Ethereum Deposit Tracker Project

Amal Vatsa | 21BCE3194

Introduction

This presentation outlines the Ethereum Deposit Tracker project, which is designed to monitor and record deposits made to the Ethereum 2.0 Beacon Deposit Contract. The project uses Web3.py for interacting with the blockchain, Django for managing deposits, and optional features like Telegram notifications and a Grafana dashboard for alerting and visualizations.

Environmental Setup

* *Requirements*

1. Python 3.7
2. Django Framework
3. Web3
4. MySQL
5. Alchemy

* *Setup*

1. Setup terminal of Windows , Linux or MacOS [In my case using MacOS terminal].
2. Setup Python and install Django (pip install django).
3. Setup web3.py for blockchain interaction.
4. Setup Database PostgreSQL/MySQL/SQLite [In my case used SQLite].
5. Alchemy API key setup.

# *Dependencies Installation*

1. web3: `pip install web3`
2. dotenv: `pip install python-dotenv`
3. requests: `pip install requests`
4. Ensure that Django is also installed for database management: `pip install django`.

# *Configuration*

1. Create a `.env` file for securely storing sensitive information such as API keys.
2. connection details. Example configuration in `.env`:
3. ALCHEMY\_API\_KEY=your-alchemy-api-key
4. DATABASE\_URL=your-database-url

# *Usage Instructions*

1. Set up your Django project:  
   {django-admin startproject eth\_tracker\_project}  
   {python manage.py startapp tracker}
2. Apply migrations to set up the database:  
   {python manage.py migrate}
3. Start the deposit tracker by running:  
   {python manage.py runserver}

* Django Setup

1. **Models**: Define the structure of your data using Django’s ORM.
2. **Views**: Handle requests and return responses by querying the database and passing data to templates.
3. **URLs**: Map URLs to specific views.
4. **Templates**: Render HTML pages with dynamic data from views.
5. **Admin**: Use Django’s built-in admin to manage models.
6. **Environment Variables**: Securely store sensitive information like API keys.
7. **Setup .env File**

* Procedure

1. After setting up the project we need to make migrations and then migrate. After the migrations are complete we then runserver.
2. After a successful server run we can http://127.0.0.1:8000/ to see your application in action.
3. We define the app in settings.py —> tracker.
4. We setup models with all the requirement.
5. We create api key at alchemy and use the dashboard for eth observation.
6. We setup views, utils and urls.
7. We make migrations and migrate.
8. Then the last step is to runserver.

ALCHEMY DASHBORDS:

A screenshot of a graph

Description automatically generated

Application:  
A screenshot of a facebook profile

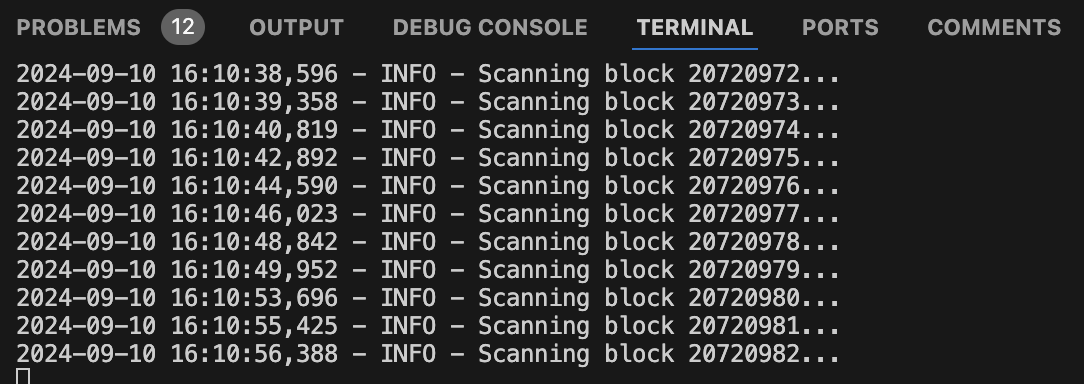
Description automatically generated

FILES CREATED:

