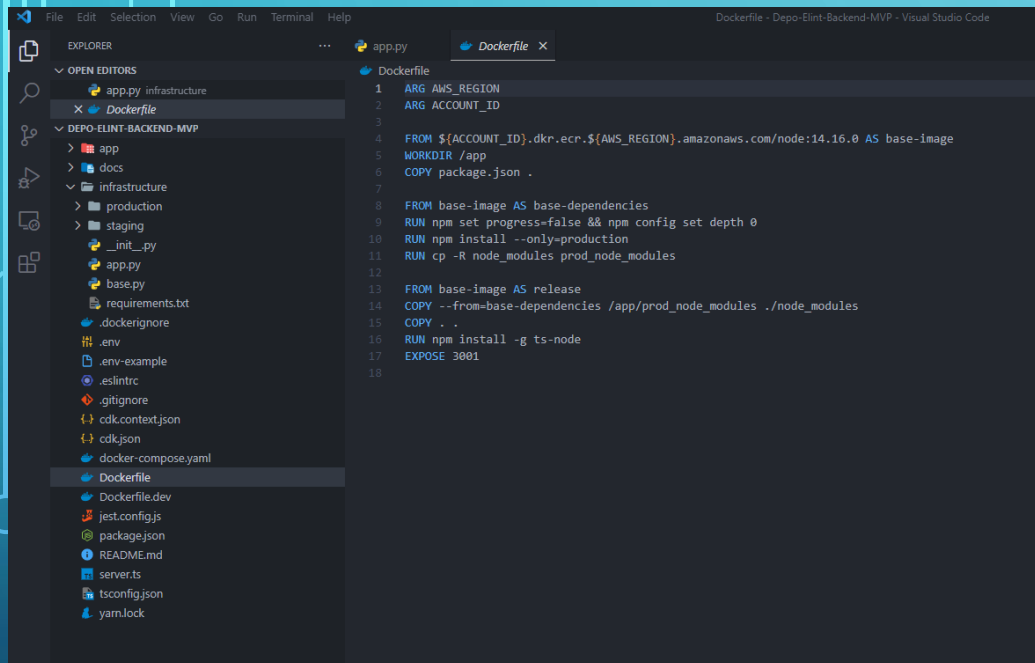
Demonstrate the tasks above can be deployed end to end on cloud (preferably on AWS)

- 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

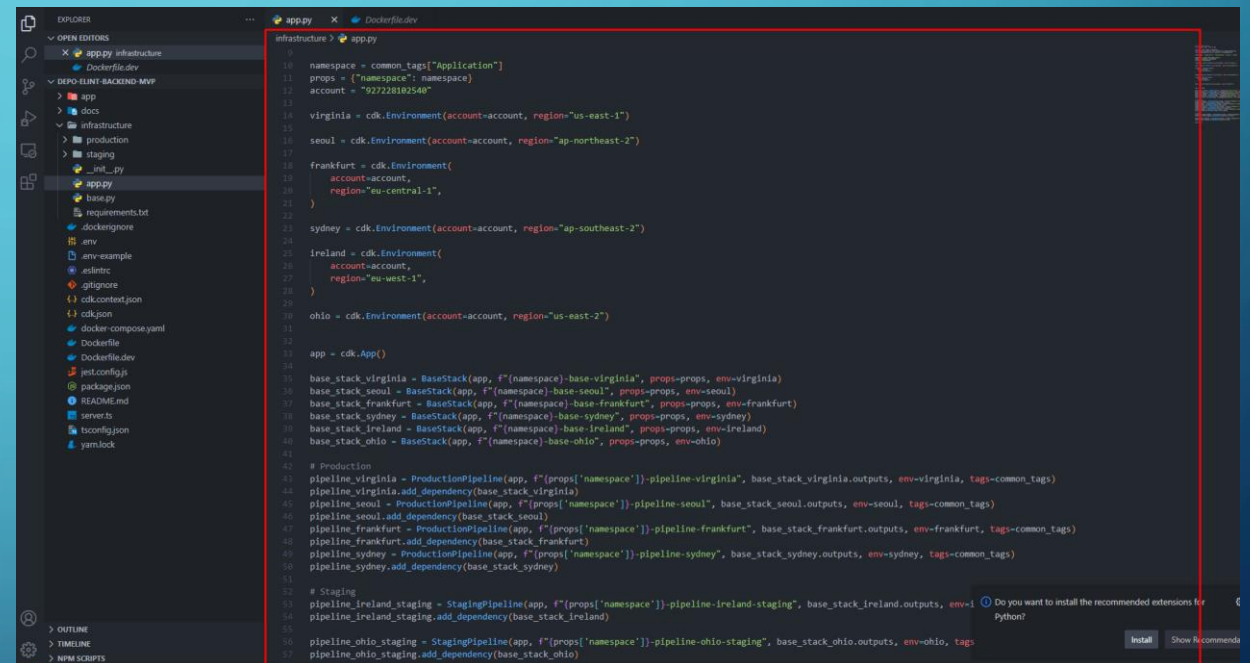
Deployment to cloud

I uploaded Depo-Elint-Backend-MVP project.



