**KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY COLLEGE OF SCIENCE**

|  |  |
| --- | --- |
| MINI Project Documentation  Supervisor: Dr. Owusu Agyemang | Name: Fiasorgbor George Etornam Kobla  Index No: 9404419 |

**COMPUTER SCIENCE DEPARTMENT**

\\

A simple approach to email classification(spam or not spam) using BERT

Contents

[CHAPTER ONE 3](#_Toc44754318)

[1.1 GENERAL OVERVIEW 3](#_Toc44754319)

[1.2 PROBLEM STATEMENT 5](#_Toc44754320)

[1.3 AIMS AND OBJECTIVES 6](#_Toc44754321)

[1.4 JUSTIFICATION 7](#_Toc44754322)

[1.5 RESEARCH QUESTIONS 8](#_Toc44754323)

[1.6 LIMITATIONS OF MY STUDY 8](#_Toc44754324)

[1.7 SCOPE OF THE PROJECT 9](#_Toc44754327)

[1.8 PROJECT LIMITATION 10](#_Toc44754328)

[1.9 SUMMARY 13](#_Toc44754333)

[CHAPTER TWO 10](#_Toc44754330)

[2.1 INTRODUCTION 10](#_Toc44754331)

[2.2 MACHINE LEARNING 12](#_Toc44754332)

[2.3 SPAM DETECTION 13](#_Toc44754333)

[2.4 RELATED WORK 13](#_Toc44754333)

[2.5 SUMMARY 13](#_Toc44754333)

[CHAPTER THREE 14](#_Toc44754334)

[3.1 THE PROPOSED SYSTEM 14](#_Toc44754335)

[3.2 METHODOLOGY 16](#_Toc44754336)

[3.3 DEVELOPMENT TOOLS AND ENVIRONMENT 18](#_Toc44754337)

[3.4 LANGUAGES USED FOR CODING 19](#_Toc44754338)

[3.5 REQUIREMENTS FOR USING SOLUDIM 19](#_Toc44754339)

[CHAPTER FOUR 19](#_Toc44754340)

[4.2 USE CASE DIAGRAM 20](#_Toc44754341)

[4.3 CLASS DIAGRAM 21](#_Toc44754342)

[4.4 SEQUENCE DIAGRAM 21](#_Toc44754343)

[4.5 GRAPHICAL REPRESENTATION OF DATA PROCESS IN THIS SYSTEM 22](#_Toc44754344)

[4.6 PROCESS SPECIFICATION 23](#_Toc44754345)

[4.7 STRUCTURED ENGLISH FOR THE SOLUDIM. 24](#_Toc44754346)

[4.8 PROBLEM IDENTIFICATION 24](#_Toc44754347)

[CHAPTER FIVE 25](#_Toc44754348)

[5.1 DATABASE DESIGN 26](#_Toc44754349)

[5.2 USER INTERFACE DESIGN 27](#_Toc44754350)

[5.3 INPUT DESIGN 35](#_Toc44754351)

[CHAPTER SIX 37](#_Toc44754352)

[6.1 IMPLEMENTATION DETAILS 38](#_Toc44754353)

[6.2 IMPLEMENTATION OF THE SOFTWARE 38](#_Toc44754354)

[6.3 TESTING 40](#_Toc44754355)

[CHAPTER 7 41](#_Toc44754356)

[7.1 PERFORMANCE EVALUATION 41](#_Toc44754357)

[7.2 PROGRAM MAINTAINACE 41](#_Toc44754358)

[7.3 RECOMMENDATION 41](#_Toc44754359)

[7.4 CONCLUSION 42](#_Toc44754360)

[7.5 REFERENCES 42](#_Toc44754361)

## CHAPTER ONE

INTORDUCTION

### GENERAL OVERVIEW

With the rising usage of the Internet and social networks, online communication has become an essential part of our daily life. One of the widely used mediums for formal and business communication is e-mail, because of its open access, quickness and reliability. With the email increasing popularity, spam emails were rapidly rising, spam is one or more unsolicited messages that take the form of advertising or promotional materials like debt reduction plans, getting rich quick schemes, online dating, and health-related products, etc. Automatic spam detection is no new topic, business and companies are always looking to improve their user’s experience. To protect their machines from potential damage; in case viruses were wrapped inside spam emails. Furthermore, it helps saving network resources (bandwidth) and time from wastage. NLP and ML Communities have been attracted to resolve this problem and provided several SPAM detection data-sets and train models to classify whole document. Different classification approaches may perform worse or better, depending on the type of data and task they are given. This has been achieved by employing the different NLP techniques; starting with the contribution of deep learning for understand-ing natural language. Convolutional neural network (CNN) that is used for sentence classification. BiLSTM for sequence tagging. Attention-based bidirectional LSTM networks for relational classification and topic-based sentiment analysis. Ending up with the state-of-the-art transfer learning models; where research focused on using pre-trained models for different tasks on NLP. BERT language model showcase the highest results compared to all other works. BERT is used to execute word embedding taking into account large left and right semantic contexts of words and can generate different semantic representations for the same word based on its context. In this research, BERT based Model is investigated for the task of automatic spam detection. The system is evaluated against different novel approaches on a publicly available corpus of spam and ham emails.

#### 1.1.1 BACKGROUND

Emails and SMSs are the most popular tools in today communications, and as the increase of emails and SMSs users are increase, the number of spams is also increases. Spam is any kind of unwanted, unsolicited digital communication that gets sent out in bulk, spam emails and SMSs are causing major resource wastage by unnecessarily flooding the network links. Although most spam mail originate with advertisers looking to push their products, some are much more malicious in their intent like phishing emails that aims to trick victims into giving up sensitive information like website logins or credit card information this type of cybercrime is known as phishing. To countermeasure spams, many researches and efforts are done to build spam detectors that are able to filter out messages and emails as spam or ham. In this project we build a spam detector using BERT pre-trained model that classifies emails and messages by understanding to their context, and we trained our spam detector model to detect spam and ham messages.

### PROBLEM STATEMENT

A tight competition between filtering methods and spammers is going on per day, as spammers have begun to use tricky methods to overcome the spam filters. There’s a lack of machine learning focuses on the model development that can predict the activity of these spammers. Spam is a waste of time to the user since they have to sort the unwanted junk mail and it consumes storage space and communication bandwidth. Rules in other existing filters must be constantly updated and maintained to make it more difficult for spammers to deceive spam filters. One of the stratagems used to deceive spam filters is to substitute vocables with synonyms or similar words that turn the message unrecognizable by the detection algorithms. Spam has cost businesses and individuals lots of money and caused loads of harm to victims. Several models and techniques to automatically detect spam emails have been introduced and developed yet non showed 100% predicative accuracy. Among all proposed models both machine and deep learning algorithms achieved more success. Natural language processing (NLP) enhanced the models’ accuracy. In this work, the effectiveness of word embedding in classifying spam emails is introduced. Pre-trained transformer model BERT (Bidirectional Encoder Representations from Transformers) is fine-tuned to execute the task of detecting spam emails from non-spam (HAM).

### 1.3 AIMS AND OBJECTIVES

**AIMS:**

To develop a BERT (Bidirectional Encoder Representations from Transformers) model-based spam detection program to be able to detect spam messages no matter how hard they are designed to beat spam filters.

**OBJECTIVES:**

i. To study how to use machine learning techniques for spam detection.

ii. To study how to incorporate and train models to perform tasks.

iii. To leverage modified machine learning algorithm in knowledge analysis software.

iv. To study different ways to develop more effective spam detection softwares.

### JUSTIFICATION

Implementing spam filtering is extremely important for any organization. Not only does spam filtering help keep garbage out of email inboxes, it helps with the quality of life of business emails because they run smoothly and are only used for their desired purpose. Spam filtering is essentially an anti-malware tool, as many attacks through email are trying to trick users to click on a malicious attachment, asking them to supply their credentials, and much more.

According to [Radicati Research Group Inc.,](http://www.radicati.com/" \t "_blank) email spam costs businesses up to $20.5 billion each year, and that number will only continue to rise. Spam filtering prevents these spam messages from ever entering an inbox in the first place, keeping organizations from adding to the growing statistic of lost revenue. Without spam filters, an organization's email setup wouldn't function properly, and internal data would have a higher risk of exposure to a cyber-attack.

### 1.5 RESEARCH QUESTIONS

1. How effective is a spam filter.
2. What issue can be found with spam filters.
3. Why is spam filtering important.
4. How do spam filters determine which emails are likely to be spam?

### 1.6 LIMITATIONS OF MY STUDY

The biggest disadvantage of using an email filter is that you may end up with messages being identified as being spam through a mistake of the algorithm that is used. According to [Steven Scott Bayesian](http://people.ischool.berkeley.edu/~hal/Papers/2012/fat.pdf) specialist, even with the very best spam filters on the market you can still end up with messages being improperly labeled.

While missing out on important emails is a nuisance, we need to think about the fact that you can also miss the same emails if you receive a lot of spam. How can you see that message from the boss if there are hundreds of emails sent every single day? You can be highly attentive and still miss out on some emails.

### 1.7 SCOPE OF THE PROJECT

This project needs a coordinated scope of work. These scopes will help to focus on this project. The scopes are:

i. Modified existing machine learning algorithm.

ii. Make use and classify of a data set including data preparation, classification and visualization.

iii. Score of data to determine the accuracy of spam detection

### 1.8 PROJECT LIMITATION

i. This project can only detect and calculate the accuracy of spam messages only.

ii. It focuses on filtering, analyzing and classifying the messages.

iii. Does not block the messages, it only determines if the message is spam or ham.

iv. May block harmless messages.

v. Needs constant updating and maintance.

### 1.9 SUMMARY

### Chapter one of this documentation discusses the project’s background, problem statement, objectives and project’s scope and limitation of work. From all of these, I can conclude that this project detects the spam messages by using the BERT model.

## CHAPTER TWO

LITERATURE REVIEW

### 2.1 INTRODUCTION

This chapter discusses the literature review for machine learning classifiers that are being used in previous researches and projects. It is not about information gathering but it summarizes the prior research that are related to this project. It involves the process of searching, reading, analyzing, summarizing and evaluating the reading materials based on the project.

Literature reviews on machine learning topic have shown that most spam filtering and detection techniques need to be trained and updated from time to time. Rules also need to be set for spam filtering to start working. So eventually it become burdensome to the user.

2.1 MACHINE LEARNING

In this project, existing machine learning algorithm is used and modified to fit the need of project. The reasons are because machine learning algorithm is adept at reviewing larges volume of data. It is typically improved over time because of the ever-increasing data that are processed. It gives the algorithm more experience and be used to make better predictions.

Machine learning allows for instantaneous adaption without human intervention. It identifies new threats and trends and implements the appropriate measures. It is also save time as it is it automated nature

#### 2.1.1 SPAM DETECTION

The overall approach and tools used to execute the spam email detection task are described in detail. Generally, any NLP task consists of five main phases: data collection, data pre-processing, feature extraction, model training, and model evaluation. Fig. 1 shows the flow for those phases. Hence, in this work, feature extraction will be done automatically as part of the deep learning model training,

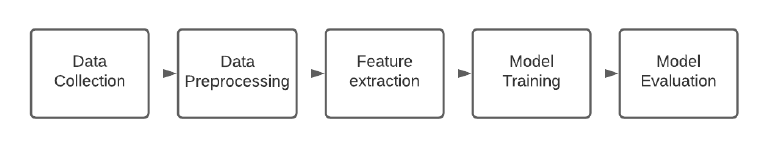


Fig. 1. Genearl Flow of Main Phases for NLP Task

*Data Collection* Tow open-source data sets were used in this work, both have two columns, the body text of the email and the class label spam or ham. The first data set is the open source Spambase data set from the UCI machine learning repository, the data set contains 5569 emails, of which 745 are spam. The second data set is the opensource Spam filter data set from Kaggle which contains 5728 emails of which 1368 are spam. Exploring the distribution for SPAM and HAM classes, both data sets are imbalanced; where the SPAM class is the rare class. In order to avoid biasing for the major class which is the HAM class, a new balanced training data set is created through merging spam samples that are randomly selected from the two data sets Spambase and Spam filter with ensuring of no duplicate records. The new data set contains 2000 samples of SPAM and 3000 samples of HAM. In addition, hold-out portion from Spambase data set was reserved for testing the persistence of the model against unseen samples. The task is a binary-classification problem to detect whether the text is classified as SPAM 1 or HAM 0. Table 2 shows the distribution for both classes HAM and SPAM in both the training and testing data sets.

