Adult Salary Prediction

Presented by:-

Iriventi Bharath Vasishta(CSC/20/36)
Mohtasim Rahman(CSC/20/39)
Nikhil Choudhary(CSC/20/42)

Our task is to analyze the dataset and predict whether the income of an adult will exceed 50k per year or not

We have been given 11 features.

Adult dataset link :

http://archive.ics.uci.edu/ml/datasets/Adult

In [111]:

```
df = pd.read_csv("adult_data.csv")
df.head()
```

Out[111]:

	age	workclass	education	occupation	race	sex	capital-gain	capital-loss	hours-per-week	native-country
0	39	State-gov	Bachelors	Adm-clerical	White	Male	2174	0	40	United-States
1	50	Self-emp-not-inc	Bachelors	Exec-managerial	White	Male	0	0	13	United-States
2	38	Private	HS-grad	Handlers-cleaners	White	Male	0	0	40	United-States
3	53	Private	11th	Handlers-cleaners	Black	Male	0	0	40	United-States
4	28	Private	Bachelors	Prof-specialty	Black	Female	0	0	40	Cuba

Exploratory data analysis and Feature Engineering

What is Explanatory data?

Exploratory Data Analysis (EDA) is the very first step before you can perform any changes to the dataset or develop a statistical model to answer business problems. In other words, the process of EDA contains summarizing, visualizing and getting deeply acquainted with the important traits of a data set.

What is Feature Engineering?

Feature Engineering is known as the process of transforming raw data (that has already been processed by Data Engineers) into features that better represent the underlying problem to predictive models, resulting in improved model accuracy on unseen data.

Specifically, the data scientist will begin building the models and testing to see if the features achieve the desired results. This is a repetitive process that includes running experiments with various features, as well as adding, removing, and changing features multiple times

Alternatively, feature engineering is the advanced step that is used to derive some extra important features from existing features, which are then used for better data modelling, whereas preprocessing involves cleaning the data and preparing the raw data in a way that allows a data scientist to begin creating those optimal features.

Exploratory data analysis and Feature Engineering

In [120]:

```
## Here we will check the missing values in our dataset
df.isnull().sum()
```

Out[120]:

```
age               0
 workclass        0
 education        0
 occupation       0
 race             0
 sex              0
 capital-gain     0
 capital-loss     0
 hours-per-week   0
 native-country   0
 salary           0
dtype: int64
```

```
numerical
```

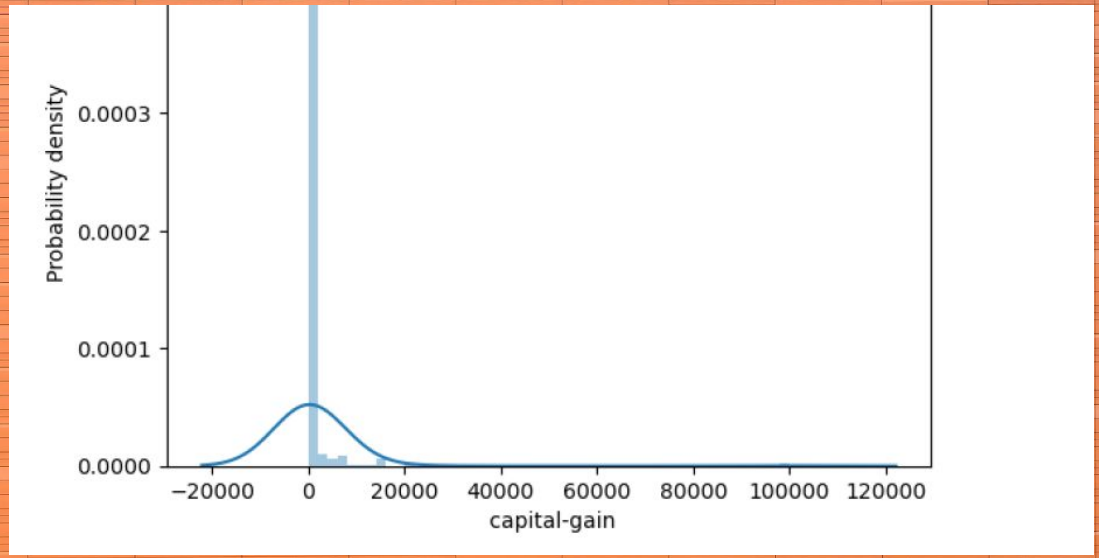
Out[122]:

```
['age', ' capital-gain', ' capital-loss', ' hours-per-week']
```