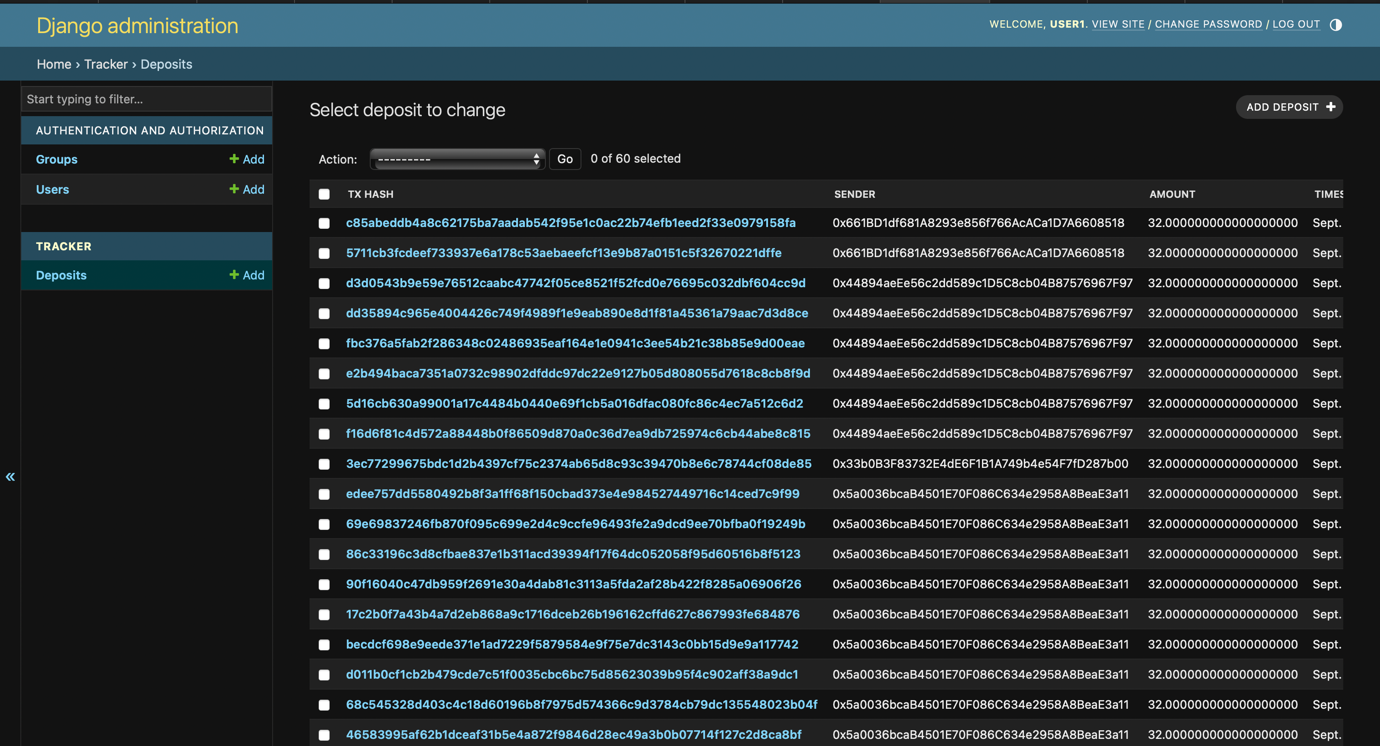
A screenshot of a black screen

Description automatically generated

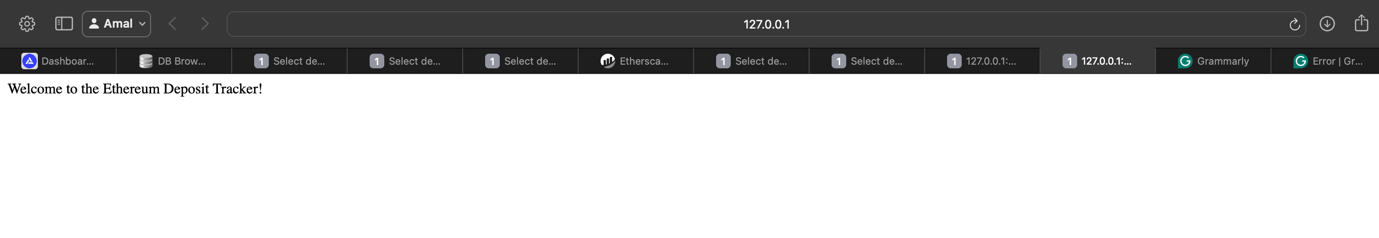
TERMINAL OUTPUT:



Django Administration[SQL DATA]---Admin



PAGE--ETH DEPOSIT RUNNING



HOME PAGE DEPLOYED 