Cryptocurrency Management System

CS3305

Team Software Project

2022-2023

Submitted by

Karim Abdul Ulmann 119329701

Eoghan Daly 119323056

Nikolajs Lozko 118512266

Liam Thomas Lenihan 119357813

Hamza Hassan Khan 119401422

University College Cork,

Ireland

Abstract

Over the last few decades there has been a significant number of new technological-based markets emerging. One such market is the cryptocurrency market, since its debut in 2009, has grown immeasurably with over 14 million Bitcoins circulating in today's market. With many different platforms supporting the buying, selling and trading of cryptocurrency is has never been easier for the "five key market participants — merchants and consumers, tech developers, investors, financial institutions and regulators" (Source: https://www.pwc.com/us/en/industries/financial-services/library/cryptocurrency-evolution.html) to enter the market. Binance and Coinbase are examples of such platforms. As will be discussed in this document, the interest in owning cryptocurrency is growing and the aim of this project is to create a beginner friendly system to explore and manage digital currencies, handle its transactions, and to offer an introduction into the new technology known as Non-Fungible Tokens (NFTs).

Table of Contents

Chapter 1

Introduction

1.1 Digital Currencies

Cryptocurrency is in some ways like the government-issued currencies that we deal with in everyday life, both hold monetary value and are stored in "wallets". The differences lie in the form that the currencies take and how they are transferred. Instead of handling physical money, cryptocurrency exists digitally and uses a decentralised system to handle transactions rather than passing money through banks that have a regulating authority. A distributed public ledger, or otherwise known as a "blockchain", records the encrypted cryptocurrency transactions and is often used to verify the authenticity of cryptocurrency during trades or market purchases. The value of each cryptocurrency is completely based on supply and demand, leading to significant drops/gains in value.

Despite the increased risk in an uncertain and constantly fluctuating market, the number of people engaging with cryptocurrency has increased exponentially. According to research carried out by crypto.com, "It only took four months to double the global crypto population from 100 million to 200 million. By comparison, it took nine months to reach 100 million from 65 million since we began tracking these numbers." This increase shows that the interest in this market is growing at an astonishing rate, this is highlighted in figure 1.

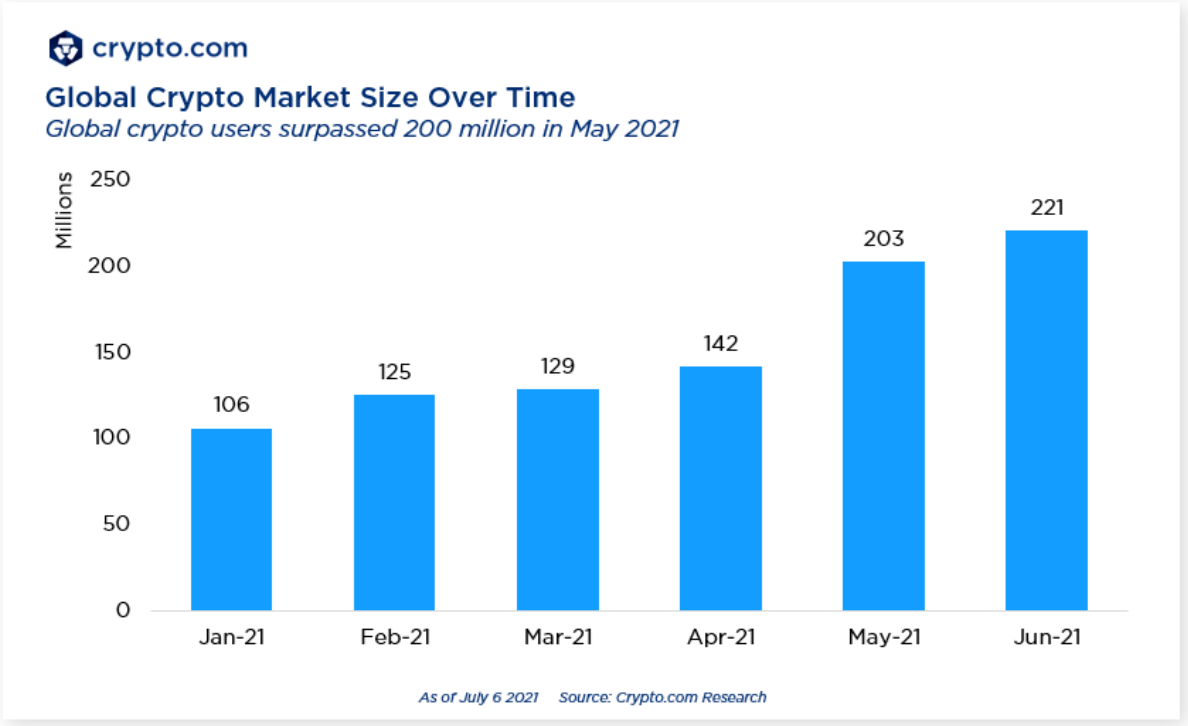


Figure 1 Increase in Cryptocurrency users from January – June 2021 (Source: https://blog.crypto.com/global-crypto-users-over-200-million/)

Additionally, as the normalisation of cryptocurrency continues to expand and more and more industries accept it for payment, the value of the market will increase, and more people will choose to explore the option of digital currency payments in the coming future. Projections show that the market will grow from "The global cryptocurrency market is projected to grow from $910.3 million in 2021 to $1,902.5 million in 2028 at a CAGR of 11.1% in forecast period, 2021-2028" (Source: https://www.fortunebusinessinsights.com/industry-reports/cryptocurrency-market-100149). Figure 2 shows the predicted increase for the North

America cryptocurrency Market between the previously mention years.

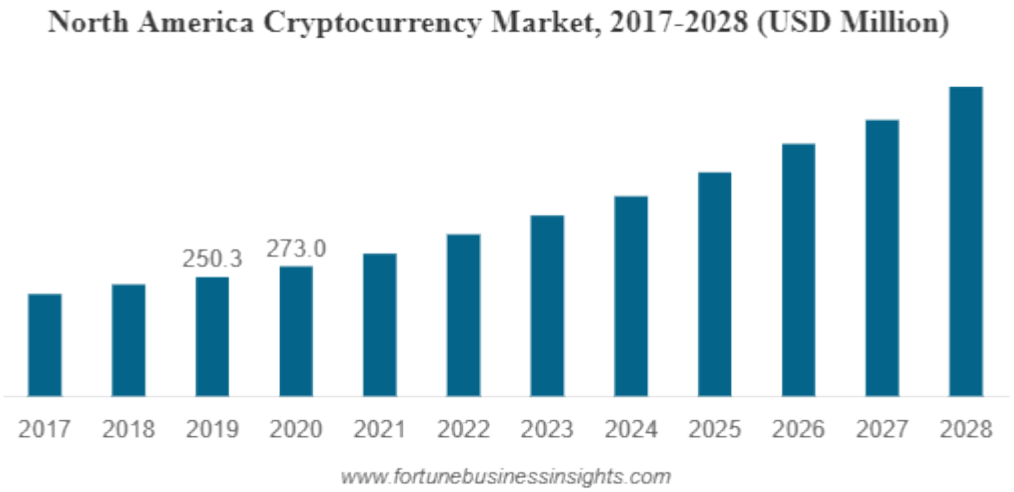


Figure 2 Predicted growth in cryptocurrency sales in North America (Source: https://www.fortunebusinessinsights.com/industry-reports/cryptocurrency-market-100149

1.2 Non-Fungible Tokens

With the recent surge in popularity of Non-Fungible Tokens over the last year. We were determined to integrate NFTs while keeping the project's beginner-friendly nature. Therefore, considering NFTs are a newer technology, we strove to provide an overview of how they operate and allow our users to learn more about them.

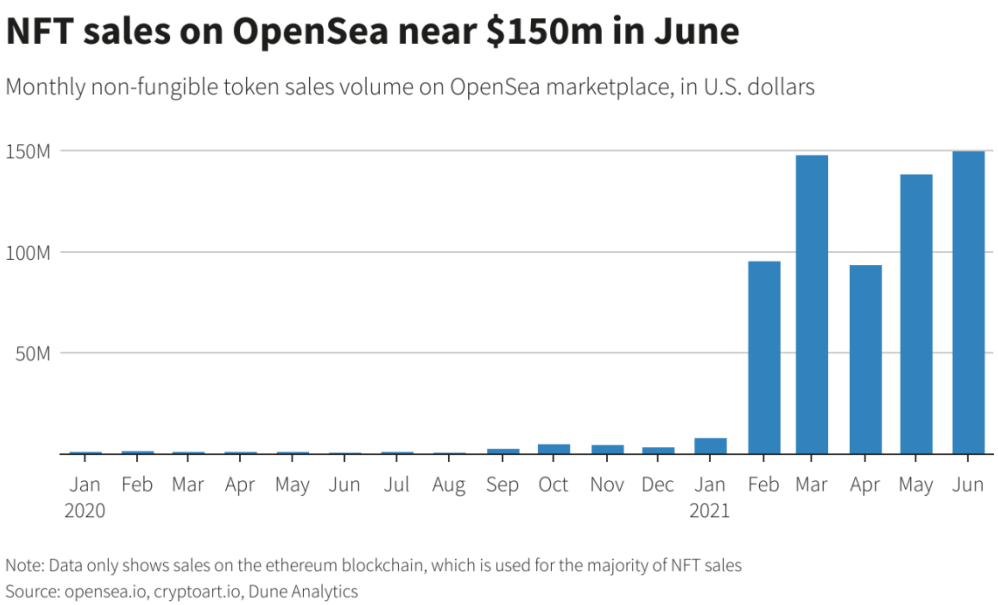


figure 3 Record of NFT sales from January 2020 – June 2021 (Source: https://www.reuters.com/article/us-fintech-nft-data-idCAKCN2EB1I8

We believe from the details discussed in this first chapter that there is a clear opportunity for a beginner friendly cryptocurrency management platform that includes an introduction to NFTs to aid the millions of new people starting their excursion into this market. This project will focus on bringing this idea to life.