In [123]:

```
for i in numerical:
    print(f" {i}  :  {len(df[i].unique())}")
```

```
 age  :  73
  capital-gain  :  119
  capital-loss  :  92
  hours-per-week  :  94
```



Handling categorical data

Categorical data is the statistical data comprising categorical variables of data that are converted into categories. One of the examples is a grouped data.

```
In [127]:
categorical = [i for i in df.columns if df[i].dtypes == "O"]
```

```
for feature in categorical:
    print(f" {feature}  :  {len(df[feature].unique())}")
```

```
 workclass  :  9
 education  :  16
 occupation  :  15
 race  :  5
 sex  :  2
 native-country  :  42
 salary  :  2
```

```
for feature in  categorical:
    print(df[feature].value_counts())
    print("\n \n ")
```

```
 Trinadad&Tobago                 19
 Laos                            18
 Thailand                        18
 Yugoslavia                      16
 Outlying-US(Guam-USVI-etc)      14
 Honduras                        13
 Hungary                         13
 Scotland                        12
 Holand-Netherlands               1
Name:  native-country, dtype: int64



 <=50K    24720
 >50K      7841
Name:  salary, dtype: int64
```

Handling missing values in categorical features

When missing values is from categorical columns such as string or numerical. Then the missing values can be replaced with the most frequent category. If the number of missing values is very large then it can be replaced with a new category.

```
workclass : 6.43%
occupation : 5.66%
native-country : 1.79%
```

```
Out[131]:

' Prof-specialty'
```

	age	workclass	education	occupation	race	sex	capital-gain	capital-loss	hours-per-week	nat cou
0	3.663562	State-gov	Bachelors	Adm-clerical	White	Male	2174	0	40	Uni St
1	3.912023	Self-emp-not-inc	Bachelors	Exec-managerial	White	Male	0	0	13	Uni St
2	3.637586	Private	HS-grad	Handlers-cleaners	White	Male	0	0	40	Uni St
3	3.970292	Private	school	Handlers-cleaners	Black	Male	0	0	40	Uni St
4	3.332205	Private	Bachelors	Prof-specialty	Black	Female	0	0	40	C

Handling missing values in categorical features

```
workclass   :  8
education   :  6
occupation   :  14
race   :  5
sex   :  2
native-country   :  41
salary   :  2
```

```
array([' Bachelors', ' HS-grad', ' school', ' Masters', ' higher',
       ' Doctorate'], dtype=object)
```

Handling missing values in categorical features

In [140]:

```
df = df.apply(LabelEncoder().fit_transform)
df .head()
```

Out[140]:

	age	workclass	education	occupation	race	sex	capital-gain	capital-loss	hours-per-week	native-country	sala
0	22	6	0	0	4	1	25	0	39	38	
1	33	5	0	3	4	1	0	0	12	38	
2	21	3	2	5	4	1	0	0	39	38	
3	36	3	5	5	2	1	0	0	39	38	
4	11	3	0	9	2	0	0	0	39	4	

Pre Processing

Data preprocessing is an important step in the data mining process. It refers to the cleaning, transforming, and integrating of data in order to make it ready for analysis. The goal of data preprocessing is to improve the quality of the data and to make it more suitable for the specific data mining task.

Some common steps in data preprocessing include:

- Data cleaning: this step involves identifying and removing missing, inconsistent, or irrelevant data. This can include removing duplicate records, filling in missing values, and handling outliers.
- Data integration: this step involves combining data from multiple sources, such as databases, spreadsheets, and text files. The goal of integration is to create a single, consistent view of the data.
- Data transformation: this step involves converting the data into a format that is more suitable for the data mining task. This can include normalizing numerical data, creating dummy variables, and encoding categorical data.
- Data reduction: this step is used to select a subset of the data that is relevant to the data mining task. This can include feature selection (selecting a subset of the variables) or feature extraction (extracting new variables from the data).
- Data discretization: this step is used to convert continuous numerical data into categorical data, which can be used for decision tree and other categorical data mining techniques.