The screenshot shows the Visual Studio Code interface with the Dockerfile open. The Dockerfile contains the following instructions:

```
1 ARG AWS_REGION
2 ARG ACCOUNT_ID
3
4 FROM ${ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/node:14.16.0 AS base-image
5 WORKDIR /app
6 COPY package.json .
7
8 FROM base-image AS base-dependencies
9 RUN npm set progress=false && npm config set depth 0
10 RUN npm install --only=production
11 RUN cp -R node_modules prod_node_modules
12
13 FROM base-image AS release
14 COPY --from=base-dependencies /app/prod_node_modules ./node_modules
15 COPY . .
16 RUN npm install -g ts-node
17 EXPOSE 3001
18
```



The screenshot shows the Visual Studio Code interface with the app.py file open. The app.py file contains the following code:

```
9
10 namespace = common_tags["Application"]
11 props = [{"namespace": namespace}]
12 account = "92728192548"
13
14 virginia = cdk.Environment(account=account, region="us-east-1")
15
16 seoul = cdk.Environment(account=account, region="ap-northeast-2")
17
18 frankfurt = cdk.Environment(
19     account=account,
20     region="eu-central-1",
21 )
22
23 sydney = cdk.Environment(account=account, region="ap-southeast-2")
24
25 ireland = cdk.Environment(
26     account=account,
27     region="eu-west-1",
28 )
29
30 ohio = cdk.Environment(account=account, region="us-east-2")
31
32 app = cdk.App()
33
34 base_stack_virginia = BaseStack(app, f"{props['namespace']}-base-virginia", props=props, env=virginia)
35 base_stack_seoul = BaseStack(app, f"{props['namespace']}-base-seoul", props=props, env=seoul)
36 base_stack_frankfurt = BaseStack(app, f"{props['namespace']}-base-frankfurt", props=props, env=frankfurt)
37 base_stack_sydney = BaseStack(app, f"{props['namespace']}-base-sydney", props=props, env=sydney)
38 base_stack_ireland = BaseStack(app, f"{props['namespace']}-base-ireland", props=props, env=ireland)
39 base_stack_ohio = BaseStack(app, f"{props['namespace']}-base-ohio", props=props, env=ohio)
40
41 # Production
42 pipeline_virginia = ProductionPipeline(app, f"{props['namespace']}-pipeline-virginia", base_stack_virginia.outputs, env=virginia, tags=common_tags)
43 pipeline_virginia.add_dependency(base_stack_virginia)
44 pipeline_seoul = ProductionPipeline(app, f"{props['namespace']}-pipeline-seoul", base_stack_seoul.outputs, env=seoul, tags=common_tags)
45 pipeline_seoul.add_dependency(base_stack_seoul)
46 pipeline_frankfurt = ProductionPipeline(app, f"{props['namespace']}-pipeline-frankfurt", base_stack_frankfurt.outputs, env=frankfurt, tags=common_tags)
47 pipeline_frankfurt.add_dependency(base_stack_frankfurt)
48 pipeline_sydney = ProductionPipeline(app, f"{props['namespace']}-pipeline-sydney", base_stack_sydney.outputs, env=sydney, tags=common_tags)
49 pipeline_sydney.add_dependency(base_stack_sydney)
50
51 # Staging
52 pipeline_ireland_staging = StagingPipeline(app, f"{props['namespace']}-pipeline-ireland-staging", base_stack_ireland.outputs, env=ireland, tags=common_tags)
53 pipeline_ireland_staging.add_dependency(base_stack_ireland)
54
55 pipeline_ohio_staging = StagingPipeline(app, f"{props['namespace']}-pipeline-ohio-staging", base_stack_ohio.outputs, env=ohio, tags=common_tags)
56 pipeline_ohio_staging.add_dependency(base_stack_ohio)
57
```



