# Catalyst Fund 11 Project Closeout Report

ID: 110024 Python Based Open Source Permissionless Marketplace and Documentation

While0x1

# Contents

# Introduction

The following report discusses the experience of completing Catalyst Fund11 Project 110024 based upon the Nescrow marketplace on the Cardano blockchain. Nescrow is a permissionless smart contract marketplace featuring python-based tooling. The smart contract and offchain code are based on Opshin and Pycardano respectively, which massively reduce barriers to entry for new developers.

# Project Details

- Name: Python Based Open Source Permissionless Marketplace and Documentation
- URL: https://projectcatalyst.io/funds/11/cardano-open-developers/python-based-open-source-permissionless-marketplace-and-documentation-a94d4
- Project ID: 110024
- Project Manager: Marc Purvis
- Project Start Date: 27/02/2024 (Official kick-off 11/03/2024)
- Project Completion Date:

Note: The research and code development for Nescrow (the underlying dApp behind the project) took roughly 6 months to complete.

# Challenge KPI's

## KPI 1

*Cardano Open: Developers (technical track) category aims to support developers and engineers to contribute to or develop open-source technology centred around enabling and improving the Cardano developer experience.*

By delivering comprehensive documentation and an entire decentralized application Project 110024 has reduced the burden on developers . With our repository under an MIT licence, developers are free to view, modify and copy our code, or parts therein for their own purposes. By delivering the first open-source project based on Opshin smart contracts and Pycardano tooling we have improved the developer experience of a massive subset of developers; namely python enthusiasts. Python is one of the most widely used and accessible languages in existence, and with our project the barrier to entry has been significantly reduced.

## KPI 2

*The goal of this category is to create developer-friendly tooling and approaches that streamline an integrated development environment, help to create code more efficiently and provide an ease of use for developers.*

Cardano's developer documentation and open-source code is often limited to just smart contract examples or tutorials, while our project has delivered the entire dApp and documentation in plain English explaining how each element works and overcomes typical

dApp design problems, like security concerns, coin-selection/UTxO management, and wallet management techniques. This tooling can be integrated into other projects easily through cloning our repository.

## Project KPI's

*Cardano has a lack of approachable open-source* **full-stack** *decentralized application code and documentation to attract new developers and makers*

Our project has created high quality outputs which comprehensively explain the entire dApp development stack. We have bridged a gap between plain-English explanations of a full-stack application and raw programming syntax. Our project could be used as a basis for new or seasoned developers wanting to bootstrap a python Cardano application.

## Key Achievements

1. We have been able to exceed our milestone commitments in quality and delivery time.
2. Provided the first full-stack open-source Cardano dApp based on Opshin smart-contracts.
3. We were early collaborators with the Opshin(formerly eopsin) project being actively involved with refining the documentation and resources on the Opshin GitHub. We have also collaborated closely with Pycardano while developing our application. I anticipate our documentation will be referenced by both Opshin and Pycardano which will enable new developers access to quality resources and increase engagement with out project.

## Key Learnings

Quality documentation increases your own understanding of your project and will reduce not only developer friction but also increases your ability to make continuous improvements. After time not looking at an aging codebase your familiarity decreases and your ability to make changes can be hindered. By creating quality full-stack documentation you can quickly reference key components and their role to reduce the time needed to create upgrades/refinements.

## Future Developments

Now that Nescrow is Open source with an issue tracker we are open to collaborations from the open-source community. We welcome open-source contributors to improve and extend our Decimals functionality and adjust the offchain code UTxO management systems to reduce wallet error edge-cases. Nescrow has been in talks with other projects with the possibility of integrating the Nescrow smart contract into their project offerings. Nescrow also begun development on a permissionless batcher algorithm.

## Developer Recommendations

1. Integrate Nescrow with your existing service as is by importing the Listing type and the Redeemers from the contract.

2. Improve on the UI – create an entirely new UI for Nescrow. We have used minimal to no CSS in the application so there is great room for improvement of the UI/UX design of Nescrow.
3. The Nescrow contract has been made completely portable – you could take our contract and setup your own marketplace without having to have ANY smart contract knowledge. Simply change the fee address and fee levels and create your own Marketplace!

# Dependencies

## Next.JS(13.2.4)

- “@emotion/react": "^11.10.6",
- "@emotion/styled": "^11.10.6",
- "@mui/icons-material": "^5.11.16",
- "@mui/material": "^5.11.16",
- "@types/node": "18.15.11",
- "@types/react": "18.0.33",
- "@types/react-dom": "18.0.11",
- "cbor": "^8.1.0",
- "eslint": "8.37.0",
- "eslint-config-next": "13.2.4",
- "next": "13.2.4",
- "react": "18.2.0",
- "react-dom": "18.2.0",
- "typescript": "5.0.3"

## Python 3.10.6

Package Version

---------- ----------

- bcrypt 3.2.0
- blockfrost-python 0.5.3
- cbor2 5.4.6
- certbot 1.21.0
- certbot-nginx 1.21.0
- cryptography 3.4.8
- Flask 2.2.3
- Flask-Cors 3.0.10
- gunicorn 20.1.0
- httplib2 0.20.2
- jsonpatch 1.32
- jsonpointer 2.0
- jsonschema 3.2.0
- opshin 0.12.1
- pycardano 0.8.1
- PyNaCl 1.5.0
- python-secp256k1-cardano 0.2.3

# Deployment Guide

To clone or to adapt this project you will need to install Next.JS, Node or (NVM), and python3. As most modern Ubuntu systems come preinstalled with python, we will briefly cover the setup of the Next.JS / nvm procedure. I would recommend first setting up a new Next.JS project, then replacing the index.tsx file with the file from the repository.