Pre Processing

Side Note:

Variance inflation factor (VIF)

Variance inflation factor measures how much the behavior (variance) of an independent variable is influenced, or inflated, by its interaction/correlation with the other independent variables.

Variance inflation factors allow a quick measure of how much a variable is contributing to the standard error in the regression.

```
Variance Inflation Factor (VIF):
```

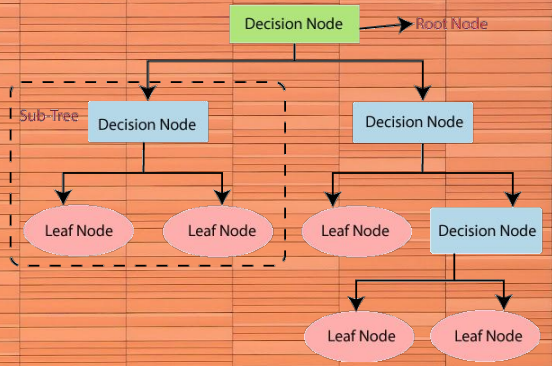
```
Out[145]:
```

	VIF Factor	features
0	1.031483	age
1	1.008832	workclass
2	1.024136	education
3	1.003528	occupation
4	1.024756	race
5	1.079126	sex
6	1.036486	capital-gain
7	1.015167	capital-loss
8	1.084125	hours-per-week
9	1.016311	native-country

MODEL EVALUATION

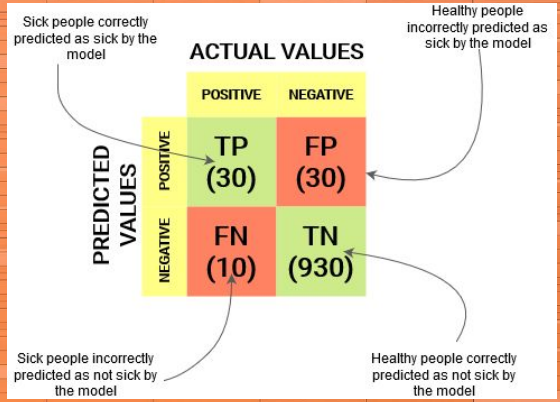
Decision Tree Classifier

A decision tree is a class discriminator that recursively partitions the training set until each partition consists entirely or dominantly of examples from one class. Each non-leaf node of the tree contains a split point that is a test on one or more attributes and determines how the data is partitioned.



Confusion Matrix

The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known. The matrix itself can be easily understood, but the related terminologies may be confusing.



Decision Tree Classifier

Precision Score

Precision is defined as the ratio of correctly classified positive samples (True Positive) to a total number of classified positive samples (either correctly or incorrectly).

1. Precision = True Positive/True Positive + False Positive
2. Precision = TP/TP+FP

   TP- True Positive

   FP- False Positive

$$Precision = \frac{True\ Positive(TP)}{True\ Positive(TP) + False\ Positive(FP)}$$

Recall Score

The recall is calculated as the ratio between the numbers of Positive samples correctly classified as Positive to the total number of Positive samples. The recall measures the model's ability to detect positive samples. The higher the recall, the more positive samples detected.

1. Recall = True Positive/True Positive + False Negative
2. Recall = TP/TP+FN

   TP- True Positive

   FP- False Positive

$$Recall = \frac{True\ Positive(TP)}{True\ Positive(TP) + False\ Negative(FN)}$$

Decision Tree Classifier

F1 Score

F1 score is a machine learning evaluation metric that measures a model's accuracy. It combines the precision and recall scores of a model. The accuracy metric computes how many times a model made a correct prediction across the entire dataset.

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Accuracy Metric

Accuracy is the ratio of the total number of correct predictions and the total number of predictions.