Chapter 2

Use Cases & Requirements

In this chapter, we will give a high-level overview of the main aspects of the project. Firstly, we will introduce our use case diagram to give an outline of the software, before going into more detail in the requirements section.

2.1 Use Case Diagram

The diagram below shows the various features available on the application. Figure 3 highlights these and reinforces that these features are only available to authenticated users. The practice of trading between application users and the global market are examples of these. However, this does not mean that the website only caters to the registered users. You can view the remaining aspects of the application in the previously mentioned diagram.

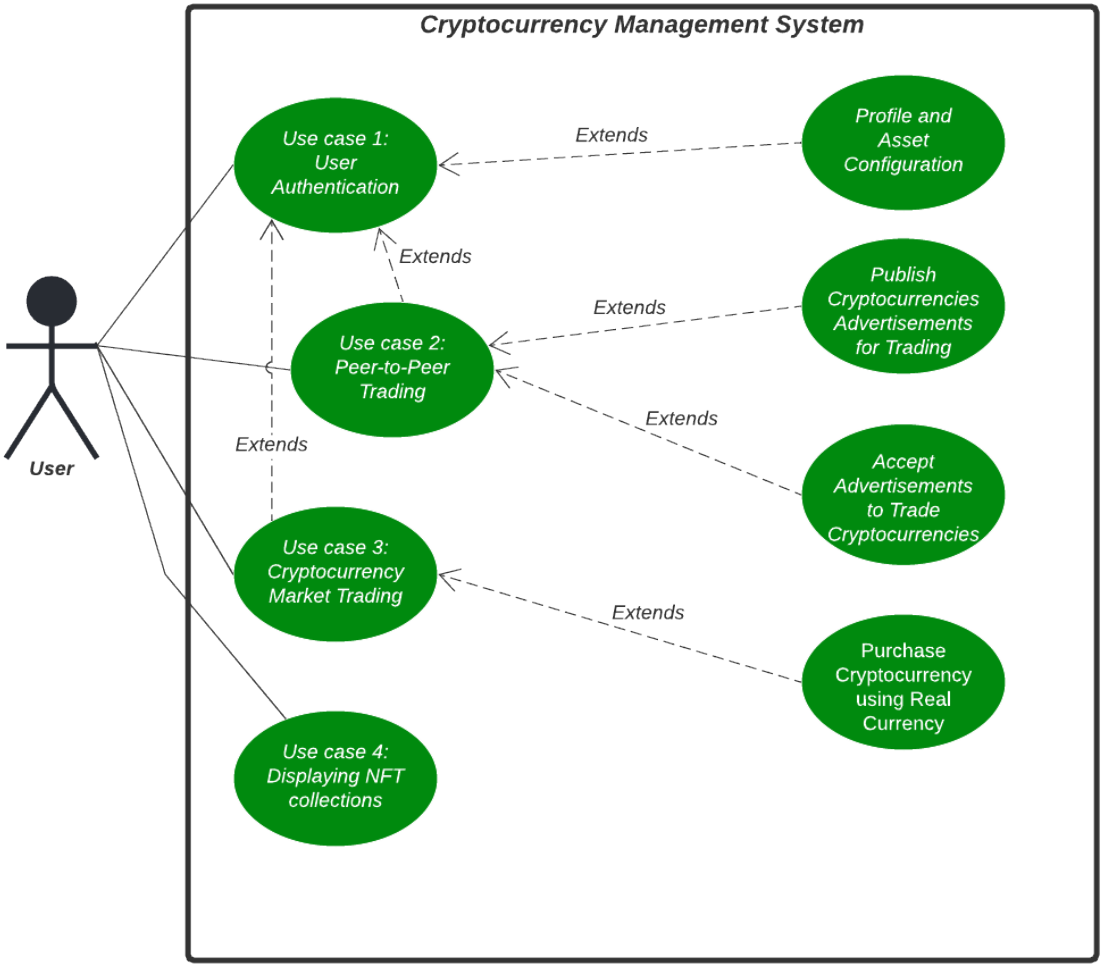


Figure 3: Use Case Diagram of the Cryptocurrency Management System

2.2 Requirements

As outlined in the Use Case Diagram, there are several components in the design to be aware of.

These components will be talked about in more detail below, within the following headings:

- User Authentication
- Peer-to-Peer Cryptocurrency Trading
- Cryptocurrency Market Trading
- Displaying Collections of NFTs

2.2.1 Use-Case 1 User Authentication

To begin, a user must log into the Cryptocurrency Management System using their login details. Upon entering the correct details, the user will have access to both peer-to-peer trading and market trading while having their own profile on the platform. If this is the user's first successful authentication, then a profile will be created for them including the entered user credentials. If a user's log in attempt fails, an error message will be displayed.

2.2.2 Use-Case 2 Peer-to-Peer Cryptocurrency Trading

After authentication the number of features available to the user will increase; one of the new features being Peer-to-Peer Cryptocurrency Trading. This allows the user to view advertisements of the different cryptocurrencies other users are willing to trade. This will show the cryptocurrencies up for sale, the cryptocurrencies that will be accepted and the total worth of the transaction. The user will also have the option to publish an advert of their own to sell the cryptocurrencies that are in their wallet.

2.2.3 Use-Case 3 Cryptocurrency Market Trading

Another new feature that opens to authenticated users is the Cryptocurrency Market Trading. This will allow users to view information about different cryptocurrencies including their current market value, total volume, top tier volume and direct volume. This information will be presented in graphs and/or tables. The user will have the option to purchase cryptocurrency using government-issued currency which will be added to the assets in their wallets.

2.2.4 Use-Case 4 Displaying Collections of NFTs

The user is also able to see a list of NFT collections that have been ranked top over the last seven days regardless of whether they are signed in or not. The data was obtained using the Rarible Data Scraper API, which was made available by nftsmaker.io. The user can see the name of the collection, its cost, and the number of NFTs it contains. This feature serves as to help familiarise users as to what NFTs are and how they work, since NFTs are relatively new concept within digital trading this will be relevant to all users. By clicking on the collection, the user will be sent to nftsmaker.io's page dedicated to the NFT collection, where they will be able to interact with the NFT further.

Chapter 3

Architectural overview

This chapter will cover three types of diagrams used in the design of our project. These include structural as well as object-oriented diagrams. They provide a broad overview of the system and how its various elements interact and will serve as a key component of documentation for future

updates and refactorizations. Some diagrams can also be a base for future documentation which would provide a more in-depth view of the system.

3.1 Entity Relationship Diagram

We make use of a relational database to store information pertaining to users, cryptocurrencies, and the relationships between them. Naturally during the design phase, we created an entity-relationship diagram to model this database. This ensured that the database was sufficient to meet the requirements set out in chapter 2 and that no redundant information was stored.

