Pusan National University
Computer Science and Engineering
Technical Report 2022-0X

업비트 API 기반 디지털 가상자산 거래소 개발 및 자체적 코인 발행 최종보고서

지도교수 김호원
팀명 언체인(UN-CHAIN)
조원1 201624526 유상욱
조원2 201624521 용태완
분과 D.지능형 융합보안

목 차

1. 서론

1.1 연구 배경

최근 몇 년간 비트코인 및 여러 종류의 알트코인이 등장하였고 언론에서도 많은 주목을 받고 있다. 이에 따라 암호화폐 투자에 관심을 가진 사람들이 계속해서 늘어나고 있으며, 가상화폐의 실용화를 논하게 된 2016년 이래로 코인의 사용처는 계속 늘어나고 있다. 비트코인을 법정화폐로 쓰는 나라도 생겼고, 기부금 납부, 음식점, 영화관, 교육기관 등록금 등 다양한 실생활에 사용되고 있다. 새로운 가상화폐의 종류가 늘어나고 사용처도 늘어나는 만큼, 가상화폐 거래소 또한 늘어나는 추세이다. 실제로 국내 주요 암호화폐 거래소 중 하나인 '업비트'는 2022년 현재 수수료 수익이 일일 약 100억원에 달할 만큼 큰 규모가 되었다.

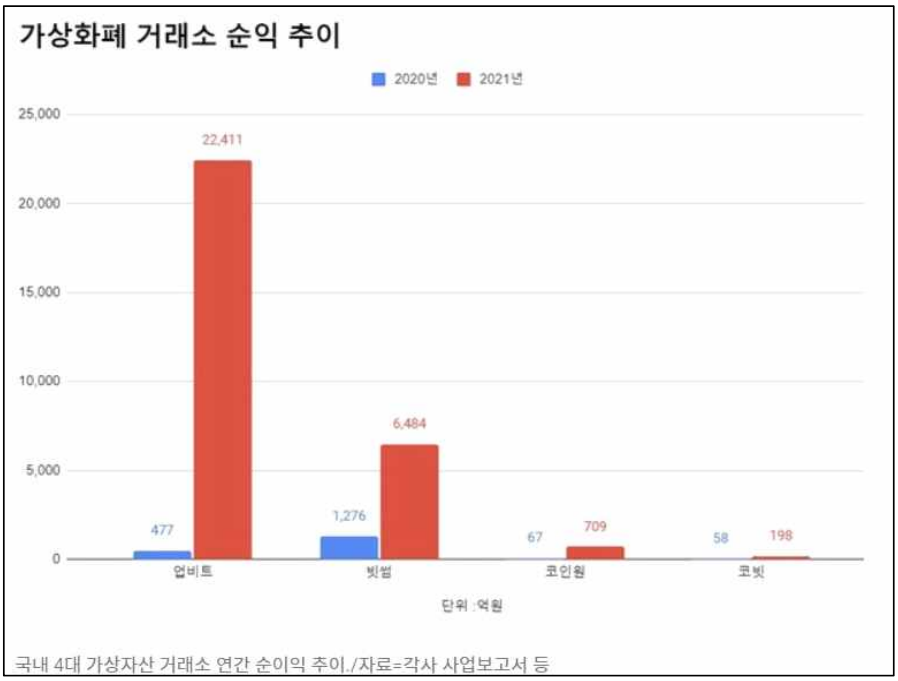


그림 1. 21년 가상화폐 거래소 순익 추이

가상화폐가 상용화 되고있는 만큼, 이에 관련된 지식이나 기술이 다수 필요하게 될 것으로 보인다. 따라서 암호화폐를 직접 만들거나, 거래소를 만들 수 있는 능력은 훌륭한 경쟁력이 될 것이다.

1.2 기존 문제점

기존의 가상화폐 거래소는 코인 관련 정보가 짧게 요약되어 있고, 관련 기사 또한 많이 찾아볼 수 없다. 코인이라는 자산을 사러 왔지만, 그것에 대한 정보는 차트밖에 없는 것이다. 그래서 사람들은 정보를 얻기 위해 각종 유료/무료 리딩방, 코인 유튜브 등으로 찾아가게 된다. 그들이 공개하는 수익은 매우 높고, 승률이 높아 손해를 보는 날이 없다고 주장한다. 하지만 우리는 어떤 방식으로 그런 기록들을 만들었는지 알 수 없다. 실제로 거래를 해서 생긴 수익인지, 거래 수량, 평단가 등을 수정 할 수 있는 매매내역인지, 다계정으로 계정마다 다른 종목을 투자해 그중에서 수익률이 가장 높은 것만 공개한 것인지 우리는 모른다. 거래소가 나서서 중재하지 않기 때문에 이 정보들은 진실인지 이용자가 투명하게 확인할 방법이 없다.

1.3 연구 목표

기존 문제점은 거래소에서 확인할 수 있는 정보 부족, 거래소가 중재하지 않는 매매내역의 불확실성으로 요약할 수 있다. 이를 보완하기 위해 뉴스를 통한 코인 여론 분석 기능을 개발하고, 거래소 내에서 사용할 수 있는 코인과 거래소 유저의 매매내역을 교환할 수 있는 기능을 만들 것이다.

그리고 거래 중개 사이트(업비트)에서 제공하는 API를 받아 각종 코인의 차트, 호가창, 코인의 시세 파악 및 매도, 매수 기능을 거래소 웹사이트에 구현하고 웹사이트를 운영하는 방식으로 거래소를 만든다. 또한, 거래소 구현 후 이더리움을 기반으로 하여 자체적으로 코인을 발행하고 역할 부여 후 거래소에 등록한다.

2. 연구 배경

2.1 연구 개발 환경

1. Vscode

웹사이트 개발 및 사이트에 게재할 내용을 계산하는 python 코드를 실행하는데 VScode를 사용하였다.

2. React

중간 진행까지 vanilla js를 이용해 사이트를 구성하고자 했지만, 자료를 찾

아보는 과정에서 일반 js를 사용한 프론트엔드보다 react를 활용한 것이 많아 중간 보고서 작성 이후 학습하면서 교체했다. vue.js를 써보려고도 했으나, html, js, css를 분리해서 사용해야 하는 vue.js에 비해 주로 jsx 형태로 작성되는 react가 더 편리하게 느껴졌다. 또한 프론트에서 가장 많이 활용하는 라이브러리를 사용해 보고 싶어 선정하게 되었다.

3. 기타 라이브러리

○ recoil

전역 상태를 관리하기 위한 라이브러리이다. 사용자의 인증 상태와 코인의 정보를 전역으로 관리해서 상태가 변경 될 때 모든 페이지에서 변경된 상태가 적용되도록 한다. 또 다른 상태 관리 라이브러리인 redux의 경우 action과 reducer를 사용한 동작이 강제되기 때문에 코드가 복잡해지는데 반해 recoil은 React의 State처럼 핸들링이 가능하고 코드가 좀더 간단하게 작성된다는 장점이 있어서 채택했다.

