A Project/Dissertation Review-ETE Report

on

FRAUD CREDIT CARD DETECTION

Submitted in partial fulfillment of the
requirement for the award ofthe degree of

BTECH IN COMPUTER SCIENCE AND ENGINEERING

Under The Supervision of
Name of Supervisor: Dr SPS Chauhan
Designation: Program Chair

Submitted By
Karishma Dhameja 18SCSE1010742 / 18021190006
Rishi Kumar 18SCSE1010224 / 18021011464

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
October, 2021

Table of Contents

Title
Abstract **1**
Chapter 1- Introduction **1**
1.1 Introduction **2**
1.2 Formulation of Problem **3**
1.2.1 Tool and Technology Used

Chapter 2- Literature Survey / Project Design **5**

ABSTRACT

It is important for credit card companies to be able to detect fraudulent credit card transactions so that customers are not charged for items they did not purchase. Such problems can be addressed with Scientific Data and its value, as well as machine learning, cannot be overstated. This project aims to demonstrate data processing using credit card learning with Credit Card Fraud Detection. The Credit Card Determination Issue involves modeling a previous credit card transaction with details of those that have been identified as fraud. This model is used to determine whether a new transaction is fraudulent or not. Credit Card Fraud Detection is a common sample of segmentation. The process is focusing on analyzing and analyzing data sets and deploying multiple detection algorithms.

INTRODUCTION

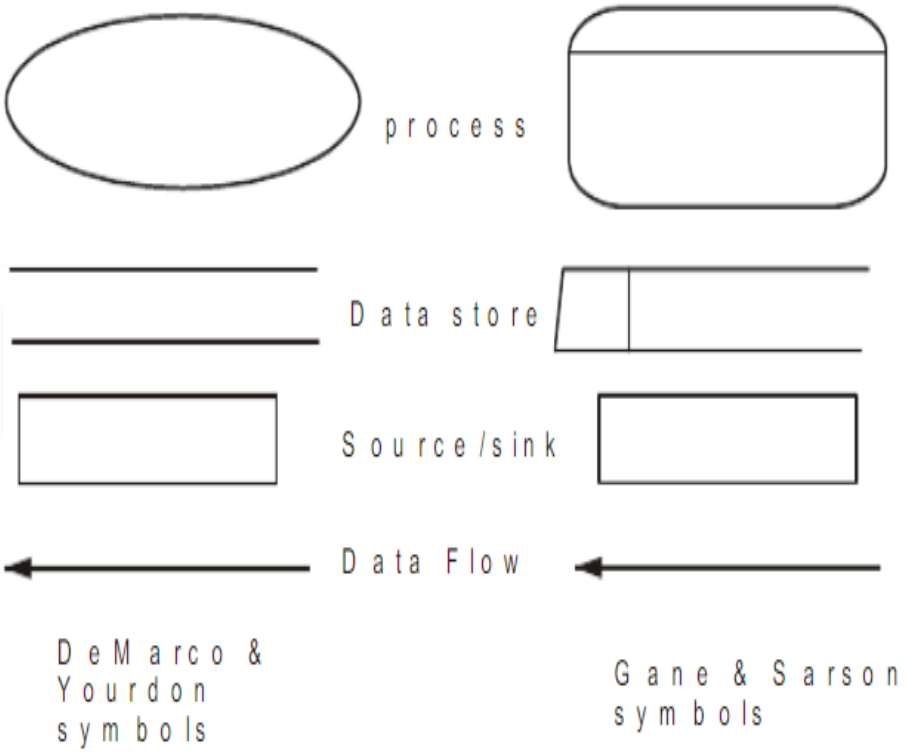
In recent years, the prevailing data mining concerns people with credit card Fraud detection model based on data mining. Since our problem is approached as a classification problem, classical data mining algorithms are not directly applicable. So an alternative approach is made by using general purpose meta heuristic approaches like genetic algorithms. This project is to propose a credit card fraud detection system using genetic algorithm. Genetic algorithms are evolutionary algorithms which aim at obtaining better solutions as time progresses. When a card is copied or stolen or lost and captured by fraudsters it is usually used until its available limit is depleted. Thus, rather than the number of correctly classified transactions, a solution which minimizes the total available limit on cards subject to fraud is more prominent. It aims in minimizing the false alerts using genetic algorithm where a set of interval valued parameters are optimized.

Data Flow Diagram

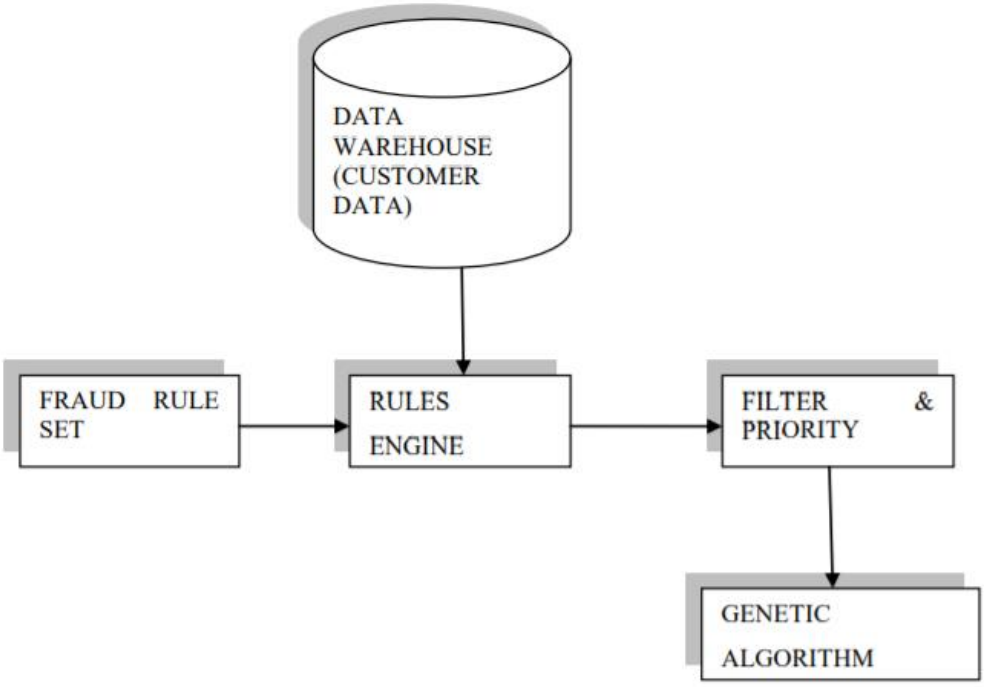
LITERATURE SURVEY

Fraud detection has been usually seen as a data mining problem where the objective is to correctly classify the transactions as legitimate or fraudulent. For classification problems many performance measures are defined most of which are related with correct number of cases classified correctly. A more appropriate measure is needed due to the inherent structure of credit card transactions. When a card is copied or stolen or lost and captured by fraudsters it is usually used until its available limit is depleted. Thus, rather than the number of correctly classified transactions, a solution which minimizes the total available limit on cards subject to fraud is more prominent. 4 Since the fraud detection problem has mostly been defined as a classification problem, in addition to some statistical approaches many data mining algorithms have been proposed to solve it. Among these, decision trees and artificial neural networks are the most popular ones. The study of Bolton and Hand provides a good summary of literature on fraud detection problems. However, when the problem is approached as a classification problem with variable misclassification costs as discussed above, the classical data mining algorithms are not directly applicable; either some modifications should be made on them or new algorithms developed specifically for this purpose are needed. An alternative approach could be trying to make use of general purpose meta heuristic approaches like genetic algorithms.

