



**SHAHEED ZULFIKAR ALI BHUTTO
INSTITUTE OF SCIENCE AND TECHNOLOGY**

Final Year Project Report

NFT Generator, Token Generator & NFT Bulk Uploader

Project Team:

Guman Singh 1812297
Kashif 1812309

Date: 31st, July, 2022

Project Supervisor:

Dr. Imran Amin

Submitted in the partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science in the

Faculty of Computing and Engineering Sciences

Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology (SZABIST)
Karachi Campus

Declaration of Authorship

We Guman Singh (1812297) and Kashif (1812309) declare that this report “NFT Generator, Token Generator& NFT Bulk Uploader” and the work presented in this report is our own.

The work has been done completely while in the candidacy for a bachelor’s degree at Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology (SZABIST) Karachi, any report previously submitted on this topic in this university or any other institution is clearly mentioned in this report. Everything we used in this report which is submitted by others or belong to any other person or organization is stated in this report.

We have always cited the work we have used. We have acknowledged all source of help we used for this report. The report is based on the research work done by the team members with, we have clearly stated the sources where we took help from to conduct the research.

Signed: Guman Singh (1812297)
Kashif (1812309)

Date:
27th July, 2022

Project Description

NFT market is a booming industry these days, everyone wants to generate NFTs to sell them on marketplaces and earn profit, but due to no availability of any GUI tool many non-developers and non-programmers are unable to generate their desired collections, the available tools are basically ruby and

JavaScript based written codes that non coders cannot operate due to that they pay high prices on Fiverr and other freelancing platforms to get their NFTs generated also non developers have to pay gas fee on NFT marketplaces to mint their NFTs, we are making a GUI based tool using that they can easily generate NFTs without having prior programming knowledge with few simple clicks and can bulk upload their collections to any existing marketplaces.

We are developing a GUI based NFT Generator that is aimed to target non- developers audience to get start in the NFT market without paying gas fee and make their own personalized collections and get a share of the pie by owning or selling their NFTs. The tool will be made using python and the minting process of the NFT's will be done through Solidity using smart Contracts and deployed on Open Sea marketplace using bulk uploading. NFT stands for on-fungible token that is a unique and non-interchangeable which will be stored on block-chain to backtrack its source. NFT can be any digital asset, photo, audio, video or any such file. NFTs use block-chain technology, each NFT is owned by someone and its proof of ownership is stored on block-chain and the ownership can be transferred to the next owner.

Acknowledgement

In the name of ALLAH, the most beneficent and merciful, who gave us the knowledge and courage to work on this project.

We are grateful for the outcome and success of this project over the year are gratitude towards the people who have provided us with the guidance and assistance to be able to complete this project in such a difficult time.

We would like to thank our supervisor “Dr. Imran Amin” head of the Computer Science at Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology. He was integral part in the project as he was always there when we would get stuck at a point in our project. He consistently guided us, motivated us and cooperated with us throughout the duration of this project.

We would like to thank to Dr. Adeel Ansari who guided us and helped us in documentation, Sir Asif Khalid who motivated us and guided through out the journey of FYP, he was always there whenever we needed him, we would also like to thank the teachers at Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology, who guided us and taught us throughout our time in the university.

We would like to express our gratitude to our parents and family members who helped and encouraged us during this time. Furthermore, we would like to thank the staff at SZABIST for allowing us to use their labs and services to be able to complete the project.

Lastly, we would like to extend our gratitude to everyone at Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology for creating and environment for students to thrive in. The quality of education, the cooperative faculty members and the motivation provided by them.

Contents

Declaration of Authorship	2
Project Description	3
Acknowledgement	4
Project Proposal	8
1 Introduction	9
2 Objective	10
3 Problem Description	10
4 Methodology	10
5 Project Scope	10
6 Features	11
7 Feasibility Study	11
8 Solution Application Areas	12
9 Tools/Technology	12
10 Expertise of the Team Members	12
11 Milestone	13
12 Project Schedule	14
13 Work breakdown structure	15
Software Design Specification	16
1. Introduction	17
1.1 Purpose of this document	17
1.2 Scope of the development project	17
1.3 Definitions, acronyms, and abbreviations	17
1.4 References	17
1.5 Overview of this document	17
2. System architecture description	19
2.1 Section Overview	19
2.2 General Constraints	19
2.3 Data Design	19
2.4 Program Structure	20
2.5 Alternatives Considered	20
3. Detailed description of components	20
3.1 Section Overview	20
3.2 Component Detail (include a sub-section for each component)	20
4. User Interface Design	24
4.1 Section Overview	24

4.2 Interface Design Rules	24
4.3 Detailed Description	25
5. Reuse and relationships to other products	26
6. Design decisions and tradeoffs	26
7 Pseudocode for components	28
Software Design Specification	34
1. Introduction	35
1.1 Purpose of this document	35
1.2 Scope of the development project	35
1.3 Definitions, acronyms, and abbreviations	35
1.4 References	35
1.5 Overview of this document	36
2. System architecture description	37
2.1. Section Overview	37
3. Detailed description of components	39
4. User Interface Design	40
Software Requirements Specification	48
1. Introduction	49
1.1 Purpose	49
1.2 Document Conventions	49
1.3 Intended Audience and Reading Suggestions	49
Software Requirements Specification	59
1. Introduction	60
1.5 References	60
2. Overall Description	61
2.1 Product Perspective	61
2.2 Product Functions	61
2.3 User Classes and Characteristics	61
2.4 Operating Environment	61
2.5 Design and Implementation Constraints	61
2.6 User Documentation	61
2.7 Assumptions and Dependencies	63
3. External Interface Requirements	63
3.1 User Interfaces	63
3.2 Hardware Interfaces	67
3.3 Software Interfaces	67

3.4 Communications Interfaces	67
4. System Features	67
5. Other Nonfunctional Requirements	71
5.1 Performance Requirements	71
5.2 Safety Requirements	71
5.3 Security Requirements	71
5.4 Software Quality Attributes	71
5.5 Business Rules	71
6. Other Requirements Appendix A: Glossary	72
Test Cases	73
User Manual	80
Student Log Form	81
Iteration Plan	83
Gantt Chart	84
Plagiarism Report	85
PLAGIARISM FREE CERTIFICATE	86

Project Proposal for NFT Generator

**Prepared by
Guman Singh 1812297
Muhammad Kashif 1812309**

**Faculty of Computing and Engineering Sciences
Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology, Karachi**

7/28/2022

1 Introduction

We are developing a GUI based NFT Generator that is aimed to target non- developers audience to get start in the NFT market without paying gas fee and make their own personalized collections and get a share of the pie by owning or selling their NFTs. The tool will be made using python and the minting process of the NFT's will be done through Solidity using smart Contracts and deployed on Open Sea marketplace using bulk uploading.

NFT stands for on-fungible token that is a unique and non-interchangeable which will be stored on block-chain to backtrack its source. NFT can be any digital asset, photo, audio, video or any such file. NFTs use block-chain technology, each NFT is owned by someone and its proof of ownership is stored on block-chain and the ownership can be transferred to the next owner.

- The highest sold NFT isEverydays: the First 5000 Days in (\$69.3 million)



The process required to make a NFT requires knowledge of tools such as the Adobe Illustrator or to create a Collection the creator must be a programmer to generate the collection of say 1000 or 10,000 NFTs. Our Tool is aimed to help Non-coders to make their own NFTs and not get ripped off by paying high Gas fee on popular websites without any promises of growth

2 Objective

To develop a GUI based NFT generator for non-developers and programmers to generate NFT collections and mint them to any existing platforms with solidity and python.

3 Problem Description

NFT market is a booming industry these days, everyone wants to generate NFTs to sell them on marketplaces and earn profit, but due to no availability of any GUI tool many non-developers and non-programmers are unable to generate their desired collections, the available tools are basically ruby and

JavaScript based written codes that non coders cannot operate due to that they pay high prices on Fiverr and other freelancing platforms to get their NFTs generated also non developers have to pay gas fee on NFT marketplaces to mint their NFTs, we are making a GUI based tool using that they can easily generate NFTs without having prior programming knowledge with few simple clicks and can bulk upload their collections to any existing marketplaces.

4 Methodology

To accomplish this goal we are using Python and Solidity, for python we are going to use PyCharm IDE and for Smart Contracts we will use Solidity on remix - Ethereum IDE and VS Code IDE to write code for bulk uploading. Using Python we will create a GUI that gives options like randomize NFTs, choose any particular images to generate NFT(face1, hair1, mouth1 and so on), to save all NFTs in one click, to save particular NFT, to get started with generating first we will be storing base images in a dedicated folder with appropriate naming. Once the NFTs are generated we will write smart contract on solidity on remix IDE and then use that smart contract along with the code for bulk uploading all images to Open Sea marketplace.

5 Project Scope

First we will design the GUI, how many features should be there and what its UI design will be, we will not add much features, as our target audience is non-programmers though the idea is to implement important features, like generating random NFTs, which is very unique one, this feature is yet not available in all the tools available for NFT generating, it will generate NFTs in random order instead of usual sequential order, other feature we will be using is to save all images in one click instead of saving each image manually with renaming, also the user can save a particular image if he doesn't want to save all generated images, the user can also select particular attributes to generated particular type of image like(head1, eyes2, nose3, hair 2 and so on), for- example there are 5 face traits in base images folder then on GUI user will have choice to select face from the range of face1 to face5, this way the user can test particular attributes just to make sure how that will look after mass generating. After images got generated user can open output folder with single click on GUI. Having completed the

generator, we will be drawing base images on illustrator for sample work, once the base images are ready we will put them to input folder along with dedicated folder names, like all face images go to face folder and hair images go to hair folder and so on. After generating all the NFTs we will write smart contract for it to get the ownership of images on block-chain and will write a code to mint all images at once and bulk upload to Open Sea marketplace.

6 Features

FYP 1:

1. Input Folder
2. Test
3. Number of NFTs(slider)
4. Amount(input)
5. Output folder
6. Generate

FYP 2:

1. Check the Web3 Wallet
2. Sign in with wallet
3. Fetch the selected network from wallet
4. Fill the required fields
5. Deploy the Smart Contract
6. Contract Deployment Gas fee
7. Display transaction hash after deployment
8. Display contract address at the bottom

7 Feasibility Study

With above defined scope, would you be able to meet your project schedule? Do mention following aspects:

- i. **Risks Involved:** There are lots of risks involved once we generate NFTs, then we want to sell it using our own crypto currency, we will also be using some tool to manage our NFTs.
- ii. **Resource Requirement:** The program requires Laptop/PC for generating NFTs, for Bulk Uploading it requires VS Code, Internet Connection, Meta Mask account, Open Sea Account.

8 Solution Application Areas

Our target audience is NFT buyers and sellers, as we discussed above, the NFT is a booming industry since the Crypto Punk collection was sold in 5 billion dollars. Since then everyone wants to buy and sell the NFTs. To buy an NFT the buyers need to pay gas fee on marketplace let's say OpenSea (largest NFT marketplace) not only buyer the sellers also need to pay gas fee on uploading their collection to any Marketplace this makes it too costly. The programmers can use available ruby and JavaScript based codes to generate their collections but non-programmers cannot operate those codes due to the lack of programming skills so they have to hire NFT generators but taking the advantage of rise in NFT sell, the NFT generator also charges high cost on NFT collection, after getting the collection the seller has to hire NFT minter to write smart contract for his collection and upload all images to existing market place. This all makes NFT work more costly, to reduce this we are generating a GUI based tool which will be easier to use for non-programmers also, everyone can generate NFTs free of cost and using our Bulk uploading method they can upload their collection to Open Sea marketplace.

9 Tools/Technology

Our project is based on following Tools & Technologies:

- 1) PyCharm for Python
- 2) Remix - Ethereum IDE for Solidity
- 3) VS Code: to write code for bulk uploading
- 4) Anaconda, to setup environment (if needed) we will be using that environment in PyCharm.
- 5) Adobe Illustrator for drawing the base images for sample work.

10 Expertise of the Team Members

Team Member (Guman): Expertise in Web Development, Python, ML, block-chain, GameDevelopment, Java, C, Solidity, Cyber Security, AI, Graphics & Networking.

Team Member (Kashif): Expertise in Python, AI, C, Java, Solidity, Networking, block-chain.

11Milestone

Milestone 1

- Search for the Project
- Discuss the idea with the supervisor
- Get idea approved by the advisor
- Create Project proposal
- Create Gantt. Chart
- Create the Presentations for the project defense
- Get approved the project

Milestone 2

- Analyzing FR and NFR
- Finalize Requirements
- Prototyping the GUI
- Proto-typing FRs
- Implementing the Features

Milestone 3

- Creating base Images to feed the tool
- Testing the base Images on the tool
- Resolving issue if any with Images or Functionality\
- Adding NFR's and other miscellaneous tasks

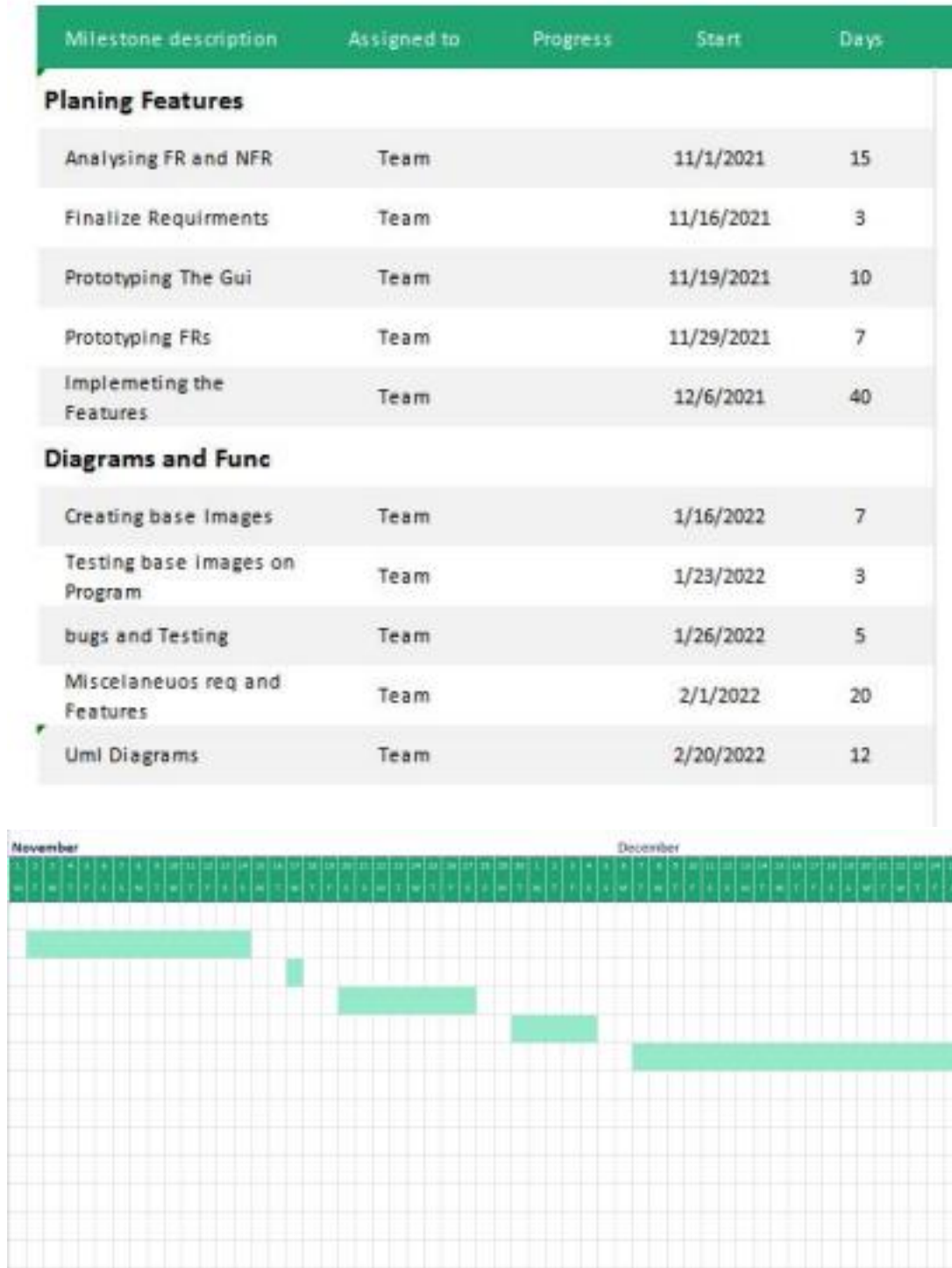
Milestone 4

- Analyzing the program for UML diagrams
- Creating all necessary diagrams that match the project description -Revising the Diagrams

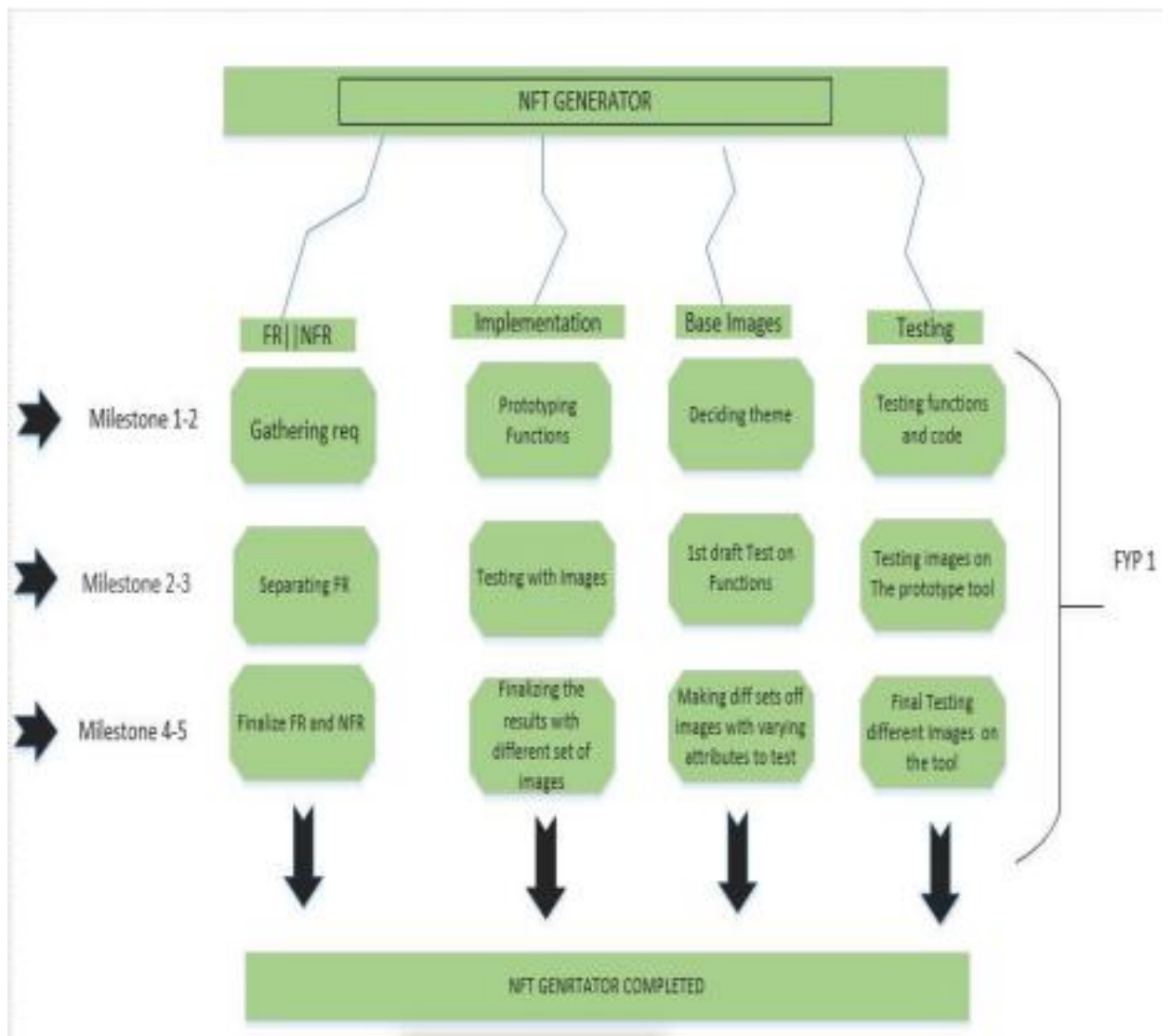
Milestone 5

- Minting the NFT's using solidity
- Meeting Project advisor for further feature adding (as advised by advisor) - Planning Measures to integrate features as directed -Learning and Integrating the Features with the NFT Generator

12 Project Schedule



13 Work breakdown structure



Software Design Specification for NFT Generator

**Prepared by
Guman Singh 1812297
Muhammad Kashif 1812309**

**Faculty of Computing and Engineering Sciences
Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology, Karachi**

4/28/2022

1.Introduction

1.1 Purpose of this document

This software design specification is made with the purpose of outlining the software architecture and design of the NFT Generator tool in detail. The document will provide developers an insight in meeting client's needs efficiently and effectively. Moreover the document facilitates communication and understanding of the tool by providing several views of the tool design. The document will provide a framework to the programmers through describing the high level components and architecture and interfaces.