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{(\text{True Positive} + \text{False Positive} + \text{True Negative} + \text{False Negative})}$$

Decision Tree Classifier

Decision Tree Classifier

```
X_train.shape: (26048, 10)
y_train.shape: (26048,)
X_test.shape: (6513, 10)
y_test.shape: (6513,)
```

```
Train Accuracy: 95.98817567567568
Accuracy Score: 79.57930293259633
```

Confusion Matrix

```
[[4330  621]
 [ 709  853]]
```

	precision	recall	f1-score	support
0	0.8593	0.8746	0.8669	4951
1	0.5787	0.5461	0.5619	1562
accuracy			0.7958	6513
macro avg	0.7190	0.7103	0.7144	6513
weighted avg	0.7920	0.7958	0.7937	6513

Precision Score

```
Precision score of macro is:  71.9
Precision score of micro is:  79.58
Precision score of weighted is:  79.2
```

Recall Score

```
recall_score score of macro is:  71.03
recall_score score of micro is:  79.58
recall_score score of weighted is:  79.58
```

F1 Score

```
f1_score score of macro is:  71.44
f1_score score of micro is:  79.58
f1_score score of weighted is:  79.37
```

2) Random Forest

Random Forest is one of the most popular and commonly used algorithms by Data Scientists. Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables, as in the case of regression, and categorical variables, as in the case of classification. It performs better for classification and regression tasks. In this tutorial, we will understand the working of random forest and implement random forest on a classification task.

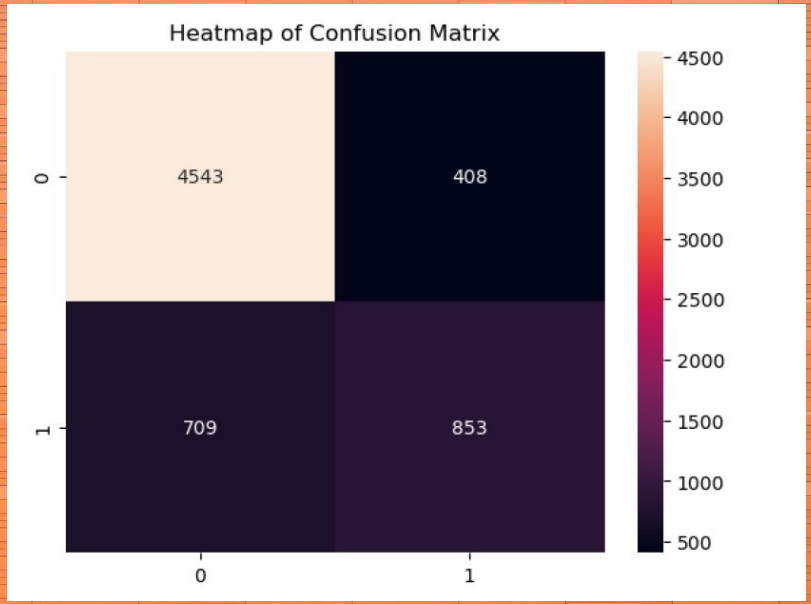
Random Forest

Random Forest

82.84968524489483

	precision	recall	f1-score	support
0	0.87	0.92	0.89	4951
1	0.68	0.55	0.60	1562
accuracy			0.83	6513
macro avg	0.77	0.73	0.75	6513
weighted avg	0.82	0.83	0.82	6513

Confusion Matrix

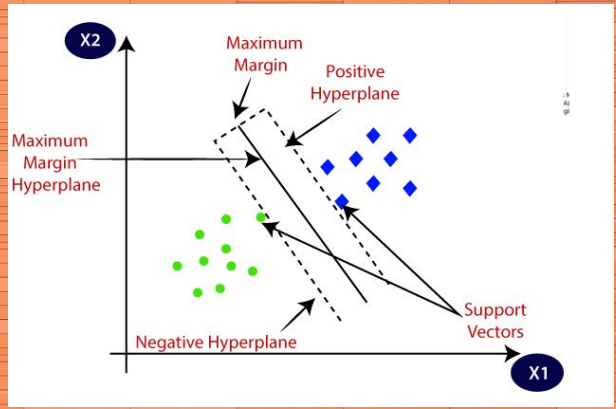


3)Support Vector Machine(SVM)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.