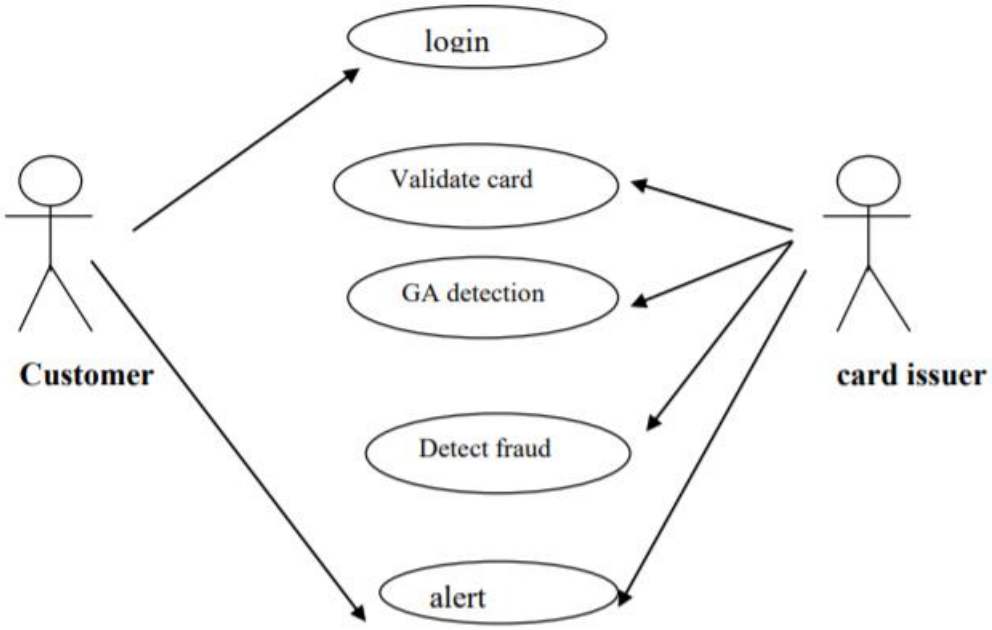
Data Flow Diagram



ARCHITECTURE SURVEY

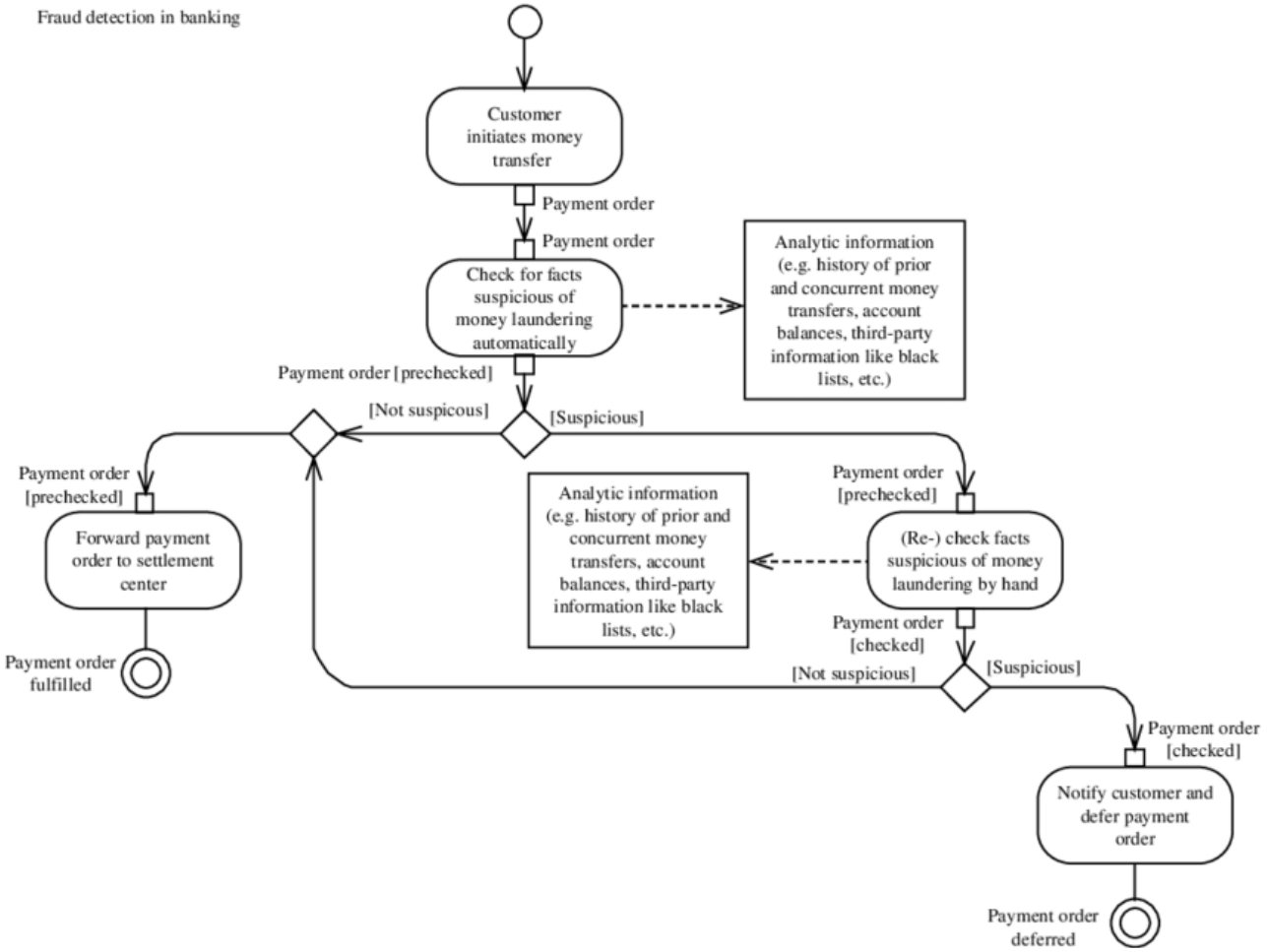


USE CASE DIAGRAM

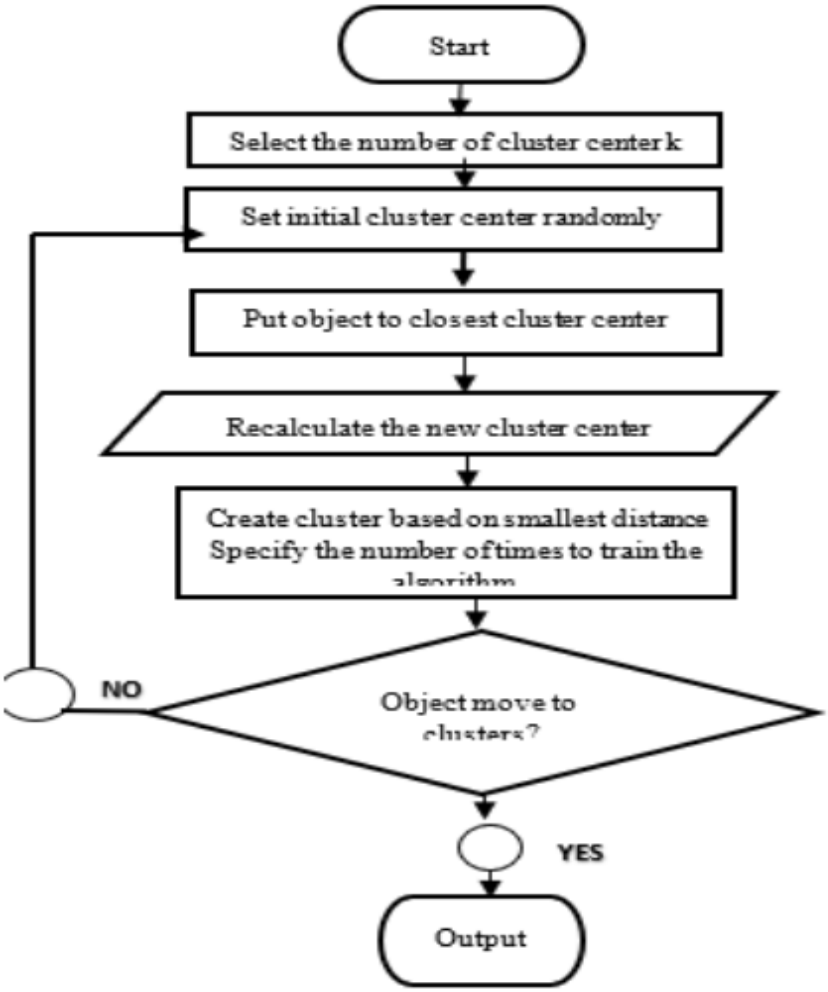


`

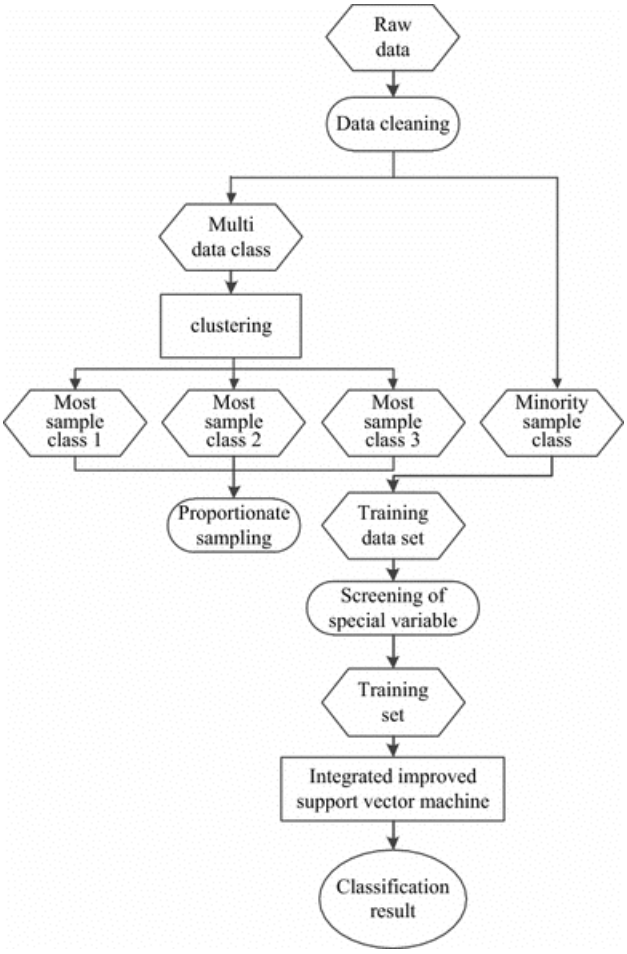
UML DIAGRAM


Fraud detection in banking
Customer initiates money transfer
Payment order
Payment order
Check for facts suspicious of money laundering automatically
Analytic information (e.g. history of prior and concurrent money transfers, account balances, third-party information like black lists, etc.)
Payment order [prechecked]
[Not suspicous]
[Suspicious]
Payment order [prechecked]
Forward payment order to settlement center
Payment order fulfilled
Analytic information (e.g. history of prior and concurrent money transfers, account balances, third-party information like black lists, etc.)
Payment order [prechecked]
(Re-) check facts suspicious of money laundering by hand
Payment order [checked]
[Not suspicious]
[Suspicious]
Payment order [checked]
Notify customer and defer payment order
Payment order deferred