1.2 Scope of the development project

Fonts: Times New Roman

Main heading: bold

Main heading font size: 18

Sub heading font size: 14

Font size heading1: 12

Other stuff: Bullet points

Main headings are in bold because user can easily distinguish it from others sub-headings, bullets use to identify special information about the website.

The rest of the document is written in Time New Roman.

1.3 Definitions, acronyms, and abbreviations

Acronyms:

- 1) SRS: Software Requirements Specification
- 2) SDS: Software Design Specification
- 3) NFT: Non Fungible Token
- 4) GUI: Graphical User Interface

Abbreviations

- 1) Number: No.
- 2) Document: doc

1.4 References

http://www.cs.iit.edu/~oaldawud/CS487/project/software_design_specification.htm
<https://thakurguman.medium.com/>
<https://www.presentationzeze.com/presentations/software-validation/software-validation-full-details/software-design-specification/>

1.5 Overview of this document

It is organized into 8 of sections, each section contains some info about the tool like section-1 contains Introductions of the tool like purpose, scope,

definitions, acronyms and references etc. Like wise section-2 is about the architecture description of the tool, including overview general constraints, data design, alternative considered and program structure, which contains functional hierarchy diagram of the tool. Similarly section-3 is about the detailed description of the components, like APIs their purpose, function and resources etc. Section-4 is about the user interface design, like design rules, GUI components and a detailed description. Section-5 is about reuse and relationships to other products, means if we have re-used our API or any code that we used somewhere else previously. Section-6 is about Design decisions and tradeoffs and section-7 contains Pseudocode for components and the last section-8 contains all the UML diagrams.

2. System architecture description

2.1 Section Overview

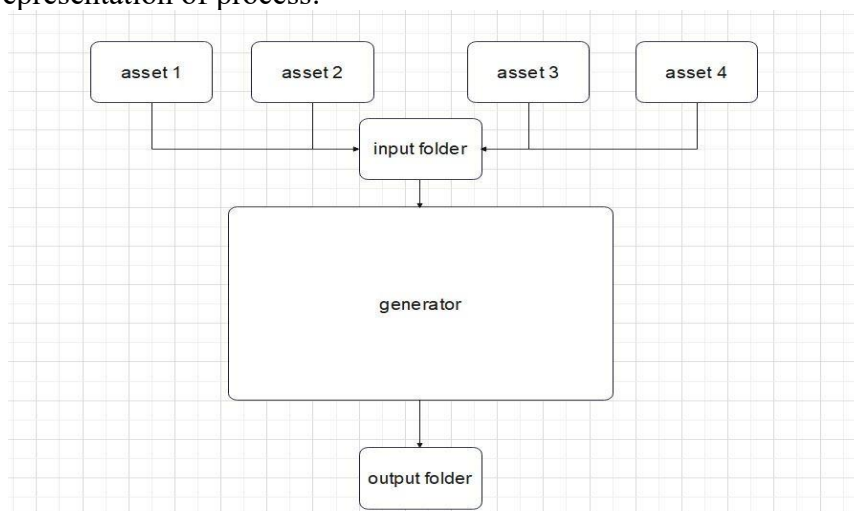
This section describes architectural design of the NFT Generator. The high level components and their interactions, suitable architectural patterns and design decisions applied to the complete tool. The next content of this section is General Constraints, which contains the requirements about the section, which includes hardware and software environments, interface requirements, external data representations, performance requirements, network requirements, etc. The Data Design section, describe the structure of any databases, external files, and internal data structures and the Program Structure describes the architectural model chosen and the pictorial representation major components. In last section alternatives considered, its justified if I have used any alternative model.

2.2 General Constraints

- ✓ The HTTP protocol will be used to facilitate communications between the UI and user.
- ✓ The UI of tool supports Google Chrome, Mozilla Firefox and all web browsers.
- ✓ The software and hardware requirements for the development of this project are not many and are open source.
- ✓ The work for the project is done with the current equipment and existing software technology.

2.3 Data Design

The tool is not using any database, it just inputs images from local storage and after generating NFTs dumps them to a default output folder. Following is the representation of process:



All the assets will be present in appropriate sub folders under a main folder, from program user will enter input folder name and the generator will combine the assets to generate all possible combinations out of those and will dump in default output folder.

2.4 Program Structure

First of all user has to put all the base images of NFTs in a particular folder under appropriate sub folders like all the weapon assets goes to weapon folder and all the soldier assets goes to soldier folder and so on. After that user will run the program with streamlit command on CMD or anaconda, the streamlit library will display the UI of tool on localhost of default browser. The user needs to input the name of folder which contains NFT base images and set the amount of NFTs he wants to generate, the user can also check if base images are perfect by clicking on test button which will generate an NFT. After specifying the amount the user wants he can click on generate button to generated specified number of NFTs in default output folder.

2.5 Alternatives Considered

A two-tier architecture is a software architecture in which a presentation layer or interface runs on a client, and a data layer or data structure gets stored on a server. Separating these two components into different locations represents a two-tier architecture, as opposed to a single-tier architecture. It also provides direct and faster communication. Here in our case we are considering user as a one layer and the interface as a second layer, the combination is called two-tier architecture, unlike traditional two-way architecture here data is not stored in any server instead all the data is inputted from local storage and dumped there too.

3.Detailed description of components

3.1 Section Overview

In this section we have disused the modules, subprograms, data files, control procedures, classes that we have used or created in our tool and what is the purpose, functionalities, dependencies of these components along with pseudo code of each component.

3.2 Component Detail (include a sub-section for each component)

1. Input_folder

Identification	Input_folder
Type	A data file

Purpose	The purpose of this feature is to input the folder which contains the assets
Function	Its function is to provide the base images to the program so the generator can generate more images by combining all the base images.
Dependencies	Input_folder is independent, it just loads the images of given folder to program, the given folder must contain required images, but even if that doesn't contain images, it will load that to

	program and throw an error later.
Interfaces	Interfaces are defined in detail below with attached screenshots.
Resources	No external resources are required.
Processing	<pre> 1 start 2 input_folder = path 3 check if images exist on given path 4 if "images exist" 5 then "load assets to generate" 6 else 7 return "FileNotFoundError" 8 end </pre>
Data	The images will be loaded, which will be used to generate NFTs later.

2. Test

Identification	Test
Type	
Purpose	The purpose of this feature is to test if the provided assets are able to generate an NFT.
Function	Its function is to check if the provided folder is empty and also to check if the images can be generated from the provided assets from input folder, if the assets are meeting the criteria of assets required for NFT.
Dependencies	Test function is dependent on input_folder, we cannot test images if we haven't load the images to program.
Interfaces	Interfaces are defined in detail below with attached screenshots.
Resources	No external resources are required.
Processing	<pre> 1 start 2 input_folder = path 3 check if images exist on given path 4 if "images exist" 5 return "sample image" 6 else 7 "throws an Error" 8 end </pre>
Data	Sample images will be generated to make sure we are ready to bulk generate by increasing the number of NFTs we want in amount sect

3. Amount

Identification	amount
Type	
Purpose	The purpose of this feature is to set the amount of NFTs user wants to generate
Function	Its function is to set the number of NFTs user wants to generate,

	the user can simply enter the NFTs number in amount field or can increase/decrease with slider option, user can also use + and - to increase single image or decrease single image.
Dependencies	Test function is dependent on input_folder, we cannot test images if we haven't load the images to program.
Interfaces	Interfaces are defined in detail below with attached screenshots.
Resources	No external resources are required.
Processing	<pre> 1 start 2 input_folder = path 3 check if images exist on given path 4 if "images exist" 5 then "user can increase/decrease amount of NFTs" 6 else if "images < 1" 7 return "Images must be greater than or equal to 1" 8 else 9 "Throws an error" 10 end </pre>
Data	Amount of NFTs will be set which the user wants to generate.

4. Generate

Identification	Generate
Type	
Purpose	The purpose of this feature is to generate the number of NFTs user set in amount section.
Function	Its function is to generate NFTs by the combining the different assets and making all possible combinations out of them.
Dependencies	Generate feature is dependent on input_folder, we cannot generate images if we haven't load the images to program.
Interfaces	Interfaces are defined in detail below with attached screenshots.
Resources	No external resources are required.
Processing	<pre> 1 start 2 input_folder = path 3 check if images exist on given path 4 if "images exist" 5 then "user can generate N number of NFTs" 6 else 7 "Throws an error" 8 end </pre>
Data	NFTs will be generated and saved to default output folder.

5. Output

Identification	Output
Type	
Purpose	The purpose of this feature is to save the generated NFTs
Function	Its function is to save the generated NFTs in the output directory, the loop will run till the no. Set in amount section it keeps saving images to output directory along with loop number, ex: 1,2,3... N.
Dependencies	Output feature is dependent on generate function, we cannot save the images to output folder, if we haven't generated the images.
Interfaces	Interfaces are defined in detail below with attached screenshots.

Resources	No external resources are required.
Processing	<pre> 1 start 2 if "NFTs generated" 3 then "NFTs will be stored in default folder" 4 else if "User changed output folder" 5 then "NFTs will be stored in new dir" 6 else 7 "Throws some error" 8 end </pre>
Data	NFTs will be saved to default output folder or to the some other folder incase user changed the output directory.

4. User Interface Design

4.1 Section Overview

This section contains details about User Interface Design of the tool, like what UI is used in what functionality, the interface design rules section contains info about the rules of interface design, suppose, Guideline for error messages, or if user is new so the tool can tour him/her complete tool, or we provide a tutorial video in popup, or coding rules to make sure user enters correct information in input fields. In Detail Description, screen shots of all the features are provided along with short description of each feature.

4.2 Interface Design Rules

Here are some of the interface design rules that user needs to take care about while using the interface features:

1. Internet is not required to use this tool.
2. Images in input folder must be properly arranged in an appropriate sub folder
3. Images must be in PNF format
4. Images must be of fixed size, ex: 300*300
5. The input folder box cannot be left empty
6. The user needs to press enter or click on test after entering the folder name
7. The minimum amount cannot be less than 1
8. The default folder to dump NFTs is output, but user can change it
9. The user can check if NFTs are properly generating by clicking on test option
10. The user must click on generate option to save all generated NFTs specified in amount section.

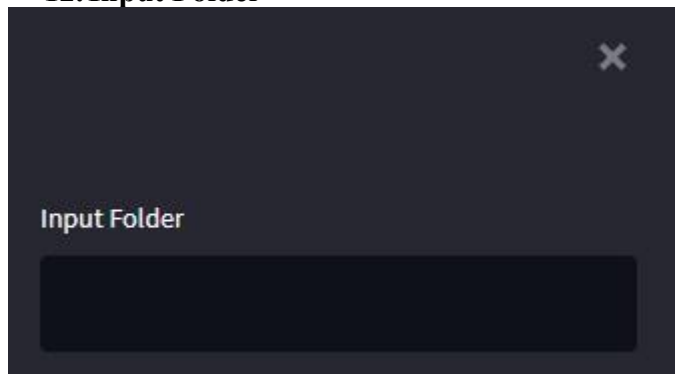
4.3 Detailed Description

11. UI



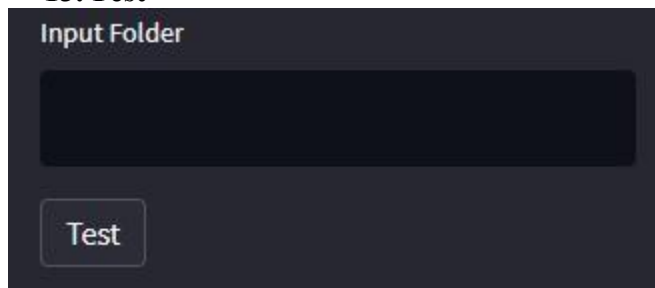
This is the UI of tool, that contains some sample NFTs and name of program and its creator, on the left side panel it contains all the features that are required to generate NFTs, on the right top corner it contains a menu that is default menu by streamlit library, using that we can do UI customization.

12. Input Folder



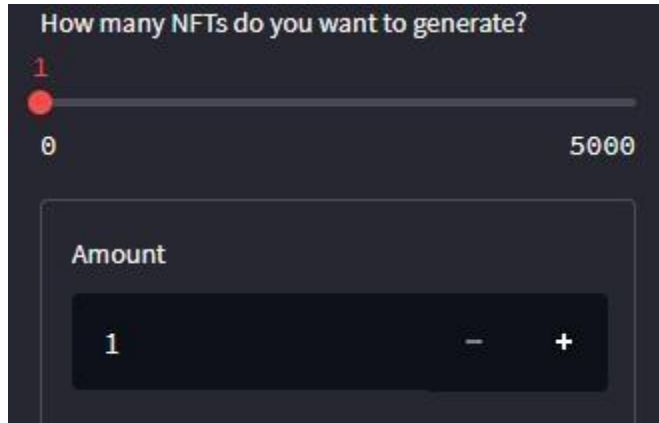
This feature will allow the user to input the folder that contains layers for NFTs, user has to enter the name of folder to load it in the program.

13. Test



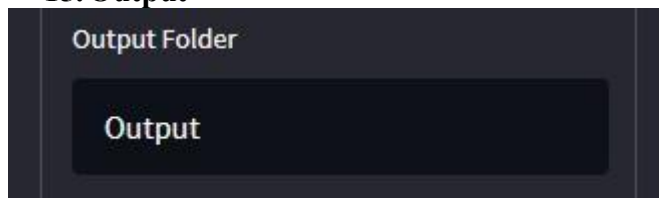
This feature will allow the user to test the layers he got from input folder, to verify if the images are generating from provided layers just before going for bulk generating.

14. Amount



These both features will allow the user to set number of NFTs they want to generate, either they can use above bar or can simply input the integer, they can also use + and - to simply increase or decrease images to certain number.

15. Output



This is the output directory, where all generated NFTs will be dumped. Folder output has been set as default folder for generated NFTs if user wish to change he can simply change the name from Output to any other name and new folder will be created automatically.

5. Reuse and relationships to other products

We have not reused any API, library, module or any such component in this project.

6. Design decisions and tradeoffs

For this tool we have used streamlit library, our initial idea was to go with traditional style of web development either go by react or angular. Since we had planned to use python on back-end, we decided to go with python for front-end also. Thus we considered Django and Flask but still we were looking for some advance and more efficient option and we come across streamlit library which is specially built for ML, DL and Data Science related projects, as it is a new library so it has limited functions so far but as compare to available web development frameworks and libraries it is one of the powerful library. Regarding the ideas that were abandoned, we were planning to add more features like the tool stores user's information and provide option for animated NFTs etc but that might get complicated for normal user as our aim was just to provide end user the simplest interface so he don't even have to think for second about how the tool will work and can easily generate NFTs with three clicks and here it is, the user only have to input the folder name and specify amount of NFTs he wants. Also the initial idea was to live this tool online but that comes in between our aim to of providing

NFT Generator offline and free, if we take this live we need to buy domain and hosting plus the proper promotion that causes if not much but at least of some

amount, and not getting it to online reduces risk of end user's NFTs got published online before he could even arrange the gas fee for deployment, since it is fully decentralized the end user need to worry about anything.