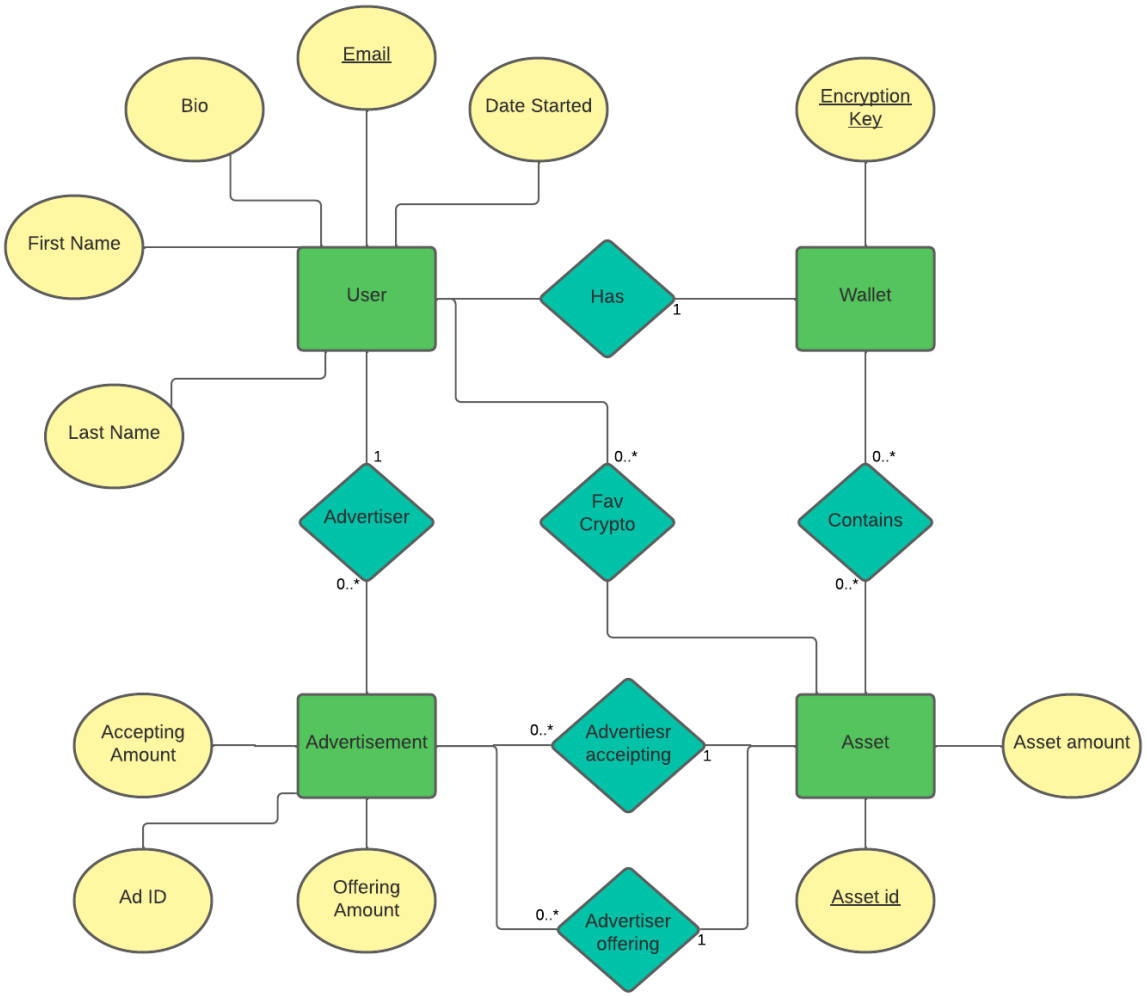


Figure 1 Entity Relationship Diagram

3.2 Data Flow Diagrams

In order to model the flow of data from external sources to data sinks we decided to create two data flow diagrams (Level 0 and Level 1). These diagrams aided in the understanding of where data is drawn from and where is goes. We opted for the Yourdon and code notation as a) it was most familiar to us and b) it is more commonly used for system analysis and design.

3.2.1 Data flow diagram Level 0

The Below diagram illustrates at a high level how the system communicates with external tools and users. It receives market information (in the form of real time and historic cryptocurrency), user advertisements (including amount, currency offered, and currency accepted), and orders (both buy and sell). It stores and processes this information for use in web pages displayed back to the user.

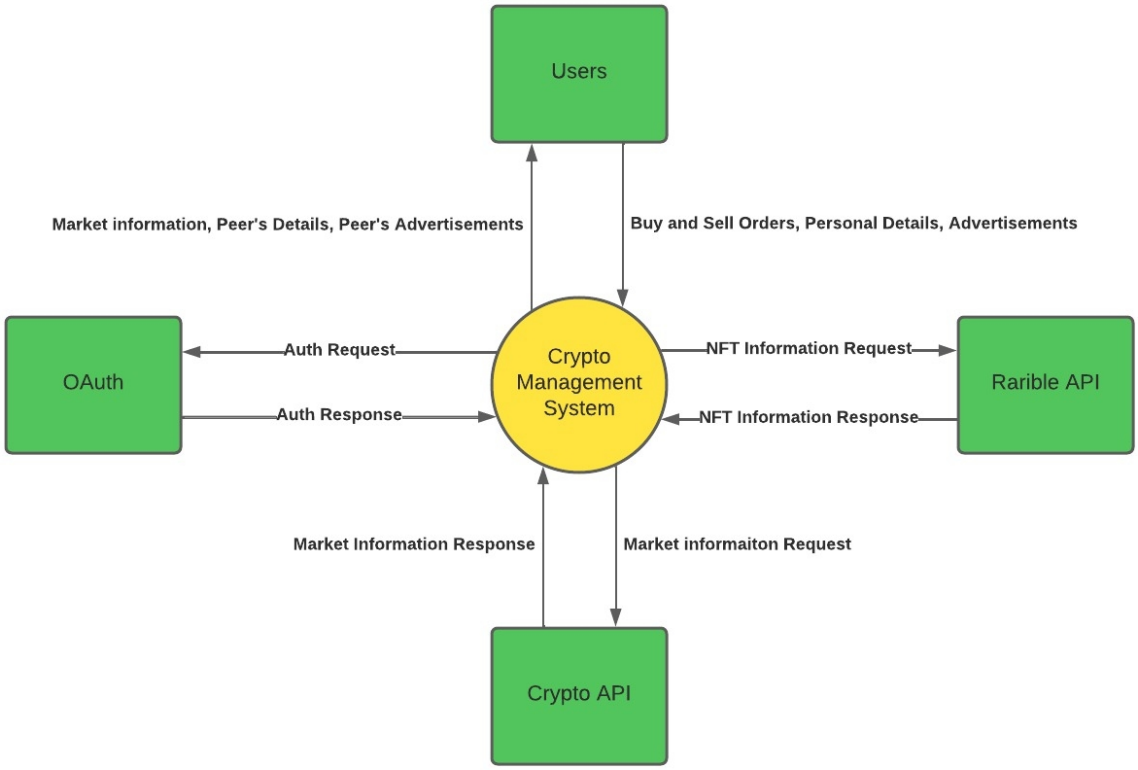


Figure 2 Data Flow Diagram Level 0

3.2.2 Data flow diagram Level 1

The below diagram illustrates in greater detail how the information flows involved in the authentication process, peer to peer trade, market trade and the NFT retrieval. Authentication, given that it has been outsourced to OAuth, is not part of our system. However, it has been included in the diagram for the sake of clarity.

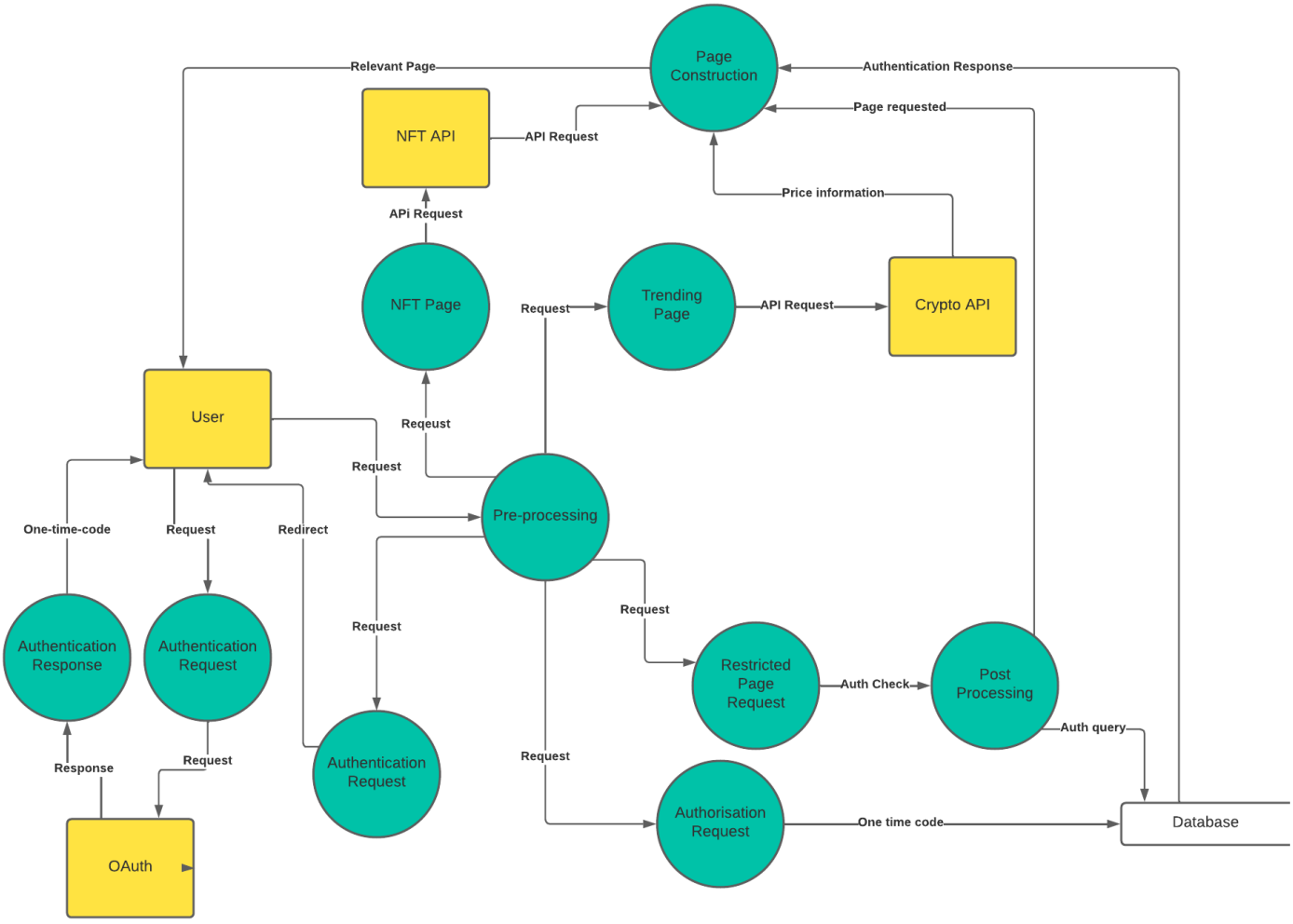


Figure 3 Data Flow Diagram Level 1

3.3 State Transition Diagram

To represent the various states in which our application can be in we have chosen to create a state transition diagram. Each state is a page that the user is currently viewing. User transitions between states by clicking various links in the page.