○ axios

브라우저와 js간 사용하는 http통신 라이브러리이다. 비동기로 http통신을 할 수 있으며 해당 프로젝트에서 업비트 api등을 연동하는 데 사용하였다. fetch api와 비교하여 사용이 더 간결하다고 판단하였기에 axios를 채택하게 되었다.

○ use-upbit-api

프로젝트 내에서 업비트의 실시간 오더북 데이터를 받아오고 차트를 그리는 과정에서 사용하기 위해 해당 라이브러리를 채택하였다.

3. 연구 내용

3.1 코인 생성 및 역할 부여

3.1.1 코인 생성

웹 코인 지갑인 metamask를 생성 후 [1]이더리움 사이트에서 제공한 코인 제작 방법을 통해 테스트용 이더리움 토큰을 생성했다. 상세 방법은 다음과 같다.

1) https://ethereum.org/en/developers

해당 프로젝트에서는 이더리움 사이트에서 제공하는 오픈소스를 활용해 자신이 만들 코인을 제작한다. 이름은 TDC코인으로, 발행량은 10만 개, 내 지갑 주소로 들어오게끔 했다.

```
constructor() public {
    symbol = "TDC";
    name = "TRADING TOKEN";
    decimals = 2;
    _totalSupply = 10000000;
    balances[0xd64Dc175606fAe83F9412b44298CC1632A6Fc979] = _totalSupply;
    emit Transfer(address(0), 0xd64Dc175606fAe83F9412b44298CC1632A6Fc979, _totalSupply);
}
```

그림 2. TDC코인 발행 코드

그 후, http://remix.ethereum.org/에 접속해 코드를 업로드하고 코인을 발행한다. metamask에 나의 erc20 토큰을 추가하고 발행된 것을 확인 후, 이를 다른 metamask 계정에 송금한 후, 블록체인 정보를 확인할 수 있는 이더스캔에서 다시 확인한다.

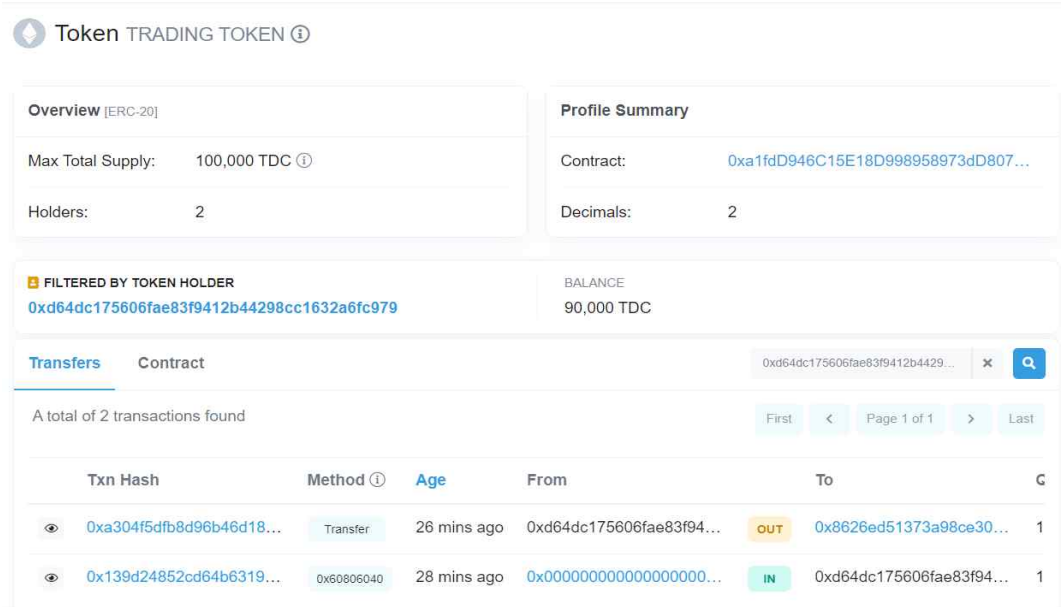


그림 3. 이더스캔에서 조회한 TRADING TOKEN의 정보, 거래내역 조회

3.1.2 코인 역할 부여-트레이딩 코인(TDC)

거래소 내에서 사용할 수 있는 코인(TDC)와 거래소 유저의 매매내역을 교환할 수 있는 기능을 만든다. metamask를 이용하여 코인을 전달할 수 있으며. 이후 페이지의 관리자가 이더스캔을 통해 TDC 전달을 확인한 후, 매매내역을 전달하는 방식을 이용할 것이다.

3.1.3 TDC 페이지

자체발행 코인인 TDC를 거래하는 페이지이다. 해당 과정에서 'ethers'모듈을 사용하였으며 관련된 코드는 아래와 같다.

```
const handleProfileClick = (item ) => () => {
 formRef .current .setValue ("receiver", item .id );
 formRef .current .setName (item .name );
 updateEthers ();
};
```

그림4. 프로필을 클릭하여 지갑 주소를 등록하는 코드

```
const handleConnectWallet = async () => {
 if (window .ethereum &&window .ethereum .isMetaMask ) {
  const [value ] = await window .ethereum .request ({
   method :"eth_requestAccounts",
  });
  formRef .current .setValue ("sender", value );
 } else {
  toastRef .current .open ();
 }
};
```

그림 5. Metamask를 통해 본인의 지갑 주소를 등록하는 코드

```
const handleSubmit = ({ message }) => {
 contract .set (message );
};
// reload the page to avoid any errors with chain change mid us
e of application
const chainChangedHandler = () =>window .location .reload ();
```

그림 6. TDC 코인 거래 팝업을 띄우는 코드

상위 트레이더들을 선정하여 화면에 띄워주고, 마음에 드는 트레이더의 '주소 선택'버튼을 누르면 TDC 전송 폼에 해당 트레이더의 지갑 주소가 등록된다. 이후 Connect를 통해 크롬 브라우저 확장프로그램 Metamask를 실행시켜 본인의 지갑 주소를 폼에 등록한다. 이후 contract 메시지를 자유롭게 작성한 뒤 전송 버튼을 누르면 Metamask를 통해 TDC를 전송할 수 있다. 해당 과정을 통해 거래를 진행한 내용은 메타마스크의 활동 -> 블록 탐색기에서 보기 또는 이더스캔으로 직접 접속하여 확인할 수 있다.