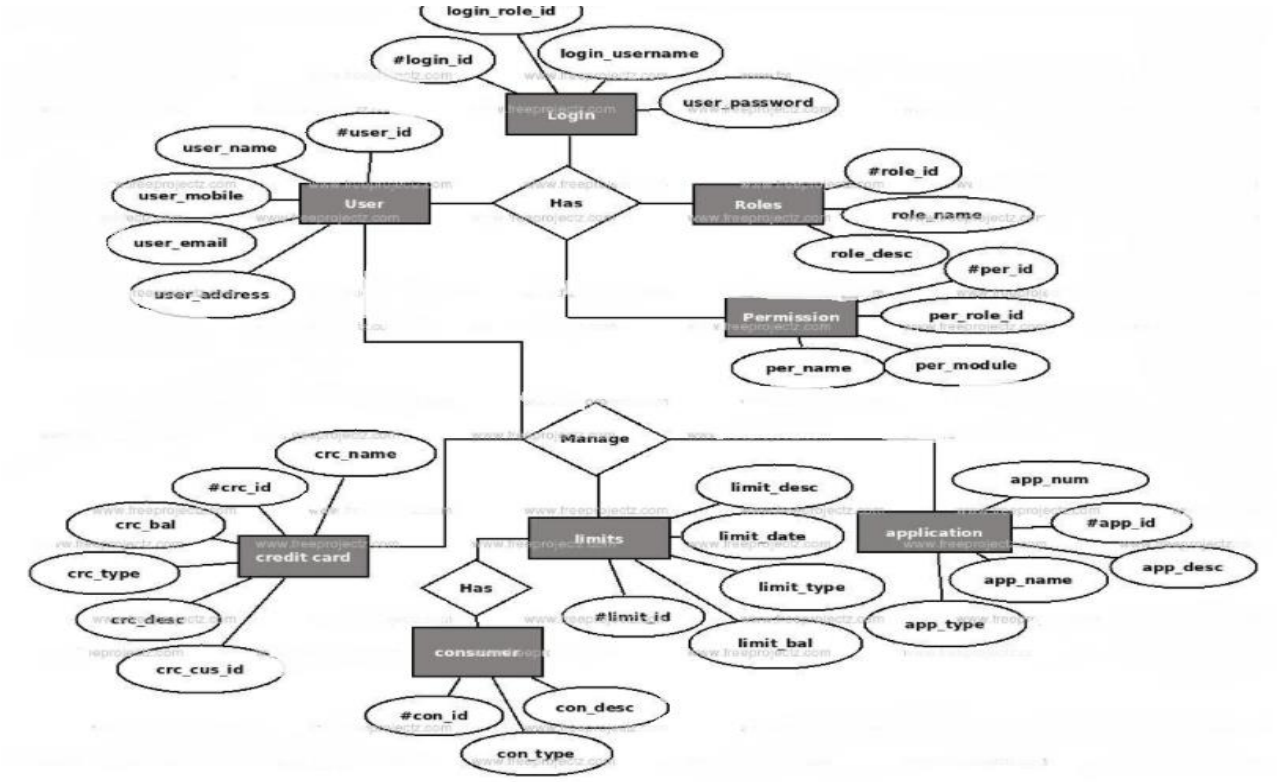
Flowchart Diagram



Schemetic Diagram



ER Diagram


login_role_id
#login_id
login_username
user_password
Login
user_name
#user_id
user_mobile
User
Has
Roles
#role_id
role_name
role_desc
user_email
user_address
#per_id
Permission
per_role_id
per_name
per_module
Manage
crc_name
#crc_id
crc_bal
credit card
crc_type
crc_desc
crc_cus_id
Has
consumer
#con_id
con_desc
con_type
limits
limit_desc
limit_date
limit_type
#limit_id
limit_bal
app_num
application
#app_id
app_desc
app_name
app_type

Module Description

System Requirements

The Software Requirements is produced at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by establishing a complete information description,a detailed functional and behavioral description,an indication of performance requirements and design constraints,appropriate validation criteria and other data pertinent to requirements

The Proposed system has the following requirements

1. System needs store information about new entry of credit card
2. System needs to help the internal staff to keep information of Transactions and find them as per variour queries
3. System needs to maintain quantity record
4. System needs to keep record of Datasets

Software Requirements:

- Windows Xp, Windows 7(ultimate, enterprise)
- Sql 2008
- Visual studio 2010

Hardware Components:

- Processor – i3
- Min Hard Disk – 4 GB
- Min Memory – 1GB RAM

Functional Requirements

- Product and component based
- Creating & changing Issues at ease
- Accuracy in work
- Robust Database Back-end
- It contain better storage capacity
- Well designed reports
- Easy and Fast retrieval of information
- Simple Status & resolution
- Multi-Level priorities &severities
- Query issue list to any depth

Source Code

Importing all the necessary Libraries

```
# import the necessary packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import gridspec
```

Code : Loading the Data

```
# Load the dataset from the csv file using pandas
# best way is to mount the drive on colab and
# copy the path for the csv file
data = pd.read_csv("credit.csv")
```

Code : Understanding the Data

```
# Grab a peek at the data
data.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801	-0.991390	-0.311169
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235	0.489095	-0.143772
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084	0.717293	-0.165946
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228	0.507757	-0.287924
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196	1.345852	-1.119670

Code : Describing the Data

```
# Print the shape of the data

# data = data.sample(frac = 0.1, random_state = 48)

print(data.shape)

print(data.describe())
```

Output :

```
(284807, 31)
                Time            V1  ...         Amount         Class
count  284807.000000  2.848070e+05  ...  284807.000000  284807.000000
mean    94813.859575  3.919560e-15  ...      88.349619       0.001727
std     47488.145955  1.958696e+00  ...     250.120109       0.041527
min         0.000000 -5.640751e+01  ...       0.000000       0.000000
25%     54201.500000 -9.203734e-01  ...       5.600000       0.000000
50%     84692.000000  1.810880e-02  ...      22.000000       0.000000
75%    139320.500000  1.315642e+00  ...      77.165000       0.000000
max    172792.000000  2.454930e+00  ...   25691.160000       1.000000
```

Code : Imbalance in the data

```
Time to explain the data we are dealing with.
# Determine number of fraud cases in dataset
fraud = data[data['Class'] == 1]
valid = data[data['Class'] == 0]
outlierFraction = len(fraud)/float(len(valid))
print(outlierFraction)
print('Fraud Cases: {}'.format(len(data[data['Class'] == 1])))
print('Valid Transactions: {}'.format(len(data[data['Class'] == 0])))
```

```
0.0017304750013189597
Fraud Cases: 492
Valid Transactions: 284315
```

Only *0.17%* fraudulent transaction out all the transactions. The data is highly Unbalanced. Lets first apply our models without balancing it and if we don't get a good accuracy then we can find a way to balance this dataset. But first, let's implement the model without it and will balance the data only if needed.

Code : Print the amount details for Fraudulent Transaction

print("Amount details of the fraudulent transaction")

fraud.Amount.describe()

Output :

```
Amount details of the fraudulent transaction
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

Code : Print the amount details for Normal Transaction

```
print("details of valid transaction")

valid.Amount.describe()
```

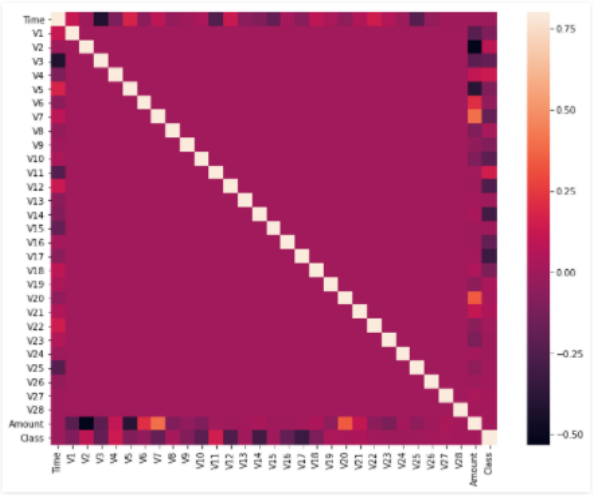
Output :

```
Amount details of valid transaction
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

As we can clearly notice from this, the average Money transaction for the fraudulent ones is more. This makes this problem crucial to deal with.

Code : Plotting the Correlation Matrix
The correlation matrix graphically gives us an idea of how features correlate with each other and can help us predict what are the features that are most relevant for the prediction.