When a user clicks on a coin's name, they are taken to the specific coin details page. This page will still have all the tabs on it and the user can navigate back to any tab they like. Likewise with all other pages the user will have access to all other tabs

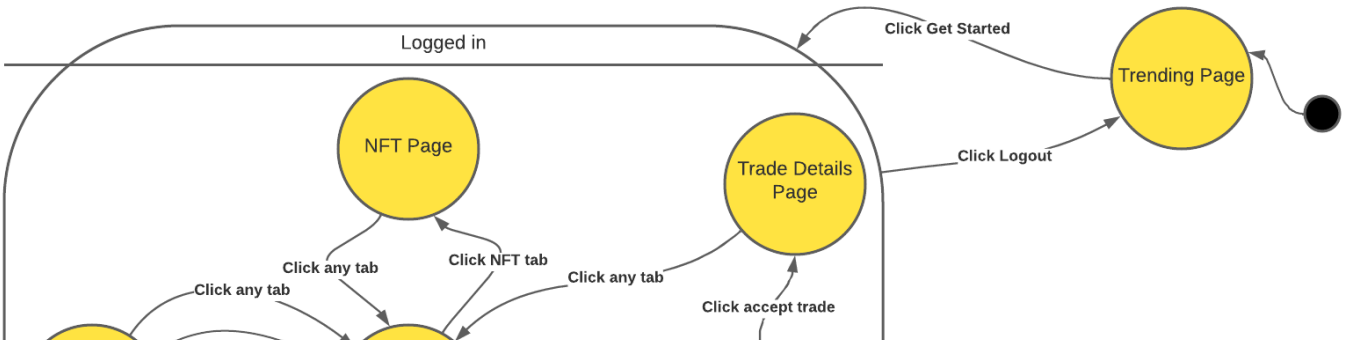


Figure 4 State Transition Diagram

3.4 Sequence Flow Diagrams

To illustrate the order of interactions between entities in our system we created three sequence flow diagrams. Each diagram models a typical interaction of one of our use cases.

3.4.1 Use Case 1

For use case 1 we will model a typical logon interaction. The user is attempting to logon but is first redirected to Google's OAuth server to get a valid one-time code (OTC). The user then sends the OTC to our server, and we validate it with google before creating and account or showing a user their account.

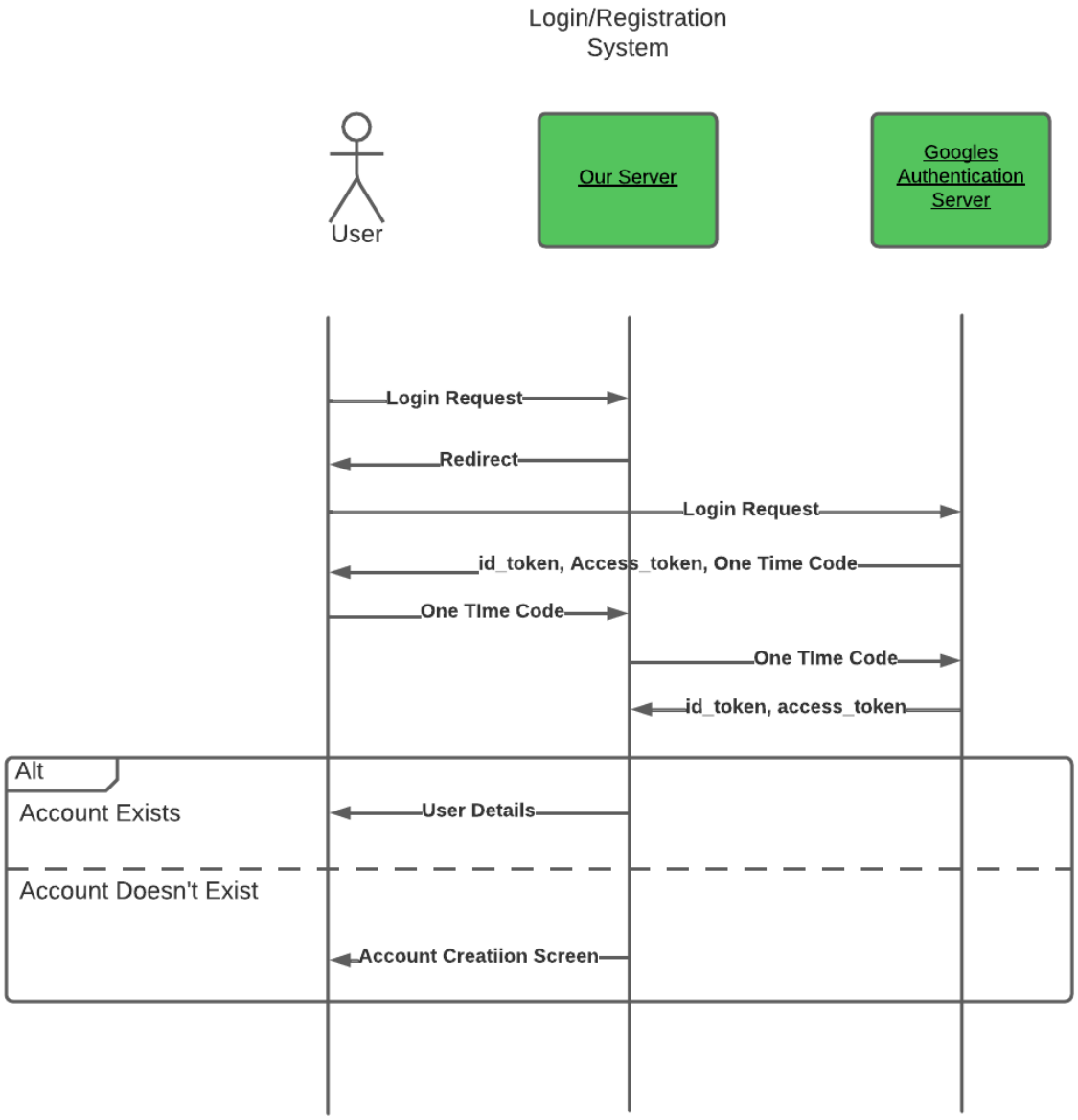


Figure 5 Sequence Flow Diagram for Use Case 1

3.4.2 Use Case 2

The below diagram models peer to peer interaction. First the advertiser must post an add on the website. The accepter will then be able to view all advertisements and accept the best offer. Once the advertisement is accepted a confirmation is sent to each user.

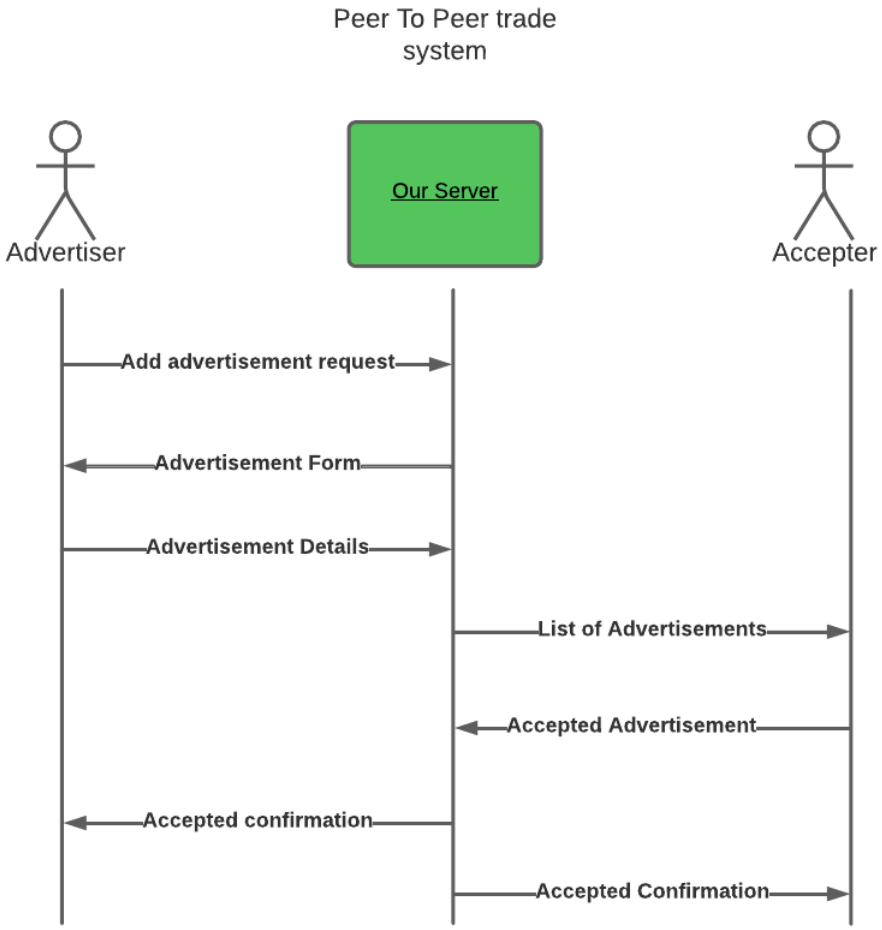


Figure 6 Sequence Flow Diagram for Use Case 2

3.4.3 Use Case 3

The below diagram models a market purchase. First the user must authenticate. They are then shown the market trade page. To purchase an amount of cryptocurrency they can click on any coin's name. They are then sent a specific coin page for that coin. They can then buy a specified amount of that cryptocurrency. Once they do, they will receive a confirmation of purchase.