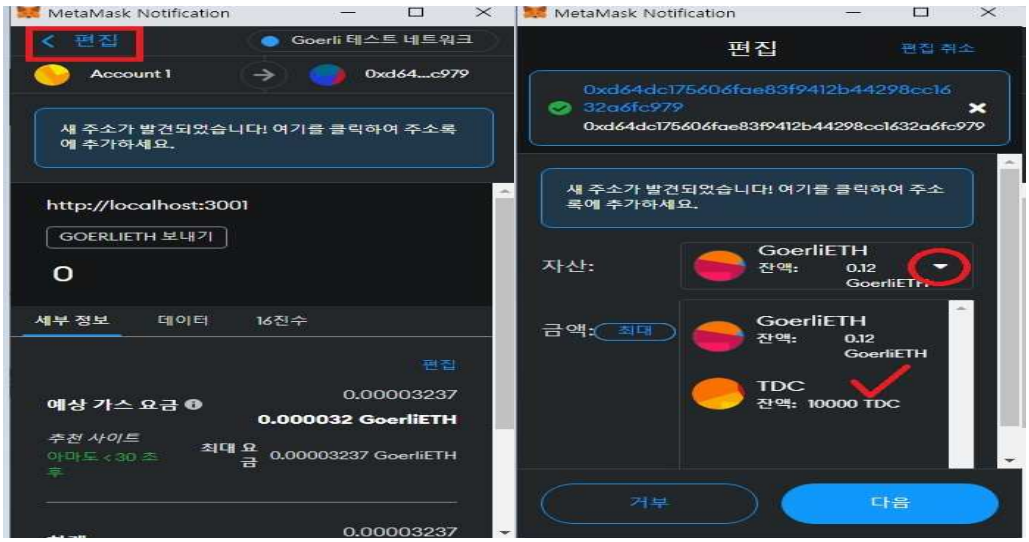


그림 7,8. Metamask TDC코인 거래 과정

해당 과정에 대한 순서 설명은 TDC 거래 페이지의 우측 하단에 위치한 도움말 버튼을 클릭하면 웹페이지 내에서도 열람할 수 있다.

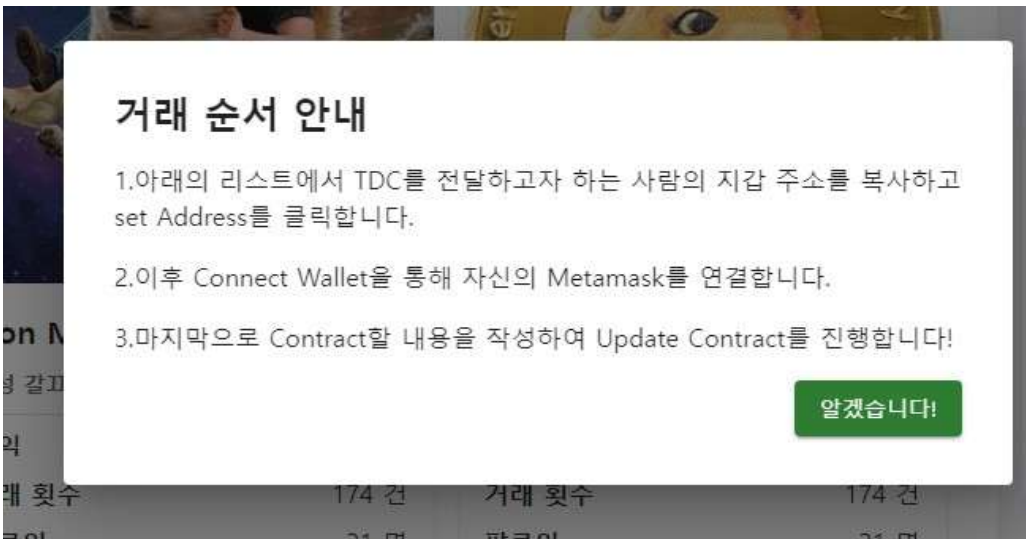


그림 9. TDC코인 거래 순서 안내문구

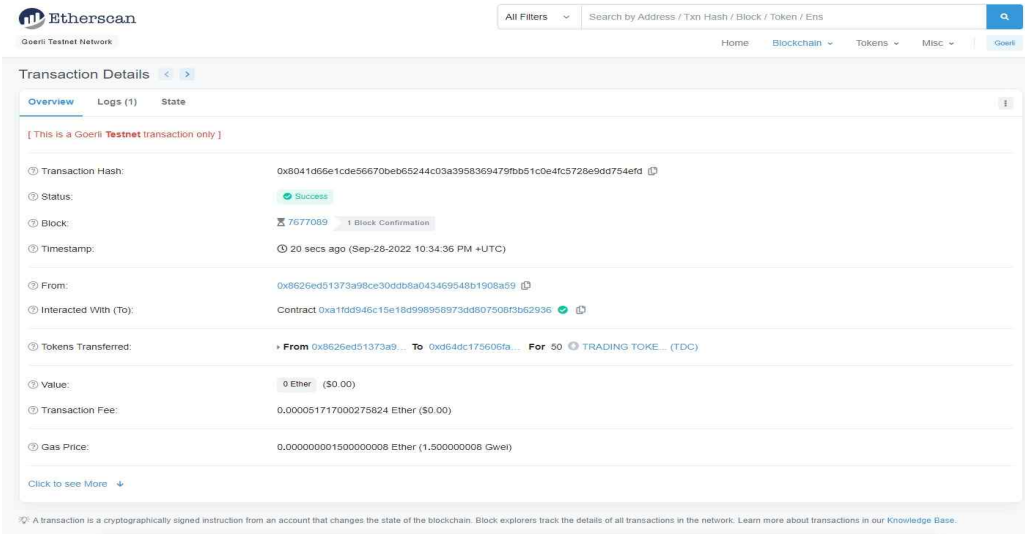


그림 10. TDC코인을 전달한 이더스캔 기록

현재 거래소 내에서 metamask와 연동 후 TDC를 전달하고 개인이 전달한 내역을 확인할 수는 있지만 설계 시 계획했던 TDC의 기능인 트레이더의 거래 내역을 전달하는 과정이 구현되어있지 않다. 후속 연구로 자체적인 DB를 서버와 연동한 이후에는 DB에 저장한 기록을 사용자에게 전달하여 개인의 마이페이지 내에서 열람할 수 있게끔 할 예정이다.

3.2 코인 동향 페이지

3.2.1 기사 여론 분석

python을 통해 특정 키워드에 해당하는 뉴스 기사의 제목을 분석하여, 해당 키워드에 대한 여론이 긍정적인지 부정적인지 평가하는 기능을 개발했다. 현재 두 가지 키워드 비트코인, 이더리움을 검색한 자료를 통해 각각의 모델을 생성하였으며, 이를 keras를 통해 학습시켰다. 해당 코드는 지정한 키워드를 해당 여론이 긍정적인지, 부정적인지, 중립적인지의 동향을 neg, neau, pos값으로 나누어 csv파일로 생성하고 해당 csv 파일을 통해 그래프 이미지를 생성한다.

```
import requests
from bs4 import BeautifulSoup
import re
import pandas as pd
from tqdm import tqdm
import matplotlib.pyplot as plt
import time
import konlpy
from konlpy.tag import Okt
import numpy as np
import keras
from keras.layers import Embedding, Dense, LSTM
from keras.models import Sequential
from keras_preprocessing.sequence import pad_sequences
```