```
# Correlation matrix
corrmat = data.corr()
fig = plt.figure(figsize = (12, 9))
sns.heatmap(corrmat, vmax = .8, square = True)
plt.show()
```



In the HeatMap we can clearly see that most of the features do not correlate to other features but there are some features that either has a positive or a negative correlation with each other. For example, *V2* and *V5* are highly negatively correlated with the feature called *Amount*. We also see some correlation with *V20* and *Amount*. This gives us a deeper understanding of the Data available to us.

Code : Separating the X and the Y values

Dividing the data into inputs parameters and outputs value format

```
# dividing the X and the Y from the dataset

X = data.drop(['Class'], axis = 1)

Y = data["Class"]

print(X.shape)

print(Y.shape)

# getting just the values for the sake of processing

# (its a numpy array with no columns)



 xData = X.values

 yData = Y.values
```

Output :

```
(284807, 30)
(284807, )
```

Training and Testing Data Bifurcation

We will be dividing the dataset into two main groups. One for training the model and the other for Testing our trained model's performance.

```
 # Using Skicit-learn to split data into training and testing sets

 from sklearn.model_selection import train_test_split
```

```
# Split the data into training and testing sets
xTrain, xTest, yTrain, yTest = train_test_split(
        xData, yData, test_size = 0.2, random_state = 42)
```

Code : Building a Random Forest Model using skicit learn

```
# Building the Random Forest Classifier (RANDOM FOREST)
from sklearn.ensemble import RandomForestClassifier
# random forest model creation
rfc = RandomForestClassifier()
rfc.fit(xTrain, yTrain)
# predictions
yPred = rfc.predict(xTest)
```

Code : Building all kinds of evaluating parameters

```
# Evaluating the classifier
# printing every score of the classifier
# scoring in anything
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score, matthews_corrcoef
from sklearn.metrics import confusion_matrix
```

```
n_outliers = len(fraud)
n_errors = (yPred != yTest).sum()
print("The model used is Random Forest classifier")

acc = accuracy_score(yTest, yPred)
print("The accuracy is {}".format(acc))

prec = precision_score(yTest, yPred)
print("The precision is {}".format(prec))

rec = recall_score(yTest, yPred)
print("The recall is {}".format(rec))

f1 = f1_score(yTest, yPred)
print("The F1-Score is {}".format(f1))

MCC = matthews_corrcoef(yTest, yPred)
print("The Matthews correlation coefficient is{}".format(MCC))
```

Output :

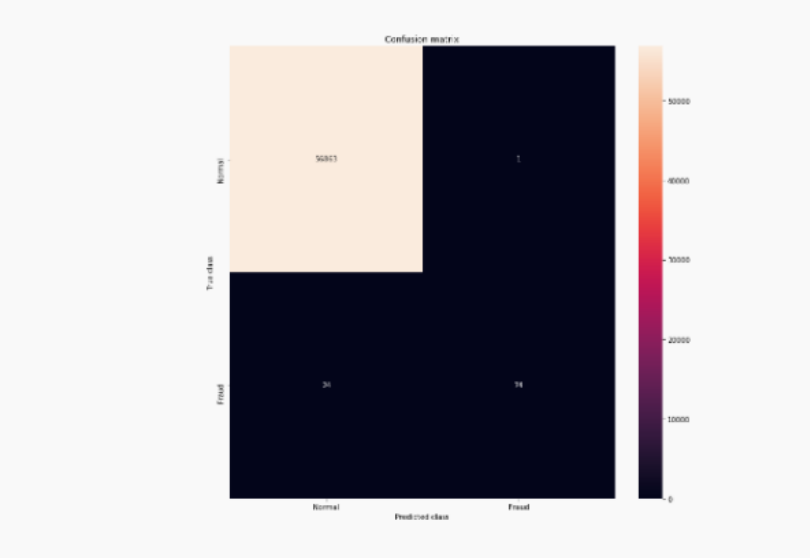
```
The model used is Random Forest classifier

The accuracy is  0.9995611109160493

The precision is 0.9866666666666667

The recall is 0.7551020408163265

The F1-Score is 0.8554913294797689

The Matthews correlation coefficient is0.8629589216367891
```

Code : Visulalizing the Confusion Matrix

```
# printing the confusion matrix

LABELS = ['Normal', 'Fraud']

conf_matrix = confusion_matrix(yTest, yPred)

plt.figure(figsize =(12, 12))

sns.heatmap(conf_matrix, xticklabels = LABELS,
```

```
            yticklabels = LABELS, annot = True, fmt ="d");

plt.title("Confusion matrix")

plt.ylabel('True class')

plt.xlabel('Predicted class')