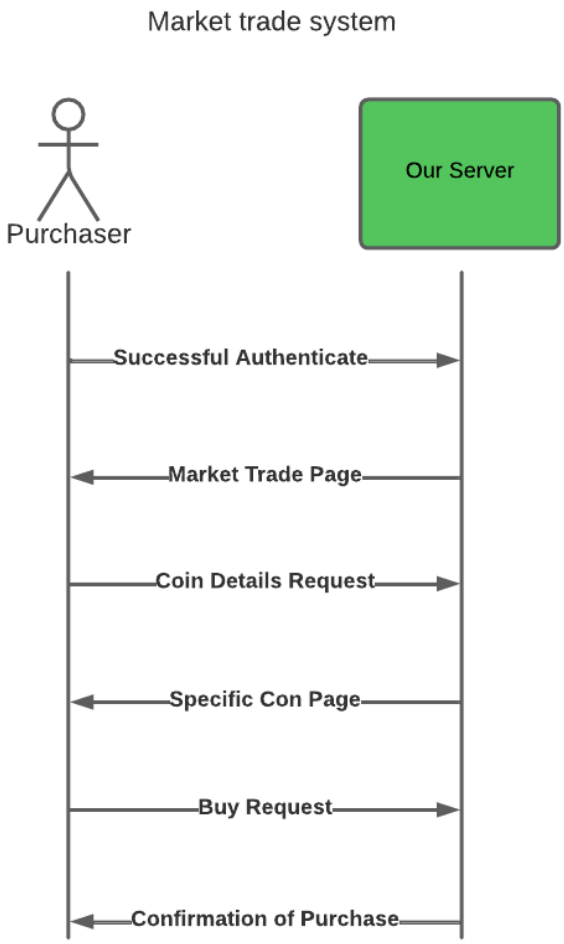


Figure 7 Sequence Flow Diagram for Use Case 3

3.4.4 Use Case 4

The below diagram models a user attempting to view the gallery of NFTs. First the user must request the initial page. This is comprised of the initial header including the navbar and logo section. Part of this page is a javascript module which is loaded separately. The User will then request metadata of the NFT collections, including the total count of NFTs, the image url of the NFT collection, and the name of the NFT collection. This metadata is retrieved from the RapidAPI server and then passed directly on to the user. The reason the user cannot directly request this resource is because they would need an API key. The user will then request each image directly from the server on which they are hosted.

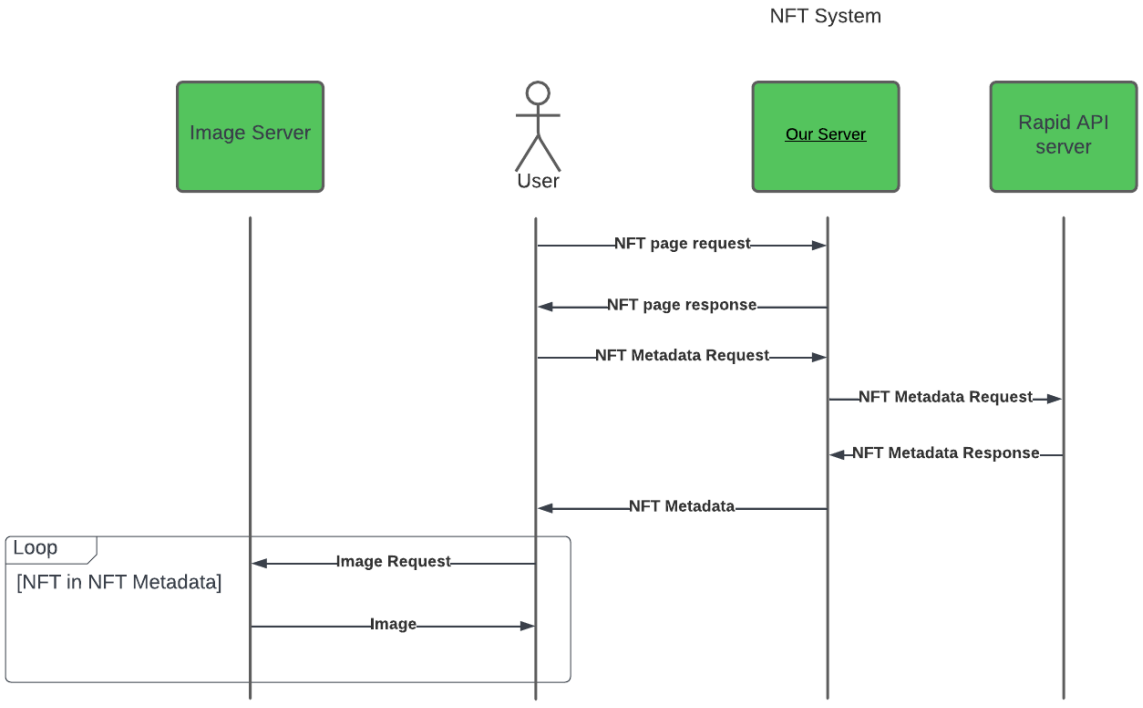


Figure 8 Sequence Flow Diagram for Use Case 4

Chapter 4

The Major Components

In this chapter we will discuss the activities taken to implement the main components during our project's development. Our implemented product performs as described, accomplishing its function

Our project was developed using the Python programming language and the Flask framework and used Jinja2 as a template engine to handle the HTML Templates. We utilised GitHub for version control enabling consistency among all our team members, see the link to our shared repository below. https://github.com/hamzakhan0077/CryptoManagementSystem

There are 7 major components underlying our product: User authentication, a Crypto wallet, The market crypto trading page, The coin's specific page, Coin Candlestick Graphs with current market prices, Peer-to-peer cryptocurrency trading and finally a page displaying NFTs

4.1 User Authentication

We enabled users to authenticate themselves via google authentication.

Using Flask oAuthlib we were able to implement user authentication. Through this we were able to control user access to view authorised pages, meaning only users who were signed in could view all pages and could interact fully with all functionalities.

First time users signing in will have a wallet instantiated and assigned to their account. Users with existing accounts can sign in and continue from where they left off the last session, with their wallet and contents saved.

```
"""  ******************** Auth  ********************  """
@app.route('/login')
def login():

    google = oauth.create_client('google')
    redirect_uri = url_for('authorize', _external=True)
    return google.authorize_redirect(redirect_uri)

@app.route('/authorize')
def authorize():

    google = oauth.create_client('google')
    token = google.authorize_access_token()
    user_info = oauth.google.userinfo()
    session['email'] = user_info['email']
    if not User.query.filter_by(email = user_info['email']).all(): # if user is not already in DB
        wallet_enc_key = sha256(user_info['email'].encode()).hexdigest()
        wallet = Wallet(encryption_key=wallet_enc_key, wallet_holder_email=user_info['email'])
        cryp_user = User(first_name=user_info['name'], last_name=user_info['family_name'], email=user_info['email'],
                         date_started=date.today(), wallet_hash=wallet_enc_key)
        db.session.add(wallet)
        db.session.add(cryp_user)
        db.session.commit()
    else:
        print("this user is already there")

    return redirect('/market')


@app.route('/logout')
@login_required
def logout():

    for key in list(session.keys()):
        session.pop(key)
    return redirect('/')
```

Figure 1 login and authorisation code

4.2 Wallet

From the Home page, if the user is currently signed in, they can navigate their way to view their own crypto wallet page. Here the page displays the wallet's encryption key using SHA256 hashing.

It is this encryption key that is used to transfer assets among users.