그림 11. 기사 분석에 사용한 모듈 및 라이브러리 목록

```
#비트코인----------------------------------------------------------------------------
bit_titles = []
bit_urls = []
keyword = "비트코인"
#pageNum = 1
for num in range(1, 70, 10):
  #print(f"{pageNum}페이지입니다.----------------")
  # pd 4 = 1일, pd 1 = 일주일, pd 2 = 1달
  response = requests.get(f"https://search.naver.com/search.naver?where=news&sm=tab_jum&query={keyword}&pd=4&start={num}")

  html = response.text
  soup = BeautifulSoup(html, 'html.parser')
  links = soup.select(".news_tit")
  for link in links:
    title = link.text
    url = link.attrs['href']
    clean_title = re.sub('[-=+,#/\?:^$.@*\"※~&%ㆍ·!』\\‘|\(\)\[\]\<\>`\'…\"\“”》]', '', title)
    #print(clean_title,"\n", url, "\n")
    bit_titles.append(clean_title)
    bit_urls.append(url)
  #pageNum = pageNum + 1

positive = []
negative = []
bit_labels = []

with open("./words/negative_words_self.txt", encoding='utf-8') as neg:
  negative = neg.readlines()

negative = [neg.replace("\n", "") for neg in negative]

with open("./words/positive_words_self.txt", encoding='utf-8') as pos:
  positive = pos.readlines()

negative = [neg.replace("\n", "") for neg in negative]
positive = [pos.replace("\n", "") for pos in positive]
```

```
label = [0] * 4000

bit_title_dic = {"title":[], "label":label}

for title in tqdm(range(len(bit_titles))):
  negative_flag = False
  neutrality_flag = True

  label = 0
  for i in range(len(negative)):
    if negative[i] in bit_titles[title]:
      label = label-1
      negative_flag = True
      neutrality_flag = False

  for i in range(len(positive)):
    if positive[i] in bit_titles[title]:
      label = label + 1
      neutrality_flag = False
      negative_flag = False

  if (label==0):
    bit_labels.append("neut")
  elif label < 0:
    label = -1
    bit_labels.append("neg")
  elif label > 0:
    bit_labels.append("pos")

bit_title_df = pd.DataFrame({"title":bit_titles, "url":bit_urls, "label":bit_labels})
dftoCsv(bit_title_df, num)
```

그림 12,13. 기사 분석에 사용한 모듈 및 라이브러리 목록

기사의 여론 분석에 사용한 python 코드는 위와 같다. BeautifulSoup 모듈을 통해 입력한 주소의 기사 리스트를 받아오고 자체적으로 제작한 positive, negative를 구분할 수 있는 단어 목록을 사용하여 긍정, 중립, 부정의 3단계로 자료를 분류하였다.

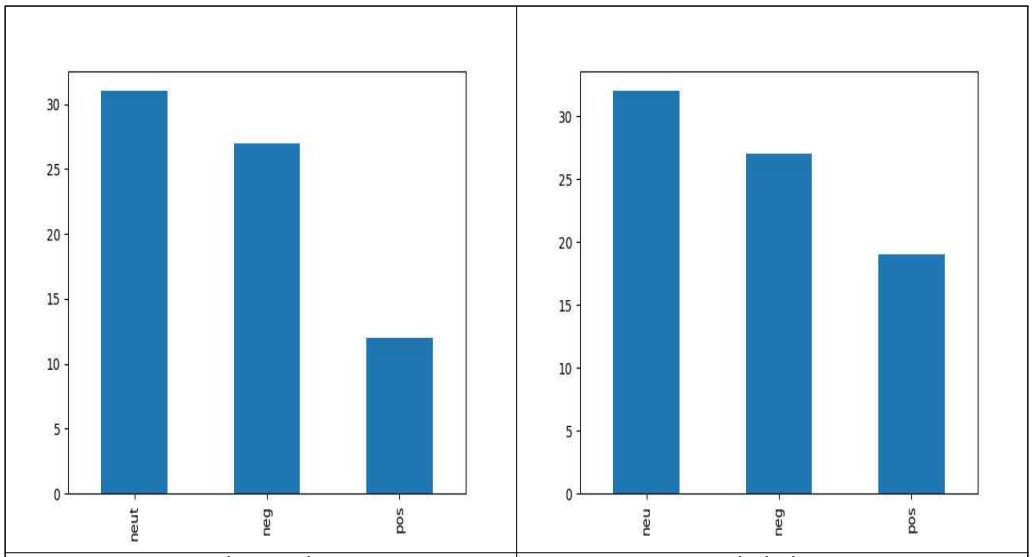
```
if __name__ == "__main__":

  bit_data = pd.read_csv(f'./article_datas/data_비트코인_{now_time}.csv')
  bit_data['label'].value_counts().plot(kind='bar')
  plt.savefig(f'./article_datas/graph/비트코인_{now_time}.png')

  eth_data = pd.read_csv(f'./article_datas/data_이더리움_{now_time}.csv')
  eth_data['label'].value_counts().plot(kind='bar')
  plt.savefig(f'./article_datas/graph/이더리움_{now_time}.png')


  print(bit_data.groupby('label').size().reset_index(name='count')) #train data = 비트코인
  print(eth_data.groupby('label').size().reset_index(name='count')) #test data = 이더리움
```

그림 14. 분석한 자료를 csv 및 이미지파일로 저장하는 코드



비트코인 | 이더리움

그림 15,16 네이버 기사를 분석한 결과 그래프

최종적으로 위에서 생성한 결과 그래프를 웹페이지에 등록하여 제공한다.

3.2.2. RSI, Stochastic, 공포-탐욕지수

또한 코인동향 페이지에서는 거래의 지표가 될 수 있는 RSI, Stochastic 지표 및 공포-탐욕지수 수치를 시각화하여 제공한다. RSI 및 Stochastic 지수는 아래의 python 코드를 통하여 계산한 후 웹페이지에 시각화하여 제공한다.

```python
import pandas
import time
import pyupbit

def rsi_func(ohlc: pandas.DataFrame, period: int = 14):
    delta = ohlc["close"].diff()
    ups, downs = delta.copy(), delta.copy()
    ups[ups < 0] = 0
    downs[downs > 0] = 0
    AU = ups.ewm(com = period-1, min_periods = period).mean()
    AD = downs.abs().ewm(com = period-1, min_periods = period).mean()

    RS = AU/AD
    return pandas.Series(100 - (100/(1 + RS)), name = "RSI")

if __name__ == "__main__":
    data = pyupbit.get_ohlcv(ticker="KRW-BTC", interval="minute60")
    now_rsi = rsi_func(data, 14).iloc[-1]
    print("now rsi value is : ",now_rsi)
```

그림 17. rsi 계산 python 코드

```python
import pyupbit
def tickers_db(ticker):
    df = pyupbit.get_ohlcv(ticker, interval='minute60', count=120)
    return df
def get_stochastic_fast_k(close_price, low, high, n):
    fast_k = ((close_price - low.rolling(n).min()) / (high.rolling(n).max() - low.rolling(n).min())) * 100
    return fast_k

# Slow %K = Fast %K의 m기간 이동평균(SMA)
def get_stochastic_slow_k(fast_k, n):
    slow_k = fast_k.rolling(n).mean()
    return slow_k

try:
    # 조회 원하는 코인 데이터 베이스 생성 ex) KRW-BTC, KRW-ADA 등
    df = tickers_db('KRW-BTC')
    # 기간 20 fast_k 획득
    df['fast_k'] = get_stochastic_fast_k(df['close'], df['low'], df['high'], 20)
    # 기간 5 slow_k 획득
    df['slow_k'] = get_stochastic_slow_k(df['fast_k'], 5)
    # 출력
    print(df)

except KeyboardInterrupt:
        print('시세조회 초과')
```

그림18. Stochastic 계산 python 코드

rsi와 Stochastic의 계산 두가지 모두 pyupbit모듈을 사용하여 진행하였다. 공포탐욕지수는 (https://alternative.me/)에서 제공하는 자료를 이용하였다. 해당 자료들은 웹사이트에서 아래와 같이 제공된다.