## NVM/Node

Install Node Version Manager so you can download a specific version of Node JS.

Follow the guide here to download NVM:
https://www.digitalocean.com/community/tutorials/how-to-install-node-js-with-nvm-node-version-manager-on-a-vps

Once nvm is installed use command: nvm ls-remote

This will display the node versions available

install the same node version used in this project: nvm install 16.7.0

*note as multiple nextJS releases have taken place you may need to install node version compatible with updated nextJS: nvm install 18.17.0

## Next.JS

We can create a next.JS application by following the guide here -
https://nextjs.org/docs/pages/api-reference/create-next-app

On your development Machine:

```
mkdir dex_nescrow
cd dex_nescrow
mkdir nescrow_dex_app
cd nescrow_dex_app
npx create-next-app@latest --typescript .
#follow prompts Yes,No,Yes,No,No

cd ..
git clone https://github.com/while0x1/Nescrow-Catalyst.git
mv Nescrow-Catalyst/flask flask
mv Nescrow-Catalyst/v3_script/ v3_script

#in nescrow_dex_app/src/pages directory
mv _app.tsx bk_app.tsx
mv _document.tsx bk_document.tsx
mv index.tsx bk_index.tsx

#in dex_nescrow directory
mv Nescrow-Catalyst/nescrow_dex_app/src/pages/index.tsx
nescrow_dex_app/src/pages/index.tsx
mv Nescrow-Catalyst/nescrow_dex_app/src/pages/_app.tsx
nescrow_dex_app/src/pages/_app.tsx
```

mv Nescrow-Catalyst/nescrow_dex_app/src/pages/_document.tsx
nescrow_dex_app/src/pages/_document.tsx

#in dex_nescrow/nescrow_dex_app
npm install @mui/material
npm install @mui/icons-material
npm install cbor
npm install @emotion/react
npm install @emotion/styled

#in ~/dex_necsrow/nescrow_dex_app/src/styles
mv globals.css bk_globals.css
mv Home.module.css bk_Home.module.css
#from /dex_necsrow
mv Nescrow-Catalyst/nescrow_dex_app/src/styles/globals.css
nescrow_dex_app/src/styles/globals.css
mv Nescrow-Catalyst/nescrow_dex_app/src/styles/Home.module.css
nescrow_dex_app/src/styles/Home.module.css
#from nescrow_dex_app folder
npm run dev

```
cardano@precision:~/testproject$ npx create-next-app@latest --typescript .
✔Would you like to use ESLint? … No / Yes
✔Would you like to use Tailwind CSS? … No / Yes
✔Would you like to use `src/` directory? … No / Yes
✔Would you like to use App Router? (recommended) … No / Yes
✔Would you like to customize the default import alias (@/*)? … No / Yes
Creating a new Next.js app in /home/cardano/testproject.

Using npm.

Initializing project with template: default


Installing dependencies:
- react
- react-dom
- next

Installing devDependencies:
- typescript
- @types/node
- @types/react
- @types/react-dom
- eslint
- eslint-config-next

(                    ) ⁞ idealTree:eslint-plugin-import: timing idealTree:node_modules/eslint-plugin-import
```

## Environment

Create a .env.local file in dex_nescrow/nescrow_dex_app folder

NEXT_PUBLIC_FLASK_AUTH = <your_flask_api_key>

## Parameterise Flask

#Replace with Your API password and contract parameters in
~/dex_nescrow/flask/static_vars.py

FLASKAUTH = '<your_auth>'
FEE = '<your_fee>'
CANCEL_FEE = '<your_fee>'

```
FEEHASH = '<your_fee_hash>'
#Replace MAINNET WithYourBlockfrostKeys
BF_HEADERS = {'project_id': '<your_key>'}
BF_PROJ_ID =  '<your_key>'
FEE_ADDRESS = '<your_fee_address>'
#Replace PreProd
BF_HEADERS = {'project_id': '<your_preprod_key>'}
BF_PROJ_ID = '<your_preprod_key>'
#If Contract has changed updater REF_CBOR to your contracts CBOR reference utxo

#once your blockfrost keys and environment static variables are set from ~/dex_nescrow/flask and your contract or the original contract is referenced you can install dependencies and run the back-end
#dependencies
pip3 install certbot certbot-nginx flask Flask-Cors opshinin==0.12.1 pycardano==0.8.1
python3 flask_nescrow_dex.py
```

# Final Thoughts

Nescrow is the currently the cheapest hybrid marketplace and escrow service which would benefit from more users to be able to refine the site and increase adoption while delivering a secure and quality service to the Cardano Community.

Project Catalyst has been instrumental in allowing Nescrow to continue to function and for that I am incredibly grateful.

# Links

https://nescrow.xyz

https://github.com/while0x1/Nescrow-Catalyst/tree/main

# Closeout Video

https://youtu.be/Gjptriyzkjs