The page also exhibits the different assets owned by the user and its amounts, the value of each asset and the total value of the assets in USD.

```python
def asset_data(asset):
    asset = asset.lower()
    crypto_data = [coin for coin in cg.get_coins_markets(vs_currency=chosen_currency) if coin["symbol"] == asset][0]
    return crypto_data

@app.route('/wallet')
@login_required
def showWallet():
    owned_cryptos = {}
    overall_value = 0
    the_user = User.query.filter_by(email=session['email']).first()
    wallet_handler = Wallet_Handler(the_user.wallet_hash)

    for asset in Asset.query.filter_by(wallet_encryption_key=the_user.wallet_hash).all():
        data = asset_data(asset.asset_id)
        owned_cryptos[asset.asset_id.upper()] = {'amount':asset.asset_amount,'value': '${:,.2f}'.format(asset.asset_amount * data['current_price']),'data':data}
        overall_value+= asset.asset_amount * data['current_price']
    overall_value_formatted = '${:,.2f}'.format(overall_value)
    # for val in all_assets:
    #     wallet_handler.fillAssets(val[0],val[1])
    return render_template("wallet.html", wallet=wallet_handler,owned_cryptos = owned_cryptos,overall_value=overall_value_formatted,email_for_nav = session['email'])
```

Figure 2 ShowWallet() source code

4.3 Market Crypto Trading

The Market Trading page displays a table of the top crypto currencies ranked in decreasing order based on their respective market cap. In addition to the currencies' name and logo, each row of the table displays the currency's volume (number of times the coin has changed hands over the last 24 hours). The HTML table element was used to construct the table and all the currency's data has been retrieved via the API. By use of a for loop we were able to populate the table with the received data quickly.

```
<table class="bitland-table table table-striped table-hover">
    <thead>
        <tr>
            <th scope="col">Rank</th>
            <th scope="col">Coin</th>
            <th scope="col" width="10px">Name</th>
            <th scope="col" width="10px">Price</th>
            <th scope="col" >Volume</th>
        </tr>
    </thead>
    <tbody>
        {% for crypto in cryptos %}
            <tr>
                <td>{{ crypto["rank"] }}</td>
                <td><img src="{{ crypto['image'] }}" alt="{{ crypto['name'] }}" width="50" height="50"/></td>
                 <td>{{ crypto["name"] }} (<a href="/coin/{{ crypto['name'] }}">{{ crypto["symbol"]|upper }}</a>)</td>
                    <td><a href="/coin/{{ crypto['name'] }}">{{ crypto["symbol"] }}</a></td>-->
                <td>{{ crypto["price"] }}</td>
                <td>{{ crypto["volume"] }}</td>

            </tr>

        {% endfor %}
    </tbody>
</table>
```

Figure 3 HTML Table with Crypto Data

```
230    @app.route('/market')
231    @login_required
232    def market():
233        return render_template("market.html",currency = chosen_currency, cryptos = cryptos)
234
```

Figure 4 Rendering market.html

4.4 The Coin Page

The user can then navigate to any of the coin's webpage by clicking on the coin. The page displays more information, such as the current price, the currency's market cap change and when did the last update occur. Like market.html, coin.html displays its information in a HTML table populated by the data collected from the API using Jinja2.

If the user has signed in, they will be able to purchase the presently selected currency.

The values submitted in the purchase form are dynamically converted from USD to the Currency's units or vice versa using JavaScript at the backend.

The user can then buy the asset by going to the payment page (Not required to enter card details). After that, assets are directly added to the wallet.

In the case that the user has not yet logged in, they will be prompted to do so in order to continue purchasing that currency.

```
<form  action=" " id="buy" class="form" name="buy" method="POST" >
    {{ form.csrf_token }}
    <div class="row">
        <div class="form-group col-12 mb-3">
            {{ form.amount.label }}
            {{ form.amount(style = "border: 1px solid Black") }}
        </div>
        <div class="form-group col-12 mb-3">
             {{ form.amount_receive.label }}
             {{ form.amount_receive(style = "border: 1px solid Black") }}
        </div>

        <div class="form-group col-lg-12 mb-0 text-center">
            <div class="actions">
                {{ form.submit(class = "btn btn-contact-bg") }}


            </div>
        </div>
    </div>

</form>
<script>
    var inputHtml = '';


    var amount = document.getElementById('amount');
    var amount_receive = document.getElementById('amount_receive');
    amount.onchange = function() {
    amount = amount.value;
    out = amount * {{ crypto_details["current_price"]|tojson }};
    document.getElementById("amount_receive").value = out;
    }
    amount_receive.onchange = function() {
    var out1 = 0;
    amount_rec = amount_receive.value;
    out1 = amount_rec / {{ crypto_details["current_price"]|tojson }};
    document.getElementById("amount").value = out1;
    }
```

Figure 5 Dynamic conversions of USD-Crypto code

4.5 Candle Stick Graphs

Candle stick graphs are embedded within the top of the Market Trade page scrolling horizontally displaying many of the currently trending currencies.

In addition to this each Currency' s page also displays a graph showing recent trends in the specific crypto currency' s market. This will prove to be very useful as it enables user to see past dips and growths aiding them to decide purchasing or selling.

Advanced charts were taken from Binance api widget. We integrated them within the website by making changes to the JavaScript to fit our website.

4.6 Peer-to-Peer Cryptocurrency Trading

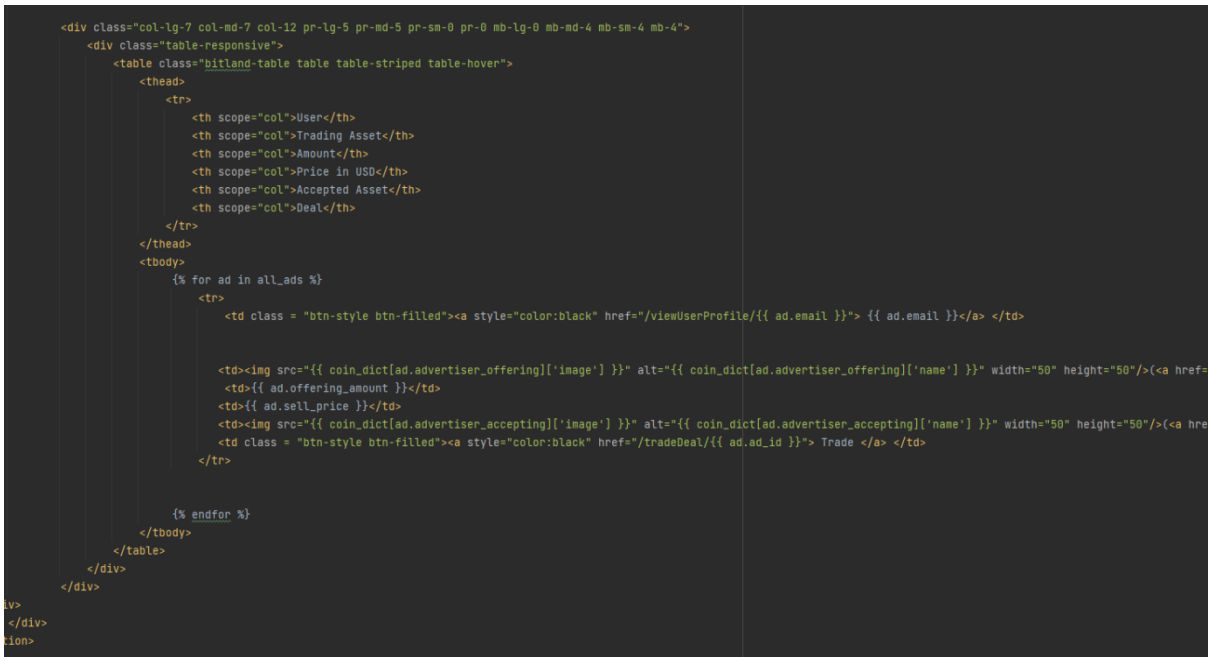
Developed to facilitate the exchange of cryptocurrency between users, the user may select to trade any of their owned currencies, establish the asking price, and the currency the user is willing to accept in return for the listed cryptocurrency, all from the user's wallet. When a user selects Upload, the trade deal is published on our website's Peer-to-Peer Trading page.

All submitted adverts are displayed on the Peer-to-Peer Trading page, along with the user who is selling it, the currency being sold, its USD value, and lastly the accepted currency.

If a user is interested in an advertised trade deal and has enough of the asking currency, they can initiate a trade and the transaction takes place and a trade summary is sent to both parties.

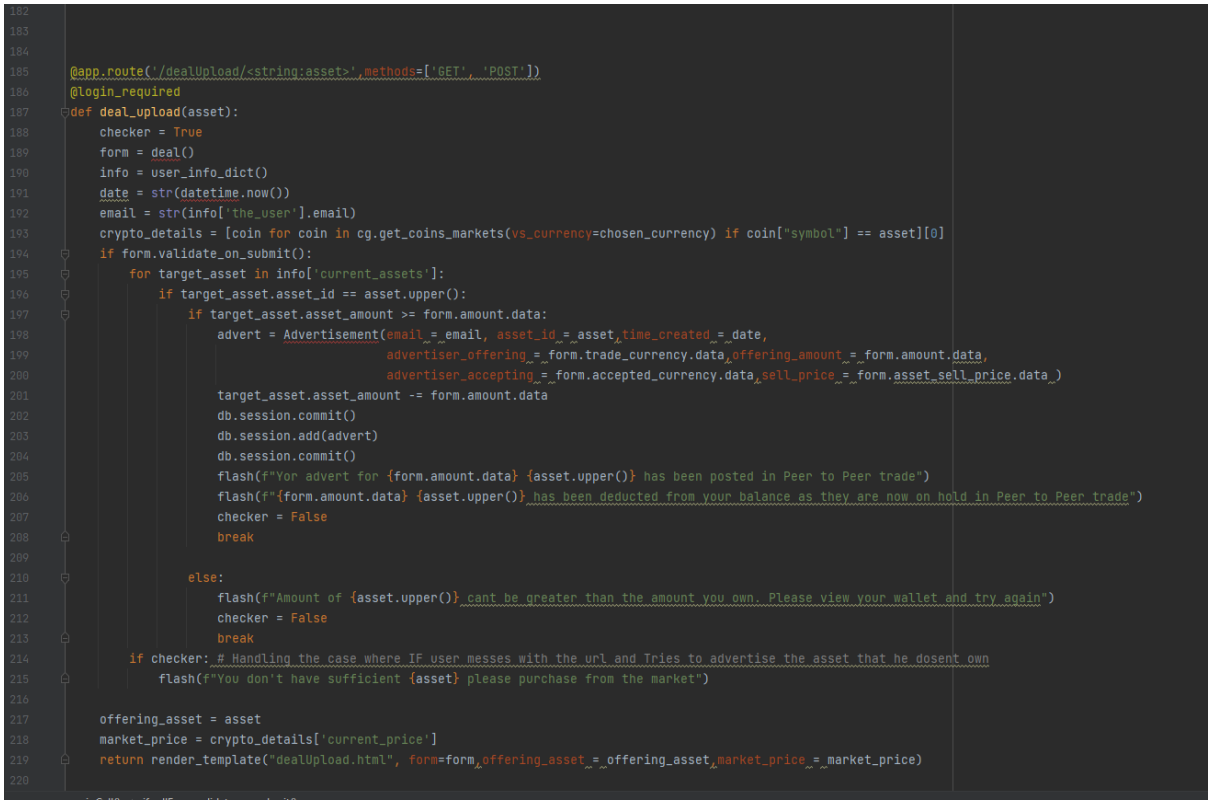
Proper error handling has been implemented to handle case in which the user has insufficient funds and to prevent self-trading.