그림19. 웹사이트 내 Rsi, Stochastic, 공포-탐욕지수

3.2.3 코인동향 페이지

python 코드를 통해 분석한 비트코인, 이더리움 관련 기사의 여론 분석 결과 그래프 및 rsi, stochastic, 공포-탐욕 지수의 지표를 제공한다. 추가로 Crypto hub 사이트의 가상화폐 기사를 제공한다. 가상화폐 기사를 받아오는 데는 Crypto hub 사이트에서 제공하는 API(그림 28)를 사용하였다. 현재 python과 react를 직접적으로 연결하는 것은 구현하지 못하여 python으로 계산한 결과를 직접 코드에 입력하여 아래와 같이 시각화만 해 주고 있다.

```
  const data1 = [
   { opinion :"긍정", positiv :12 , positivColor :"green" },
   { opinion :"중립", neutral :31 , neutralColor :"orange" },
   { opinion :"부정", negative :27 , negativeColor :"red" },
  ];
...
      <TrendGraph
       value ={61.20 }
       title ="강도 지수(RSI)"
       keyword ={getKeyword (61.20 )}
       color ={getColor (61.20 )}
      />
...
       <img
        width ="100%"
         src  ="https://alternative.me/crypto/fear-and-greed-inde
x.png"
        alt ="feer-and-greed-graph"
       />
```

그림20. 계산값을 웹페이지에 시각화하는 코드의 일부

```
const fetchNews = async () => {
  const request = axios .create ({
   baseURL :"https://api.cryptohub.or.kr/api/news/list?page=1",
   timeout :5000 ,
   headers : {
    "Content-Type":"application/json",
   },
  });
  const { data } = await request ({
   method :"GET",
  });
  return data ;
};
export default fetchNews ;
```

그림21. 기사 리스트를 불러오는 News API

3.3 거래소 기능 구현

3.3.1 거래소 페이지

```
import { useFetchMarketCode , useUpbitWebSocket } from "use-u
pbit-api";
import { useRecoilState } from "recoil";
import { useEffect } from "react";
import styled from "@emotion/styled";
import RealTimeChart from "@/components/RealTimeChart";
import RealTimeOrderBook from "@/components/RealTimeOrderBoo
k";
import {
  marketCodesState ,
  selectedCoinInfoState ,
  selectedCoinState ,
} from "@/recoil/state";
import CoinInfo from "@/components/CoinInfo";
import CoinList from "@/components/CoinList";
import TradeForm from "@/components/TradeForm";
```

그림22. trade.js에 사용한 모듈, 컴포넌트 및 라이브러리

```
function TradePage () {
  const [selectedCoin , setSelectedCoin ] = useRecoilState (selec
tedCoinState );
  const [marketCodes , setMarketCodes ] = useRecoilState (marketC
odesState );
  const { marketCodes : targetMarketCodes } = useFetchMarketCode
();
  const { socketData } = useUpbitWebSocket (marketCodes , "ticker
", {
   throttle_time :400 ,
   max_length_queue :100 ,
  });
  const [, setSelectedCoinInfo ] = useRecoilState (selectedCoinIn
foState );
```

그림23. Tradepage 관련 코드 1

```
useEffect (() => {
 if (socketData ) {
  const targetData = socketData .filter (
   (data ) =>data .code === selectedCoin [0 ].market
  );
  setSelectedCoinInfo (...targetData );
 }
}, [selectedCoin , socketData ]);
useEffect (() => {
 const MarketCodesKRW = targetMarketCodes .filter ((code ) =>
  code .market .includes ("KRW")
 );
 setMarketCodes (MarketCodesKRW );
}, [targetMarketCodes ]);
const handleCoinClick = (event ) => {
 const currentTarget = marketCodes .filter (
  (code ) =>code .market === event .currentTarget .id
 );
 setSelectedCoin (currentTarget );
};
```

그림24. Tradepage 관련 코드 2

해당 프로젝트에서는 upbit의 가상화폐 목록에서 "KRW" 필터를 적용하여 원화로 거래되는 상품만을 추출하여 제공하였다. 위에서 사용한 use-upbit-api의 useUpbitWebSocket 및 useFetchMarketCode 함수의 정보는 아래와 같다.

```
interface ImarketCodes {
    market: string;
    korean_name: string;
    english_name: string;
}
export declare function useUpbitWebSocket(targetMarketCodes?: ImarketCodes[], type?: string, options?: {
    throttle_time: number;
    max_length_queue: number;
}): {
    socket: WebSocket | null;
    isConnected: boolean;
    socketData: any;
};
export declare function useFetchMarketCode(): {
    isLoading: boolean;
    marketCodes: ImarketCodes[];
};
export {};
```

그림25. use-upbit-api 정보

3.3.2 거래소 페이지의 컴포넌트

CoinList.js 파일은 가상화폐의 이름, 현재가, 전일대비 상승 %, 거래대금 등을 가져온다. 코인리스트의 코인을 클릭하면 해당 코인의 차트와 orderbook

이 출력된다.

```
function CoinList ({ data , onClick }) {
  const selectedCoin = useRecoilValue (selectedCoinState );
  const marketCodes = useRecoilValue (marketCodesState );
  //useRecoilValue는 저장된 state의 값을 반환할 때,
  //useRecoilState는 값을 불러오고 변경도 해야될 때 사용.
...
        <CoinBox
         key ={coin .code }
         id ={coin .code }
         onClick ={onClick }
         selected ={selectedCoin [0 ].market === coin .code }
        >
...
```