Support Vector Machine(SVM)

SVM

```
Train Accuracy: 82.8585687960688
Accuracy Score: 82.081989866421
```

Classification Report

	precision	recall	f1-score	support
0	0.8279	0.9649	0.8911	4951
1	0.7658	0.3643	0.4937	1562
accuracy			0.8208	6513
macro avg	0.7969	0.6646	0.6924	6513
weighted avg	0.8130	0.8208	0.7958	6513

Confusion Matrix

```
[[4777  174]
 [ 993  569]]
```

Handling Imbalanced data

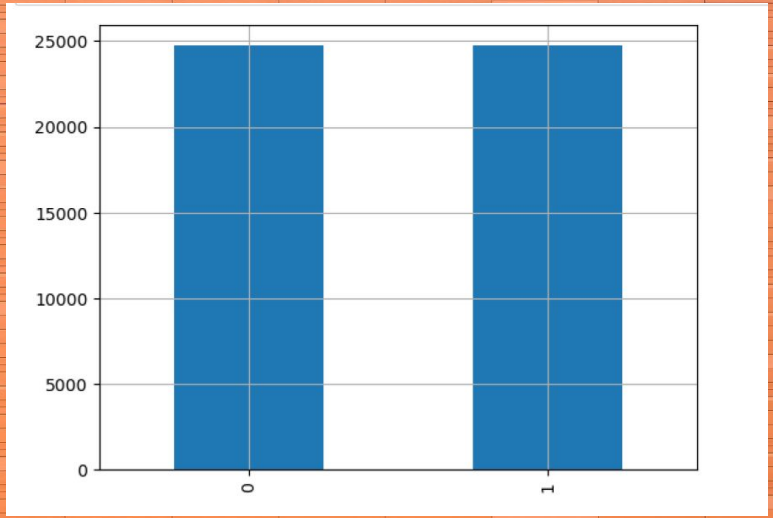
```
0    24720
1     7841
Name:  salary, dtype: int64
```

```
X_res1.shape: (49440, 10)
y_res1.shape: (49440,)
```

Support Vector Machine(SVM)

Handling Imbalanced data

```
Original y value counts:
0    24720
1     7841
Name:  salary, dtype: int64
Resampled y value counts:
0    24720
1    24720
Name:  salary, dtype: int64
```



Retraining and Re-evaluation

1) Decision Tree Classifier

```
X_train.shape: (39552, 10)
y_train.shape: (39552,)
X_test.shape: (9888, 10)
y_test.shape: (9888,)
```

```
Decision Tree Classifier (after SMOTE) Train Accuracy:  96.93315129449837
Decision Tree Classifier (after SMOTE) Accuracy Score:  83.59627831715211
```

Confusion Matrix

```
[[4224  723]
 [ 899 4042]]
```

Classification Report

	precision	recall	f1-score	support
0	0.8245	0.8539	0.8389	4947
1	0.8483	0.8181	0.8329	4941
accuracy			0.8360	9888
macro avg	0.8364	0.8360	0.8359	9888
weighted avg	0.8364	0.8360	0.8359	9888