7 Pseudocode for components

1. Input folder

```
1 start
2 input_folder = path
3 check if images exist on given path
4 if "images exist"
5     then "load assets to generate"
6 else
7     return "FileNotFoundError"
8 end
```

2. Test

```
1 start
2 input_folder = path
3 check if images exist on given path
4 if "images exist"
5     return "sample image"
6 else
7     "throws an Error"
8 end
```

3. Amount

```
1 start
2 input_folder = path
3 check if images exist on given path
4 if "images exist"
5     then "user can increase/decrease amount of NFTs"
6 else if "images < 1"
7     return "Images must be greater than or equal to 1"
8 else
9     "Throws an error"
10 end
```

4. Generate

```

1 start
2 input_folder = path
3 check if images exist on given path
4 if "images exist"
5     then "user can generate N number of NFTs"
6 else
7     "Throws an error"
8 end

```

5. Output

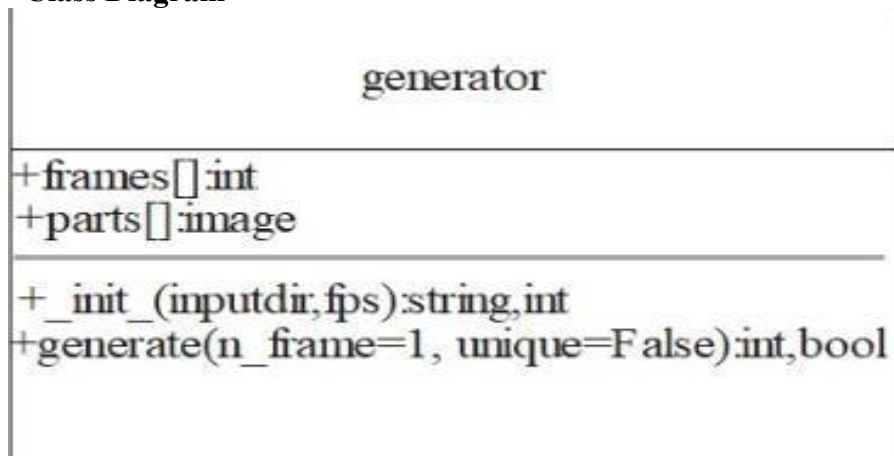
```

1 start
2 if "NFTs generated"
3     then "NFTs will be stored in default folder"
4 else if "User changed output folder"
5     then "NFTs will be stored in new dir"
6 else
7     "Throws some error"
8 end

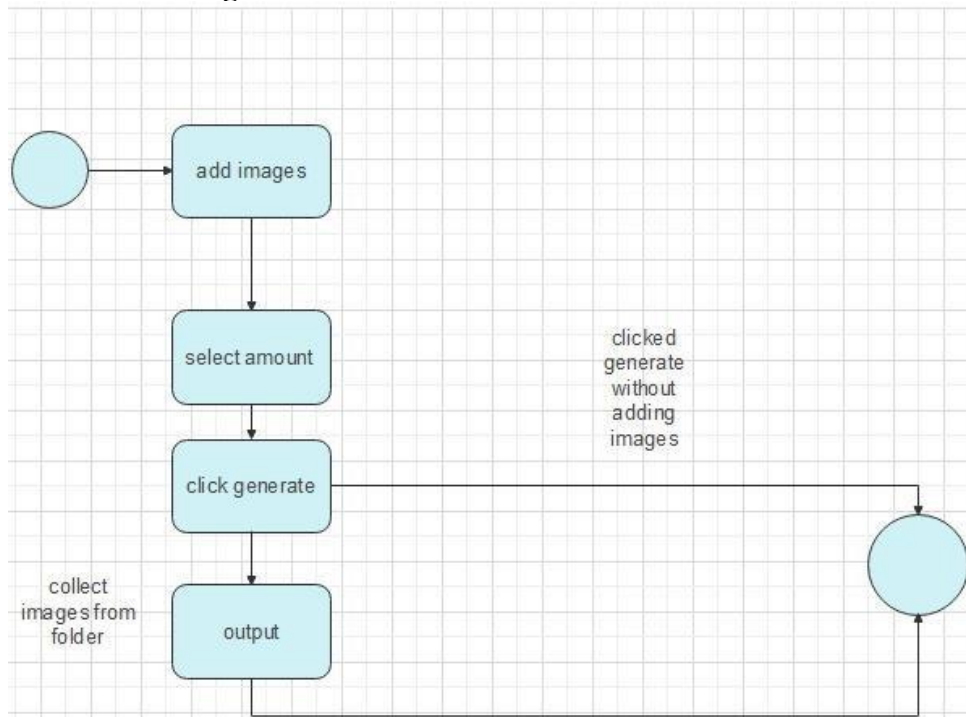
```

8.0 Appendices

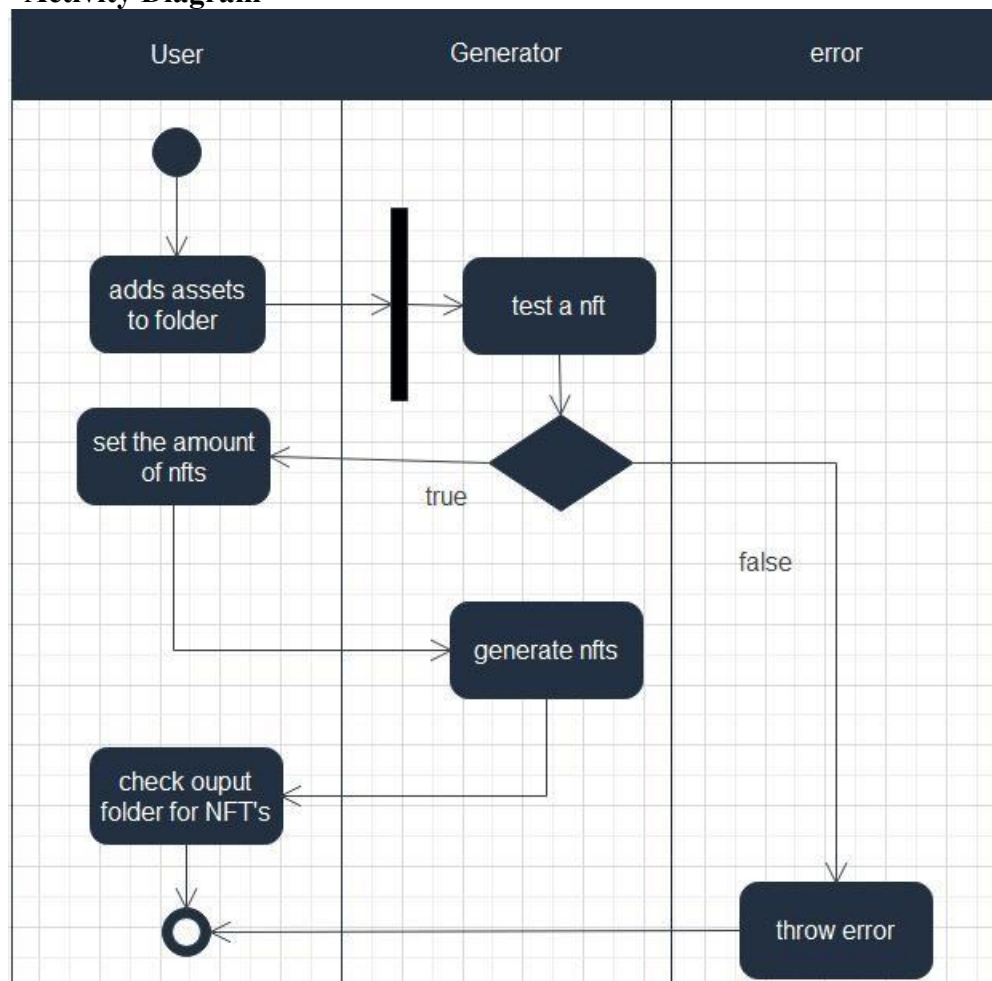
1. Class Diagram



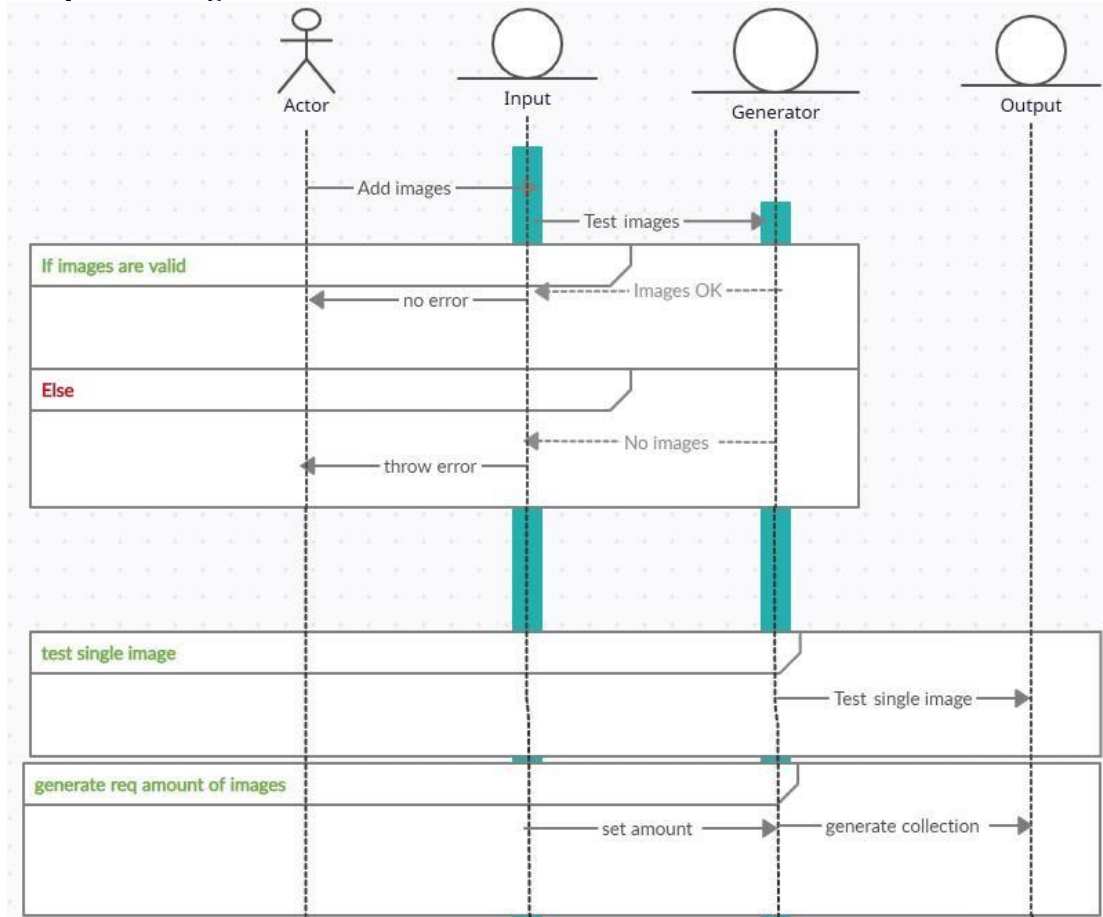
2. Statechart Diagram



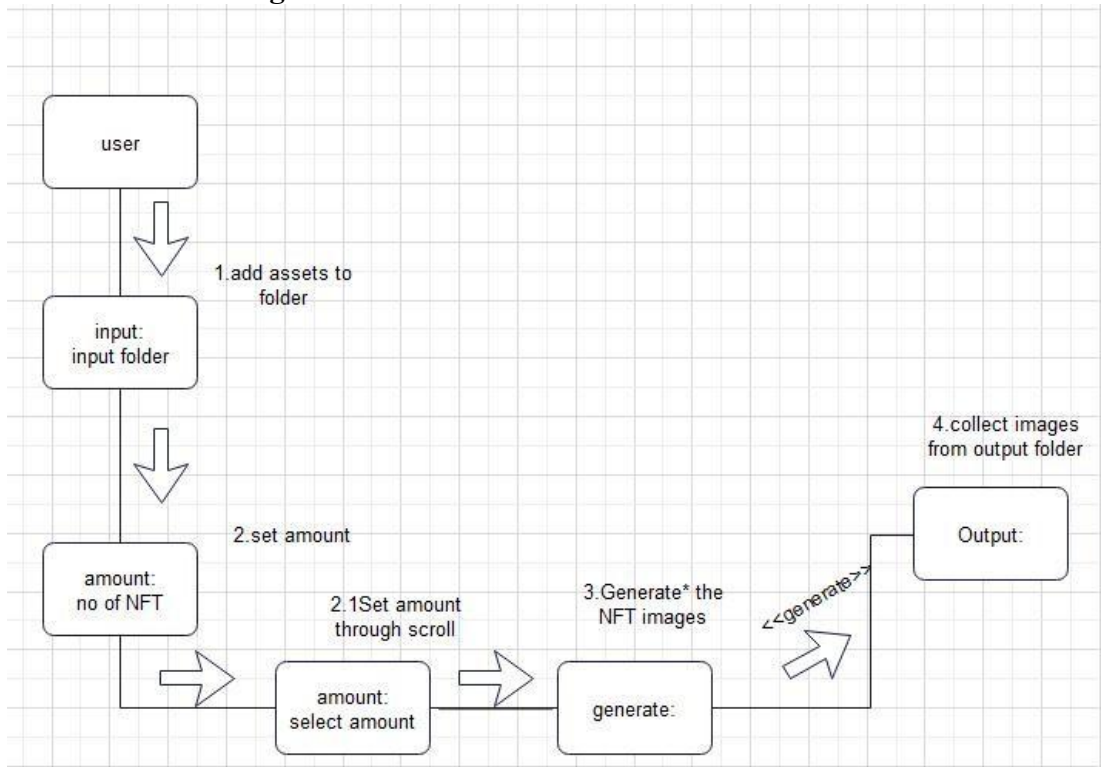
3. Activity Diagram



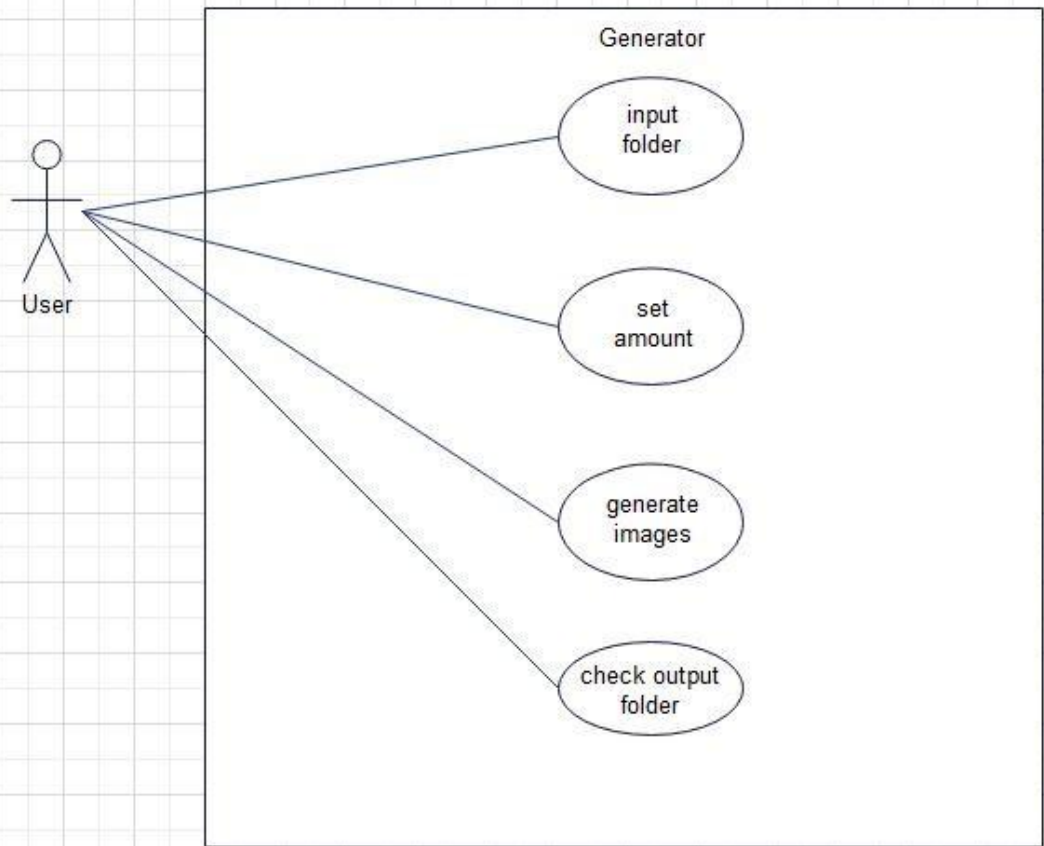
4. Sequence Diagram



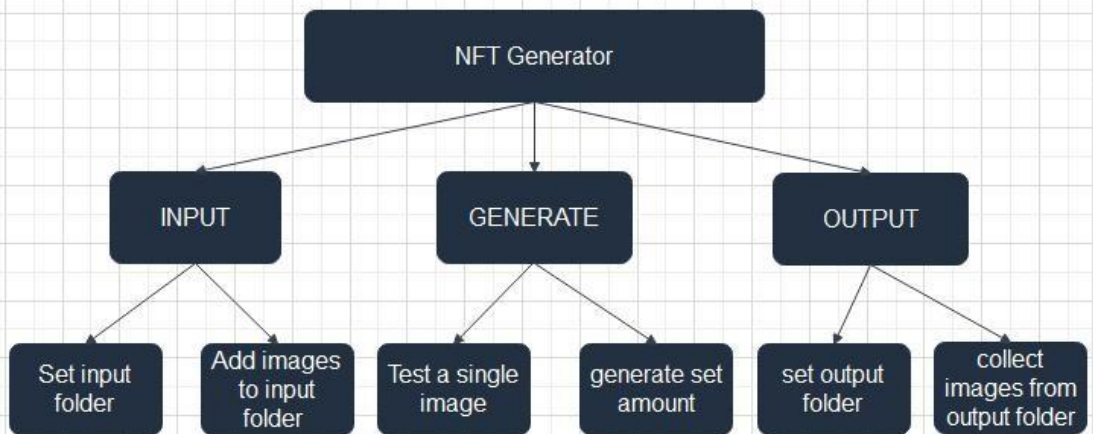
5. Collaboration Diagram



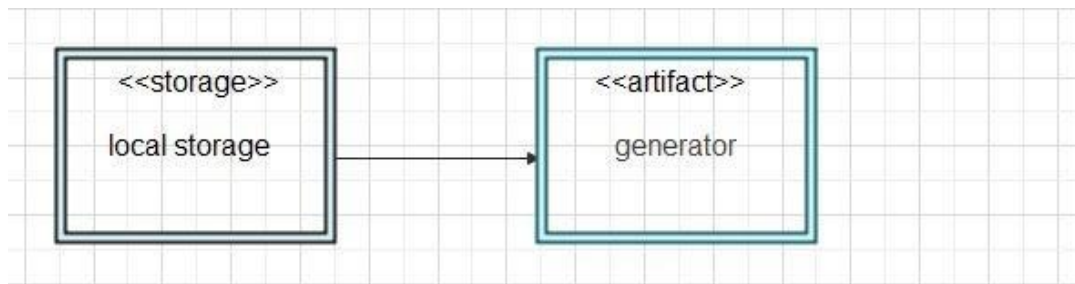
6. Use-case Diagrams



7. Functional Hierarchy Diagram



8. Deployment Diagram



Software Design Specification for Token Generator

**Prepared by
Guman Singh 1812297
Muhammad Kashif 1812309**

**Faculty of Computing and Engineering Sciences
Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology, Karachi**

5/8/2022

1. Introduction

1.1 Purpose of this document

This software design specification is made with the purpose of outlining the software architecture and design of the Token Generator tool in detail. The document will provide developers an insight in meeting client's needs efficiently and effectively. Moreover the document facilitates communication and understanding of the tool by providing several views of the tool design. The document will provide a framework to the programmers through describing the high level components and architecture and interfaces.

1.2 Scope of the development project

Fonts: Times New Roman

Main heading: bold

Main heading font size: 18

Sub heading font size: 14

Font size heading1: 12

Other stuff: Bullet points

Main headings are in bold because user can easily distinguish it from others sub-headings, bullets use to identify special information about the website.

The rest of the document is written in Time New Roman.