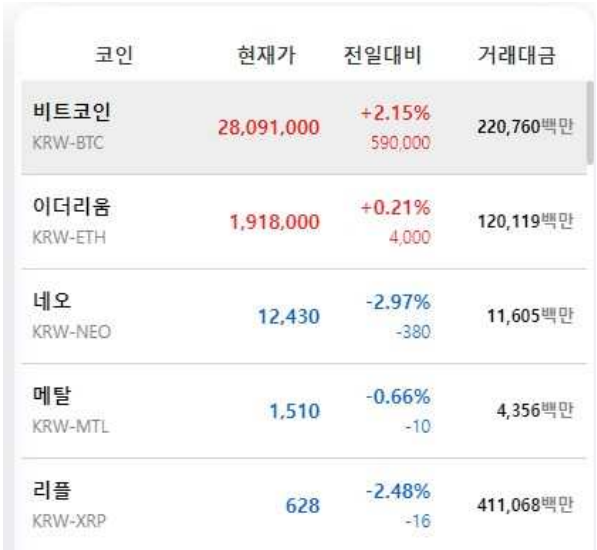
그림26. CoinList 코드의 일부

해당 코드에서는 recoil을 사용하여 거래소 페이지에서 선택한 코인의 정보를 볼 수 있는 틀을 제공한다. 자세한 가상화폐의 정보는 아래의 CoinInfo 코드를 기반으로 화면에 출력하게 된다.

```
function CoinInfo () {
  const selectedCoin = useRecoilValue (selectedCoinState );
   const selectedCoinInfo = useRecoilValue (selectedCoinInfoState
);
  return (
...
     <Text variant ="h4"weight ={700 }>
      {selectedCoin ? selectedCoin [0 ].korean_name : null }
     </Text >
     <Text variant ="subtitle1"color ="#888"weight ={700 }>
      {selectedCoinInfo ? selectedCoinInfo .code : null }
     </Text >
...
        <Text variant ="subtitle2"weight ={600 }>
         52주 저가
        </Text >
        <Text color ="#1261c4"weight ={600 }>
         {selectedCoinInfo .lowest_52_week_price
          ? selectedCoinInfo .lowest_52_week_price .toLocaleString
           (
            "ko-KR"
           )
          : null }
        </Text >
...
```

그림27. CoinInfo.js 코드

마찬가지로 해당 코드에서도 recoil을 사용하며 페이지에서 선택한 코인의 상세 정보를 제공하는 역할을 수행한다. 위의 코드들을 활용하여 웹페이지에서 제공되는 형태는 아래와 같다.



코인	현재가	전일대비	거래대금
비트코인 KRW-BTC	28,091,000	+2.15% 590,000	220,760백만
이더리움 KRW-ETH	1,918,000	+0.21% 4,000	120,119백만
네오 KRW-NEO	12,430	-2.97% -380	11,605백만
메탈 KRW-MTL	1,510	-0.66% -10	4,356백만
리플 KRW-XRP	628	-2.48% -16	411,068백만

그림28. 거래소 페이지의 coinlist 레이아웃

```
const getMaxSize = (orderbook ) => {
  const askSizes = [];
  const bidSizes = [];
  orderbook .forEach ((unit ) => {
   askSizes .push (unit .ask_size );
   bidSizes .push (unit .bid_size );
  });
  const maxAskSize = Math .max (...askSizes );
  const maxBidSize = Math .max (...bidSizes );
  return [maxAskSize , maxBidSize ];
};
```

그림29. orderbook 관련 코드의 일부 1

```
function RealTimeOrderBook ({ style }) {
  const selectedCoin = useRecoilValue (selectedCoinState );
   const selectedCoinInfo = useRecoilValue (selectedCoinInfoState
);
  const webSocketOptions = { throttle_time :400 , max_length_queu
e :100 };
  const { socketData } = useUpbitWebSocket (
   selectedCoin ,
   "orderbook",
   webSocketOptions
  );
  const [askMaxSize , setAskMaxSize ] = useState ();
  const [bidMaxSize , setBidMaxSize ] = useState ();
  const [scrollOn , setScrollOn ] = useState (false );
  const orderBookContainerRef = useRef ();
  useEffect (() => {
   if (socketData ) {
    const orderbook = socketData .orderbook_units ;
    const [maxAskSize , maxBidSize ] = getMaxSize (orderbook );
    setAskMaxSize (maxAskSize );
    setBidMaxSize (maxBidSize );
    if (!scrollOn ) {
     orderBookContainerRef .current .scrollTop =
      orderBookContainerRef .current .scrollHeight / 3 ;
    }
   }
  }, [socketData ]);
  const scrollEventHandler = () => {
   setScrollOn (true );
  };
...
```

그림30. orderbook 관련 코드의 일부 2

use-upbit-api와 recoil을 사용하여 업비트 사이트의 가상화폐가 거래되는 금액 구간을 실시간으로 받아온다. 서버로부터 수령한 데이터를 socketData 배열에 저장한 후 시각적으로 보여준다. 해당 자료는 페이지에서 아래와 같은 모습으로 제공된다.

그림31. 페이지 내 orderbook 레이아웃

```
function RealTimeChart () {
  const selectedCoin = useRecoilValue (selectedCoinState );
   const selectedCoinInfo = useRecoilValue (selectedCoinInfoState
);
  const [fetchedData , setFetchedData ] = useState ();
  const [processedData , setProcessedData ] = useState ();
  const [updatedCandle , setUpdatedCandle ] = useState ();
  const options = { method :"GET", headers : { Accept :"applicati
on/json" } };
  async function fetchDayCandle (marketCode , date , count ) {
   try {
    const response = await fetch (
       `https://api.upbit.com/v1/candles/days?market=${marketCode
}&to=${date   }T09:00:00Z&count=${count   }&convertingPriceUnit=KR
W`,
     options
    );
    const result = await response .json ();
    setFetchedData (result );
   } catch (error ) {
    console .error (error );
   }
  }
```

그림32. RealTimeChart 관련 코드의 일부 2

```
  useEffect (() => {
   if (!selectedCoin ) return ;
   fetchDayCandle (selectedCoin [0 ].market , dayjs ().format ("Y
YYY-MM-DD"), 200 );
  }, [selectedCoin ]);
  useEffect (() => {
   if (!fetchedData ) return ;
   const processed = [...fetchedData ].reverse ().map ((data ) =>
{
    return {
     time :data .candle_date_time_kst ,
     open :data .opening_price ,
     high :data .high_price ,
     low :data .low_price ,
     close :data .trade_price ,
    };
   });
   setProcessedData (processed );
  }, [fetchedData ]);
```

```
  useEffect (() => {
   if (!selectedCoinInfo ) return ;
   setUpdatedCandle ({
    time :selectedCoinInfo .trade_date
     ? {
       day :selectedCoinInfo .trade_date .slice (6 , 8 ),
       month :selectedCoinInfo .trade_date .slice (4 , 6 ),
       year :selectedCoinInfo .trade_date .slice (0 , 4 ),
      }
     : null ,
    open :selectedCoinInfo .opening_price ,
    high :selectedCoinInfo .high_price ,
    low :selectedCoinInfo .low_price ,
    close :selectedCoinInfo .trade_price ,
   });
  }, [selectedCoinInfo ]);
```

그림 33. RealTimeChart 관련 코드의 일부 2

코인 리스트에서 가상화폐를 선택하면 RealTimeChart 함수의 fetchDayCandle 함수를 이용하여 marketCode, date, count 변수를 전달한 후 response의 json 데이터를 받아온다. 해당 json 데이터를 chartComponent 함수를 이용하여 차트로 시각화한다.