*Data Cleaning and Pre-Processing* After going through the class distribution for each target, the distribution of the word number in each record is explored. SPAM texts are longer than HAM texts in general, the input sequence for our model in this case study is defined to be 300 tokens or words. The next step was to clean the input data by extracting and removing the stop words using the Sklearn library, since stop words provide little to no unique information that can be used for classification, then punctuation marks were extracted and removed as they affect the text encoding part, especially when it is attached with a word, for e.g: [users, user’s] have different encoding. Text kept in cased shape as it might be an indicator for SPAM, especially promotion scams or other types of Spams. Keras tokenization tool used to split the words as tokens based on the space. For example, [“We went to Amman.”] it will be tokenized as the following: [‘We’, ‘went’, ’to’, ’Amman.’]. Each token got encoded to vector for classical classifiers using TfidfVectorizer from Keras. The label target variable is encoded to a binary format SPAM 1 and HAM 0.

*Base Line Model* The baseline model is A state-of-the-art BiLSTM model. It takes the input from an embedding layer and its output vectors are fed to a Dense layer. The Dense layer uses Relu activation function to allows converging quickly . The next layer is a Dropout layer of 0.1 to avoid overfitting. Last layer is a Dense layer that uses Sigmoid activation function to normalize the output. Fig. 2 illustrate the baseline model structure and its layers. In addition to BiLSTM model, a set of classic classifiers KNN with n-neighbors=3 and MultinomialNB are trained to compare results with.

*Transformer Model* BERT transformer is a pre-trained model published by Google AI language. It uses attention models to learn the contextual relation between the words in a sentence. It mainly consists of two parts: an encoder that encodes the input text and a decoder for the output result based on the task. The bert-base-cased model implemented using Simple Transformers library which is an API built above Hugging face library. Bert cased model has been trained on English Wikipedia with 2500 million words and BookCorpus with 800 million words.The model was trained using hyper-parameters: 300 sequence length, three epochs, 32 batch size, 4e-5 learning rate, and AdamW as an optimizer.

### 2.2 RELATED WORK

Spam e-mails detection problem has already drawn researchers’ attention. Several significant works to detect spam e-mails have been proposed. In this section; prior related works that focus on the spam classification using ML and deep learning techniques are discussed.

Thiago S. Guzella et. Al (2009) conducted “A Review of Machine Learning Approaches to Spam Filtering”. In their paper, they found that Bayesian Filters that are used to filter spam required a long training period for it to learn before it can completely well function.

S. Ananthi (2009), conducted research on “Spam Filtering using K-NN”. In this paper, she used KNN as it is one of the simplest algorithms. An object is classified by a majority vote of its neighbours where the class is typically small.

Anjali Sharma et. Al (2014) has conducted “A Survey on Spam Detection Techniques”. In this paper, they found that Artificial Neural Network (ANN) must be trained first to categorize emails into spam or non-spam starting from the particular data sets.

Simon Tong and Daphne Koller (2001), has conducted research on “Support Vector Machine Active Learning with Applications to Text Classification”. In this paper, they presented new algorithm for active learning with SVM induction and transduction. It is used to reduce version space as much as it can at every query. They found out that the existing dataset only differ by one instance from the original labelled data set.

Minoru Sasaki and Hiroyuki Shinnou (2005) has conducted research on “Spam Detection Using Text Clustering”. They used text clustering based on vector space model to construct a new spam detection technique. This new spam detection model can find spam more efficiently even with various kinds of mail.

Aigars Mahinovs and Ashutosh Tiwari (2007) has conducted research on “Text Classification Method Review”. They test the process of text classification using different classifier which is natural language processing, statistical classification, functional classification and neural classification. They found that all the classifier works well but need more improvement especially to the feature preparation and classification engine itself in order to optimize the classification performance.

### 2.3 SUMMARY

### This chapter discussed the technique and model used in the proposed project. The method and model are chosen based on the previous research articles and journals. Some advantages and disadvantages of each model are also discussed.

## CHAPTER THREE

METHODOLOGY

INTORDUCTION

### 3.1 INTRODUCTION

This chapter will explain the specific details on the methodology being used in order to develop this project. Methodology is an important role as a guide for this project to make sure it is in the right path and working as well as plan. There is different type of methodology used in order to do spam detection and filtering. So, it is important to choose the right and suitable methodology thus it is necessary to understand the application functionality itself.

#### 3.2 **IMPLEMENTATION AND CODING**

This project is developed by using Python Language and combining with the implementation of the BERT model. Google Collaboratory is the platform used to develop the project. It contains important functions for preprocessing the dataset. Then, the dataset is going to be used to train and test the machine learning model and achieve the objectives of the project.

#### 3.3 **PROJECT REQUIREMENT AND SPECIFICATION**

System requirements are needed in order to accomplish the project goals and objectives and to assist in development of the project. That involves the usage of hardware and software. Each of these requirements is related to each other to make sure that system can be developed smoothly.

**3.3.1 HARDWARE**

The hardware used in completing this project are as follows;

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Hardware | Type | Description |
| 1. | Laptop | HP Pavilion |  Processor: Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.40 GHz   OS version: Windows 64 bit   RAM: 8 GB |
| 2. | Printer | HP Deskjet |  Printing document |
| 3. | Printed paperwork | A4 paper |  Used to study on how to implement this project from past paperwork |

**3.3.2 SOFTWARE**

The software used in completing this project are as follows;

|  |  |  |
| --- | --- | --- |
| No. | Software | Description |
| 1. | Google Collaboratory |  Machine learning platform   Deploy models   Run models in cloud |
| 2. | Tensorflow\_hub |  Place where all TensorFlow pre-trained models are stored. |
| 3. | Tensorflow |  For model creation |
| 4. | Pandas |  For data loading, manipulation and wrangling. |
| 5. | Tensorflow\_text: |  Allows additional NLP text processing capabilities outside the scope of tensorflow. |
| 6. | Skelarn |  For doing data splitting |
| 7. | Matplotlib |  For visualization support |
| 8. | Microsoft Word 2016 |  Creating and editing report |
| 9. | Microsoft PowerPoint 2016 |  For presenting finding and result of the project |
| 10. | Github |  Get dataset |
| 11. | Snipping Tool |  Captures and screenshot images |

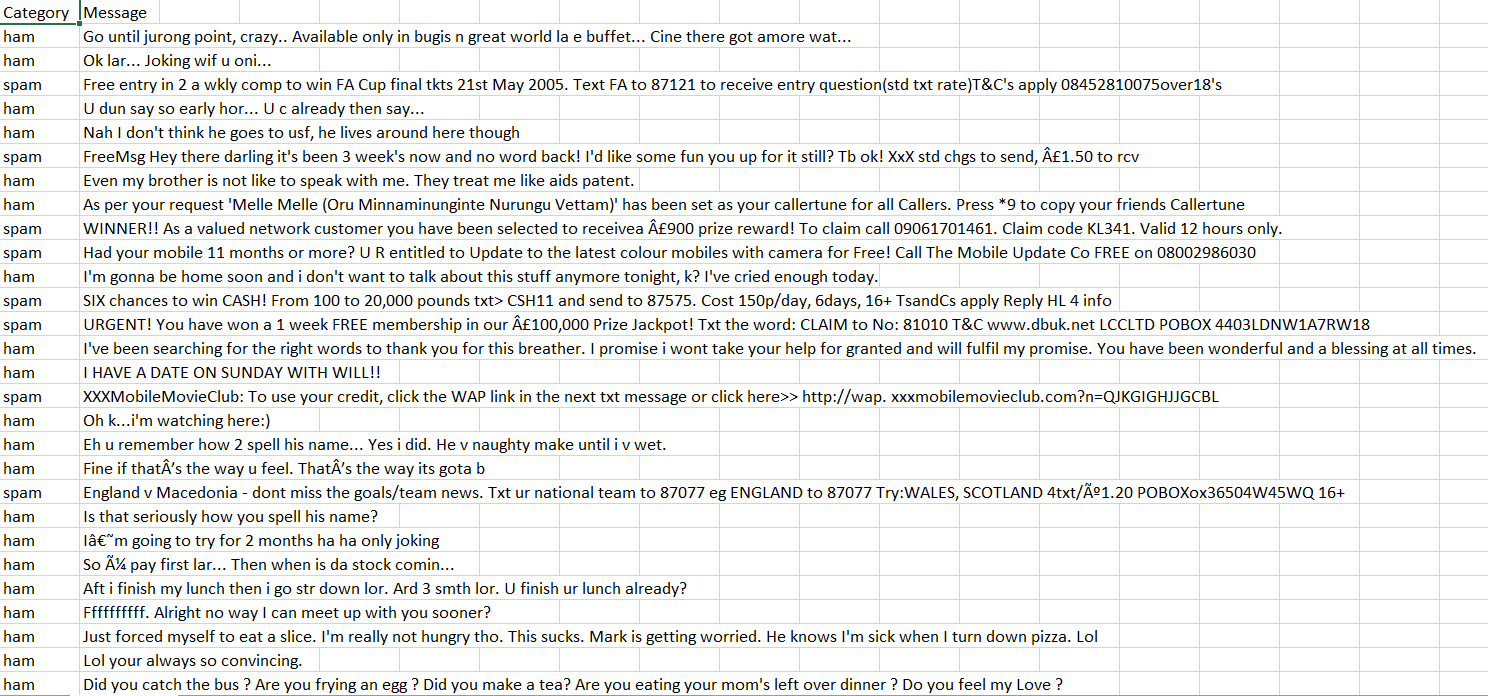
#### 3.4 FRAMEWORK

3.4.1 DATA SOURCE

#### Collecting data is utterly difficult due to numerous constraints for instances the volume of data and the throughput required for proper and timely ingestion. The dataset that I’ve used in this project is the real existing data that can be downloaded from machine learning data repository site.

3.4.2 DATA SETS

To start we will first download a readily available dataset. The data first need to be reformatted into .CSV to make it easier to use for further process. Once you downloaded it will look something like this:



#### 3.5 PROPOSED MODEL

1. Load Data – I will be loading the data which is simple [2 categories (ham and spam) along with corresponding emails] CSV file.

2. EDA – Perform some EDA to get a feel of what data looks like

3. Pre-Processing – Based on the results of EDA we will be going to apply some preprocessing to the data to make it model friendly

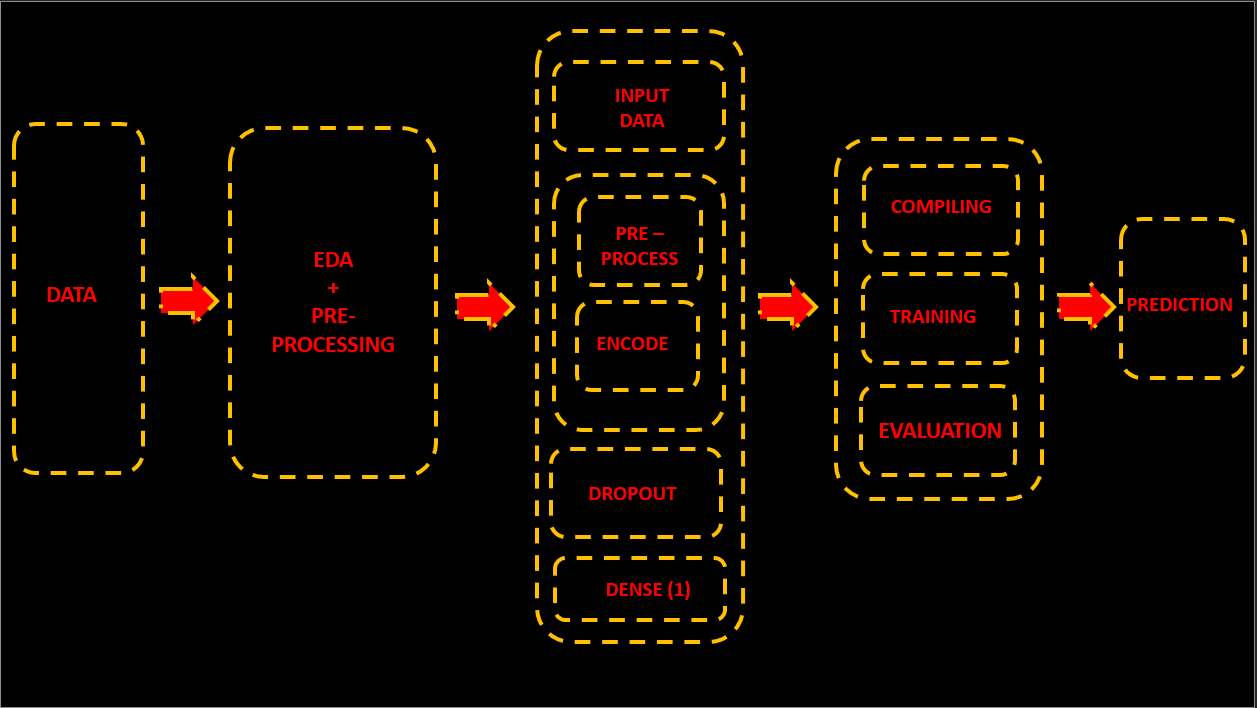
4 – Model Creation – We will use**Functional API\*** to build our Deep Learning Model which will consist of

* **INPUT** – Will create an input layer that will take in all the pre-processing data and pass it to bert\_processor.
* **BERT PREPROCESSOR** – Process our input text to be in the format which encoder accepts. (Adds MASK AND ENCODINGS).
* **BERT ENCODER** – Feed our processed text to the bert model which will eventually generate a contextualized word embedding for our training data.
* **DROPOUT**– Add a dropout layer that will randomly drop out a few neurons from the network to take care overfitting problem.
* **SOFTMAX**– At last we will add a softmax layer to predict data in 2 categories.
* + 2 more generic steps.