1.3 Definitions, acronyms, and abbreviations

Acronyms:

- 1) SRS: Software Requirements Specification
- 2) SDS: Software Design Specification
- 3) Metamask: a web3 wallet
- 4) token: crypto currency

Abbreviations

- 1) Number: No.
- 2) Document: doc

1.4 References

- 5) http://www.cs.iit.edu/~oaldawud/CS487/project/software_design_specification.htm
- 6) <https://thakurguman.medium.com/>
- 7) <https://www.presentationeze.com/presentations/software-validation/software-validation-full-details/software-design-specification/>

1.5 Overview of this document

It is organized into 8 of sections, each section contains some info about the tool like section-1 contains Introductions of the tool like purpose, scope, definitions, acronyms and references etc. Like wise section-2 is about the architecture description of the tool, including overview general constraints, data design, alternative considered and program structure, which contains functional hierarchy diagram of the tool. Similarly section-3 is about the detailed description of the components, like APIs their purpose, function and resources etc. Section-4 is about the user interface design, like design rules, GUI components and a detailed description. Section-5 is about reuse and

relationships to other products, means if we have re-used our API or any code that we used somewhere else previously. Section-6 is about Design decisions and tradeoffs and section-7 contains Pseudocode for components and the last section-8 contains all the UML diagrams.

2. System architecture description

2.1. Section Overview

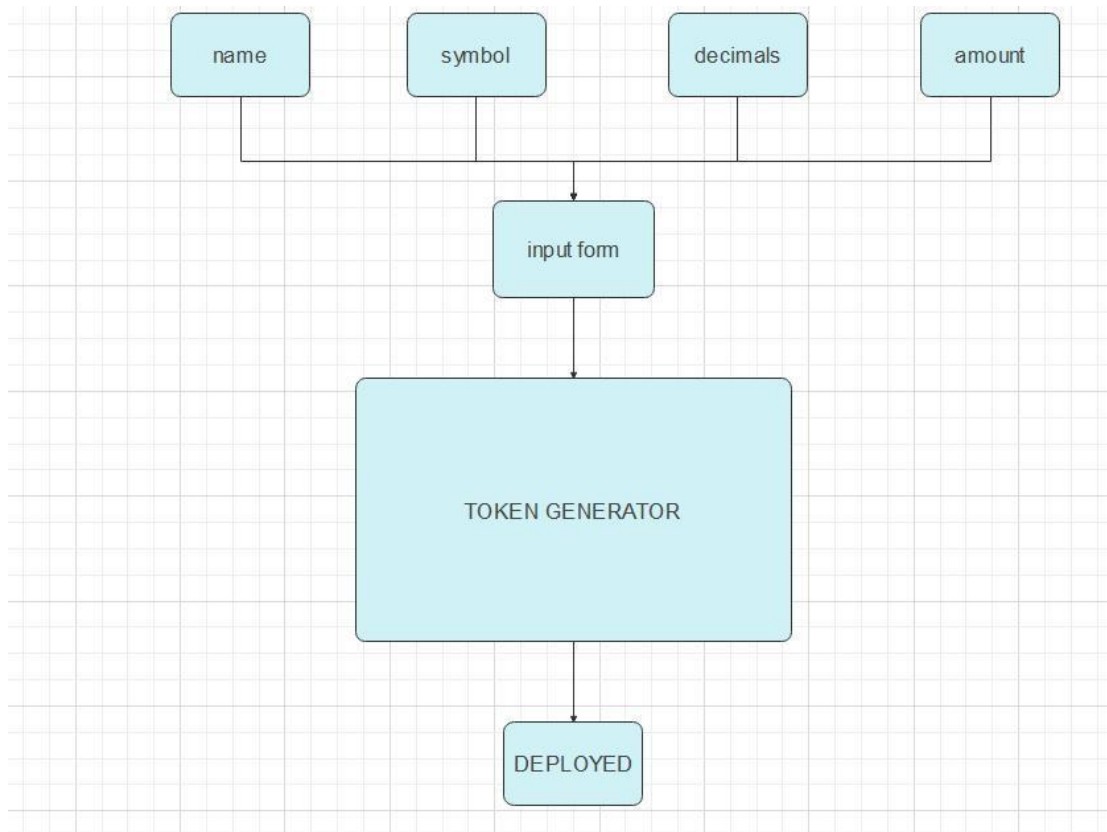
This section describes architectural design of the Token Generator. The high level components and their interactions, suitable architectural patterns and design decisions applied to the complete tool. The next content of this section is General Constraints, which contains the requirements about the section, which includes hardware and software environments, interface requirements, external data representations, performance requirements, network requirements, etc. The Data Design section describes the structure of any databases, external files and internal data structures and the Program Structure describes the architectural model chosen and the pictorial representation major components. In last section alternatives considered, its justified if I have used any alternative model.

2.2 General Constraints

- ✓ The HTTP protocol will be used to facilitate communications between the UI and user.
- ✓ The UI of tool supports Google Chrome, Mozilla Firefox and all web browsers.
- ✓ The software and hardware requirements for the development of this project are not many and are open source.
- ✓ The work for the project is done with the current equipment and existing software technology.

2.2.Data Design

The tool is not using any database, it uses the smart contract for all its storage needs and after deploying the token to the block chain it keeps them stored on it until moved by the owner them to a default output folder. Following is the representation of process:



the tokens can then be imported by adding the contract address to any web3 wallet

2.3. Program Structure

First of all user has to login with his web3 wallet then he must fill the token generator fields of the token . After that user will click the deploy button to deploy the token to the block chain then wait for the transaction to be mined and then when the user receives a notification from meta mask that the transaction was a success he can go to the respective block chains scan site to view his token.

2.4. Alternatives Considered

A two-tier architecture is a software architecture in which a presentation layer or interface runs on a client, and a data layer or data structure gets stored on a server. Separating these two components into different locations represents a two-tier architecture, as opposed to a single- tier architecture. It also provides direct and faster communication. like traditional two-way architecture here data is not stored in any server instead all the data is inputted from user and deployed to a distributed block chain.

3. Detailed description of components

3.1. Section Overview

In this section we have disused the modules, subprograms, data files, control procedures, classes that we have used or created in our tool and what is the purpose, functionalities, dependencies of these components along with pseudo code of each component.

3.2. Component n Detail (include a sub-section for each component)

1. Web3 wallet detection

Identification	Web3 check
type	Conditional check statement
Purpose	To check if user has web3 wallet
Function	To tell user to install wallet if not already installed
dependencies	Independent
interfaces	Interfaces are defined in detail below with attached screenshots.
resources	No external resources are required.
Processing	<pre>if (window.ethereum) { console.log("New ethereum provider detected"); // Instance web3 with the provided information web3 = new Web3(window.ethereum); // ask user for permission</pre>
Data	Returns an ethereum object

2. Input fields

Identification	Input form
type	Text fields
Purpose	Fill the token details
Function	Take the token details and for the contract
dependencies	Independent
interfaces	Interfaces are defined in detail below with attached screenshots.
resources	No external resources are required.
Processing	<pre>var initialSupply = \$('#total-supply').val(); var tokenName = \$('#name').val(); var decimalUnits = \$('#decimals').val(); var tokenSymbol = \$('#symbol').val();</pre>

3. Deploy

Identification	Deploy button
type	button
Purpose	Used to finalize the details
Function	Deploy the token to block chain
dependencies	Independent
interfaces	Interfaces are defined in detail below with attached screenshots.
resources	The input fields of the form
Processing	<pre> <div class="form-group"> <button align="center" class="submit-button btn btn-primary" id="submit-btn" type="submit">Deploy </button> </div> </pre>

4. User Interface Design

4.1. Section Overview

This section contains details about User Interface Design of the tool, like what UI is used in what functionality, the interface design rules section contains info about the rules of interface design, suppose, Guideline for error messages, or if user is new so the tool can tour him/her complete tool, or we provide a tutorial video in popup, or coding rules to make sure user enters correct information in input fields. In Detail Description, screen shots of all the features are provided along with short description of each feature.

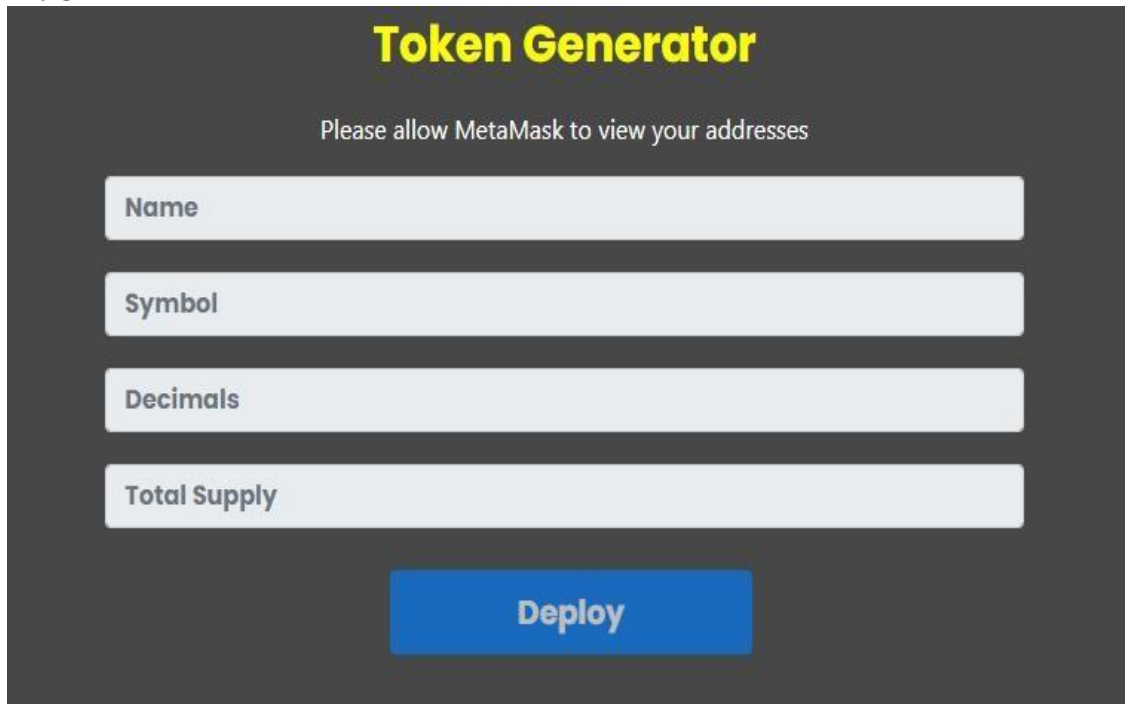
4.2. Interface Design Rules

Here are some of the interface design rules that user needs to take care about while using the interface features:

1. Internet is required to use this tool.
2. Web3 wallet must exist
3. All text fields of the form must be filled
4. Text and numbers must be filled respectively
5. No field can be left empty
6. The user needs to press deploy after filling the fields
7. The minimum amount cannot be less than 1
8. The block chain must be selected before
9. The user can check if their token was deployed by seeing their web3 wallet

4.3. Detailed Description

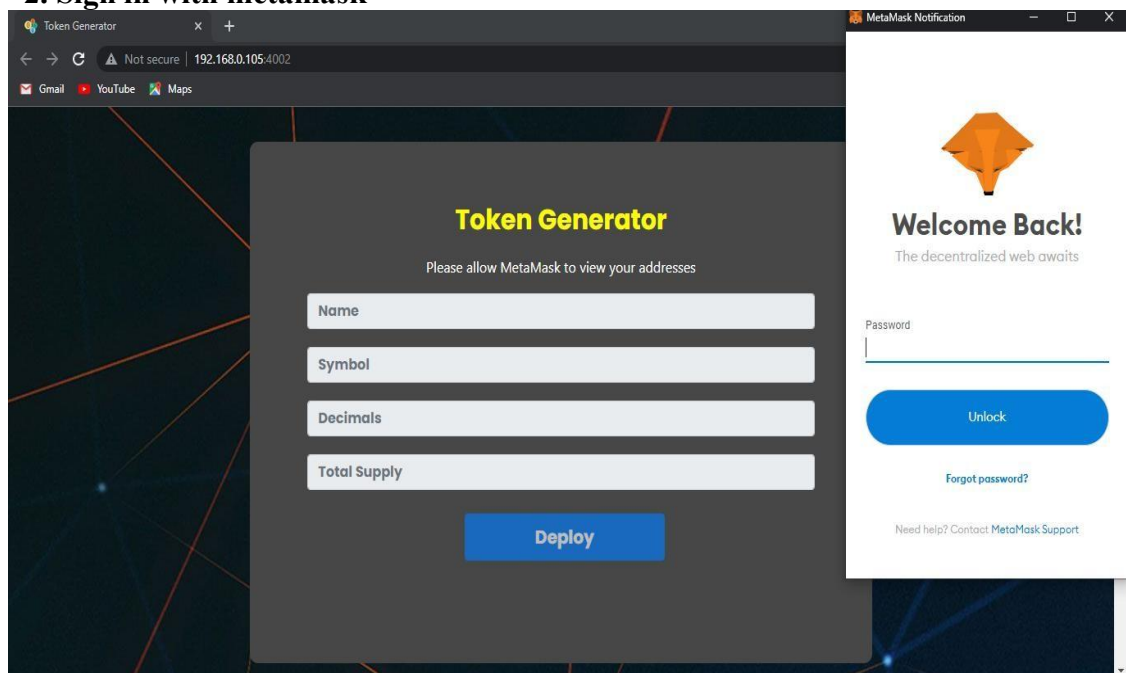
1. UI



The image shows a web interface titled "Token Generator" in yellow text on a dark gray background. Below the title, it says "Please allow MetaMask to view your addresses". There are four input fields: "Name", "Symbol", "Decimals", and "Total Supply", each with a light gray border. At the bottom, there is a blue button labeled "Deploy".

This is the UI of tool, that contains some input fields that needs to be fields in order to deploy the smart contract on ethereum blockchain network.

2. Sign in with metamask



This feature will check if the metamask exists, if it exists it pops up the metamask login page, if not it asks the user to download it first and then login, in case if user does not have the account, he must create one or he can simply import some existing account using private key or secret phrase.

3. Metamask Status Check

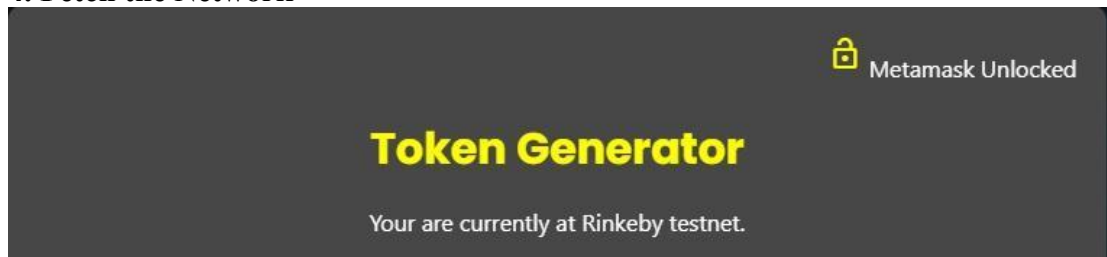


This feature will check if user has logged into the metamask account, if he is not logged into it or once logged in and then logged out, the status will display “Metamask Unlocked”.



and when logged in, it displays “Metamask Unlocked”.

4. Fetch the Network



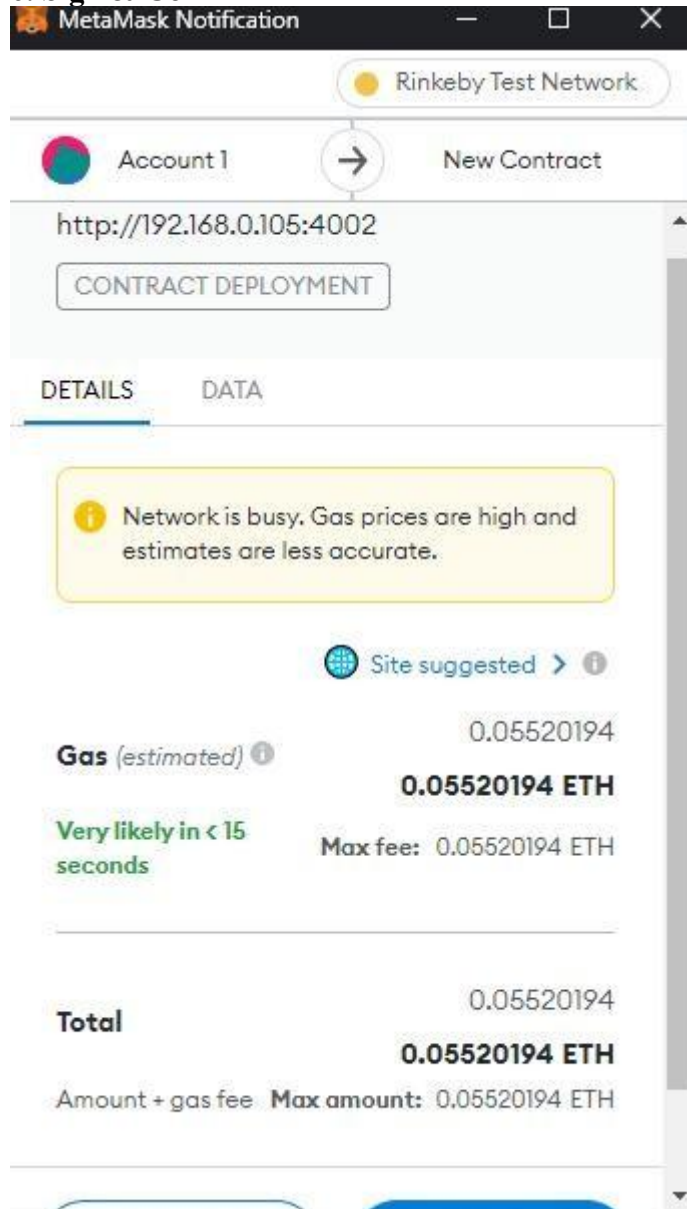
It will fetch the selected ethereum network from metamask wallet.

5. Deploy

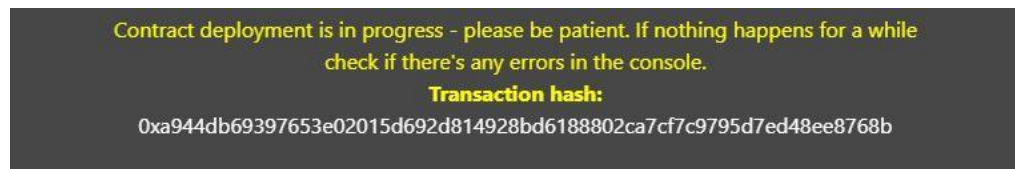
A dark gray rectangular interface for a 'Token Generator'. In the top right corner, there is a yellow open padlock icon followed by the text 'Metamask Unlocked'. In the center, the words 'Token Generator' are displayed in a large, bold, yellow font. Below this, in a smaller white font, it says 'Your are currently at Rinkeby testnet.' Below the text, there are four light blue input fields stacked vertically. The first field contains the text 'Gcoin', the second contains 'G', the third contains '5', and the fourth contains '500'. At the bottom center, there is a white rectangular button with the word 'Deploy' in a green font.

This allows to deploy the contract on ethereum blockchain but to proceed further, user must fill out all the required fields.

6. Sign & Confirm



User has to sign and confirm the transaction for gas fee, so that contract can be deployed on the ethereum blockchain.