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

Orbs

Dashboard

Project

Branch

Workflow

All Pipelines > BrowserRenderer > feature/SH-3106/Create-CircleCI-file-for-Merge-Request > 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

build_and_push_imag... 6m 49s

deploy_images_to_eks 16s

jira_comment_failed_m... 8m 8s

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

20 pipelines/day 70 min to recovery

5 pipelines/day 142 min

Learn more

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

cybersecurity measures

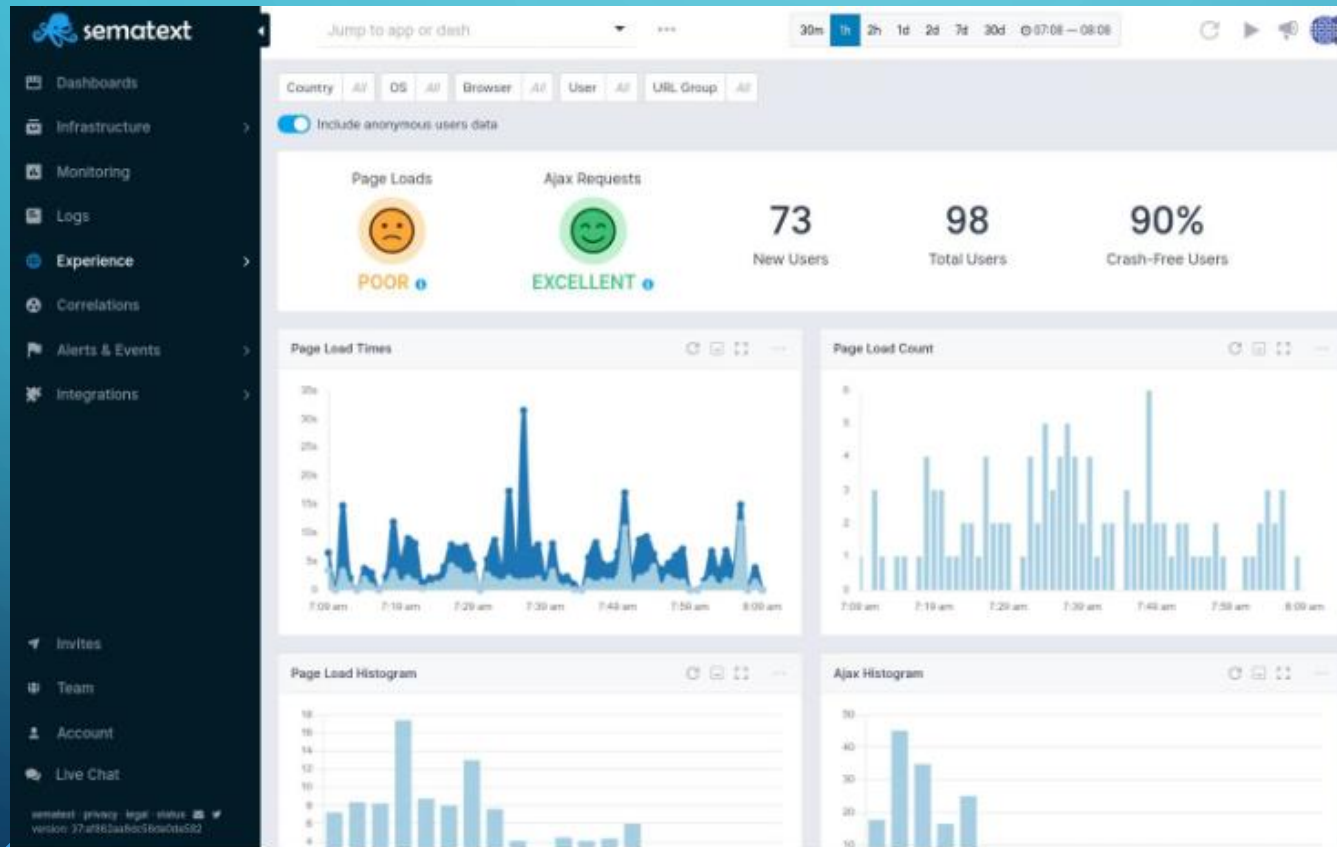
- Use strong passwords. Strong passwords are vital to good online security.
- Control access to data and systems.
- Put up a firewall.
- Use security software.
- Update programs and systems regularly.
- Monitor for intrusion.
- Raise awareness.

Part C

Bonus Task: Infrastructure optimisation

some kind of monitoring tool / alert / disaster recovery method

Sematext [freemium: free trial available]



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 are opportunities to expand upon your work!

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