5- Model Evaluation – Further we will check how the model performs on the test data and plot some relevant charts and metrics based on the observations.

6- Predict Data – Finally we will predict our own emails using the model.

Here is a general visualization of the flow of logic:



#### 3.6 SUMMARY

Methodology is one of the most important roles in system and application development. There are a lot of different software development methodology available that can be used to develop any kind of application. The right methodology can help the project to be done according to the specified time. The activities in each phase in the methodology are explained so that it can be understood easily.

## CHAPTER FOUR

**IMPLEMENTATION AND RESULT**

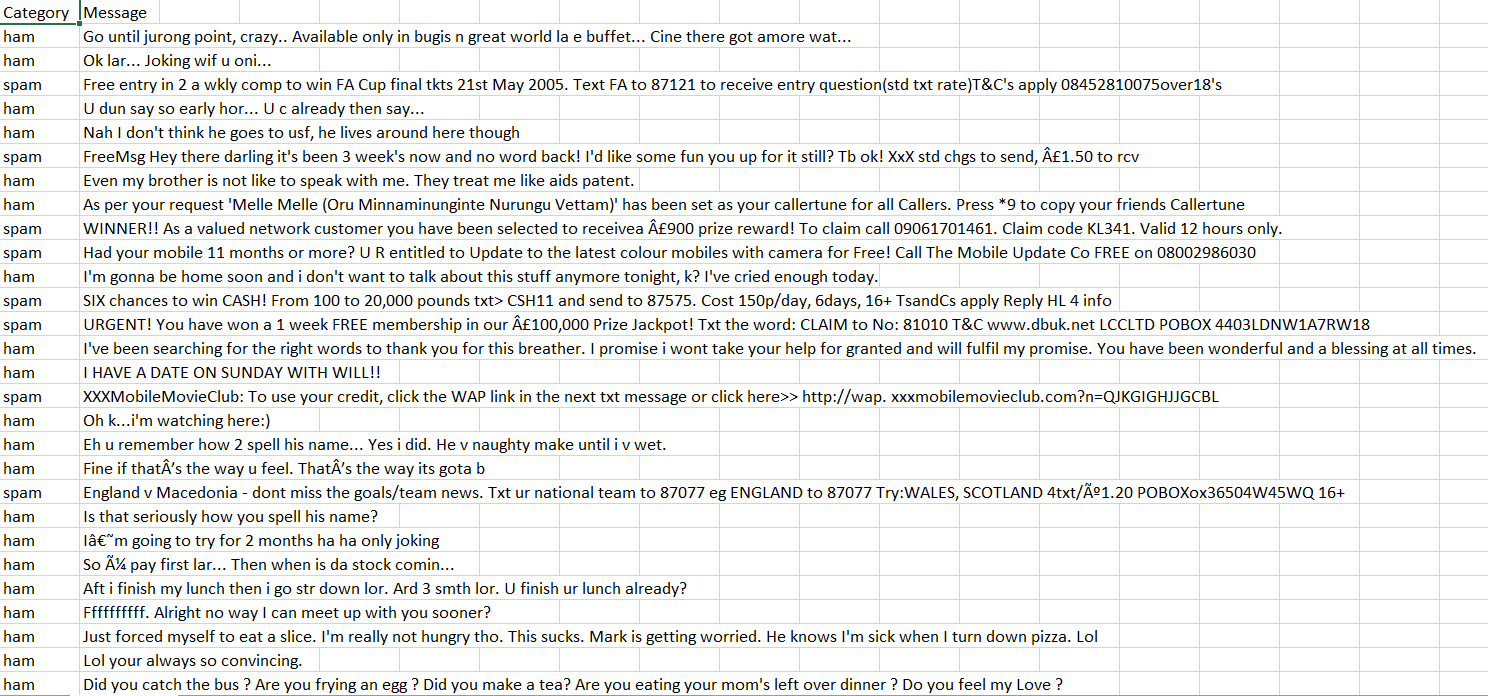
**4.1 INTRODUCTION**

In this chapter, I will discuss the project implementation and testing. Both implementation and testing result are the last stage of the project development. Implementation is necessary to verify that the project development of the trained model meet the requirement. Testing result is the process of showing the final result of the testing the testing that have been done to ensure its functionality. At this phase, it will show that the machine learning model is well functioning and to identify any weaknesses to be improved later on. So, this chapter will generally discuss the implementation, deployment and testing of the entire project after being developed.

### 4.2 IMPLIMENTATION

**Implementing Spam Detection Using BERT**

To start we will first download a readily available dataset. Once you downloaded it will look something like this:

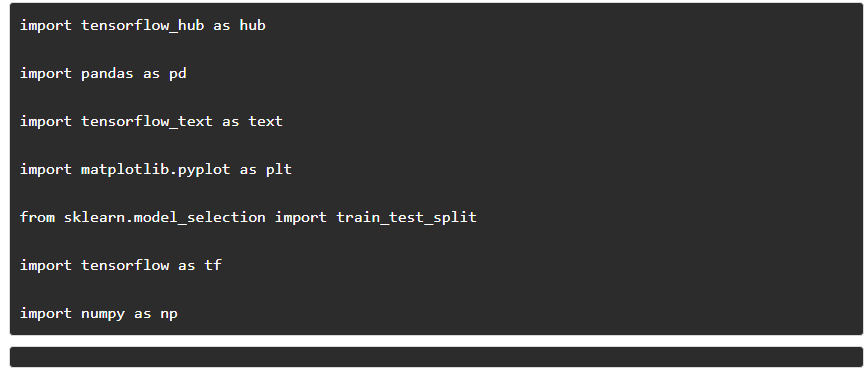


Here ham – Good Mails and Spam – Spam mails. Our job will be to build a model which can train on the data and return prediction when given new inputs.

**Loading Dependencies**

So, first of all, we are going to import few libraries namely:

* **Tensorflow\_hub**: Place where all TensorFlow pre-trained models are stored.
* **Tensorflow:** For model creation
* **Pandas:** For data loading, manipulation and wrangling.
* **Tensorflow\_text**: Allows additional NLP text processing capabilities outside the scope of tensorflow.
* **Skelarn:** For doing data splitting
* **Matplotlib**: For visualization support



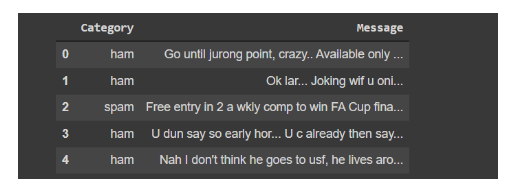
**Loading Data**

Now we will just load our data into a pandas dataframe ‘**df**‘ using its **read\_csv()** method/fn and view the

first 5 samples using the **head()** method:



Executing the above code returns a data frame with the first 5 samples:



**Spam Detection Exploratory Data Analysis**

We will now do some of the Exploratory – Data Analysis to check how data is distributed along two categories. This will give us a feel if we need to do some type of preprocessing over data or is it on the same scale.

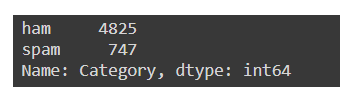
To perform this operation we will just be **grouping the data based on category** and call **value\_counts()**

method on it like:

# check count and unique and top values and their frequency

df['Category'].value\_counts()

This returns **no of samples** of each class as:



Clearly, the data is imbalanced and there are more good emails(ham) than spam emails. This may lead to a problem as a model may learn all the features of the ham emails over spam emails and thus always predict all emails as ham (OVERFITTIN!). So before proceeding, we need to take care of that.

**Downsampling Data**

Downsampling is a technique where the majority class is downsampled to match the minority class. Since our data has only one column(feature) it ok to use it.

We perform downsampling by just picking any random 747 samples from the ham class. Here is the code

to do that:

1) We first calculated the percentage of data that needs to be balanced by **dividing minority (spam) by**

**majority(ham)**:

# check percentage of data - states how much data needs to be balanced print(str(round(747/4825,2))+'%')

>> 0.15%

**It came out to be 15% of data that needs to be balanced.**

2) We then create 2 new datasets namely ham and spam and filtered the data having categories as spam

and ham and append it to the respective dataset and finally printed their shape to confirm the filtering and

creation:)

# creating 2 new dataframe as df\_ham , df\_spam df\_spam = df[df['Category']=='spam'] df\_ham =

df[df['Category']=='ham'] print("Ham Dataset Shape:", df\_ham.shape) print("Spam Dataset Shape:",

df\_spam.shape)

>> Spam Dataset Shape: (747, 2)

Ham Dataset Shape: (4825, 2)

**Seems like filtering and separation were successful.**

3) Now we will sample the ham dataset using the sample() method with the shape of our spam dataset and

to be more specific load it into a new dataframe **df\_ham\_downsampled and** print its shape to cross verify.

# downsampling ham dataset - take only random 747 example # will use df\_spam.shape[0] - 747

df\_ham\_downsampled = df\_ham.sample(df\_spam.shape[0]) df\_ham\_downsampled.shape

>> (747, 2)

4) Finally we will **concatenate** our **df\_ham\_downsampled** and **df\_spam** to create a final dataframe called **df\_balalnced.**

# concating both dataset - df\_spam and df\_ham\_balanced to create df\_balanced dataset df\_balanced =pd.concat([df\_spam , df\_ham\_downsampled])

Checking the value counts again projects downsampling we have done-

df\_balanced['Category'].value\_counts()

Returns:

>> spam 747

ham 747

Name: Category, dtype: int64

Printing few samples wouldn’t hurt either – 10 samples

df\_balanced.sample(10)

**Returns – Evenly distributed data of spam and ham (no in cade of head)**

**Category Message**

**5164** spam Congrats on 2 mobile 3G Videophones R yours. call…

**1344** ham Crazy ar he’s married. Ü like gd-looking guys …

**1875** spam Would you like to see my XXX pics they are so …

**907** spam all the latest from Stereophonics, Marley, Di…

**3848** spam Fantasy Football is back on your TV. Go to Sky…

**4973** ham I’m fine. Hope you are well. Do take care.

**1540** ham You’re not sure that I’m not trying to make a…

**1312** ham U r too much close to my heart. If u go away i…

**4086** spam Orange brings you ringtones from all-time Char…

**3010** spam Update\_Now – 12Mths Half Price Orange line ren…

**Preprocessing of Spam Detection Data**

**One Hot Encoding Categories**

As can be seen, we have only text as categorical data, and the model doesn’t understand them. So instead

of text, we can just assign integer labels to our class **ham and spam as 0 and 1 respectively,** and store it in

new column **spam. This is called- Hot-Encoding**

To achieve this we will just be filtering the column category and perform operations:

1 – If a category is a ham/ not spam

0 – if the category is spam

The best part is that we can use the **one-liner lambda function** to achieve the following result and **apply** it

to the dataframe for all values.