After the successful deployment of contract on the ethereum blockchain, it will return a transaction hash, this indicates that the transaction has been verified and added to the blockchain, the transaction hash can be used to locate that transaction or the funds.

Transaction mined! Contract address:
0xb31304AB906DbBff5307FcdccA0Fd7Cdd0a0e4AF

Once the transaction hash added to blockchain network, it returns a contract address it is usually given when a contract is deployed to the Ethereum Blockchain. The address comes from the creator's address and the number of transactions sent from that address (the “nonce”). We can add the tokens to the Web3 wallet using this contract address.

5.0 Reuse and relationships to other products

We have not reused any API, library, module or any such component in this project.

6.0 Design decisions and tradeoffs

For this tool we have used web3 react and HTML CSS for structure and styling and we have used web3.js for interacting with the ethereum block chain we have used solidity the language for smart contracts to create a token template which is deployed with the users details all the resources are free to use except for the gas fee which is the fee fro deploying the contract to the block chain we didn't have any such tradeoffs both performance wise and design wise

7.0 Pseudocode for components

1. Wallet check

```
start
check if web3 wallet exists
if "wallet exists "
then "sign in "
else "ask user to get a wallet"
```

2. Input fields

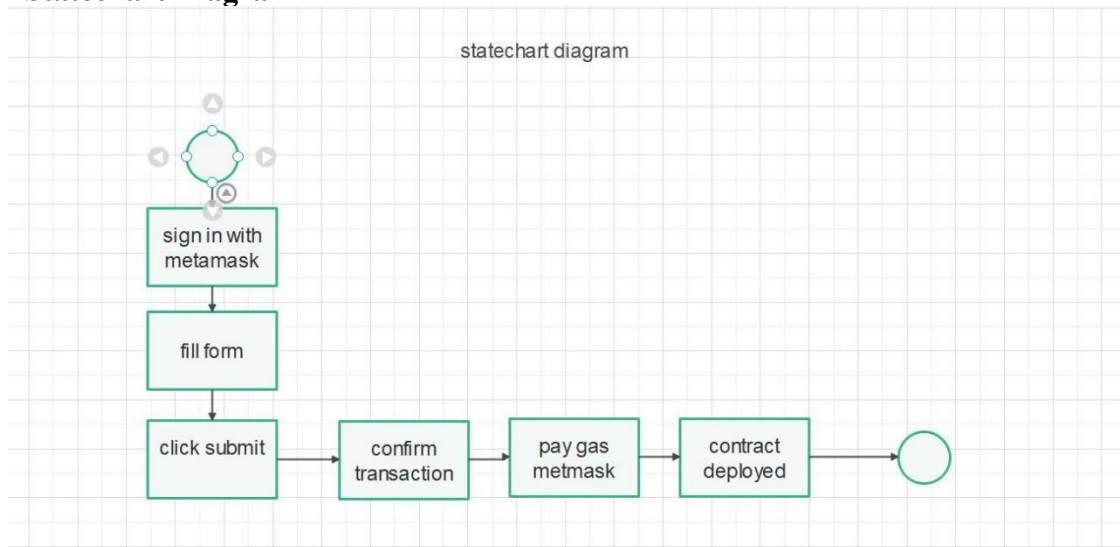
```
start
sign in with wallet
if "fields filled"
then "deploy "
else "ask user to fill the fields"
```

8.0 Appendices

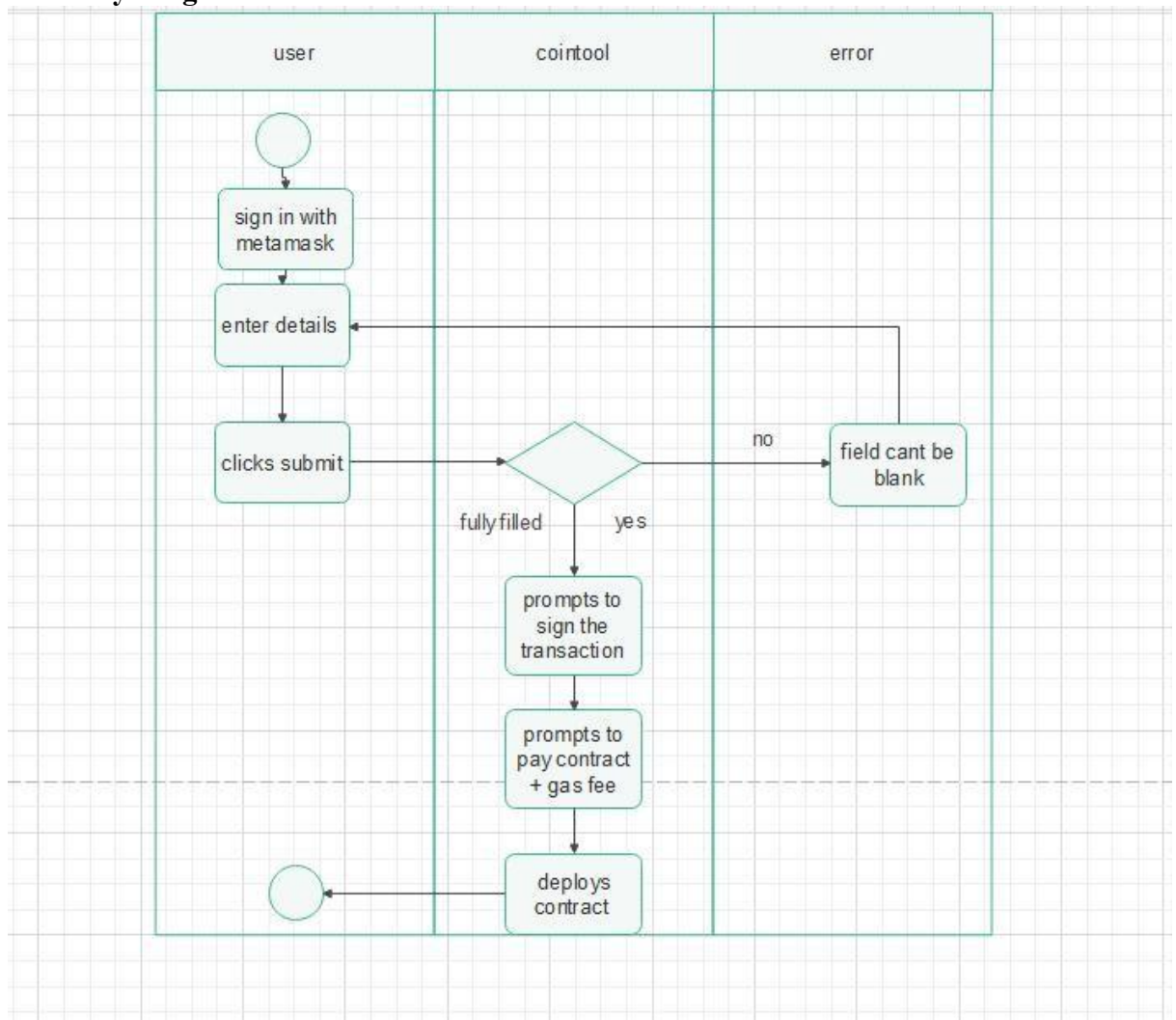
1. Class Diagram



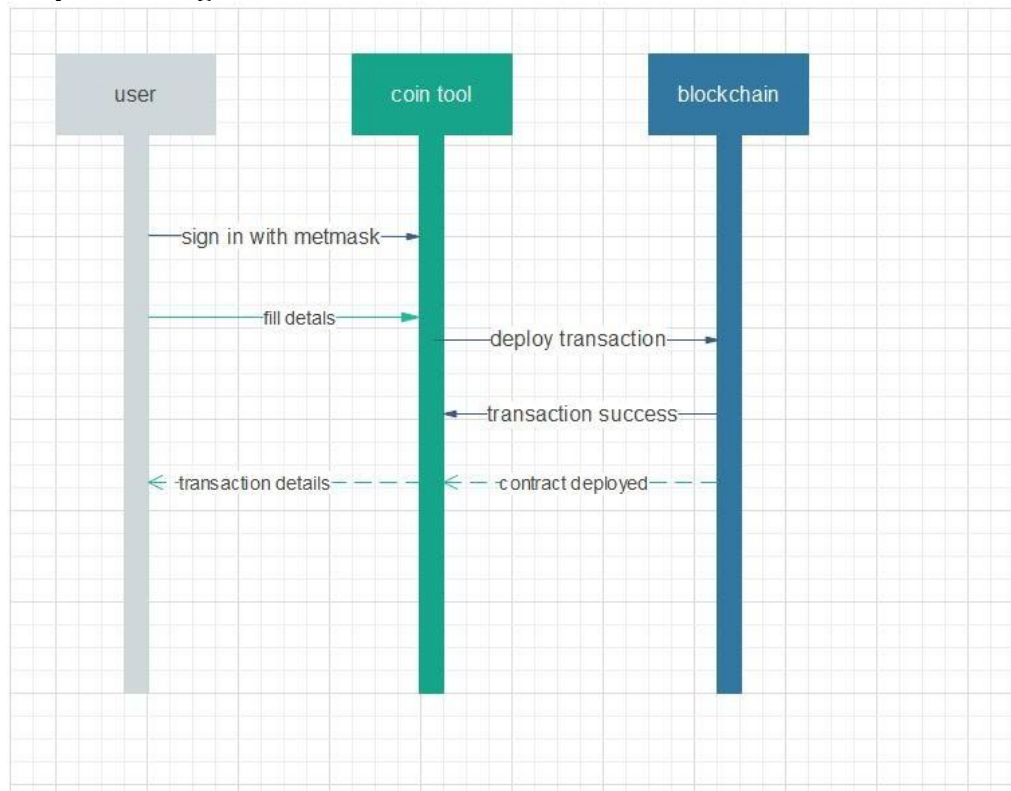
2. Statechart Diagram



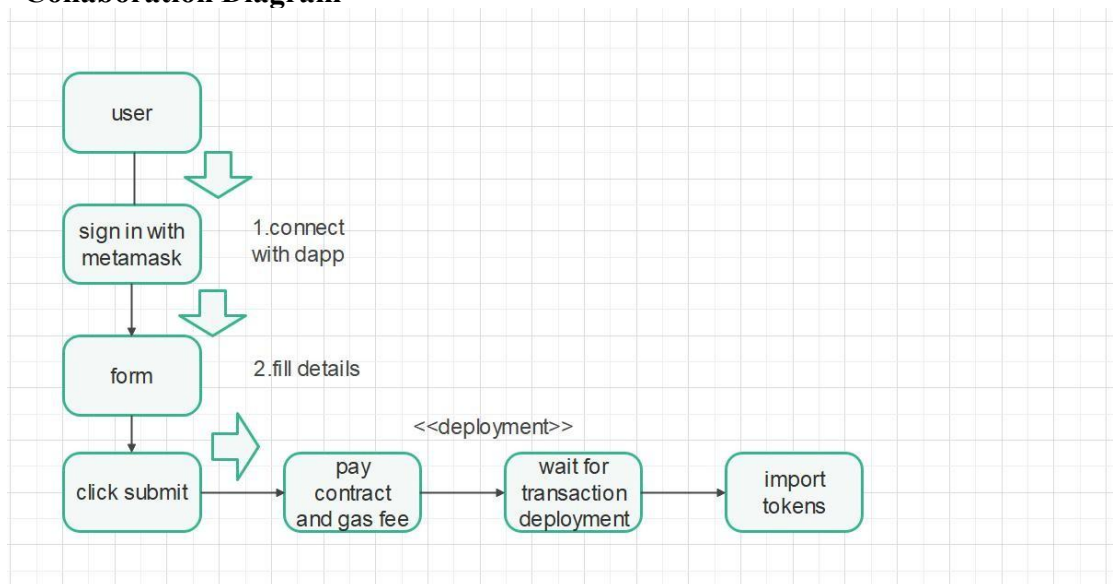
3. Activity Diagram



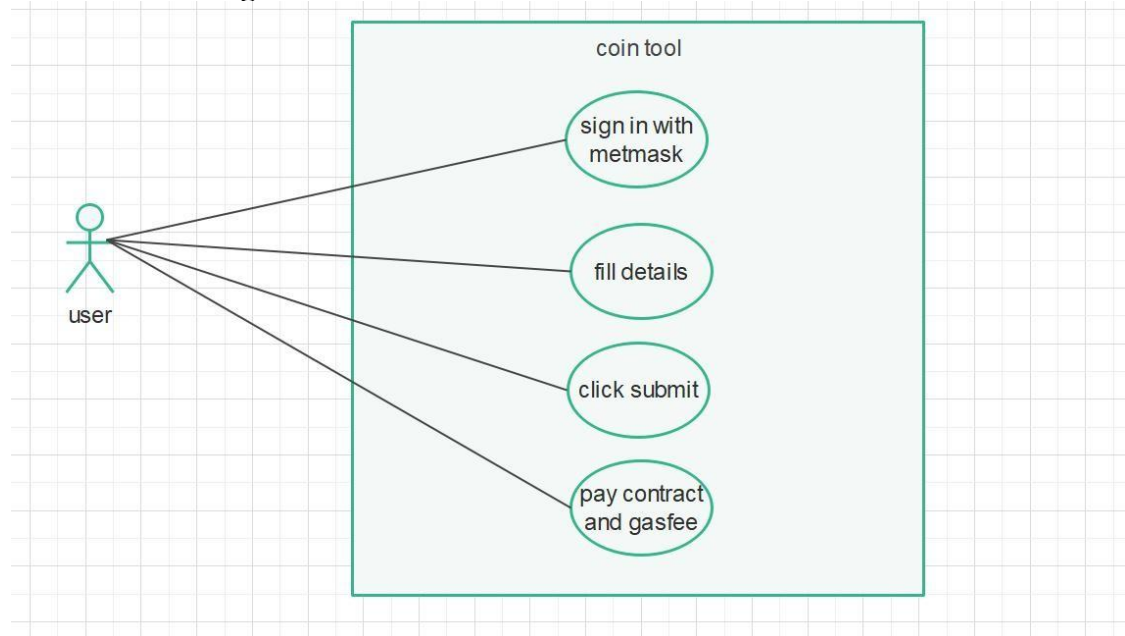
4. Sequence Diagram



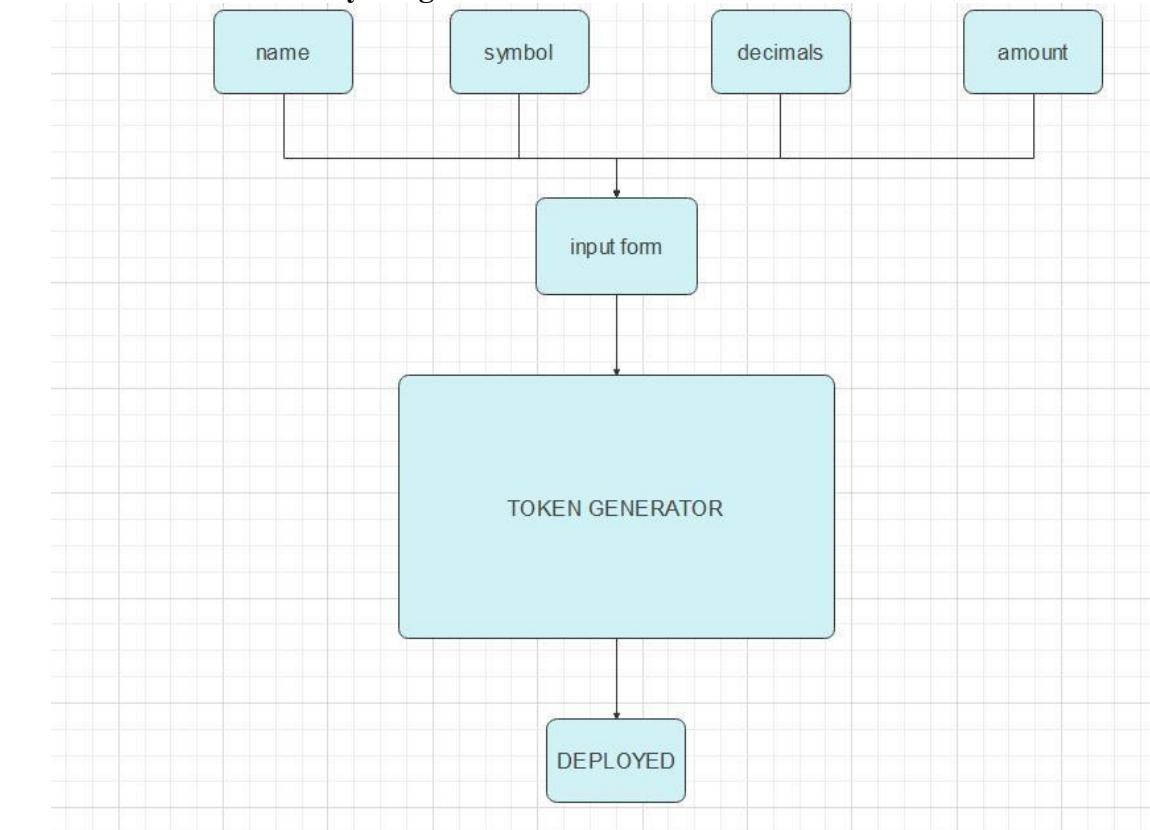
5. Collaboration Diagram



6. Use-case Diagrams



7. Functional Hierarchy Diagram



Software Requirements Specification

for

NFT Generator

Prepared by

Guman Singh 1812297

Muhammad Kashif 1812309

Faculty of Computing and Engineering Sciences

Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology, Karachi

4/29/2

1.Introduction

1.1 Purpose

The purpose of NFT Generator is to provide the platform the people who want to generate NFTs but aren't familiar with programming and algorithms, there are less tools available that provide the GUI based environment to the users, whatever tools are available are expensive, we are providing a GUI based tool, that will generate n number of NFTs with single click, we haven't added much features so no non-developer get confused how it works. Once the NFTs get generated we will also help users to bulk upload all these NFTs to OpenSea marketplace and list them for sell.

1.2 Document Conventions

This document uses the following font conventions for text:

<i>TEXT</i>	<i>FONT NAME</i>	<i>FONT SIZE</i>
Main Headings	Time	18 points
Sub Headings	Time New Roman	14 points
Paragraph Text	Time New Roman	12 points

1.3 Intended Audience and Reading Suggestions

This document is intended for multiple readers, such as users, testers and developers. This document contains the introduction, overall description, external interface requirements, system features and other nonfunctional requirements. The readers of this document can get the idea how the website works, and how many different ideas are implemented. For every user, the most ideal order to read the document is to start from the introduction.

1.4 Product Scope

NFT Generator is a python based tool that is created to generate NFTs for the ease of developers and specially non-developers. The tool provides some important features like to test the NFT after putting designed layers to dedicated folder, just to make sure that the layers have the right file type and size, it allows user to select number of NFTs he wants to generate, after this user needs to simply click on generate button and he is good to go, after the NFTs are generated, the tool shows success message and throws path to output directory down there so user can collect the generated NFTs from there. After collecting the NFTs, user needs to go to remix - Ethereum IDE to bulk upload all images to OpenSea.