plt.show()
```

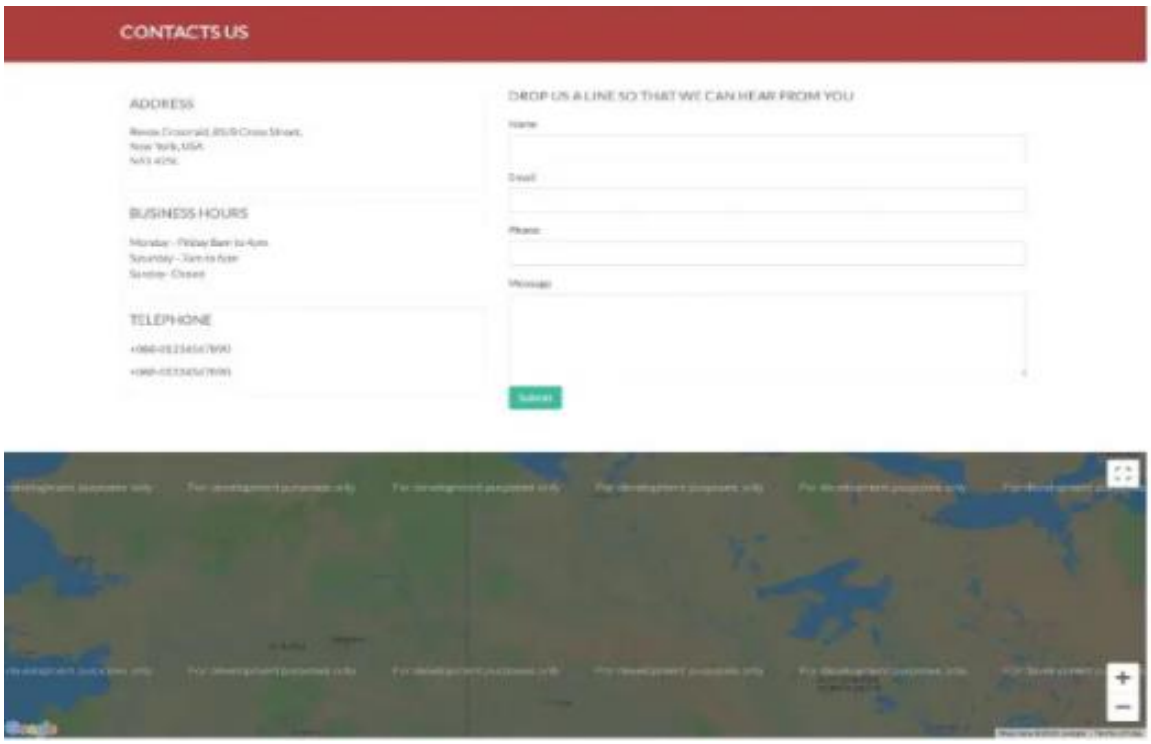
Output :

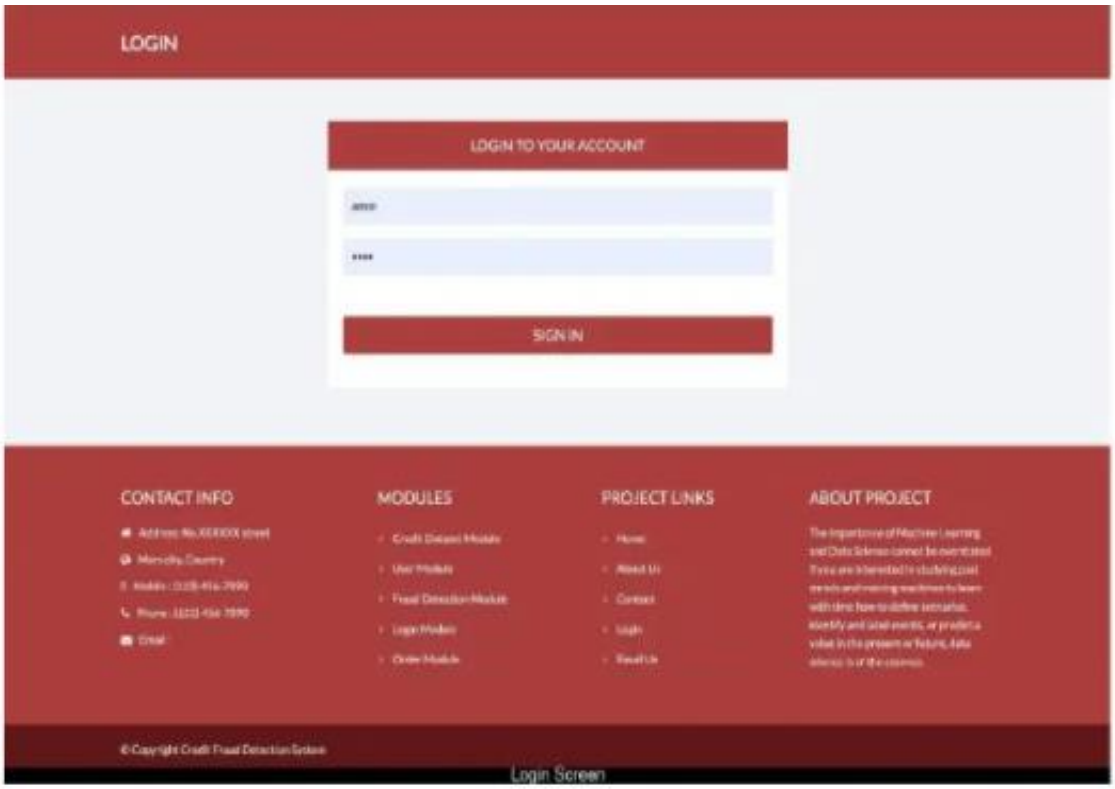


Results

Home Page Screen

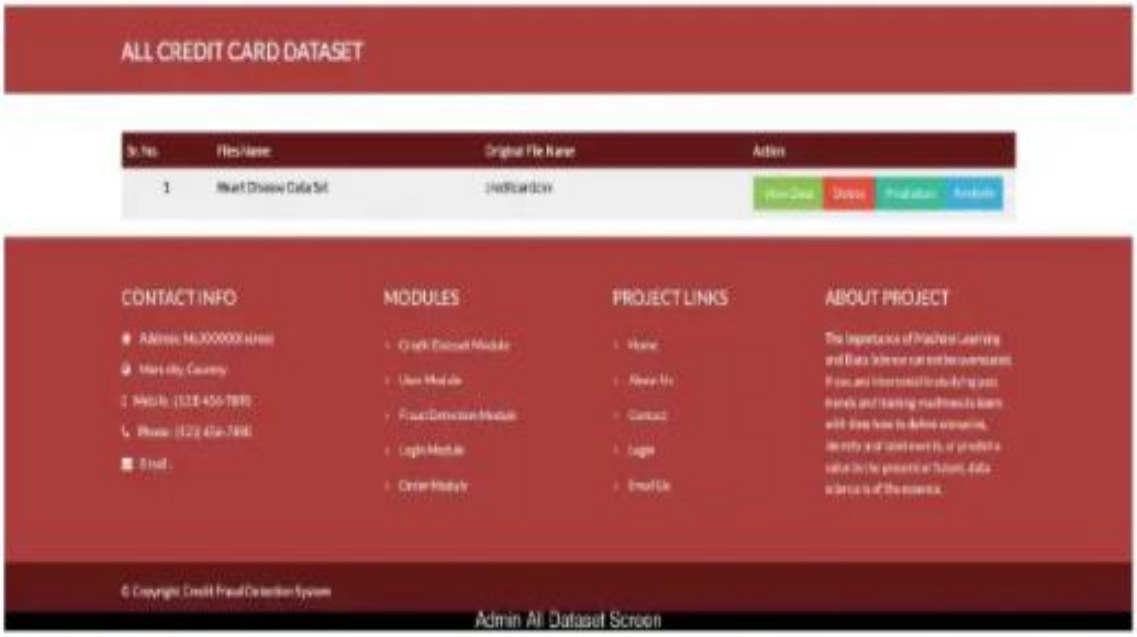
About Project Screen



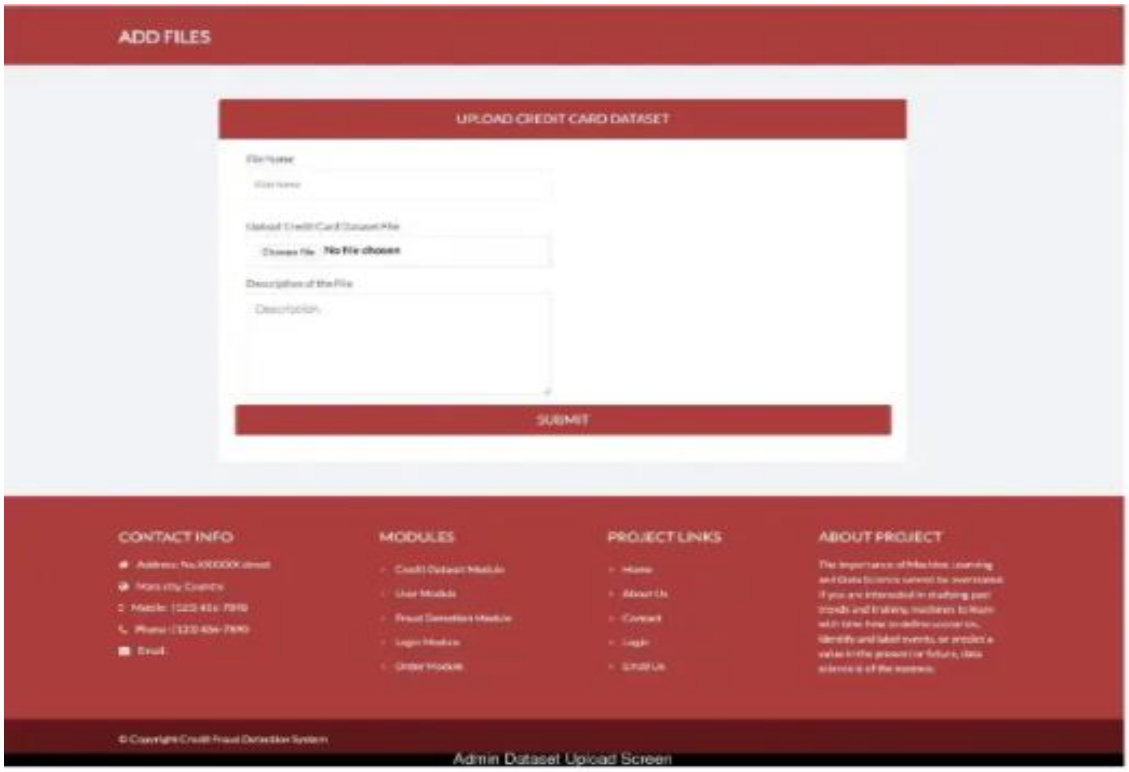


Login Screen

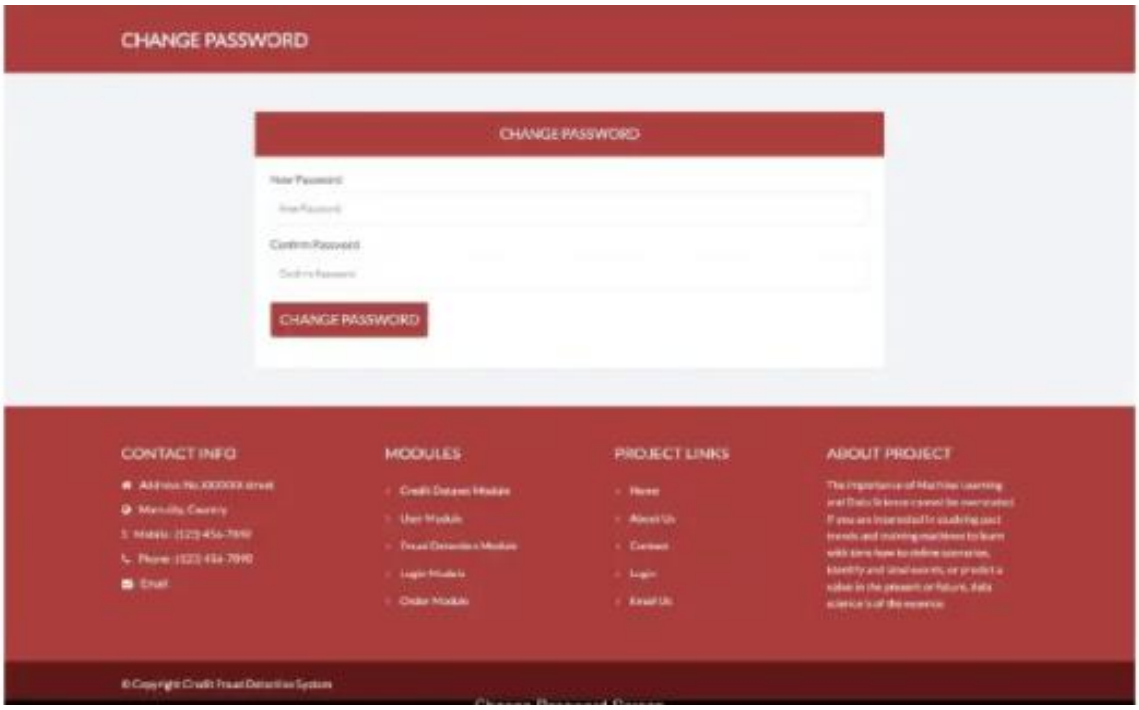
Admin Dashboard Screen



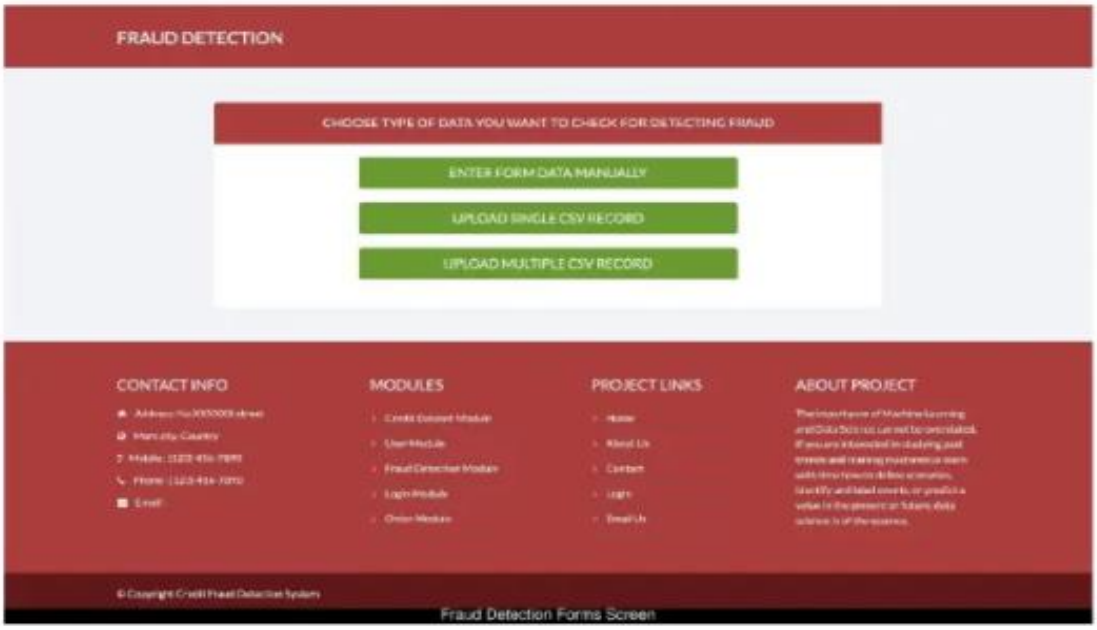
Admin All Dataset Screen



Admin Dataset Upload Screen

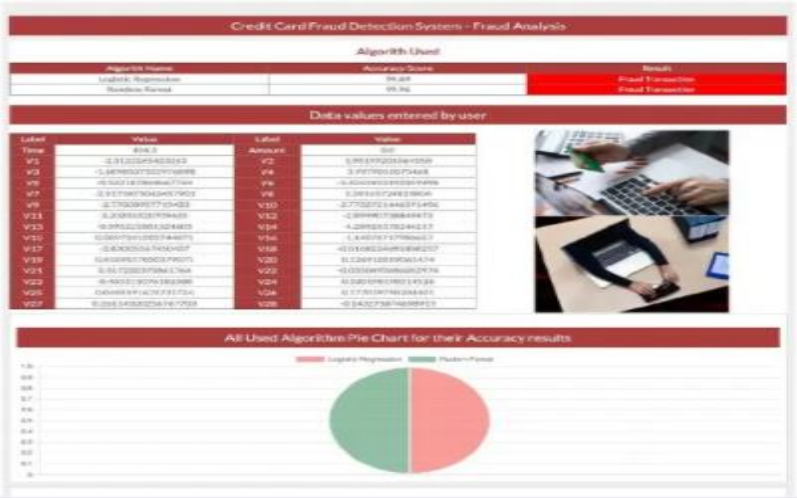


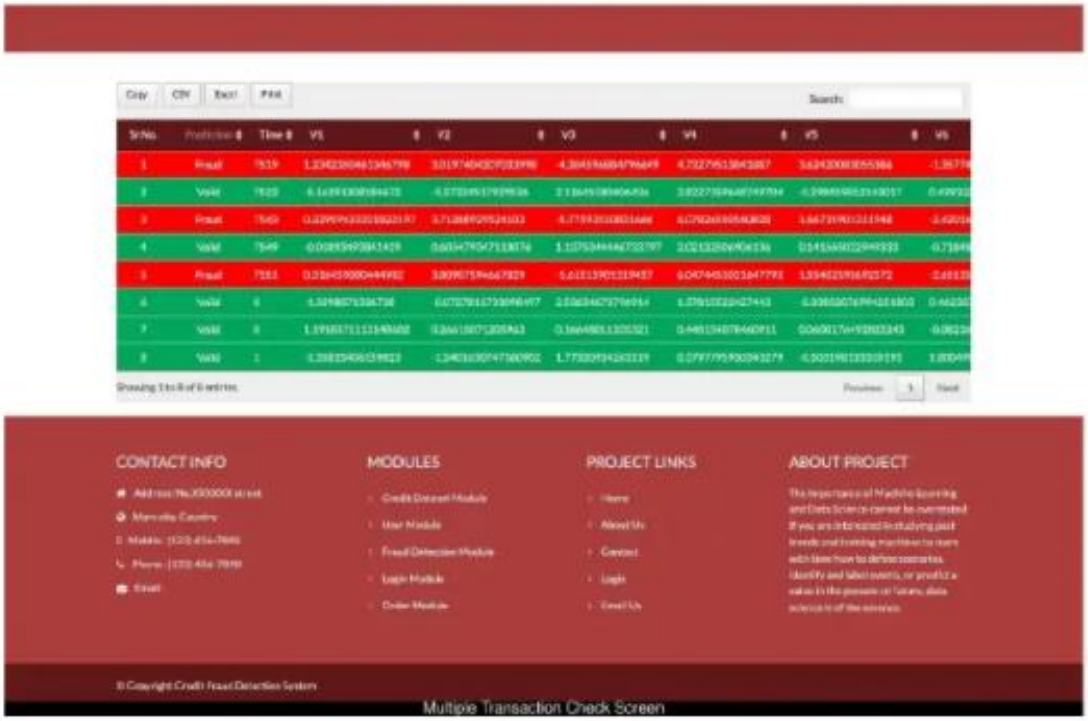
Change Password Page



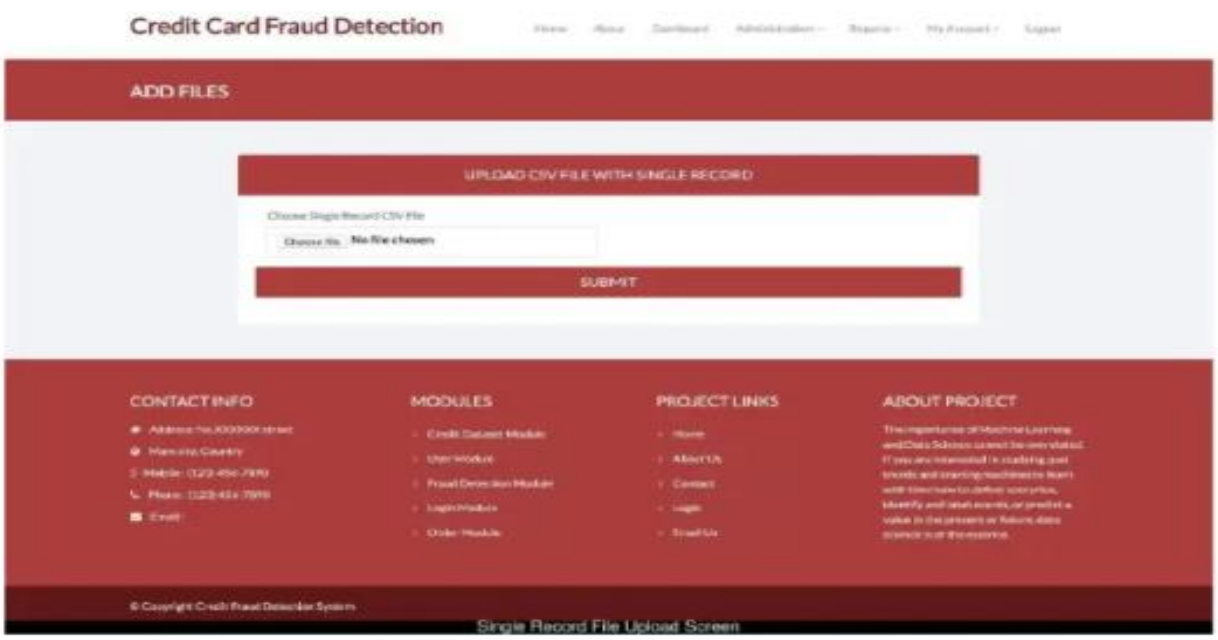
Fraud Detection Forms Screen

FRAUD DETECTION





Multiple Transaction Check Screen



Single Record File Upload Screen

Conclusion