Lambda Fn Syntax = [lambda x : value expression else value]

So we can use the following code to achieve the desired result:

# creating numerical repersentation of category - one hot encoding df\_balanced['spam'] =

df\_balanced['Category'].apply(lambda x:1 if x=='spam' else 0)

For checking the result we can now just print few samples as :

# displaying data - spam -1 , ham-0 df\_balanced.sample(4)

Returns :

**Category Message spam**

**2100** spam SMS SERVICES. for your inclusive text credits,… 1

**5120** spam PRIVATE! Your 2003 Account Statement for 078 1

**4590** ham Have you not finished work yet or something? 0

**2066** ham Cos daddy arranging time c wat time fetch ü ma… 0

Modification to the data frame was successful.

**Performing Train Test Split**

Now as our data is processed, we can feed it to the model, but if we do so it may be that model will learn

the patterns of the data, and when we evaluate it will always predict the right results, which leads to

biasing of the model. So we will follow the train test strategy.

***Train Test Split Strategy:***

It states out of all data (population) a large amount of data goes in for training(about80%) which has inputs(Messages) and Labels(1/,0)

and the remaining 20% will not be fed to the model. When we further do evaluation we can just predict on that 20% and see how it

performs as the model will not be biased now.

\* Note :- it is generally considered to split data as 80:20 as good ratio but can be experimented:)

The above can be achieved using the **train\_test\_split** fn of **sklearn.**

# loading train test split from sklearn.model\_selection import train\_test\_split X\_train, X\_test , y\_train,

y\_test = train\_test\_split(df\_balanced['Message'], df\_balanced['spam'], stratify = df\_balanced['spam'])

Here we first loaded the fn from sklearn.model\_selection module and performed the split and set **stratify**

to select almost equal no samples from the dataset.

Here:

X\_train, y\_train – training inputs and labels – Training Set

X\_test, y\_test – testing inputs and labels – Testing Set

This marks the end of the pre-processing part and now our model is ready for training. But before that, we

need to generate word embedding and that’s what we are going to see in the next section:)

### 

### 4.4 SEQUENCE DIAGRAM

Sequence diagrams are often used to illustrate the processing described in use case scenarios. In practice, sequence diagrams are derived from use case analysis and are used in systems design to derive the interactions, relationships, and methods of the objects in the system. Sequence diagrams are used to show the overall pattern of the activities or interactions in a use case. The sequence diagram for the System is shown in fig 1.2 below.

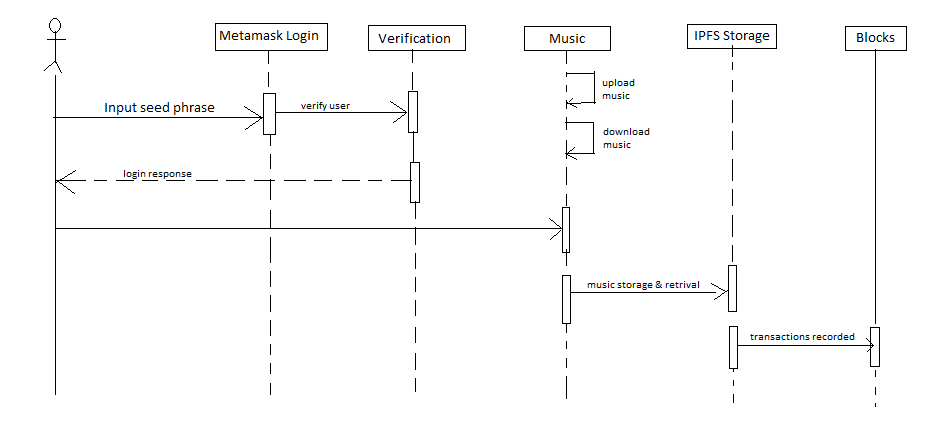


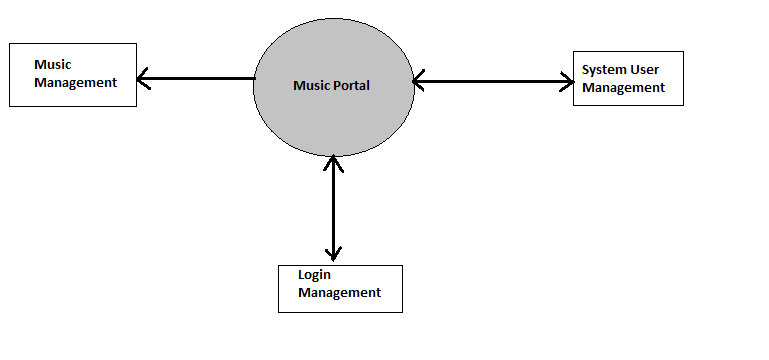
Fig 1.2

### 4.5 GRAPHICAL REPRESENTATION OF DATA PROCESS IN THIS SYSTEM

Systems analyst can put together a graphical representation of data processes throughout the organization.

By using combinations of only four symbols (Entity, Data Flow, Process, Data Store), the systems analyst can create a pictorial depiction of processes that will eventually provide solid system documentation.

Data Flow Diagram for **Soludim**



### 4.6 PROCESS SPECIFICATION

Process specifications—sometimes called *minispecs,* because they are a small portion of the total project specifications—are created for primitive processes on a data flow diagram as well as for some higher-level processes that explode to a child diagram. They also may be created for class methods in object-oriented design, and, in a more general sense, for the steps in a use case. These specifications explain the decision making logic and formulas that will transform process input data into output. Each derived element must have process logic to show how it is produced from the base elements or other previously created derived elements that are input to the primitive process.

### 4.7 STRUCTURED ENGLISH FOR THE SOLUDIM.

IF userHasMetamak

THEN sync with web app

User clicks on songs or transaction or upload Songs

IF userHasLogedIn with metamask

THEN continue

ELSE

Show metamask login page

IF seed phrase or password is correct

Then continue

ELSE

Go back to metamask login page

ENDIF

ENDIF

ELSE

Navigate to Metamask download page

ENDIF

EXIT

### 4.8 PROBLEM IDENTIFICATION

* **Vulnerability to Hackers**

With the traditional sites, users are required to give sensitive details like their

credit card numbers and all that and since the website too is not that much secured,

a hacker can make advantage of it and still that information which will be messy.

* **No Central Point**

Users have full control to the system. There is no middle point of confirmation. Users can upload their songs on their own, download etc. There is no need for users to contact admins for them to upload their songs for them.

* **Unwanted songs**

Some labels on songs sometimes confuse users to buy the wrong song. Users would only get the chance to play the song after they have bought it which is messy and painful when you realized you paid for the wrong song but in that case there is no turning back.

## CHAPTER FIVE

SYSTEM DESIGN

System Design is one of the tasking sections of the Programming. In this section of the project many previews are going to be seen and we are gradually getting close to the new system. System design is a transition from a user-oriented document to a document oriented to programmers or database personnel. The system design is structured into the following parts:

* User Interface design
* Input design
* Database design
* System’s flow (using the entity relationship diagram)

### 5.1 DATABASE DESIGN

Database is a file composed of records, each containing fields together with a set of operations it helps in organizing data in a logical order for references. Database contains related data which are organized together in a group of object, table, and file.

With the widespread use of computer technology and network technology, the development of database technology has become an important part of advanced information technology. All applications have their own database, which stores large number of key data in it. The core of the attendance system is how to use and operate it database.

The system uses IPFS was storing the songs. Transactions are stored in chain of blocks. Each block has three basic elements.

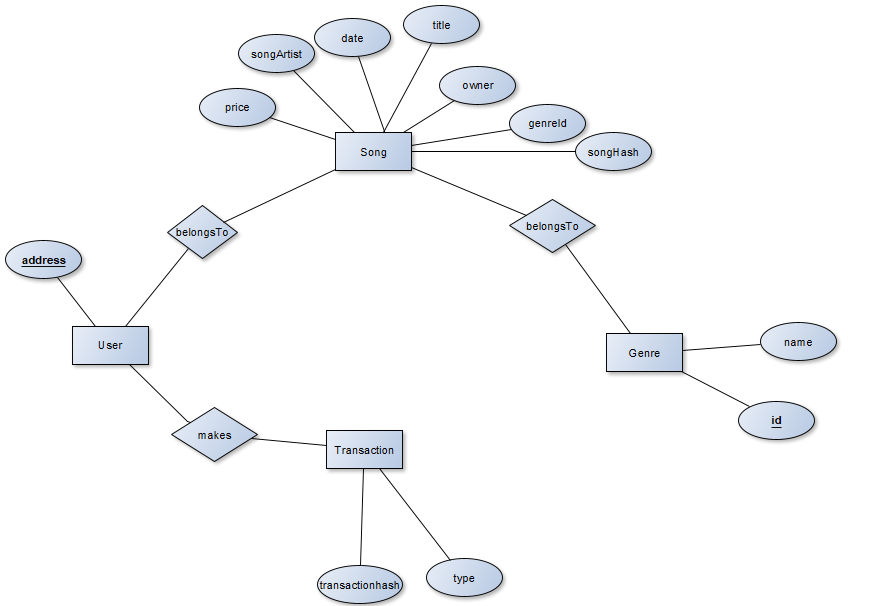
1. The data in the block.

2. A 32-bit whole number called nonce. The nonce is generated randomly when a block is created which then generate a block header hash.

3. The hash is a 256-bit number wedded to the nonce.

#### 5.1.1 ENTITY RELATIONSHIP DIAGRAM

This is the logical structure that represents the blue print of the system in other words, it defines as the algorithm of the software in a concise and logical order. The process design is represented diagrammatically in the form of ER diagram shown below