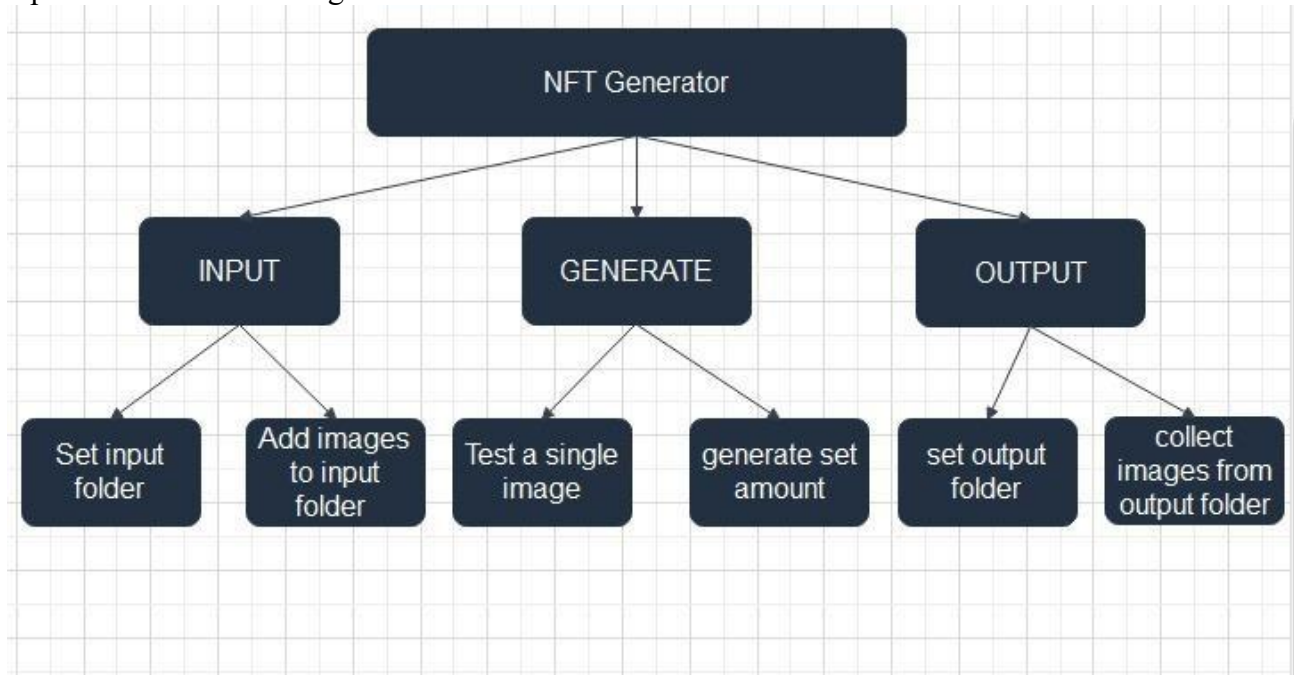
1.3 References

1. https://gephi.org/users/gephi_srs_document.pdf
2. <https://thakurguman.medium.com/>
3. <https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document>

2. Overall Description

2.1 Product Perspective

The perspective of the project is to provide the platform to the developers as well as non developers who are not familiar with programming, to generate the collection of their NFTs and list them on OpenSea with minimum gas fee.



2.2 Product Functions

1. Input Folder
2. Test
3. Number of NFTs(slider)
4. Amount(input)
5. Output folder
6. Generate

2.3 User Classes and Characteristics

- This tool is intended to target the user class of developers and non-developers, basically to the people who wants to generate NFTs with no hurdles.
- Students who want to explore NFT world, but are unable to pay high fees on generating NFTs and minting them to marketplaces.
- Artists, who knows how to design and draw the NFT but aren't ready to pay higher gas fee on minting.

2.4 Operating Environment

- Windows
- Linux
- Mac

2.5 Design and Implementation Constraints

For the design we are using Python library Streamlit, specially built for ML, DL and Data Science related projects, as it is a new library so has limited functions so far. As compare to available web development frameworks and libraries it is powerful, but still it lacks some designing features like colors, background customization and fully customize other UI features, though it will be added in upcoming versions.

2.6 User Documentation

SRS, SDS and STD documents will be provided along with the application which will include all the requirement specifications, design patterns and more technical details of the tool.

2.7 Assumptions and Dependencies

Assumptions

1. If the laptop has SSD, the generating process may be faster than usual
2. If the source images has higher size(1000*1000) it may take more time than lesser sized images
3. The folders with layers should be arranged from top to bottom accordingly, else the output image may have differences such glasses instead of upon the eyes may go behind the eyes, to tackle this issue the glasses folder should come after eyes folder like 3eyes, 4glasses and so on.

Dependencies

1. A laptop
2. Python and streamlit installed
3. Any web browser
4. Source Images(layers)
5. Source images should have PNG type
6. Source images should have fixed size(e.g 300*300)

3. External Interface Requirements

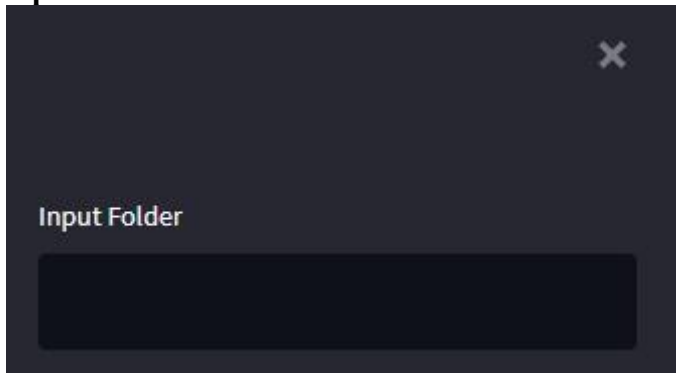
3.1 User Interfaces

1) UI



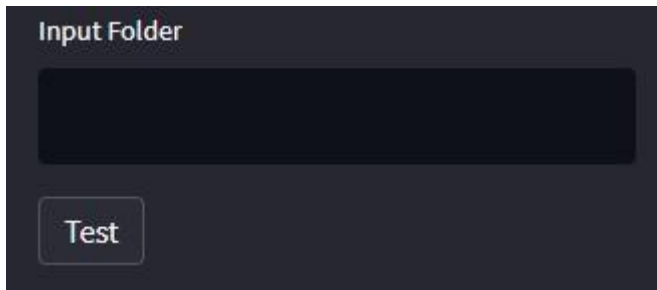
This is the UI of tool, that contains some sample NFTs and name of program and its creator, on the left side panel it contains all the features that are required to generate NFTs, on the right top corner it contains a menu that is default menu by streamlit library, using that we can do UI customization.

2) Input Folder



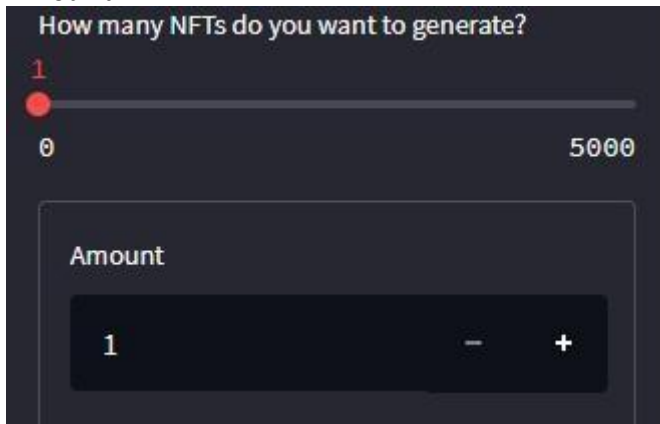
This feature will allow the user to input the folder that contains layers for NFTs, user has to enter the name of folder to load it in the program.

3) Test



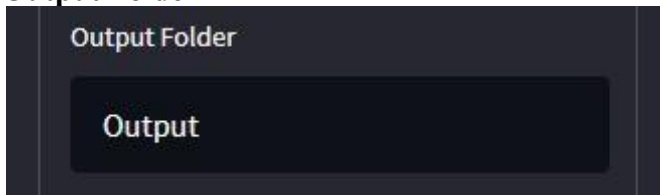
This feature will allow the user to test the layers he got from input folder, to verify if images are generating from provided layers just before going for bulk generating.

4) Amount



These both features will allow the user to set number of NFTs they want to generate, either they can use above bar or can simply input the integer, they can also use = and - to simply increase or decrease images to certain number.

5) Output Folder



This is the output directory, where all generated NFTs will be dumped.

6) Generate



This feature will allow the user to generate the number of NFTs he set in amount.

3.2 Hardware Interfaces

The tool is developed on Python, It is compatible for most of Operating Systems such as Windows, Linux and Mac. User can operate the program with mouse or touch.

3.3 Software Interfaces

- **Programming Languages:** Python for NFT Generator, Solidity for Smart Contracts and deploying them to OpenSea marketplace
- **Compilers:** PyCharm for Python and Remix - Ethereum IDE for Solidity
- **Libraries:** Streamlit
- **Environments:** Anaconda for python
- **Software:** Adobe Photoshop and Adobe Illustrator for NFT Design and Layers
- **Services:** IPFS for storing NFTs and Pinata for pinning them
- **Browsers:** Chrome
- **Marketplace:** OpenSea

3.4 Communications Interfaces

Any Web Browser is required to open the local host to view the GUI of tool. Local host is using HTTP protocol for connection. Internet is not required to generate NFTs or operate the features of this tool. The tool is using some sample NFTs in the homepage for that internet is required as the images are attached with a source link from the internet.

4. System Features

This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.

1) Input Folder

Use Case Name	Input folder(contains layers)
Version	1.0
Goal/Summary	This Use case is used to take input folder which contains layers for NFTs
Actors	User
Pre-Conditions	1. Actor must choose a folder
Basic course of Event	Getting the NFT layers to combine them and generate full image/images
Actor Action	System Response
1. Actor will write folder's name	1. System won't respond until we click test
Alternative Path	User can click on test to check if images are generating.

Post Condition	1. User can click on test to check if images are generating. 2. User can set amount of images he wants to generate 3. User can generate images
Author and Date	Guman Singh - 12/12/2021
Exception	1. No web browser in the machine 2. Images are not arranged in a required manner 3. The web browser go down due to heavy load.

2) Test

Use Case Name	Test for sample image
Version	1.0
Goal/Summary	This use case is used to test for the sample image, just to verify if images are generating
Actors	User
Pre-Conditions	1. Actor must have selected the input folder 2. The folder must contain images
Basic course of Event	Generate a sample image to verify if images has fixed size and required file type
Actor Action	System Response
1. Actor will click on test 2. If the user has entered wrong folder name that doesn't exist 3. If user entered wrong folder name that exists 4. If user entered folder name that contains non required file types	1. System will check if the input folder exists 2. The system will display "The system cannot find the path specified: 'folder_name' 3. The system will display "Cannot identify file type". 4. The system will display "Cannot identify file type"
Alternative Path	User can select amount of images he wants to generate
Post Condition	A Sample image will be generated
Author and Date	Guman Singh - 12/12/2021
Exception	1. Images has not PNG type 2. Images has not fixed size

3) Amount

Use Case Name	amount
Version	1.0
Goal/Summary	This use case is used to set the amount of images user wants to generate
Actors	User
Pre-Conditions	1. User has tested images are generating
Basic course of Event	To set the amount of images user wants to generate

Actor Action	System Response
1. Actor will scroll amount bar 2. Actor will input number of NFTs in amount section 3. Actor will click on + option 4. Actor will click on - option	1. System will increase the number of NFTs in amount section 2. System will increase the number of NFTs 3. System will increase 1 NFT 4. System will decrease 1 NFT
Alternative Path	User can generate given number of NFTs
Post Condition	Given number of NFTs will be set for generating
Author and Date	Guman Singh - 12/12/2021
Exception	1. Some source images got corrupted after setting up number of NFTs 2. Some source images got deleted after setting up number of NFTs

4) Generate

Use Case Name	Generate
Version	1.0
Goal/Summary	This use case is used to generate the NFTs
Actors	User
Pre-Conditions	Actor must have selected more than 1 number of the NFTs to generate
Basic course of Event	To generate the NFTs
Actor Action	System Response
Actor will click on generate	System will display “Wait, NFTs are getting generated...”
Alternative Path	User can view generated NFTs
Post Condition	1. System will display “Please check the output folder” 2. User can view generated NFTs
Author and Date	Muhammad Kashif - 12/12/2021
Exception	1. System crashed 2. User interrupted

5) Output

Use Case Name	Output
Version	1.0
Goal/Summary	This use case is used to view the generated NFTs
Actors	User
Pre-Conditions	Actor must have generated the NFTs
Basic course of Event	To view the generated NFTs
Actor Action	System Response
1. Output folder does not exists 2. Actor will view the output folder	1. System will generate the Output folder 2. System will show the generated NFTs

Alternative Path	User can now list the NFTs for sell
Post Condition	User can now list the NFTs for sell
Author and Date	Muhammad Kashif - 12/12/2021
Exception	1. System crashed 2. The layers are disturbed

5. Other Nonfunctional Requirements

5.1 Performance Requirements

For improving the performance of this tool, 4 GB RAM with SSD is required that will enhance the speed of image generating. A good internet connection is required to upload all the images to IPFS. Other than that if the images has lesser size than the process will be faster than usual.

5.2 Safety Requirements

The tool is designed in such a way that it will not harm or damage any person or environment, in case of any failure or termination of program, the tool won't affect the other programs or laptop thus it has no safety requirements.

5.3 Security Requirements

Any user can use this tool, they need not to authenticate or prove the identity, as it is not holding any data. While bulk uploading all the NFTs to OpenSea, user has to create account on OpenSea also needs to create Metamask wallet account and link the wallet to OpenSea.

5.4 Software Quality Attributes

We would carry out the Usability prerequisites, Flexibility prerequisites and Legal prerequisites, and to guarantee them in this tool, we have built the interface simple so user don't get confused and can easily operate the tool. We will ensure that tool doesn't crash and instead of errors the exceptions shows up.

5.5 Business Rules

1. User must have layers to generate NFTs
2. User have to write folder's name that contains NFT Layers
3. User have to set amount for minimum 1 NFT
4. If user has not the OpenSea account or Metamask account he needs to create the one in order to upload images to OpenSea.
5. User should have required amount of cryptocurrency in his metamask in order to make the transaction successful.
6. User has to mint at least one NFT in the contract
7. User has to select more than 0.001 Ether while listing the rt for sell.

6. Other Requirements

Appendix A: Glossary

WORD	STANDS	DEFINATION
RAM	Random Access Memory	Computer's short-term memory, which it uses to handle all active tasks and apps.
NFT	Non Fungible Token	A non-fungible token is a unique and non-interchangeable unit of data stored on a digital ledger.
ML	Machine Learning	It is the study of computer algorithms that can improve automatically through experience and by the use of data.
DL	Deep Learning	It is a part of a broader family of machine learning methods based on artificial neural networks with representation learning.
API	Application Programming Interface	It is connection between computers or between computer programs.
UI	User Interface	UI brings together concepts from interaction design, visual design, and information architecture.
GUI	Graphical User Interface	It is a type of user interface through which users interact with electronic devices via visual indicator representations.

Software Requirements Specification

for

Token Generator

Prepared by

Guman Singh 1812297

Muhammad Kashif 1812309

Faculty of Computing and Engineering Sciences

Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology, Karachi

4/29/2022

1. Introduction

1.1 Purpose

The purpose of Token Generator is to generate an ERC-20 token(cryptocurrency) that can be deployed to Ethereum blockchain network by providing, token name, supply, decimals and symbol, it will auto detect the connected wallet account and the blockchain selected on metamask wallet.

1.2 Document Conventions

This document uses the following font conventions for text:

TEXT	FONT NAME	FONT SIZE
Main Headings	Time	18 points
Sub Headings	Time New Roman	14 points
Paragraph Text	Time New Roman	12 points

1.3 Intended Audience and Reading Suggestions

This document is intended for multiple readers, such as users, testers and developers. This document contains the introduction, overall description, external interface requirements, system features and other nonfunctional requirements. The readers of this document can get the idea how the website works, and how many different ideas are implemented. For every user, the most ideal order to read the document is to start from the introduction.

1.4 Product Scope

Token Generator is a Solidity and Web3 based generator that generate an ERC-20 token (cryptocurrency) that can be deployed to Ethereum blockchain, it a GUI based tool that is created for the ease of developers and specially non-developers. Since creating a cryptocurrency is a complicated and lengthy process, which everyone cannot be able to do, even the developer needs to go through lots of documentation and sample work to generate his own coin, we are intended to provide a GUI based tool that makes things easy, where even a non developer can create the token by providing required feels.

1.5 References

1. https://gephi.org/users/gephi_srs_document.pdf
2. <https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document>
3. <https://www.investopedia.com/how-to-make-a-cryptocurrency-5215343>
4. <https://www.sofi.com/learn/content/how-to-create-your-own-cryptocurrency/>
5. <https://www.datadriveninvestor.com/how-to-create-your-own-cryptocurrency/>
6. <https://hackernoon.com/how-to-create-your-own-cryptocurrency-tips-to-get-started-947ba92f79f9>

2. Overall Description

2.1 Product Perspective

The perspective of the project is to provide the platform to the developers as well as non developers who are not familiar with programming, to create the cryptocurrency and deploy that on Ethereum blockchain network.

2.2 Product Functions

1. Check the Web3 Wallet
2. Sign in with wallet
3. Fetch the selected network from wallet
4. Fill the required fields
5. Deploy the Smart Contract
6. Contract Deployment Gas fee
7. Display transaction hash after deployment
8. Display contract address at the bottom

2.3 User Classes and Characteristics

- This tool is intended to target the user class of developers and non-developers, basically to the people who wants to create cryptocurrency with no hurdles.
- Students who want to learn and explore crypto world and wants to create a cryptocurrency for project using testnet of blockchain.

2.4 Operating Environment

- Windows
- Mac
- Linux

2.5 Design and Implementation Constraints

For the design we are using HTML, CSS and JavaScript, for Smart Contract we have used Solidity and the contracts were compiled on Remix - Ethereum IDE. Web3 along with JavaScript is used to interact with blockchain and Metamask wallet, for fetching current network and account address, also for displaying the displaying transaction hash, contract address and Metamask current status.

2.6 User Documentation

SRS, SDS and Test Cases documents will be provided along with the application which will include

all the requirement specifications, design patterns and more technical details of the tool.

2.7 Assumptions and Dependencies

Assumptions

1. Ethereum network should be selected on Metamask, else it will displays the account address
2. Sometimes contract deployment takes some time, because blockchain is busy with mining other transactions, so should wait until it response back.