Our project is only a humble venture to satisfy the needs to manage their project work. Several user friendly coding have also adopted.This package sDatasets prove to be a powerful package in satisfying all the requirnments of the school.The objective of software planning is to provide a framework that enables the manager to make reasonable estimates made within a limited time frame at the beginning of the software project and should be updated regularly as the project progresses.

At the end it is concluded that we hae made effort on following points

- A description of the backdatasets and context of the project and its relation to work already done in the area.
- Made statement of the aims and objective of the project
- The description of Purpose,Scope, and applicability
- We define the problem on which we are working in the project
- We can conclude that as the technology is developing day by day there are also fraudsters developing.
- Hence it is everyone's responsible to update about the technology and use it in a correct way.
- We should know about the Do's and Don'ts about the credit card before we start to use it and act accordingly to avoid any serious issues.

REFERENCES

- **Google for problem solving**
- **Head First PYTHON 2nd Edition**
- **http://www.jdbc-tutorial,com/**
- **http://www.PYTHON.net/**
- **PYTHON and SOFTWARE Design concepts by Apress**
- **https://www.tutorialspoint.com/sql/index.html**
- **https://www.docs.oracle.com/en**
- **https://www.wampserver.com/en/**
- **Database Programming with JDBC and PYTHON by O'Reilly**
- **https://www.tutorialspoint.com/python**
- **https://www.tutorialspoint.com/html**