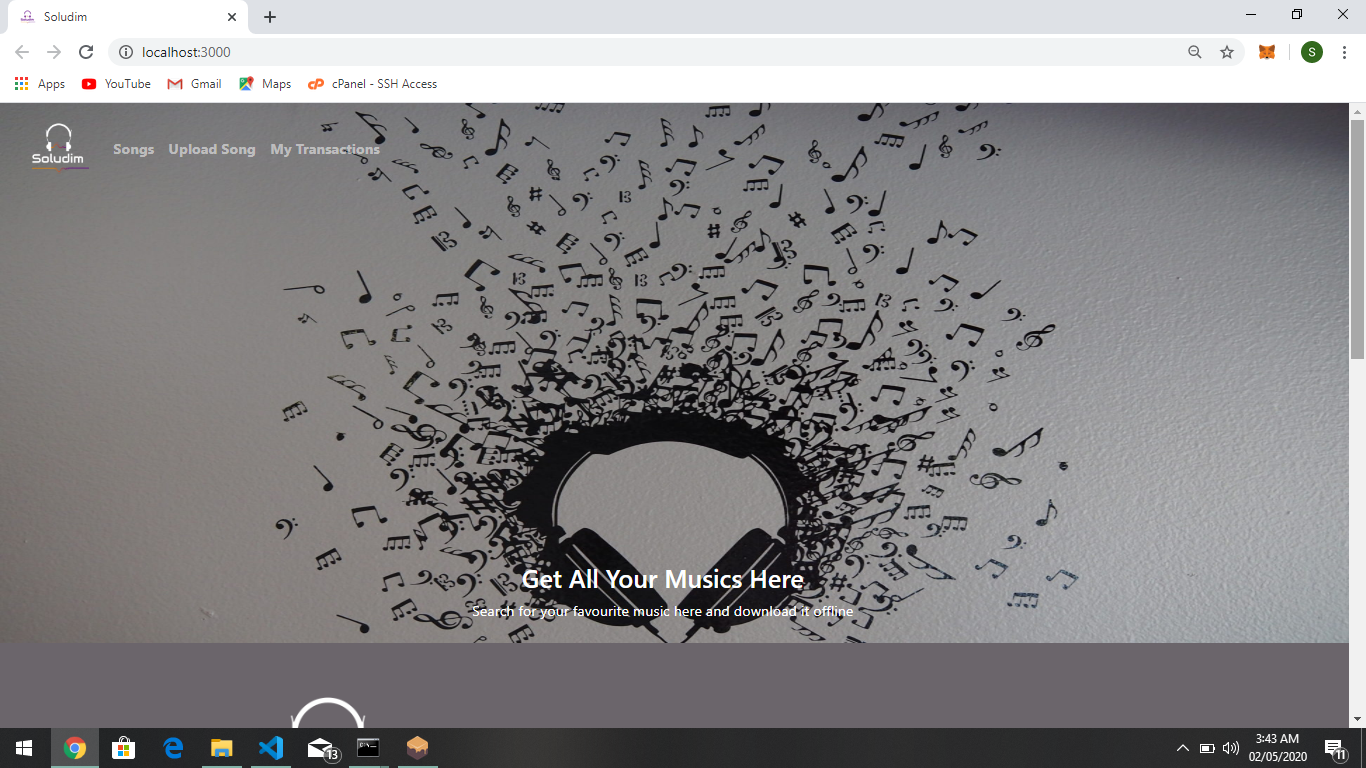
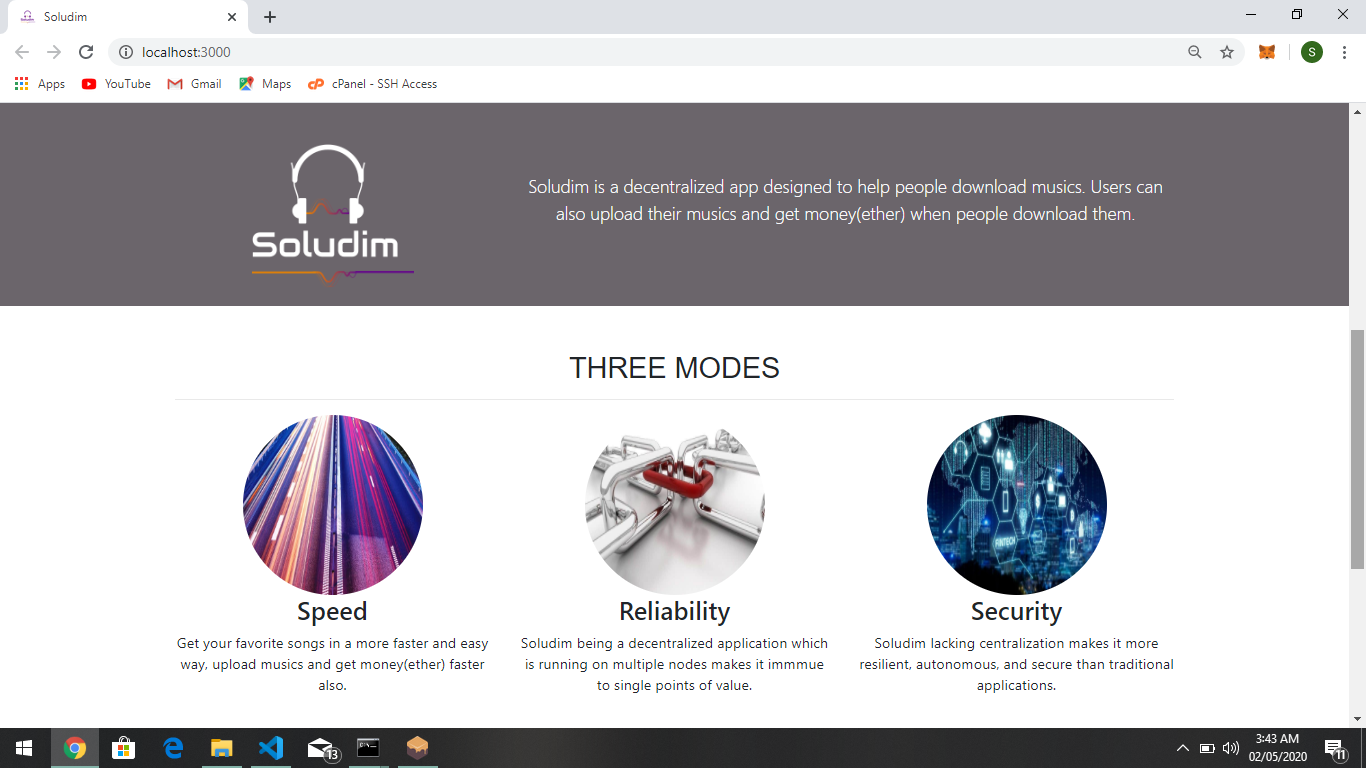


### 5.2 USER INTERFACE DESIGN

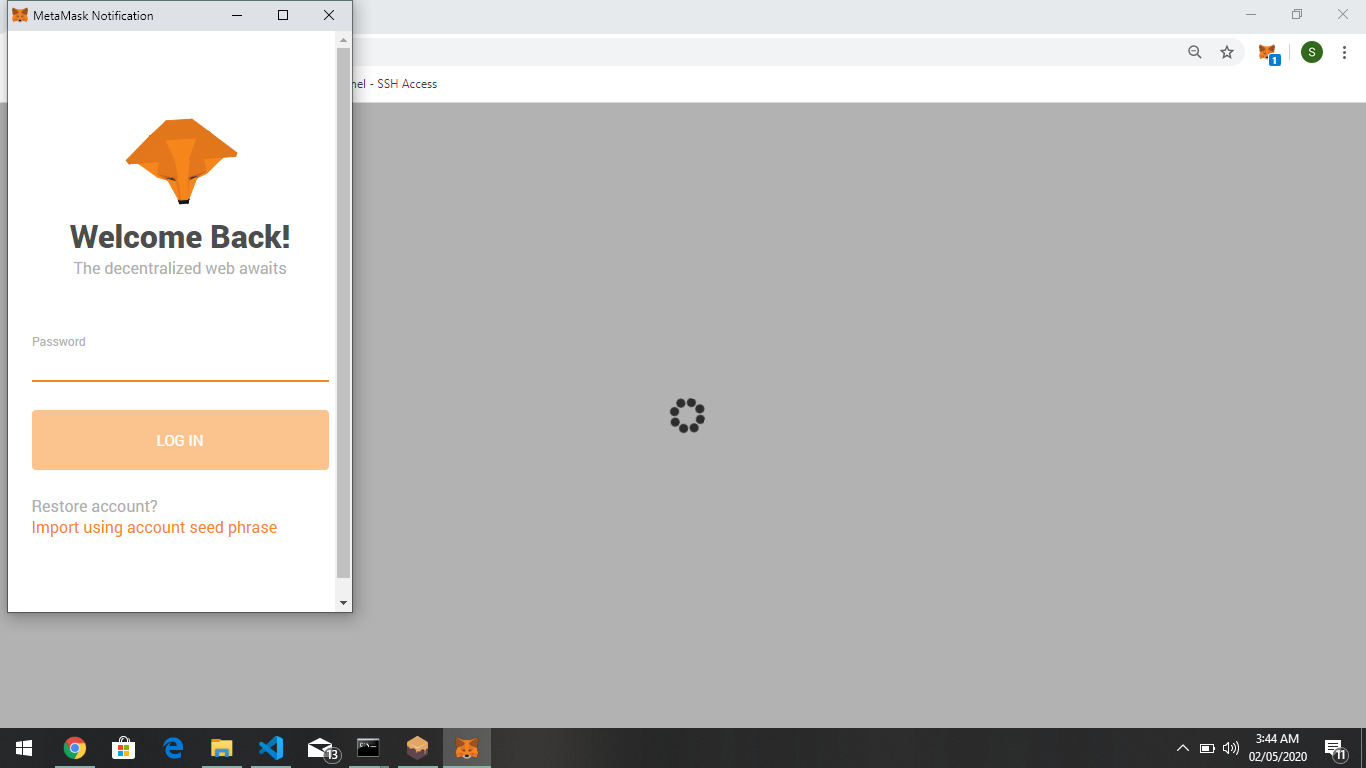
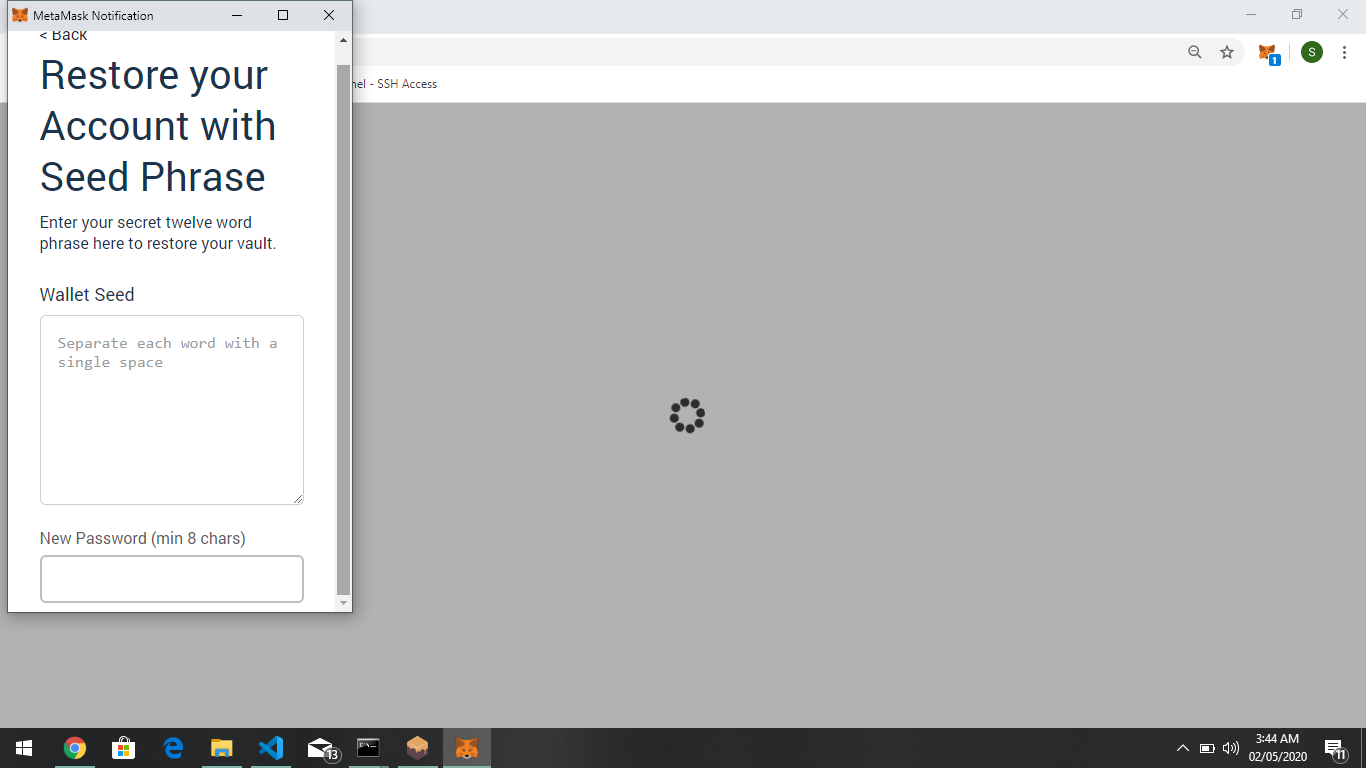
Is a type of user interface that allows users to interact with electronic devices (application) through graphical icons and visual indicators such as secondary notation, instead of text-based interfaces, typed command labels or text navigation.

An attractive, easy to use GUI attract more people to use an application. Easy to use aspect of it releases the stress of going through whole bunch of steps to get things done.