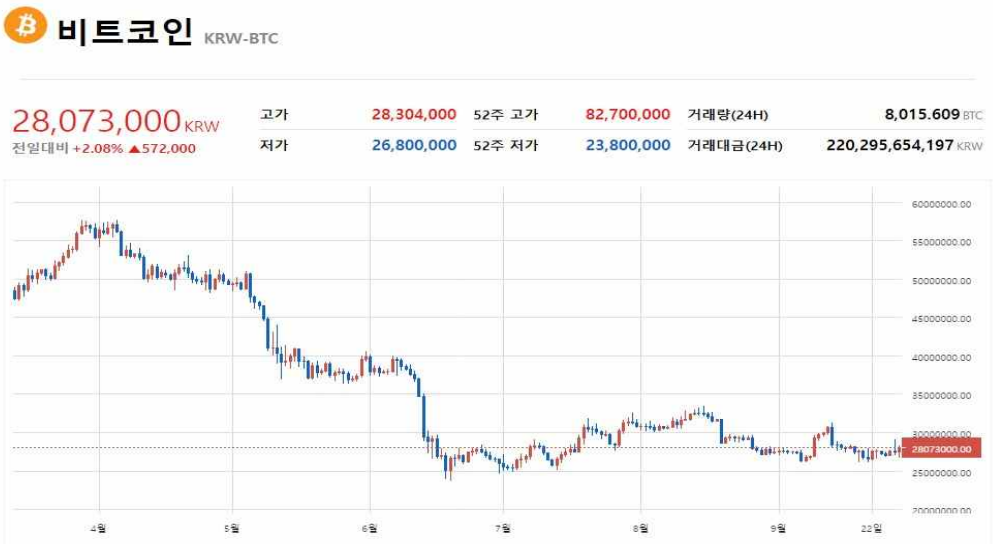


그림 34. 페이지 내 chart 레이아웃

그림35. 거래소 페이지 내 매수기능

거래소 페이지 내의 코인 목록에서 매수하고자 하는 가상화폐를 클릭하여 수량 및 가격을 입력하여 매수를 진행한다. 현재 디자인만 구현해 둔 상태이며 후속 작업으로 DB와 연동을 마무리한 후 보유 금액 및 코인의 정보를 매수매도 동작과 연계하여 사용할 수 있도록 할 것이다.

4. 연구 결과 분석 및 평가

해당 프로젝트에서 진행하였던 페이지들의 모습은 아래의 사진(그림36~39)과 같다.

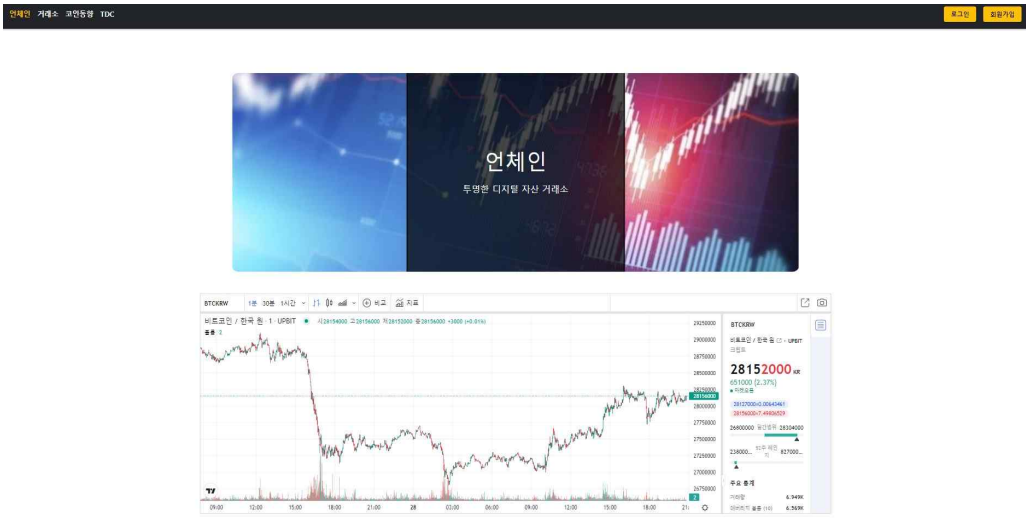


그림36. 메인 페이지

메인페이지에서는 거래소 이름, 차트를 간단하게 보여준다.

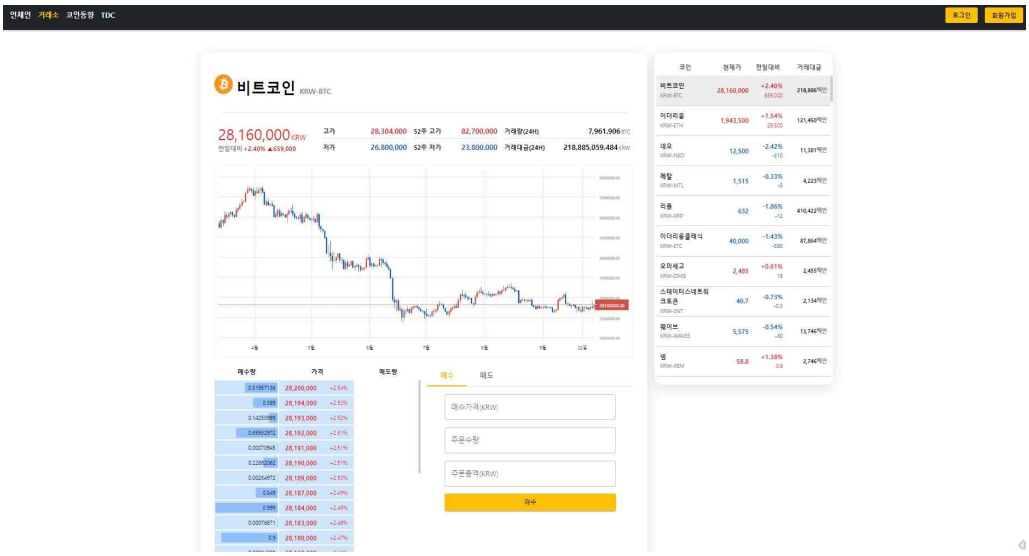


그림37. 거래소 페이지

거래소 페이지에서는 upbit api를 기반으로 한 실시간 코인 정보를 제공하고 있다.