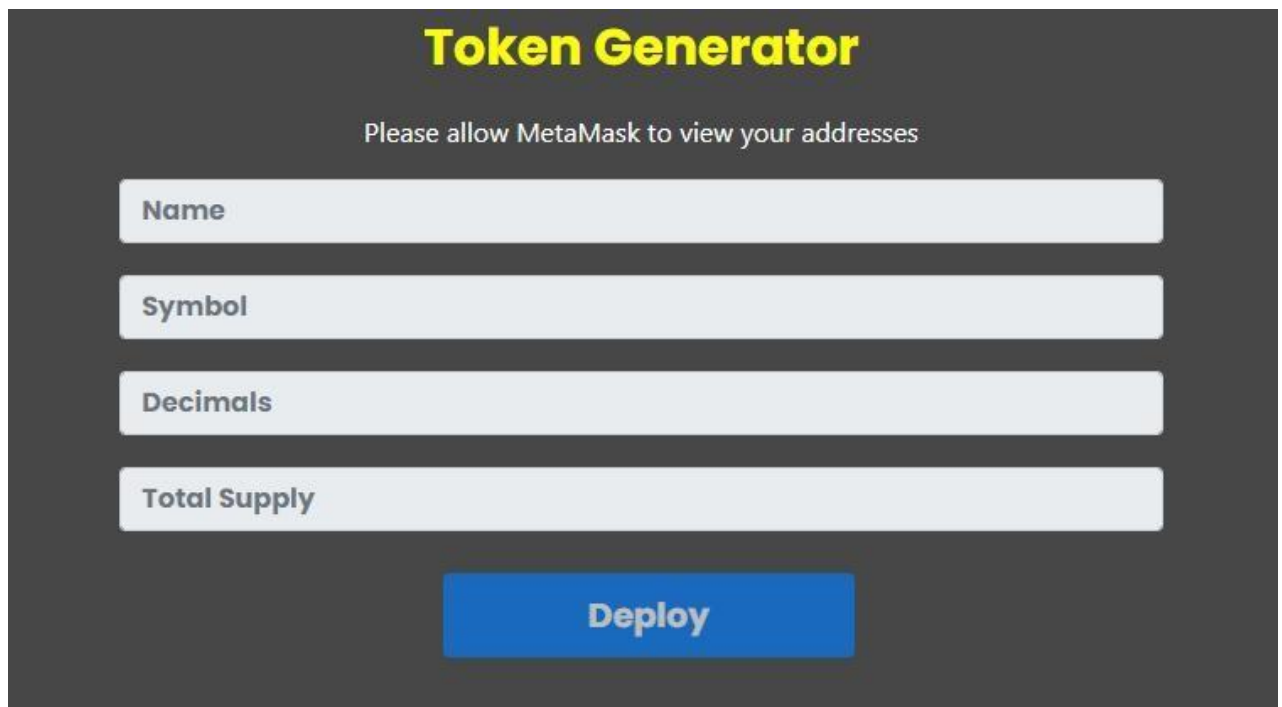
Dependencies

1. A laptop
2. Node installed
3. Any web browser
4. Metamask wallet
5. Some eth in Ethereum mainnet account or testnet account for gas fees
6. Ethereum network should be selected on Metamask
7. Required fields should be filled

3. External Interface Requirements

3.1 User Interfaces

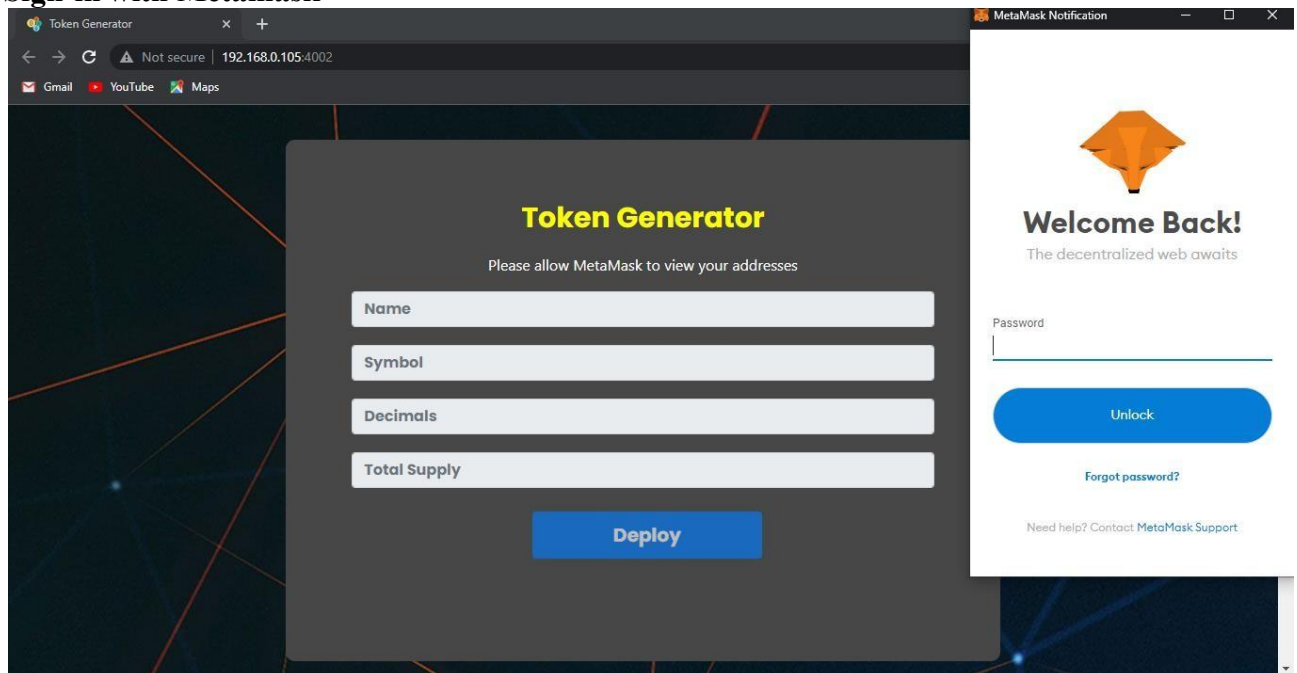
1) UI



The image shows a web interface titled "Token Generator" in yellow text on a dark gray background. Below the title, a message reads "Please allow MetaMask to view your addresses". There are four light gray input fields stacked vertically, each with a label in bold: "Name", "Symbol", "Decimals", and "Total Supply". At the bottom center, there is a blue button with the word "Deploy" in white text.

This is the UI of tool, that contains some input fields that needs to be fields in order to deploy the smart contract on ethereum blockchain network.

2) Sign-in with Metamask



This feature will check if the metamask exists, if it exists it pops up the metamask login page, if not it asks the user to download it first and then login, in case if user does not have the account, he must create one or he can simply import some existing account using private key or secret phrase.

3) Metamask Status Check

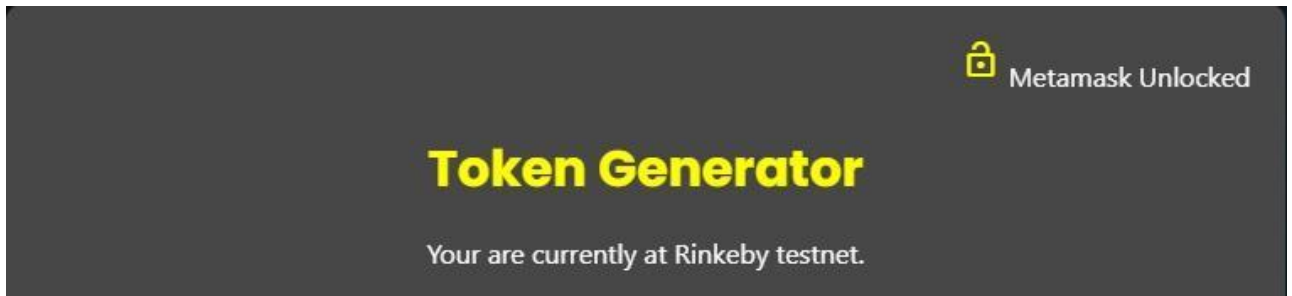


This feature will check if user has logged into the metamask account, if he is not logged into it or once logged in and then logged out, the status will display “Metamask Unlocked”.



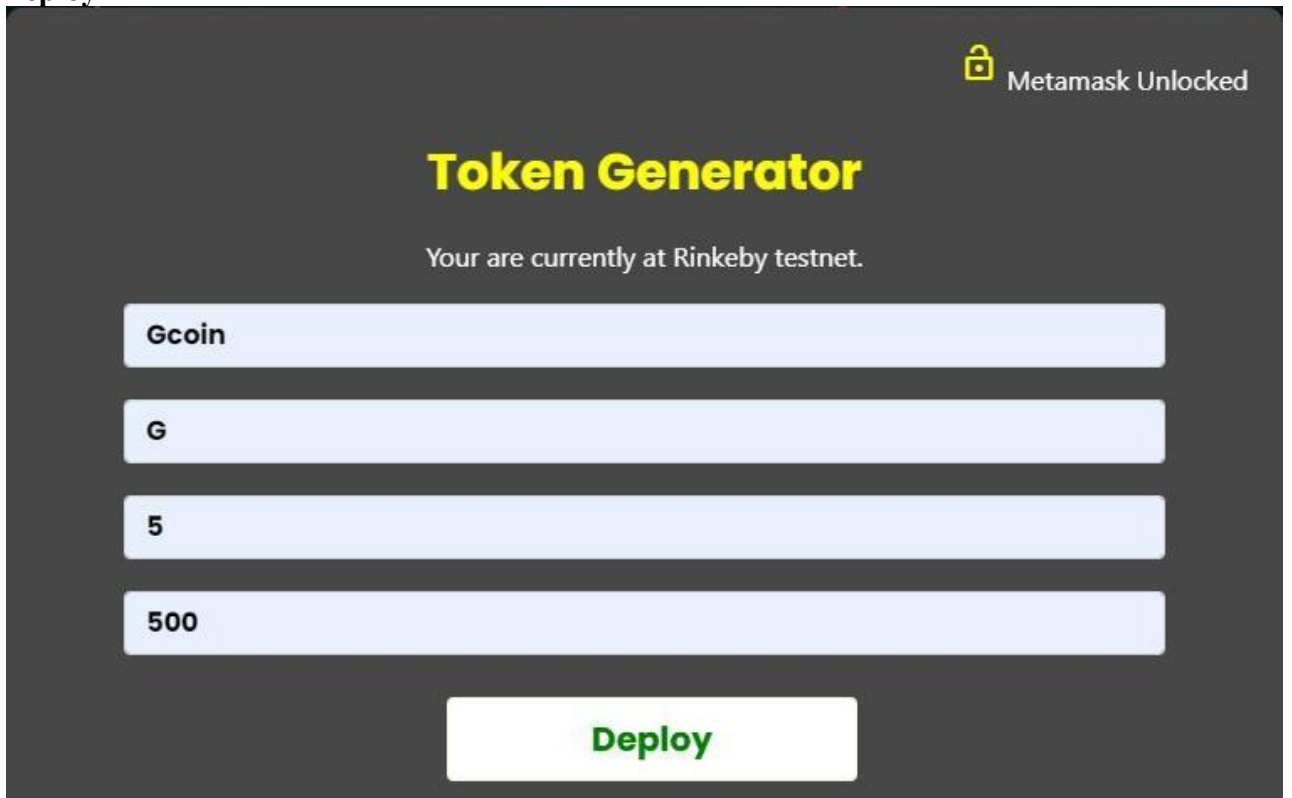
and when logged in, it displays “Metamask Unlocked”.

4) Fetch the Network



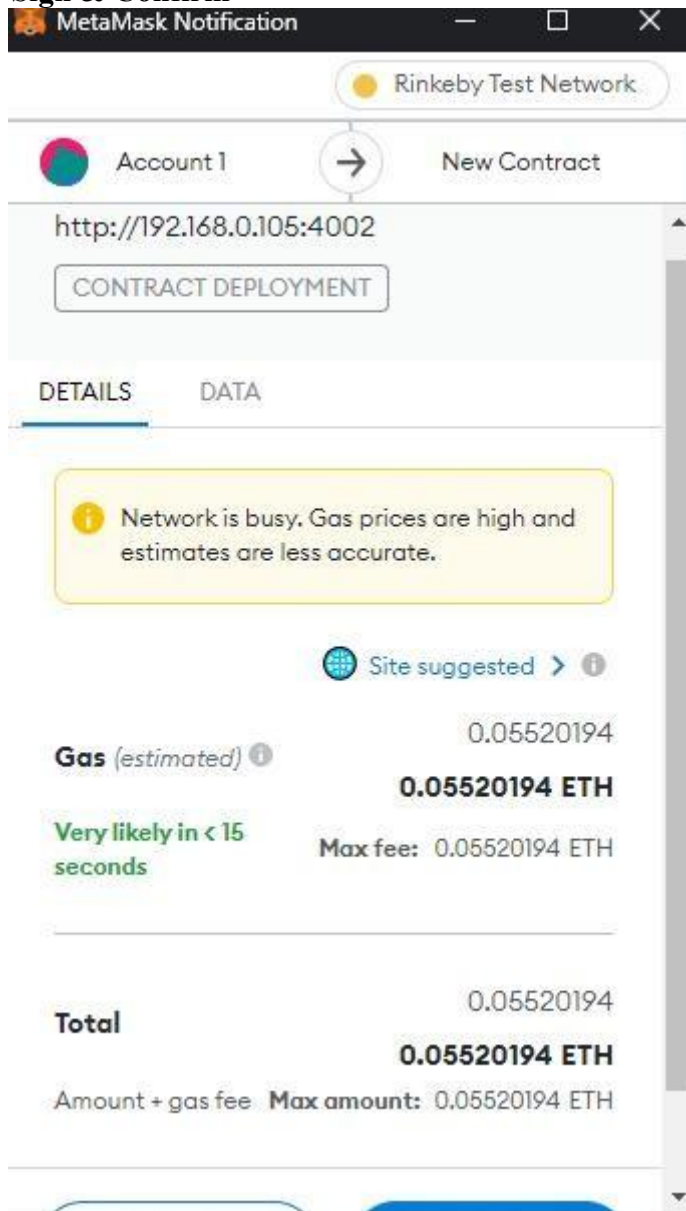
It will fetch the selected ethereum network from metamask wallet.

5) Deploy

A screenshot of the same "Token Generator" web application interface. It now shows a deployment form. The top part is identical to the previous screenshot. Below the "Your are currently at Rinkeby testnet." text, there are four light blue input fields stacked vertically. The first field contains the text "Gcoin", the second contains "G", the third contains "5", and the fourth contains "500". At the bottom center of the form, there is a white button with the word "Deploy" in green text.

This allows to deploy the contract on ethereum blockchain but to proceed further, user must fill out all the required fields.

6) Sign & Confirm



User has to sign and confirm the transaction for gas fee, so that contract can be deployed on the ethereum blockchain.

Contract deployment is in progress - please be patient. If nothing happens for a while check if there's any errors in the console.

Transaction hash:

0xa944db69397653e02015d692d814928bd6188802ca7cf7c9795d7ed48ee8768b

After the successful deployment of contract on the ethereum blockchain, it will return a transaction hash, this indicates that the transaction has been verified and added to the blockchain, the transaction hash can be used to locate that transaction or the funds.

Transaction mined! Contract address:

0xb31304AB906DbBff5307FcdccA0Fd7Cdd0a0e4AF

Once the transaction hash added to blockchain network, it returns a contract address it is usually given when a contract is deployed to the Ethereum Blockchain. The address comes from the creator's address and the number of transactions sent from that address (the “nonce”). We can add the tokens to the Web3 wallet using this contract address.

3.2 Hardware Interfaces

The tool is developed using HTML, CSS, JavaScript and Solidity. It is compatible for most of Operating Systems such as Windows, Linux and Mac. User can operate the program with mouse or touch.

3.3 Software Interfaces

- **Programming Languages:** HTML, CSS and JavaScript for design purpose, Solidity for Smart Contract and Web3 along with JavaScript is used to interact with blockchain and Metamask.
- **Compilers:** Visual Studio Code for HTML, CSS and JavaScript, Remix - Ethereum IDE for Solidity
- **Libraries:** Web3 contains collections of libraries to interact with blockchain
- **Wallet:** Metamask
- **Browsers:** Chrome
- **Blockchain:** Ethereum
- **Cryptocurrency:** Ethers

3.4 Communications Interfaces

Any Web Browser is required to open the local host to view the GUI of tool. Local host is using HTTP protocol for connection. Internet is required to deploy the contract on ethereum blockchain, in case if metamask wallet is not downloaded, it is required to download it first. Web3 is also works as a communication interface between the Ethereum blockchain and the local host.

4. System Features

This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.

1) Sign-in with Metamask

Use Case Name	Sign-in with Metamask
Version	1.0
Goal/Summary	This Use case is used to sign-in with the metamask
Actors	User
Pre-Conditions	1. Actor must have a metamask installed on his browser 2. Actor must have a metamask account
Basic course of Event	Sign-in with the metamask to fetch the current account details from the wallet
Actor Action	System Response
1. Actor will open the UI on localhost 2. Actor will login with the metamask	1. System will check if the metamask is installed 2. System will display the dashboard of metamask
Alternative Path	User can make the transactions
Post Condition	1. User can make the transactions 2. User can confirm or reject the transactions 3. User can view all the accounts 4. User can view all the crypto balance 5. User can import the account 6. User can view and change the settings
Author and Date	Guman Singh - 20/04/2022
Exception	1. Web browser crashed 2. Metamask crashed 3. User logged in and forgot the private key 4. User logged in and forgot the secret

2) Metamask Status Check

Use Case Name	Metamask Status Check
Version	1.0
Goal/Summary	This use case is used to check the status of Metamask, whether it is Locked or Unlocked.
Actors	System
Pre-Conditions	1. Actor must have metamask installed 2. Actor must have a metamask account
Basic course of Event	To check the status of Metamask, whether it is Locked or Unlocked.
Actor Action	System Response
1. Actor will open the UI on localhost	1. System will check if the metamask is connected 2. If it is connected system will display "Unlocked" 3. If it is not connected system will display "Locked" and neither the user can fill the field, nor can he deploy the contract
Alternative Path	If metamask is unlocked, user can deploy the contract, if it is locked user must have to unlock it, in order to deploy the contract

Post Condition	User can deploy the contract, if it is unlocked
Author and Date	Guman Singh - 20/04/2022
Exception	1. Web browser crashed 2. Metamask crashed

3) Fetch the Network

Use Case Name	Fetch the Network
Version	1.0
Goal/Summary	This use case is used to fetch the current selected network from metamask wallet
Actors	System
Pre-Conditions	1. System should have a metamask installed 2. User must have a metamask account 3. The metamask should be unlocked
Basic course of Event	To fetch the network from metamask
1. Actor will open the tool UI on localhost	1. System will check if Metamask is locked or unlocked 2. System will fetch the current selected network
Alternative Path	User can view the current selected network on the tool UI.
Post Condition	The current selected network will be visible to user from the tool UI.
Author and Date	Guman Singh - 21/04/2022
Exception	1. Web browser crashed 2. Metamask crashed 3. No ethereum network selected

4) Deploy

Use Case Name	Deploy
Version	1.0
Goal/Summary	This use case is used to deploy the contract on ethereum blockchain.
Actors	User
Pre-Conditions	1. User must have a web browser installed 2. User must have a metamask installed 3. User must have a metamask account 4. User must have a metamask unlocked 5. User must fill out all the required fields
Basic course of Event	To deploy the contract.
Actor Action	System Response
Actor will click on deploy	System will display "Waiting for contract to be deployed..."
Alternative Path	User can deploy the contract.
Post Condition	1. System will display "Contract deployment is in progress - please be patient"

	2. System will display the transaction hash. 3. System will display “Transaction mined! Contract address”
Author and Date	Guman Singh - 22/04/2022
Exception	1. Web browser crashed 2. Metamask crashed 3. No ethereum network selected 4. No fields were filled 5. Transaction failed 6. Transaction was rejected 7. Signature was not provided 8. System was busy 9. Contract had some issues

5) Sign & Confirm

Use Case Name	Sign & Confirm
Version	1.0
Goal/Summary	This use case is used to sign the transaction and confirm the gas fee for the transaction
Actors	User
Pre-Conditions	1. User must have a web browser installed 2. User must have a metamask installed 3. User must have a metamask account 4. User must have a metamask unlocked 5. User must fill out all the required fields
Basic course of Event	To sign & confirm the transaction
Actor Action	System Response
1.Actor will click on deploy 2.Actor clicks on confirm 3.Actor clicks on reject	1. Metamask will popup the window of gas fees, with confirm and reject option 2. System will display “Waiting for contract to be deployed...” and returns transaction hash and contract address 3. System will display “Waiting for contract to be deployed...” and remains unchanged
Alternative Path	The contract will be deployed the blockchain
Post Condition	We will get the transaction hash and contract address of deployed contract
Author and Date	Guman Singh - 23/04/2022
Exception	1. Web browser crashed 2. Metamask crashed 3. No ethereum network selected 4. No fields were filled 5. Transaction failed 6. Transaction was rejected 7. Signature was not provided 8. System was busy 9. Contract had some issues

5. Other Nonfunctional Requirements

5.1 Performance Requirements

There are no such things are required for improving the performance of this tool, this tool can operate on any such system with basic hardware and software functionalities.