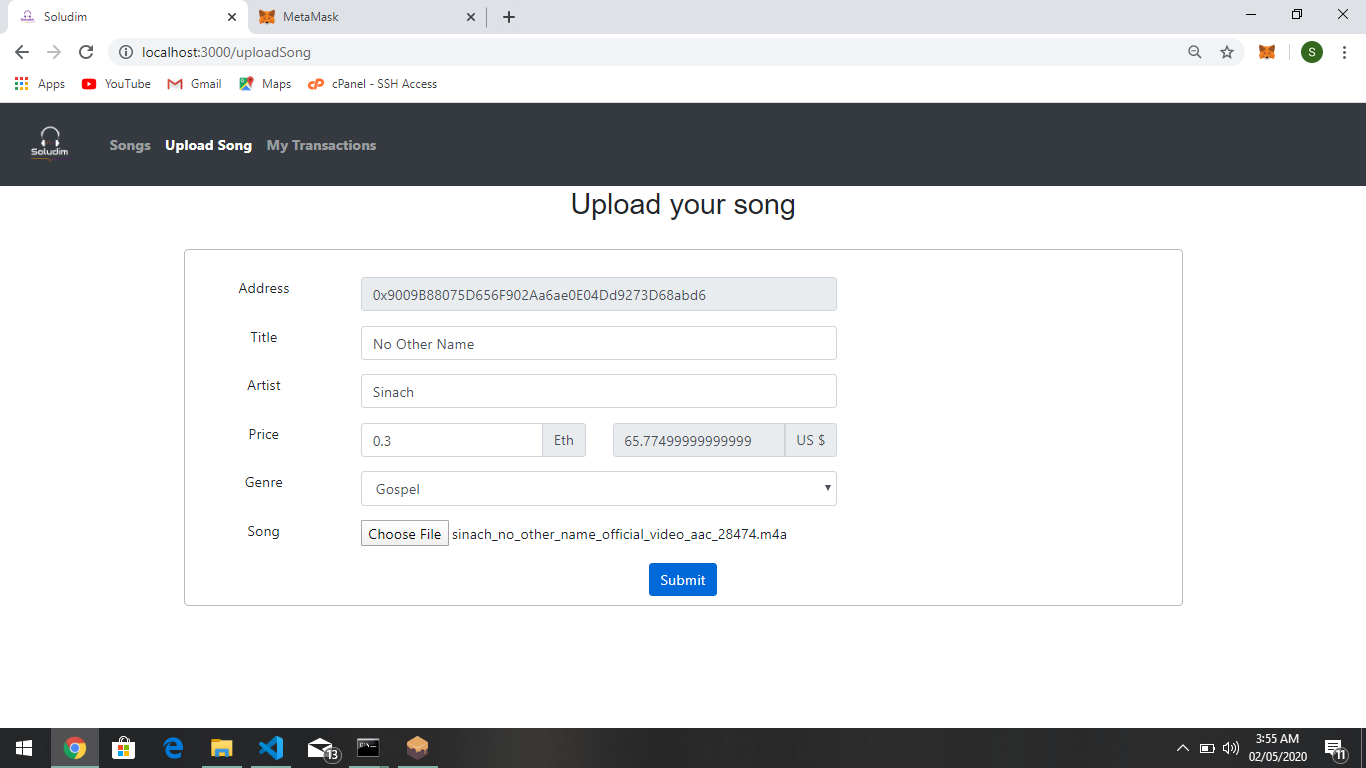
Below are previews of the output design of **Soludim**:

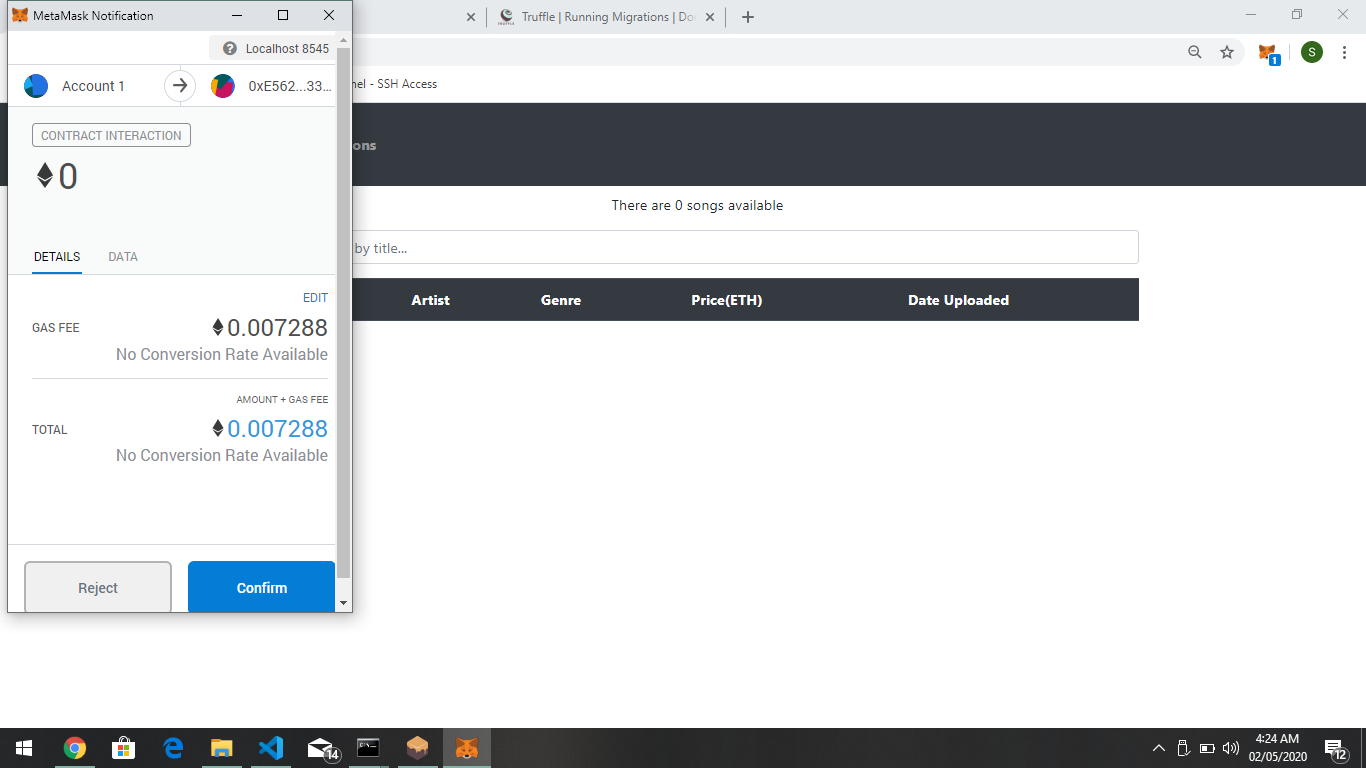
**HOMEPAGE**

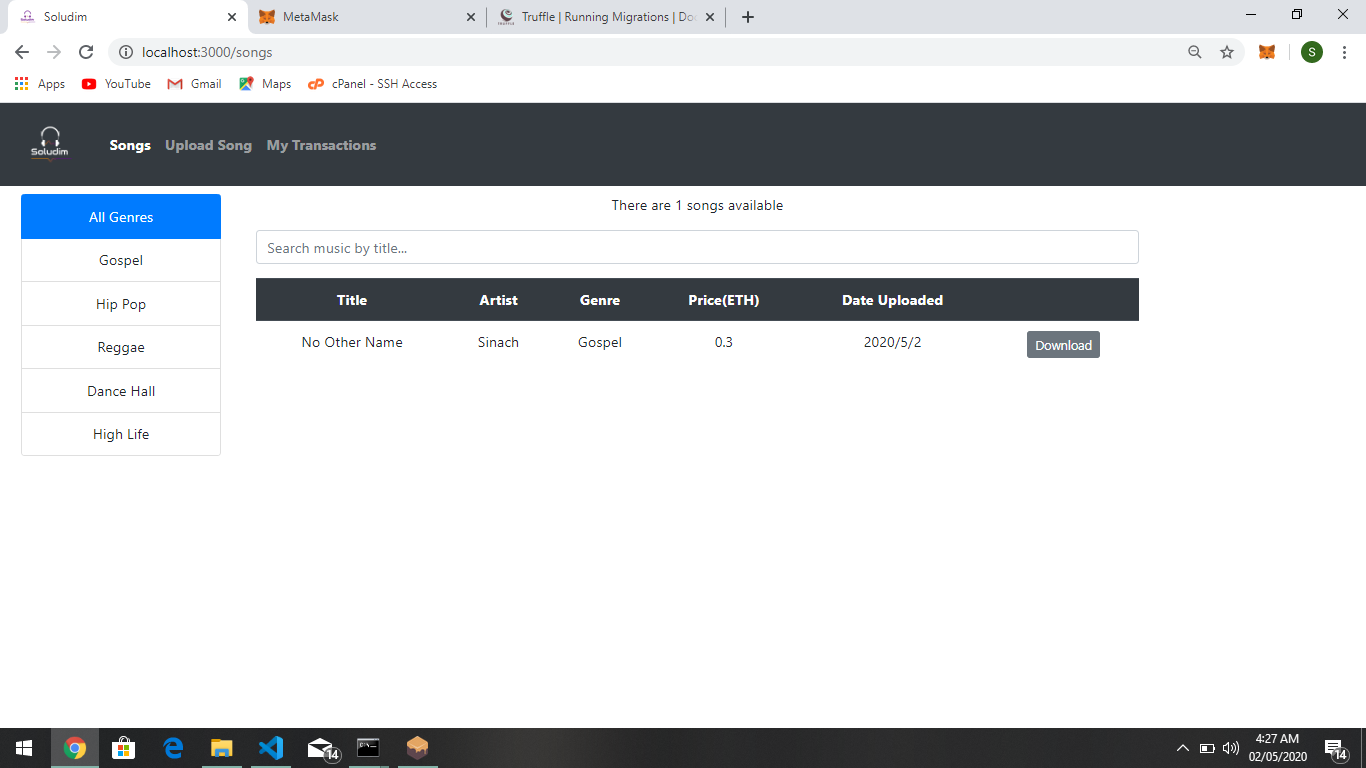
**USER AUTHENTICATION**



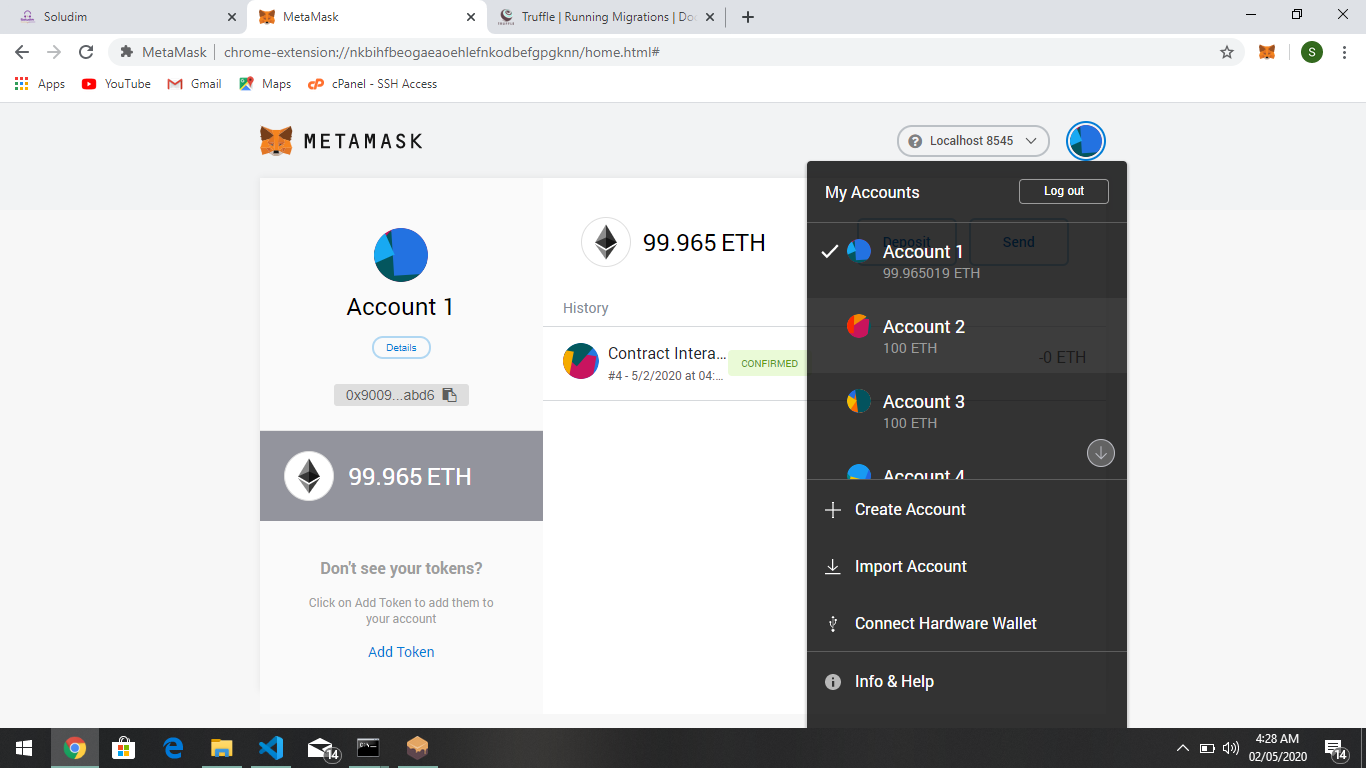
**UPLOADING SONG**

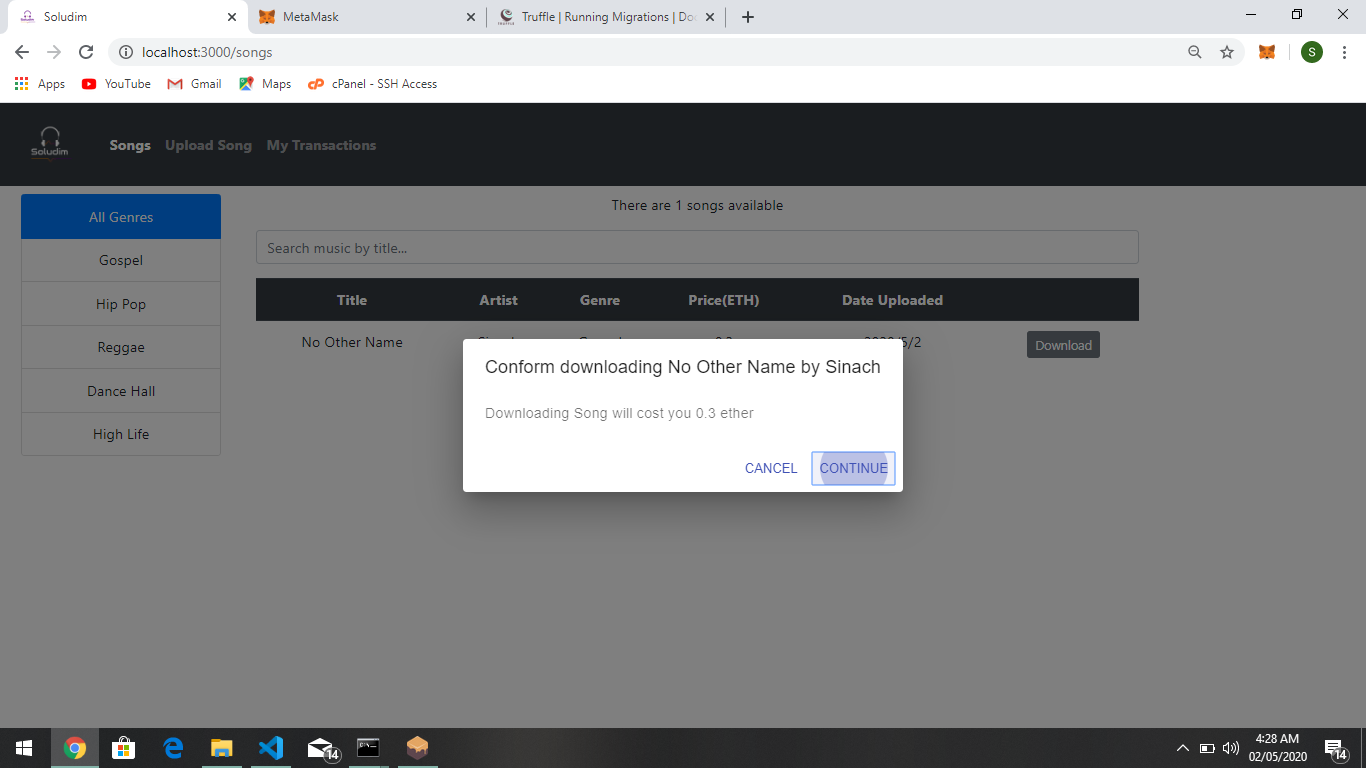


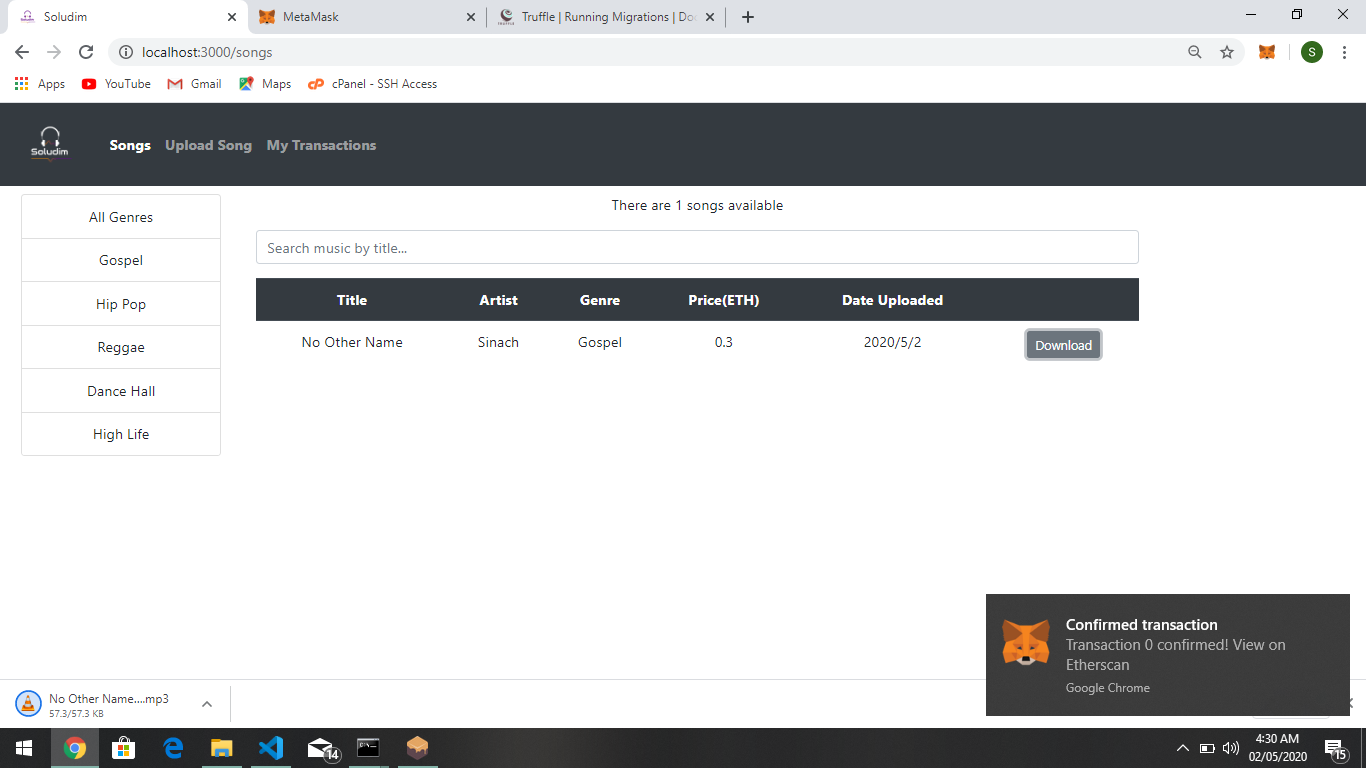


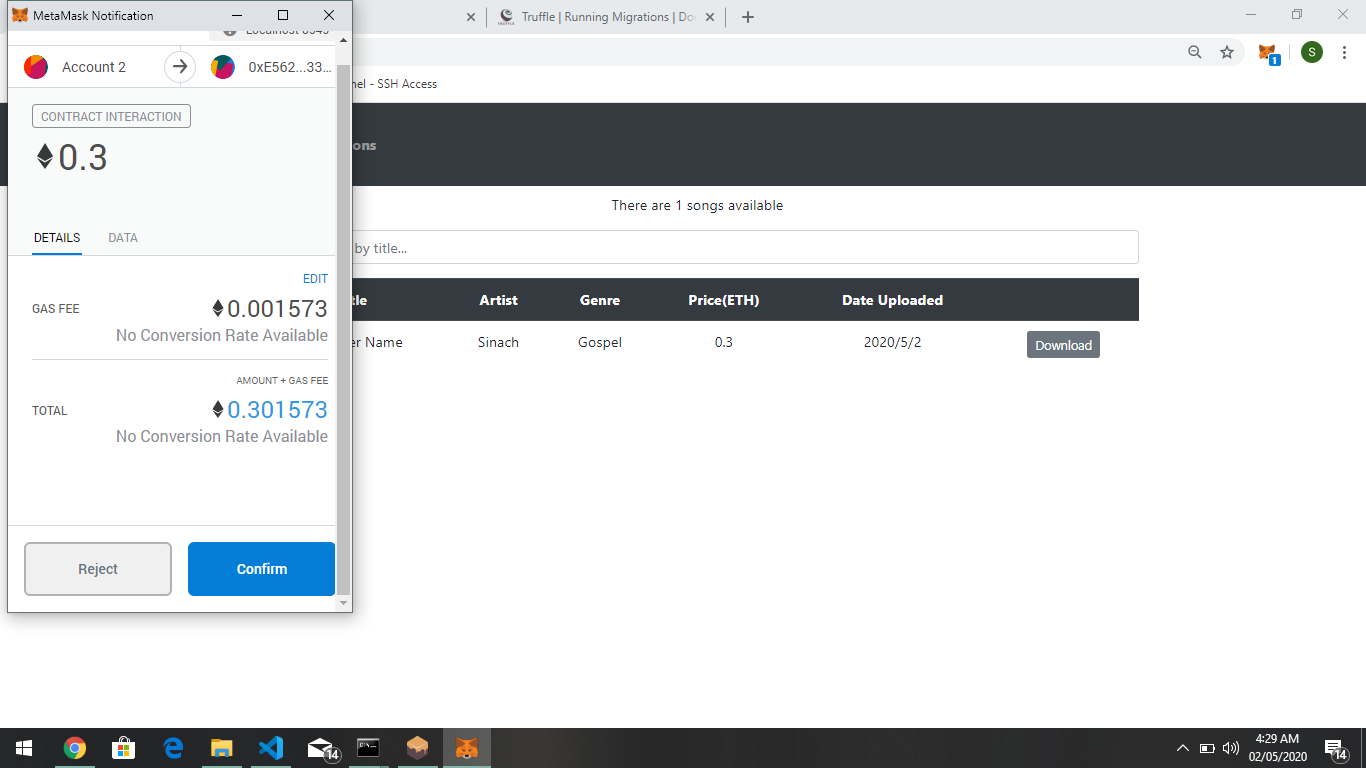


**DOWNLOADING A SONG**

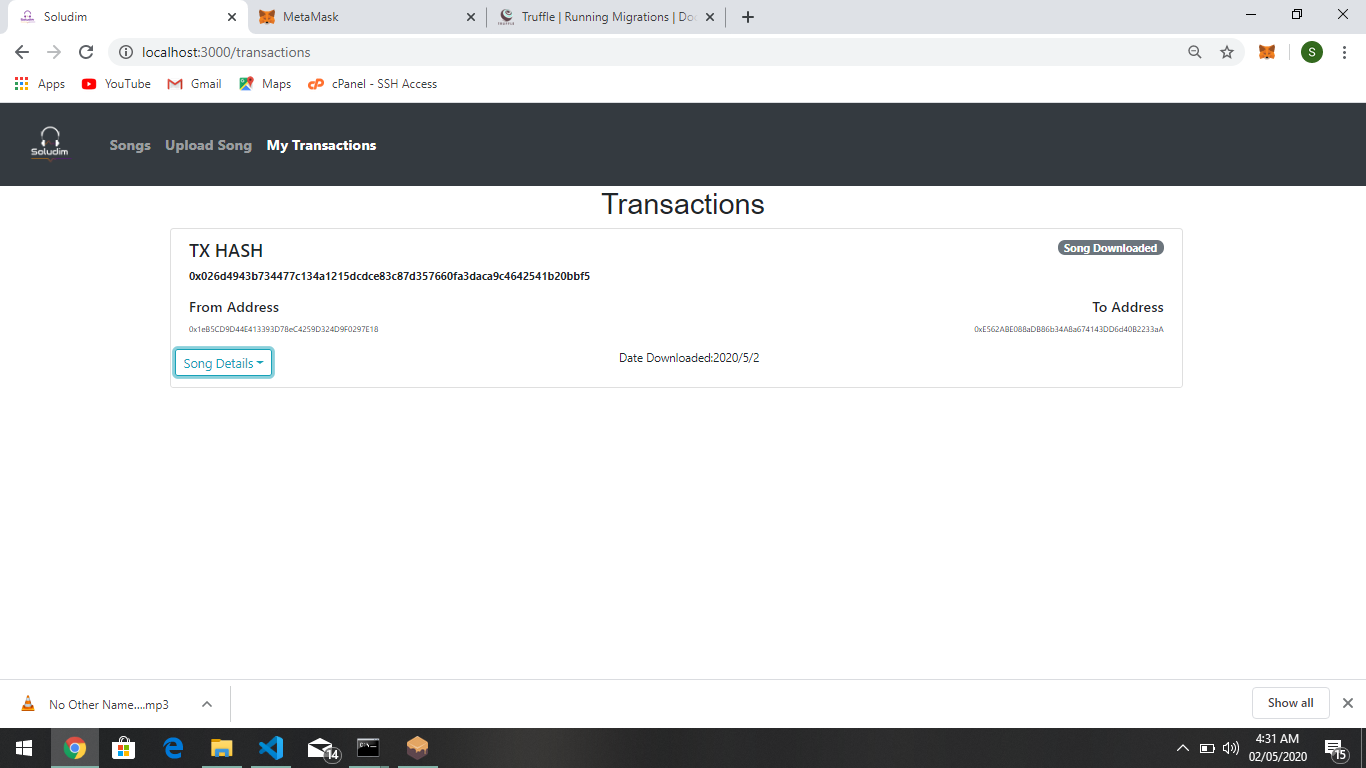


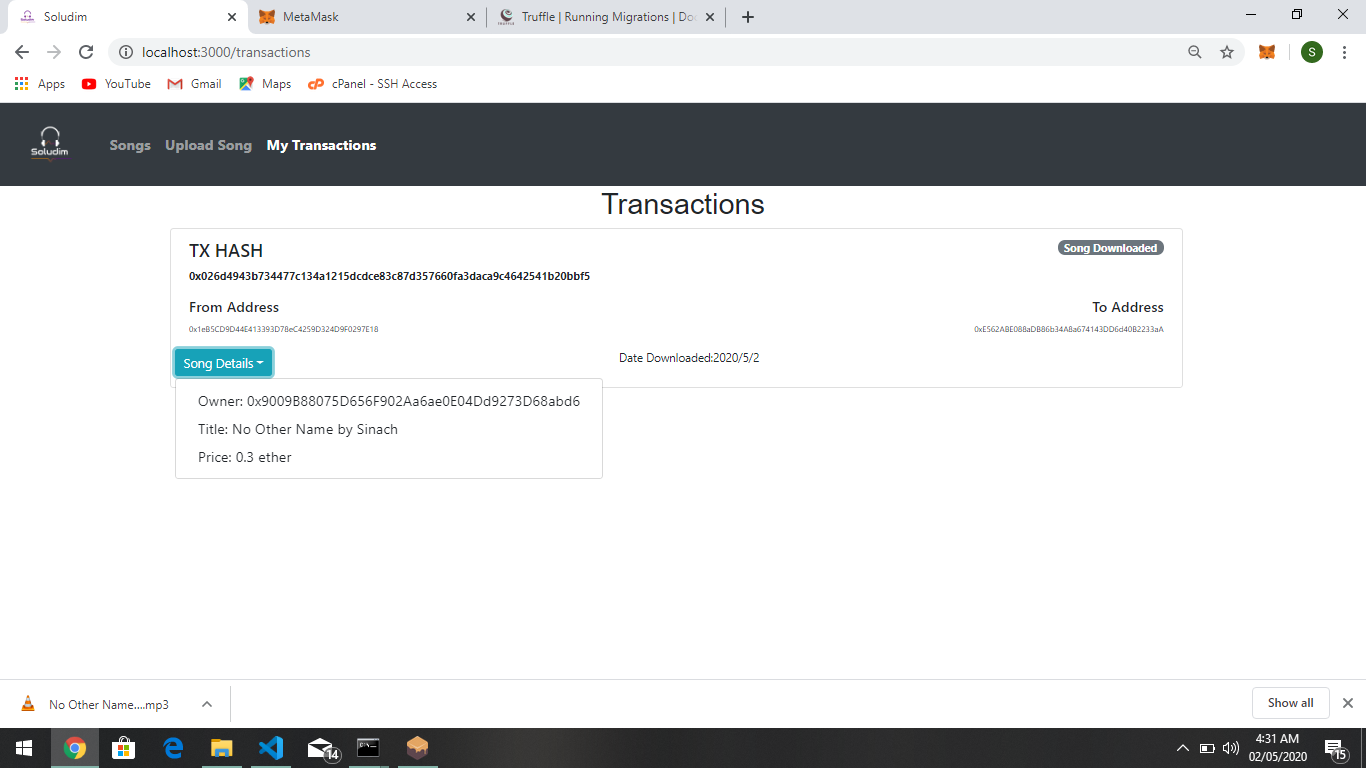


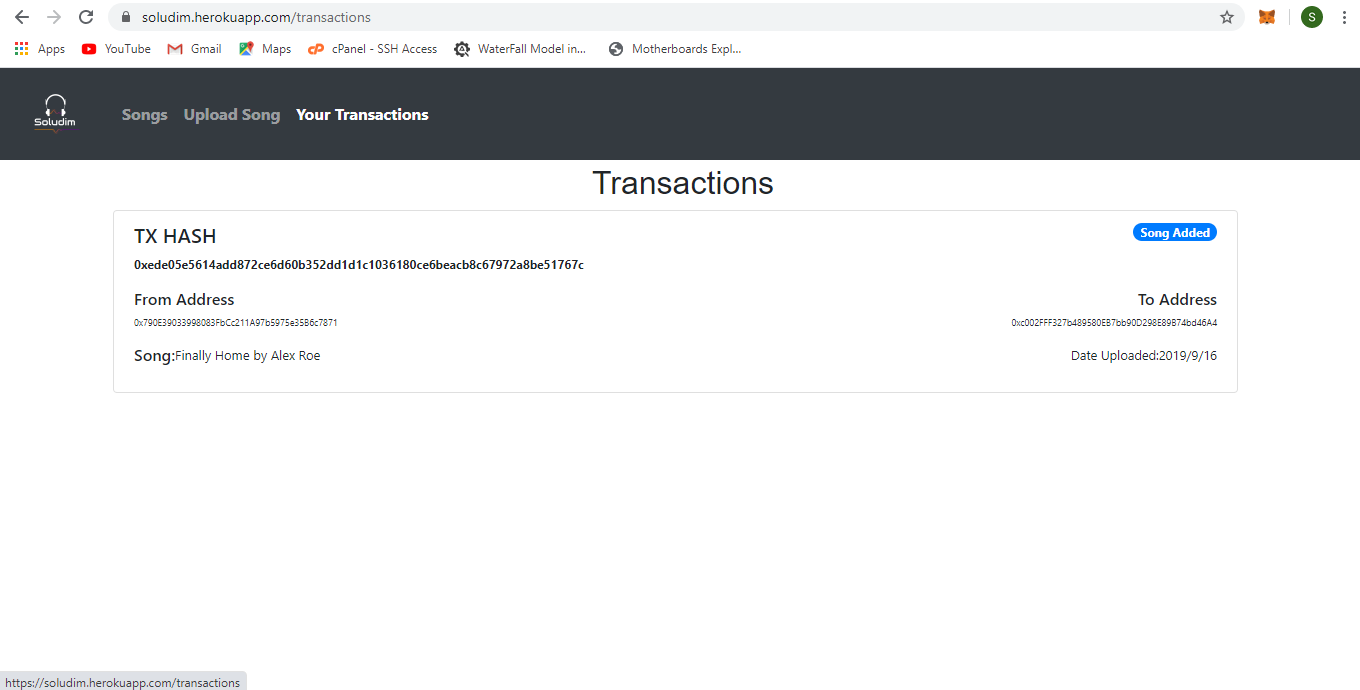




**TRANSACTIONS PAGE**







### 5.3 INPUT DESIGN

Any application that interacts with it environment takes input in order to produce required output. A wrong input implies wrong output. Hence, one has to be circumspect in data input when it comes to systems. Below are some of the input designs and events of the APP.

SONG BLOCK

|  |  |
| --- | --- |
| Fields | Datatype |
| owner | address |
| title | string |
| price | string |
| date | string |
| songHash | string |
| songArtist | string |
| genreId | uint |

GENRE TABLE

|  |  |
| --- | --- |
| Fields | Datatype |
| \_id | uint |
| name | string |

DOWNLOAD SONG EVENT

|  |  |
| --- | --- |
| Fields | Datatype |
| owner | address |
| buyer | address |
| song | Song |
| dateDownloaded | string |

SONG ADDED EVENT

|  |  |
| --- | --- |
| Fields | Datatype |
| songowner | address |
| song | Song |

## CHAPTER SIX

SYSTEM IMPLEMENTATION AND TESTING

### 6.1 IMPLEMENTATION DETAILS

The present chapter discusses the implementation of the system, highlighting the testing exercise and describing some of the main components of the APP’s Graphical User Interface. It will give an output from programming language and other tools used to develop our system. The project implements Blockchain technology.

### 6.2 IMPLEMENTATION OF THE SOFTWARE

**Login:**

Metamask will automatically pop up waiting for confirmation to sync with app when users try to access the site. Users just need to click on yes. This confirmation is one time.

**Create an Account**

This app needs an ethereum wallet to work. This site will automatically redirect users to metamask download page if no wallet is found. Users then have to download and create a personal wallet.

**Uploading song**: Uploading song will take song details like price of the song (in ether), song title, song genre, song writer and the song itself. Uploaded can leave the song price to be zero if he wants users to download it for free.

**Download song**: This option enables users to purchase a song and save it to their local storage.

**Error Handling:**

Alert boxes and notifications are built into this app which will be shown to the screen if an error occurs. If a user is trying to download a song and his money is not enough, an error message will be shown to the screen in a form of alert box to users.

#### 6.2.1 Home page

Introduction:

This is the main page of the entire system where attendance is taken. It has features like:

* HOME: This is the front page of the app. It has some details of the app including sponsors and also the purpose of this system.