User profiles are also viewable within the Peer-to-Peer trading page where they can see user’s bio profile and date joined.



```
        <div class="col-lg-7 col-md-7 col-12 pr-lg-5 pr-md-5 pr-sm-0 pr-0 mb-lg-0 mb-md-4 mb-sm-4 mb-4">
            <div class="table-responsive">
                <table class="bitland-table table table-striped table-hover">
                    <thead>
                        <tr>
                            <th scope="col">User</th>
                            <th scope="col">Trading Asset</th>
                            <th scope="col">Amount</th>
                            <th scope="col">Price in USD</th>
                            <th scope="col">Accepted Asset</th>
                            <th scope="col">Deal</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for ad in all_ads %}
                            <tr>
                                <td class = "btn-style btn-filled"><a style="color:black" href="/viewUserProfile/{{ ad.email }}"> {{ ad.email }}</a> </td>


                               <td><img src="{{ coin_dict[ad.advertiser_offering]['image'] }}" alt="{{ coin_dict[ad.advertiser_offering]['name'] }}" width="50" height="50"/>(<a href=
                                <td>{{ ad.offering_amount }}</td>
                               <td>{{ ad.sell_price }}</td>
                               <td><img src="{{ coin_dict[ad.advertiser_accepting]['image'] }}" alt="{{ coin_dict[ad.advertiser_accepting]['name'] }}" width="50" height="50"/>(<a hre
                               <td class = "btn-style btn-filled"><a style="color:black" href="/tradeDeal/{{ ad.ad_id }}"> Trade </a> </td>
                            </tr>


                        {% endfor %}
                    </tbody>
                </table>
            </div>
        </div>
iv>
 </div>
tion>
```

Figure 6 Peer-to-peer Template Core



```python
@app.route('/dealUpload/<string:asset>',methods=['GET', 'POST'])
@login_required
def deal_upload(asset):
    checker = True
    form = deal()
    info = user_info_dict()
    date = str(datetime.now())
    email = str(info['the_user'].email)
    crypto_details = [coin for coin in cg.get_coins_markets(vs_currency=chosen_currency) if coin["symbol"] == asset][0]
    if form.validate_on_submit():
        for target_asset in info['current_assets']:
            if target_asset.asset_id == asset.upper():
                if target_asset.asset_amount >= form.amount.data:
                    advert = Advertisement(email = email, asset_id = asset,time_created = date,
                                           advertiser_offering = form.trade_currency.data,offering_amount = form.amount.data,
                                           advertiser_accepting = form.accepted_currency.data,sell_price = form.asset_sell_price.data )
                    target_asset.asset_amount -= form.amount.data
                    db.session.commit()
                    db.session.add(advert)
                    db.session.commit()
                    flash(f"Yor advert for {form.amount.data} {asset.upper()} has been posted in Peer to Peer trade")
                    flash(f"{form.amount.data} {asset.upper()} has been deducted from your balance as they are now on hold in Peer to Peer trade")
                    checker = False
                    break

                else:
                    flash(f"Amount of {asset.upper()} cant be greater than the amount you own. Please view your wallet and try again")
                    checker = False
                    break
        if checker: # Handling the case where IF user messes with the url and Tries to advertise the asset that he dosent own
            flash(f"You don't have sufficient {asset} please purchase from the market")

    offering_asset = asset
    market_price = crypto_details['current_price']
    return render_template("dealUpload.html", form=form,offering_asset = offering_asset,market_price = market_price)

```

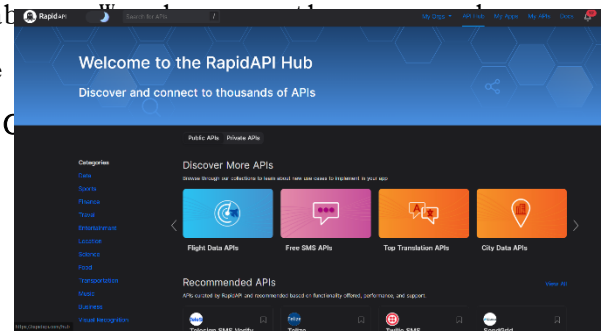
Figure 7 Deal Publishing

```python
@app.route('/tradeDeal/<int:ad_id>')
@login_required
def trade_deal(ad_id): # This method makes the deal happen between two peers

    ad = Advertisement.query.filter_by(ad_id = ad_id).first()
    ad_data = Advertisement.query.filter_by(ad_id=ad_id).first()
    user = user_info_dict()
    coin_dict = coin_info_dict()
    transaction = {'advertiser':"",'offered_asset':"",'paid':""}
    email_for_nav = user['the_user'].email

    if ad.email != user['the_user'].email: # handling the case where Same user tries to make his own deal
        for user_asset in user['current_assets']:
            if user_asset.asset_id == ad.advertiser_accepting:
                # we have record of price in Crypto to USD (NOT CRYPTO TO ANOTHER CRYPTO eg BTC -ETH)
                if user_asset.asset_amount * coin_dict[user_asset.asset_id.upper()]['current_price'] >= locale.atof(ad.sell_price.strip("$")):
                    transaction['paid'] = locale.atof(ad.sell_price.strip("$")) / coin_dict[user_asset.asset_id.upper()]['current_price']
                    user_asset.asset_amount -= transaction['paid']
                    db.session.commit()
                    addToWallet(ad.offering_amount,ad.advertiser_offering) # deal completes here
                    # I don't need to remove from the advertiser's wallet because
                    # when an advertiser uploads a deal their assets are deducted and are on hold
                    # in that deal. If the deal is successful like here. The assets are not needed to be deducted
                    # as they were already on hold.

                    transaction['advertiser'] = ad.email
                    transaction['offered_asset'] = (ad.asset_id,ad.offering_amount)
                    db.session.delete(ad)
                    db.session.commit()
                    return render_template("tradeDeal.html",coin_dict=coin_dict,transaction=transaction,ad=ad_data,email_for_nav = email_for_nav)

                else:
                    flash(f"You do not have sufficient {ad.advertiser_accepting} to make this deal Please Purchase more from the market.")
                    break
    else:
        flash(f"Please don't try to self trade.")

    return render_template("tradeDeal.html",email_for_nav = email_for_nav)
```

Figure 8 Carrying out the deal

4.7 Displaying NFTs

The webpage NFT collections, consists of our front-end (HTML, JavaScript with React library & CSS), and our back-end (Python). Behind the scenes we utilise the API that is sourced from the host https://rapidapi.com/hub which fetches all data displayed of **Rarible NFTs marketplace.** With the generated API key that we get from the host, we get access to each NFT collection that is available from their datab
total of NFTS in the specific collections we
collections we picked and obtained were **Hot C**
collection.

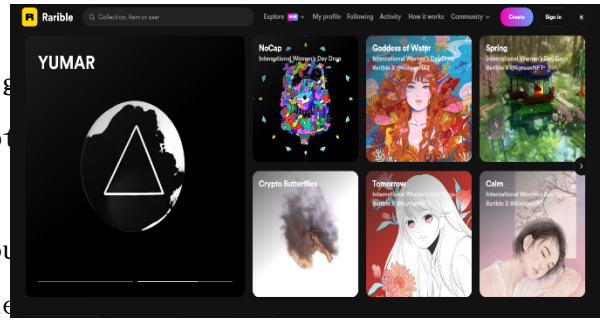


RAPID API: HOST

RapidAPI is the largest API Marketplace in the world, with over one million developers using it to find and connect to hundreds of APIs.

RapidAPI allows developers to find and test APIs, subscribe to them, and connect to them all with a single account, API key, and SDK.