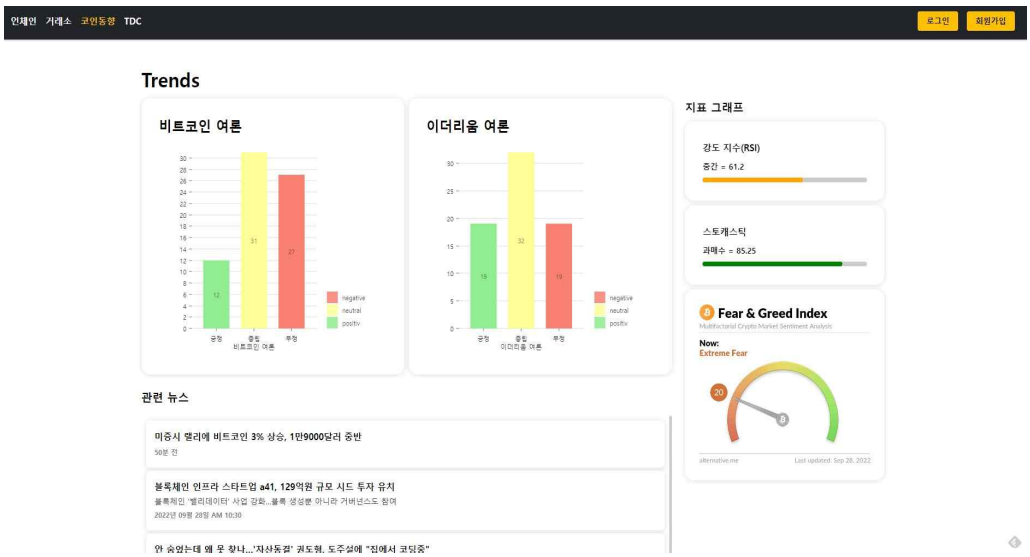


그림38. 코인동향 페이지

코인동향 페이지에서는 비트코인 및 이더리움의 여론 분석 그래프, rsi, stochastic, 공포-탐욕지수를 지표로써 제공한다. 또한 가상화폐 관련 뉴스를 타임라인의 형태로 제공한다.

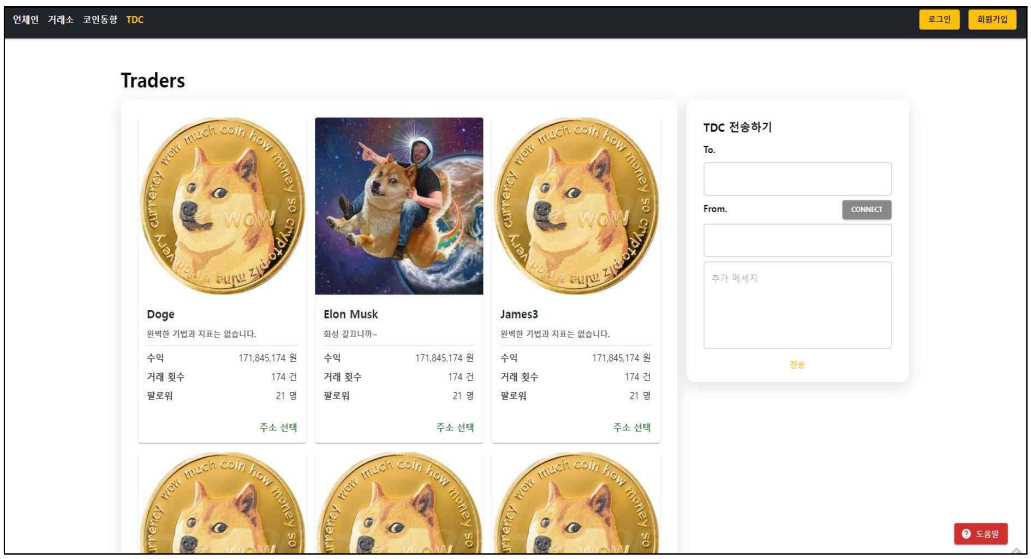


그림39. TDC 거래 페이지

TDC페이지는 수익금 순위가 상위권인 상위 트레이더들의 목록을 보여주며, 해당 트레이더들에게 TDC코인을 전달하여 거래기록을 받아볼 수 있도록 할 것이다. 지금은 DB와 연동이 되지 않은 상태라 임의의 주소를 사용하였으나 이후 실제 사용자의 주소와 연동하여 거래를 진행할 수 있을 것이다.

배경지식의 부족으로 프론트 관련 지식 습득 및 자료수집 등에 계획한 것

이상으로 시간을 많이 소모하게 되었다. 따라서 거래소의 기본 기능을 구현하는 데에도 시간이 많이 소요되었고, 완성하지 못한 부분(실제로 매도, 매수를 진행하는 부분 및 데이터베이스 연동 등)이 다소 있었다.

5. 결론 및 향후 연구 방향

해당 프로젝트에서 upbit 기반 거래소 생성 및 이더리움 기반 코인 발행 연구를 진행했다. 리액트로 페이지를 구현하고 거래소의 기본 기능을 만들면서 프론트에 대한 지식 및 경험을 많이 쌓을 수 있었다. 앞에서 얘기한 기존 거래소의 문제점을 보완해 기사 제목을 분석해 코인에 대한 여론의 긍/부정을 알려주는 도표를 만들었고, rsi, stochastic 지표를 숫자로 바로 볼 수 있게 만들어 코인동향 페이지에 구현했다.

코인을 발행하면서 거래소 안에서 어떻게 쓰일 수 있을까를 고민했다. 가상화폐와 가장 밀접한 거래소에선 화폐(원, 달러)와 화폐(코인)의 교환만 있을 뿐 가상화폐를 사용해 무언가를 구매하는 경우는 거의 없다. 따라서 '발행한 코인을 지불수단으로 활용할 수 있으면 좋지 않을까?'라는 생각에 해당 프로젝트의 TDC코인에 대해서 계획하게 되었다.

후속 연구를 통해 자체적인 데이터베이스를 구축하여, 거래소 내의 거래 횟수 및 기타 데이터를 수집하여 이를 TDC로 거래할 수 있는 환경을 갖춘다면 현재의 이름뿐인 가상화폐가 아닌 실제 '화폐'로써의 TDC기능을 사용할 수 있을 것이다.

6. 개발 일정 및 역할 분담

6.1 개발 일정

<table>
<tr><td colspan="2">5월</td><td colspan="6">6월</td><td colspan="4">7월</td><td colspan="4">8월</td><td colspan="4">9월</td></tr>
<tr><td>3주</td><td>4주</td><td>1주</td><td>2주</td><td>3주</td><td>4주</td><td>1주</td><td>2주</td><td>3주</td><td>4주</td><td>1주</td><td>2주</td><td>3주</td><td>4주</td><td>1주</td><td>2주</td><td>3주</td><td>4주</td></tr>
<tr><td colspan="6">거래소, 암호화폐 스터디</td><td></td><td></td><td></td><td colspan="5">리액트 스터디</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td colspan="14">거래소 기능 구현</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="2">중간 보고서 작성</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="6">ui디자인</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="5">코인 개발 및 발행</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="4">오류 수정</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="4">최종 발표, 보고서 준비</td></tr>
</table>

6.2 역할 분담

이름	역할
유상욱	-거래소 시스템 설계 및 구현 -여론 분석 그래프 구현
용태완	-거래소 ui디자인 -코인 생성 및 기능 정리 -코인동향 페이지 구현

7. 참고 문헌

[1]. byseop. 트레이딩 시뮬레이터 구현
(https://byseop.github.io/trading-simulator/)

[2]. kangminhyek. use upbit api 라이브러리
https://socket.dev/npm/package/use-upbit-api

[3]. mikec3. metamask react 연동
https://github.com/mikec3/my_tutorials/tree/master/MetaMask_Connection

[4]. ethere document
https://docs.ethers.io/v5/

[5]. 뉴스 기사 API
https://www.cryptohub.or.kr/

[6] 코인 발행 참고자료
https://ethereum.org/en/developers