* SONGS: The songs page is where all the songs can be found. It has a download button beside each song which users can just perform a simple click to make transaction and then get the song offline. This page has a search text field to sort songs by title. It also has a genre type to get all songs that belongs to a particular genre.
* MY TRANSACTIONS: This page is for record keeping. List all the transactions users have made so far in the app at any point in time.
* UPLOAD SONG: This page is where addition of songs is made. It has required field which need to be filled before the song can be uploaded.

### 6.3 TESTING

Testing was done in several steps of the software development process.

**Phase one (1): UNIT TESTING**

Within this phase the logic part of the system was tested. Dummy data was used to check how the system will work if sent into production. Breaking points were used to trace program logics and see where things are going wrong. Corrections were made where necessary and tested again to check validity.

Several other tests were performed on each page to check their responsiveness to pages they are linked to.

**Phase two (2): PROCESS TESTING**

Function testing was implemented at this stage to check as to whether or not the system core functionalities meet the specification of the system. By following the black box testing I got an idea as to what function to be included to meet the systems requirement and also verifying with the white board testing to confirm as to whether a function is working as expected.

**Phase 3: ULTIMATE TESTING**

Modifications were made where necessary to reduce bugs and errors to ascertain the ultimate state of the system. Fullness and non-redundancy of code was also taken into consideration in this phase.

## CHAPTER 7

CONCLUSION AND RECOMMENDATION

### 7.1 PERFORMANCE EVALUATION

The app is tested by 5 block chain developers and the response was good.

However, the test will be carried out on many people for validity.

The system will then be evaluated further based on the feedbacks that will be supplied

from these users.

### 7.2 PROGRAM MAINTAINACE

This app would be maintained during upgrades when there is a new field to be added or a new form to be added in other to serve users well. Since it is not compiled as a standalone it can be maintained at any time by changing or removing some codes or adding new ones.

### 7.3 RECOMMENDATION

I recommend users to use chrome as their choice of web browser since it has metamask module as part of its packages. A good Internet connection is needed in order to use this application, is not a standalone application.

### 7.4 CONCLUSION

The research on how to hold music firmly and also to increase funds on behalf of musicians and also to promote our music industry as well as to encourage upcoming musicians has led to the design of a computerized System to ascertain these challenges. But there is always a room for improvement in any software since ideas and research keep popping up. The important thing is that the system should be flexible enough for future modifications. Every effort has been made to cover all user requirement and make it user friendly.

### 7.5 REFERENCES

* <https://www.infura.io>
* https://en.wikipedia.org
* https://www.smartdraw.com/entity-relationship-diagram/
* <https://www.ethereum.org>