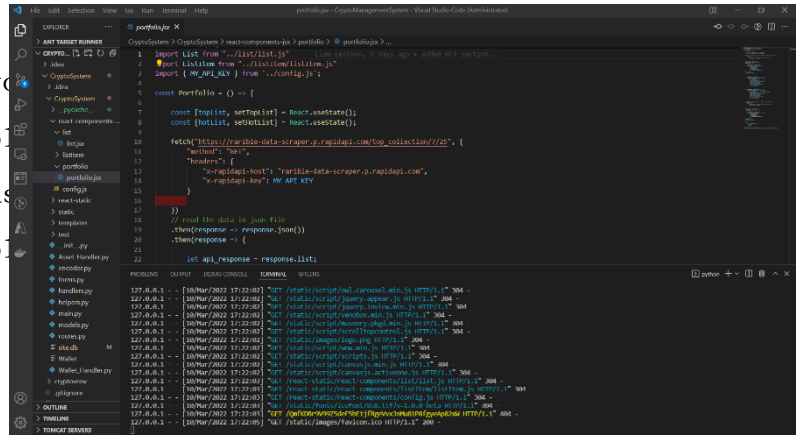
RARIBLE MARKET PLACE: API



Rarible is a piece of software that allows dig
and creators to create and sell bespoke crypt
that represent ownership of their work.
Rarible is noteworthy because it is a distribu
based on Ethereum that allows them to be trade

Non-fungible tokens are the tokens that creators create on Rarible (NFTs). Each NFT is one-of-a-kind, and unlike bitcoins (or other forms of currency), they cannot be exchanged. This attribute is referred to as fungibility, which is why Rarible tokens are referred to as non-fungible.

FRONT-END:

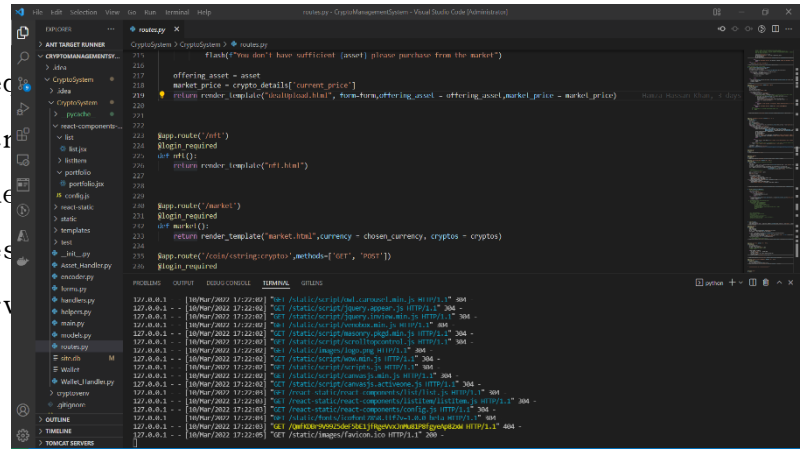


Inspired Netflix UI design where yo
And scroll sideway to view each col
like the trending movie collections
Double click to go the specific col
such “CyberBrokers” on Rarible.

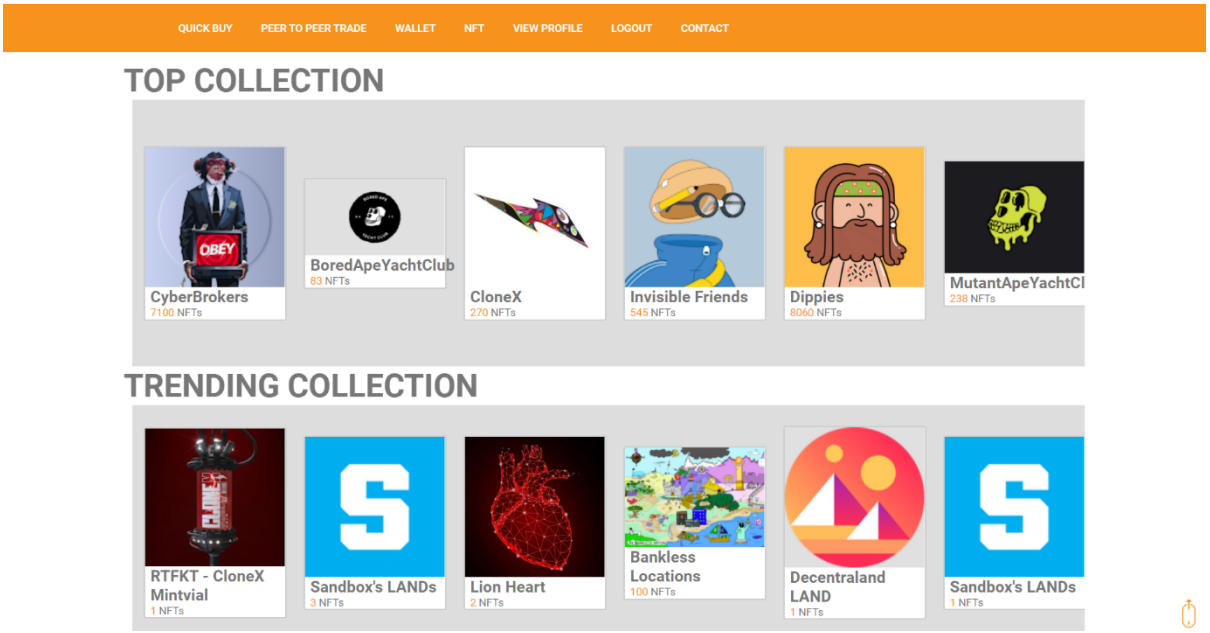
This features helps new user to get familiar
With NFT trading and range of market.

BACK-END:

Programmed in python to deliver rec
from site to view the most top & tr
collections available. This work he
to interact with back-end services
establish communications with serv



FINAL PRODUCT — NFT COLLECTION



Chapter 5

Lessons Learned

5.1 Functionalities Deemed Unfeasible

Initially, our ambition pushed us to investigate a wider range of features that we wanted to incorporate within our project. However, we realised that adding additional functions would take us away from what we had laid down in Chapter 2. Therefore, rather than wasting time on extra features which we felt were not critical to the project we decided to upgrade our existing components.

5.1.1 Prediction of Values

We were particularly interested in incorporating the concept of projecting future currency values. However, because our product was aimed at crypto trading newbies, the idea was deemed unneeded. Furthermore, value prediction would have necessitated abandoning our present candlestick graph in favour of manually implementing a line graph.

5.1.2 NTF Generating/Trading

Previously, we hoped to include the generation and trading of NFTs among our users. However, considering that our target demographic was comprised of beginners just getting started in the field of Crypto trading, as well as the fact that NFTs were a new technology. Instead, we presented an introduction to the new concept and pointed them to an outsourced specialized NFT trading webpage.

5.2 Major Blockers

Over the course of the project, we ran into some large challenges which we had to overcome in order to complete the project on schedule and to the highest quality possible. We learned many things in dealing with these challenges. For some we were able to adapt our process in order to resolve them or insure they do not happen in the future. Others are skills that will take many years to master. By dealing with such major blockers we were able to work as a team better to help overcome said challenge.

5.2.1 Halfway mark

Throughout the first half of the semester all team members kept regular meetings twice a week to code together. However, after reaching the second half of the semester we realised we needed to hold more meetings in order to maximise our productivity. Granted the early weeks of the semester required us to draw up the idea of the project and for all team members to agree on the project design. In future projects we feel an increased frequency of meetings, especially in the design phase, would aid in communication and increase productivity.

5.2.2 Irreversible Git Commit

Certain git commands, such as ‘git clean -fxd’, have effects that cannot be reversed. One such command is ‘git reset —hard <commit hash>’. It reverts the repository to the specified commit, deleting all changes made in the working tree and in subsequent commits. One of our team members made changes to the git repository and was unable to merge with the remote repository. After a long time spent trying to resolve the issue they

decided the best option was to reset the repository to the head and delete everything in the working tree. This resulted in five hours of lost code. A better solution would have been to do a ‘git reset —soft’ which would have kept the changes staged. The files which were causing the issue could then be unstaged and a push could be attempted again.

We have therefore resolved that these commands should only be used when you have a thorough understanding of what the command does in general and what it will do in your specific case. It is also advisable to search for a less permanent solution to the problem.

5.2.3 Communication Issues

As the NFT section is somewhat separate from the rest of the project it was decided that a team would form to fulfil requirement 2.2.4 (displaying collections of NFTs). The two-man team split from the main group at the start of week 6 and there was little communication between the two teams. Unfortunately, there was a large gap in expectations of the finished product. The NFT team thought of it mainly as a gallery for viewing NFTs with external links to a website where you could trade NFTs. Other team members had a different view in mind where the user could view and trade NFTs.

Due to the lack of communication this issue was not addressed at the design stage and it was only with four days left before the video of the finished product was due that the issue was uncovered. This left little time for a redesign and reimplementation of the NFT section.

Therefore, we decided to continue with the view laid out by the NFT team that the section should be a gallery for viewing NFTs.

This was the best solution we could implement at the time in order to mitigate the problems that stemmed from the lack of communication. A better approach would have been to meet at the end of the design phase of the NFT section to discuss the chosen design. Had we done this we would have uncovered the problem sooner and had more time to arrive at an appropriate solution.

5.2.4 Common Development Platform

We decided to use a python virtual environment for the project, in doing so we were able to maintain the dependencies separately from other python projects. This virtual environment was put in the git repository. This

meant we had to settle on a common operating system for use in the project. Two members of the team run Ubuntu in their daily life and three use Windows. Therefore, we settled on windows as our operating system of choice and the two Ubuntu users were able to adapt to it.