5.2 Safety Requirements

The tool is designed in such a way that it will not harm or damage any person or environment, in case of any failure or termination of program, the tool won't affect the other programs or laptop thus it has no safety requirements. As the blockchain has one disadvantage, it charges gas fees on each transaction whether the transaction failed or passed, in that case a user must use testnet first and when he is ready with is smart contract only then he must go for mainnet, so that he don't waste the gas on failed transactions.

5.3 Security Requirements

Any user can use this tool, they need have a metamask account, for that he doesn't even have to provide his personal or sensitive data, the metamask will provide him a public and a private key along with a secret phrase, he must keep the private key secure and confidential, in any case if he lost it he can reset the password using the secret phrase, that also should be kept confidential. On each of gas transaction user has to make a sign and approve the gas fee to make the transaction happen. If a user rejects the sign request, no transaction will take place.

5.4 Software Quality Attributes

We would carry out the usability prerequisites, Flexibility prerequisites and Legal prerequisites, and to guarantee them in this tool, we have built the interface simple so user don't get confused and can easily operate the tool. We will ensure that tool doesn't crash and instead of errors the exceptions shows up.

5.5 Business Rules

1. User should carefully write supply and decimals
2. User must have an Ethereum account with a mainnet while testnets are optional
3. User must have required amount of eth for gas fee
4. User must sign the transaction request
5. User must approve gas fees request

6. Other Requirements

Appendix A: Glossary

WORD	STANDS	DEFINATION
ETH	Ether	It is a cryptocurrency of Ethereum blockchain
API	Application Programming Interface	It is connection between computers or between computer programs.
UI	User Interface	UI brings together concepts from interaction design, visual design, and information architecture.
GUI	Graphical User Interface	It is a type of user interface through which users interact with electronic devices via visual indicator representations.
HTML	Hyper Text Markup Language	It is the code that is used to structure a web page and its content.
CSS	Cascading Style Sheets	CSS is used for defining the styles for web pages.
ERC	Ethereum request for comment	ERCs are application-level standards for Ethereum and can include token standards.

Test Cases for NFT Generator

**Prepared by
Guman Singh 1812297
Muhammad Kashif 1812309**

**Faculty of Computing and Engineering Sciences
Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology,
Karachi**

4/29/2022

1. Test Case #1 Input Folder

Test Case ID	Test Case Name	Test Case Summary	Test Case Steps	Expected result	Actual result	Pass/fail
1	Input_folder	Load the input folder which contains base images(assets)	Press test with blank input_folder	"File not found error" will be displayed	"File not found error" is displayed	Pass
			Press test with wrong input folder	"File not found error" will be displayed	"File not found error" is displayed	Pass
			Press test with right input folder	Displays a sample NFT	Displayed a sample NFT	Pass

2. Test Case #2 Test

Test Case ID	Test Case Name	Test Case Summary	Test Case Steps	Expected result	Actual result	Pass/fail
2	Test	Test the sample image From provided assets	Press test with blank input_folder	"File not found error" will be displayed	"File not found error" is displayed	Pass
			Press test with wrong input folder	"File not found error" will be displayed	"File not found error" is displayed	Pass
			Press test with right input folder	Displays a sample NFT	Displayed a sample NFT	Pass

3. Test Case #3 Amount

Test Case ID	Test Case Name	Test Case Summary	Test Case Steps	Expected result	Actual result	Pass/fail
3	Amount	Set amount of NFTs user wants to generate	Leave the slider at default position	No images will be generated	No images are generated	Pass
			Move the slider forward to set the amount of images.	Number of images will increase	Number of images are increased	Pass
			Move the slider backward to set the amount of images.	Number of images will decrease	Number of images are decreased	Pass
			Put no value in amount field	No images will be generated	No images are generated	Pass
			Put any N value in amount field	N number of images will be generated	N number of images are generated	Pass
			Click - with 0 value in amount field	Nothing will happen	Nothing is happened	Pass
			Click - with > 0 value in amount field	One number will decrease	One number is decreased	Pass

			Click + with default value in amount field	One number will increase	One number is increased	Pass
			Click + with any value in amount field	One number will increase	One number is increased	Pass

4. Test Case #4 Generate

Test Case ID	Test Case Name	Test Case Summary	Test Case Steps	Expected result	Actual result	Pass/fail
4	Generate	Generate N number of NFTs where N is user specified number in amount field.	Press generate with blank input_folder	"File not found error" will be displayed	"File not found error" is displayed	Pass
			Press generate with wrong input folder	"File not found error" will be displayed	"File not found error" is displayed	Pass
			Press generate with right input folder but 0 in amount field	No NFT will be generated	No NFT is generated	Pass
			Press generate with right input folder and > 0 value in amount field	Generate the number of NFTs specified in amount field	Generated the number of NFTs which were specified in amount field	Pass

5. Test Case #5 Output

Test Case ID	Test Case Name	Test Case Summary	Test Case Steps	Expected result	Actual result	Pass/fail
5	Output	Save the generated NFTs in output folder	Press generate with right input folder and > 0 value in amount field	NFTs will be saved in default output folder	NFTs are saved in default output folder	Pass
			If default output folder does not exist	The system will create the one	The system created the one	Pass
			Replace the folder name in output field	The new folder will be created with the specified name	The new folder is created with the specified name	Pass

Test Cases for Token Generator

**Prepared by
Guman Singh 1812297
Muhammad Kashif 1812309**

**Faculty of Computing and Engineering Sciences
Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology,
Karachi**

4/29/2022

1. Test Case #1 Sign-in with Metamask

Test Case ID	Test Case Name	Test Case Summary	Test Case Steps	Expected result	Actual result	Pass/fail
1	Sign-in with Metamask	Sign-in with the metamask to fetch the current account details from the wallet	If metamask is not installed	“No web3 provider detected web3 not exists.” will be displayed	“No web3 provider detected web3 not exists.” is displayed	Pass
			If metamask is installed	“Please allow MetaMask to view your addresses” will be displayed	“Please allow MetaMask to view your addresses” is displayed	Pass
			If user enter wrong password	“Incorrect password” will be displayed	“Incorrect password” is displayed	Pass
			If user enter correct password	Dashboard will be displayed	Dashboard is displayed	Pass

2. Test Case #2 Metamask Status Check

Test Case ID	Test Case Name	Test Case Summary	Test Case Steps	Expected result	Actual result	Pass/fail
2	Metamask Status Check	To check the status of Metamask, whether it is Locked or Unlocked.	If metamask is not connected	“Metamask Locked” will be displayed	“Metamask Locked” is displayed	Pass
			If metamask is connected	“Metamask Unlocked” will be displayed	“Metamask Unlocked” is displayed	Pass

Test Case ID	Test Case Name	Test Case Summary	Test Case Steps	Expected result	Actual result	Pass/fail
3	Fetch the Network	To fetch the network from metamask	If Ethereum mainnet is selected	“You are currently at mainnet” will be displayed	“You are currently at mainnet” is displayed	Pass
			If Ropsten Testnet is selected	“You are currently at Ropsten Testnet” will be displayed	“You are currently at Ropsten Testnet” is displayed	Pass
			If Rinkeby Testnet is selected	“You are currently at Rinkeby Testnet” will be displayed	“You are currently at Rinkeby Testnet” is displayed	Pass
			If Goerli Testnet is selected	“You are currently at Goerli Testnet” will be displayed	“You are currently at Goerli Testnet” is displayed	Pass
			If any other network is selected	“You are currently at NetworkId” will be displayed	“You are currently at NetworkId” is displayed	Pass

4. Test Case #4 Deploy

Test Case ID	Test Case Name	Test Case Summary	Test Case Steps	Expected result	Actual result	Pass/fail
4	Deploy	To deploy the contract on ethereum blockchain.	If no field is filled	“Please fill out this field” will be displayed	“Please fill out this field” is displayed	Pass
			If name field is filled and others not	“Please fill out this field” will be displayed	“Please fill out this field” is displayed	Pass
			If symbol field is filled and others not	“Please fill out this field” will be displayed	“Please fill out this field” is displayed	Pass
			If decimals field is filled and others not	“Please fill out this field” will be displayed	“Please fill out this field” is displayed	Pass
			If total supply field is filled and others not	“Please fill out this field” will be displayed	“Please fill out this field” is displayed	Pass
			If all fields are filled	“Waiting for contract to be deployed...” will be displayed	“Waiting for contract to be deployed...” is displayed	Pass

Test Case #5 Sign & Confirm

Test Case ID	Test Case Name	Test Case Summary	Test Case Steps	Expected result	Actual result	Pass/fail
5	Sign & Confirm	To sign the transaction and confirm the gas fee for the transaction	If user clicks on reject	"You reject the permission request, Please refresh to try again" will be displayed.	"You reject the permission request, Please refresh to try again" is displayed.	Pass
			If user clicks on deploy again while the first request is already in pending.	"Metamask permission request is already pending" will be displayed	"Metamask permission request is already pending" is displayed	Pass
			If user clicks on confirm	"Waiting for contract to be deployed" will be displayed	"Waiting for contract to be deployed" is displayed	Pass
			If user clicks on confirm	"Contract deployment is in progress" will be displayed	"Contract deployment is in progress" is displayed	Pass
			If user clicks on confirm	"Transaction hash" will be displayed	"Transaction hash" is displayed	Pass
			If user clicks on confirm	"Transaction mined! Contract address" will be displayed	"Transaction mined! Contract address" is displayed	Pass

User Manual

1. To access the tool, first of all open anaconda or CMD
2. Type“streamlit run nft-generator” and enter
3. In the input folder field, enter the name of folder that contains base images
4. Set the amount of NFTs you want to generate
5. Click on test button to check if base images are perfect which will generate an NFT.
6. Click on generate button to generate the specified number of nfts
7. Generated nfts will be saved in default output folder, you can change the folder.

Student Log Form

Title: **NFT Generator, Token Generator& NFT Bulk Uploader**

Supervisor: **Dr. Imran Amin** Batch/Sec: **8F** Group #: **23**

Reg. # (Group members): **1812297(Guman Singh) & 1812309(Kashif)**

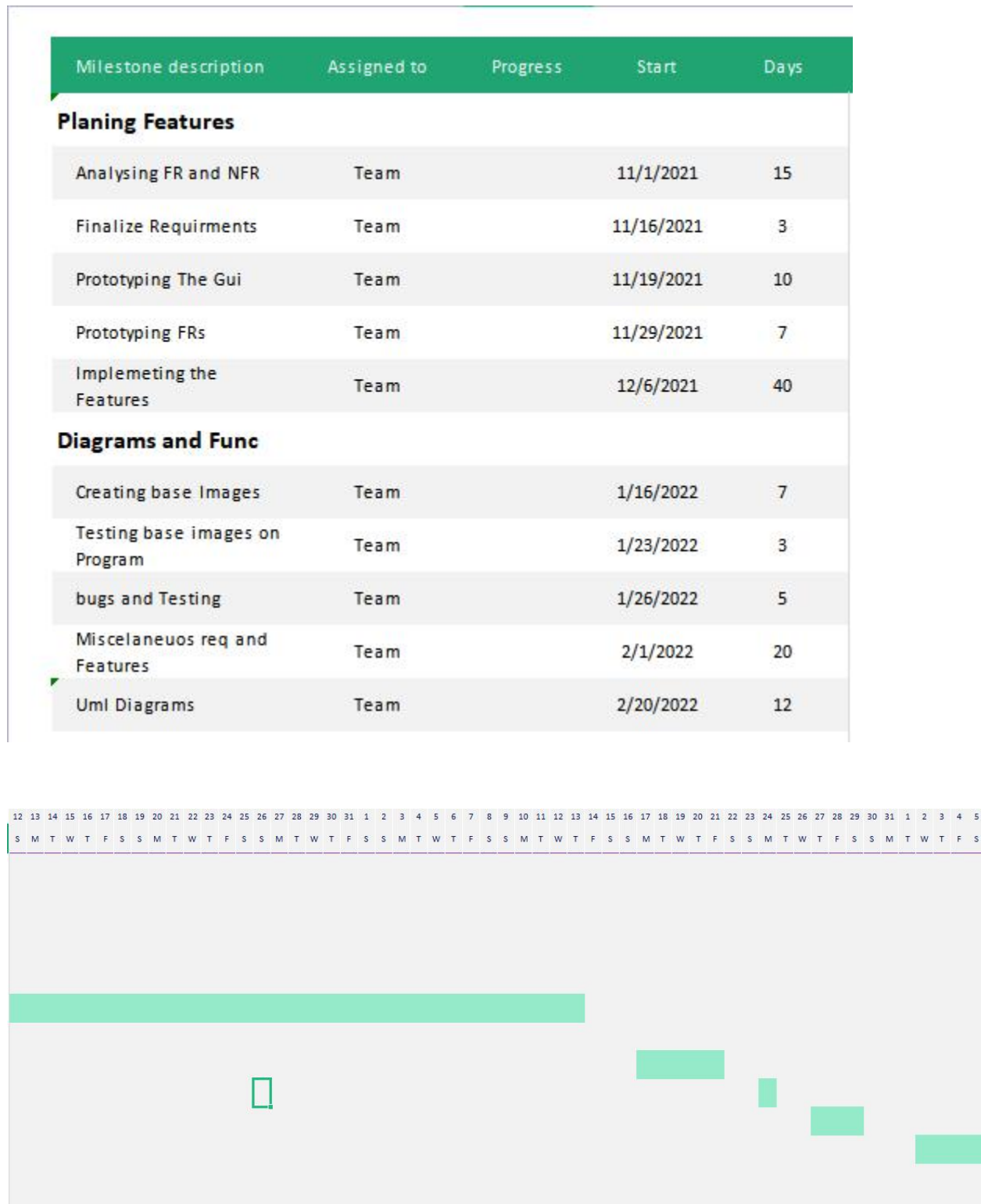
Sr	Task Assigned	Due Date	Task Completed	Supervisor
1	Prototyping Generator	07-11-2021	Basic GUI implemented	Dr. Imran Amin
2	Prototyping NFR & NFR	14-11-2021	Finalized FR and features	Dr. Imran Amin
3	SRS discussion and UML diagrams	10-12-2021	Completed	Dr. Imran Amin
4	SDS discussion and finalize diagrams	25-12-2021	Completed	Dr. Imran Amin
5	Showing & testing backend	05-01-2022	Completed	Dr. Imran Amin
6	Testing backend with base images	25-01-2022	Completed	Dr. Imran Amin
7	Add slider option	02-03-2022	Completed	Dr. Imran Amin
8	NFT bulk uploading smart contract	09-03-2022	Completed	Dr. Imran Amin
9	Mint sample NFTs using smart contract	19-03-2022	Completed	Dr. Imran Amin

10	Prototyping token generator	27-03-2022	Completed	Dr. Imran Amin
11	GUI of token generator	07-04-2022	Completed	Dr. Imran Amin
12	Implement smart contract in token generator	19-04-2022	Completed	Dr. Imran Amin
13	Deploy a token	02-05-2022	Completed	Dr. Imran Amin
14	Add token on DEX and wallet	25-05-2022	Completed	Dr. Imran Amin

Iteration Plan

GUI BASED NFT GENERATOR					
S.No.	Features	FYP-I Iterations			
		Monthly Iteration-I	Monthly Iteration-II	Monthly Iteration-III	Monthly Iteration-IV
1	documentation				
		AnalysisFR and NFR	SRS create	UML diagrams SDS draft	test cases and SDS final
		25%	50%	75%	Implementation(100%)
2	front end				
			Design and theme		
			Testing	Testing	
			Implementation(50%)	implementation(100%)	
3	Assets			Testing	
				checking on code	
		decidie size	Creating assets for test	Adjust size and varriants	
		15%	40%%	100%	
4	backend code				
				Testing with images	connect with front end
				debugging	test with new assets
				50%	Implementation(100%)
...
Output Features		1	1,2	1,2,3	1,4

Gantt Chart



Plagiarism Report

Turnitin Originality Report

Final_Year_Report_1812297,_1812309.pdf

by Anonymous



From SummerReports (Summer2022)

- Processed on 02-Aug-2022 11:05 PKT
- ID: 1878000490
- Word Count: 13733

Similarity Index

11%

Similarity by Source

Internet Sources:

10%

Publications:

2%

Student Papers:

10%

sources:

1

3% match (Internet from 21-Jun-2022)

<https://www.slideshare.net/OnlyTechForYou/online-ecommerce-website-srs>

2

2% match (Internet from 26-Apr-2022)

<https://www.coursehero.com/file/127949835/srs-1docx/>

3

1% match (Internet from 08-Jan-2022)

<https://www.coursehero.com/file/123930155/FYP-MIDTERMdocx/>

PLAGIARISM FREE CERTIFICATE

This is to certify that, I am Guman Singh S/D/0Dujai Singh ,groupleader of FYP under

Date: _____

Name of Group Leader: _____

Signature: _____

Name Of Supervisor: _____

Designation: _____

Signature: _____