Retraining and Re-evaluation

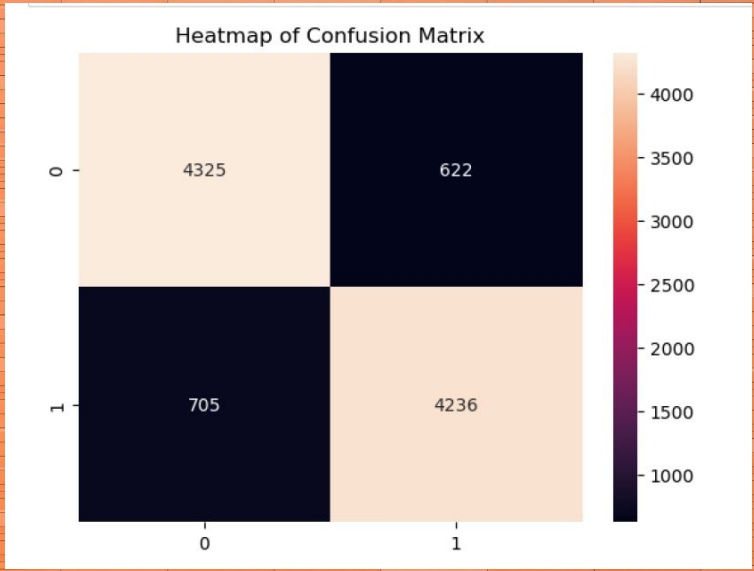
2) Random Forest

86.5796925566343

Classification Report

	precision	recall	f1-score	support
0	0.86	0.87	0.87	4947
1	0.87	0.86	0.86	4941
accuracy			0.87	9888
macro avg	0.87	0.87	0.87	9888
weighted avg	0.87	0.87	0.87	9888

Confusion Report



Retraining and Re-evaluation

3)SVM

```
Train Accuracy: 78.85821197411003
Accuracy Score: 78.47896440129449
```

Confusion Report

```
[[3752 1195]
 [ 933 4008]]
```

Classification Report

	precision	recall	f1-score	support
0	0.8009	0.7584	0.7791	4947
1	0.7703	0.8112	0.7902	4941
accuracy			0.7848	9888
macro avg	0.7856	0.7848	0.7846	9888
weighted avg	0.7856	0.7848	0.7846	9888

Problem statement

Adult Salary Prediction

Solved by:- Iriventi Bharath Vasishta(CSC/20/36), Mohtassim Rahman(CSC/20/29), Nikhil Choudhary(CSC/20/42).

Our task is to analyze the dataset and predict whether the income of an adult will exceed 50k per year or not.

- We have been given 14 features.

  Adult dataset link : http://archive.ics.uci.edu/ml/datasets/Adult (http://archive.ics.uci.edu/ml/datasets/Adult)

In [110]:

```
import numpy as np  # For mathematical calculations
import pandas as pd  # For data preprocessing
import matplotlib.pyplot as plt  # For data visualization
import seaborn as sns  # For data visualization
from sklearn.preprocessing import StandardScaler  # For feature scaling
from statsmodels.stats.outliers_influence import variance_inflation_factor  # For detect
from sklearn.model_selection import train_test_split  # For splitting data into training
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, pre
from sklearn.tree import DecisionTreeClassifier  # For decision tree classification
from sklearn.ensemble import RandomForestClassifier  # For random forest classification
from sklearn.svm import SVC  # For support vector machine classification
from imblearn.over_sampling import SMOTE  # For oversampling the minority class
import warnings
warnings.filterwarnings("ignore")  # To ignore warnings for cleaner output
```

In [111]:

```
df = pd.read_csv("adult_data.csv")
df.head()
```

Out[111]:

	age	workclass	education	occupation	race	sex	capital-gain	capital-loss	hours-per-week	native-country
0	39	State-gov	Bachelors	Adm-clerical	White	Male	2174	0	40	United-States
1	50	Self-emp-not-inc	Bachelors	Exec-managerial	White	Male	0	0	13	United-States
2	38	Private	HS-grad	Handlers-cleaners	White	Male	0	0	40	United-States
3	53	Private	11th	Handlers-cleaners	Black	Male	0	0	40	United-States
4	28	Private	Bachelors	Prof-specialty	Black	Female	0	0	40	Cuba

In [112]:

```
df.columns
```

Out[112]:

```
Index(['age', ' workclass', ' education', ' occupation', ' race', ' sex',
       ' capital-gain', ' capital-loss', ' hours-per-week', ' native-count
ry',
       ' salary'],
      dtype='object')
```

In [113]:

```
len(df.columns)
```

Out[113]:

```
11
```

In [114]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   age              32561 non-null  int64 
 1    workclass       32561 non-null  object
 2    education       32561 non-null  object
 3    occupation      32561 non-null  object
 4    race            32561 non-null  object
 5    sex             32561 non-null  object
 6    capital-gain    32561 non-null  int64 
 7    capital-loss    32561 non-null  int64 
 8    hours-per-week  32561 non-null  int64 
 9    native-country  32561 non-null  object
 10   salary          32561 non-null  object
dtypes: int64(4), object(7)
memory usage: 2.7+ MB
```

In [115]:

```
df.shape
```

Out[115]:

```
(32561, 11)
```

In [116]:

```
df.describe()
```

Out[116]:

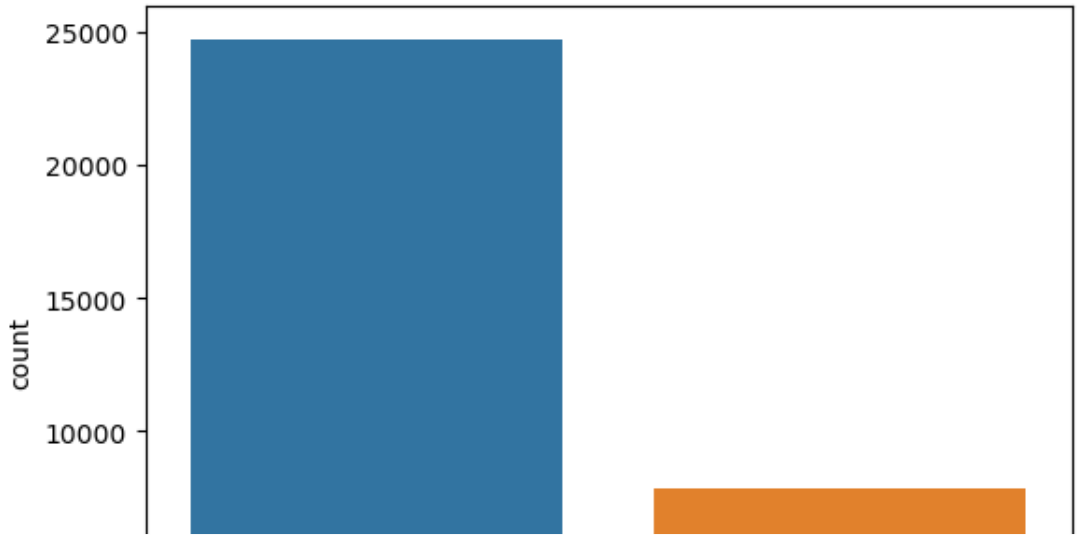
	age	capital-gain	capital-loss	hours-per-week
count	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1077.648844	87.303830	40.437456
std	13.640433	7385.292085	402.960219	12.347429
min	17.000000	0.000000	0.000000	1.000000
25%	28.000000	0.000000	0.000000	40.000000
50%	37.000000	0.000000	0.000000	40.000000
75%	48.000000	0.000000	0.000000	45.000000
max	90.000000	99999.000000	4356.000000	99.000000

In [117]:

```
sns.countplot(x=' salary',data=df )
```

Out[117]:

```
<AxesSubplot:xlabel=' salary', ylabel='count'>
```



In [118]:

```
df[" salary"].value_counts()
```

Out[118]:

```
 <=50K    24720
 >50K      7841
Name:  salary, dtype: int64
```

In [119]:

```
print(f"<= 50k : {round(24720 /32561 * 100 , 2)}")
print(f"> 50k : {round(7841 /32561 * 100 , 2)}")
```

```
<= 50k : 75.92
> 50k : 24.08
```

Exploratory data analysis and Feature Engineering

In [120]:

```
## Here we will check the missing values in our dataset
df.isnull().sum()
```

Out[120]:

```
age               0
 workclass        0
 education        0
 occupation       0
 race             0
 sex              0
 capital-gain     0
 capital-loss     0
 hours-per-week   0
 native-country   0
 salary           0
dtype: int64
```

In [121]:

```
numerical = [i for i in df.columns if df[i].dtypes != "O"]
```

In [122]:

```
numerical
```

Out[122]:

```
['age', ' capital-gain', ' capital-loss', ' hours-per-week']
```

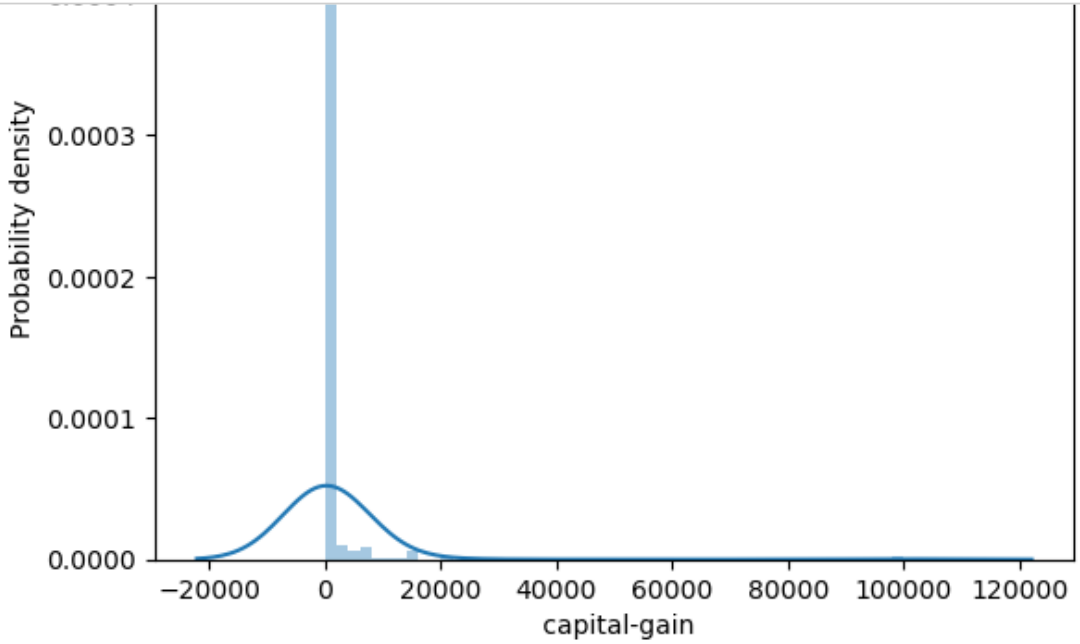
In [123]:

```
for i in numerical:
    print(f" {i}  :  {len(df[i].unique())}")
```

```
 age  :  73
  capital-gain  :  119
  capital-loss  :  92
  hours-per-week  :  94
```

In [124]:

```
for feature in numerical:
    bar = sns.distplot(df[feature] , kde_kws = {'bw' : 1})
    bar.legend(["Skewness: {:0.2f}".format(df[feature].skew())])
    plt.xlabel(feature)
    plt.ylabel("Probability density")
    plt.title(feature)
    plt.show()
```



In [125]:

```
df["age"] = np.log(df["age"])
```

In [126]:

```
df.columns
```

Out[126]:

```
Index(['age', ' workclass', ' education', ' occupation', ' race', ' sex',
       ' capital-gain', ' capital-loss', ' hours-per-week', ' native-count
ry',
       ' salary'],
      dtype='object')
```

Handling categorical data

In [127]:

```
categorical = [i for i in df.columns if df[i].dtypes == "O"]
```

In [128]:

```
for feature in categorical:
    print(f" {feature}  :  {len(df[feature].unique())}")
```

```
  workclass  :  9
  education  :  16
  occupation  :  15
  race  :  5
  sex  :  2
  native-country  :  42
  salary  :  2
```

In [129]:

```
for feature in  categorical:
    print(df[feature].value_counts())
    print("\n \n ")
```

```
 Cambodia                     19
 Trinadad&Tobago              19
 Laos                         18
 Thailand                     18
 Yugoslavia                   16
 Outlying-US(Guam-USVI-etc)   14
 Honduras                     13
 Hungary                      13
 Scotland                     12
 Holand-Netherlands            1
Name:  native-country, dtype: int64



 <=50K    24720
 >50K      7841
Name:  salary, dtype: int64
```

Handling missing values in categorical features

In [130]:

```
print(f"workclass : {round(2093 / 32561 , 4) *100}%")
print(f"occupation : {round(1843 / 32561 , 4) *100}%")
print(f"native-country : {round(583 / 32561 , 4) *100}%")
```

```
workclass : 6.43%
occupation : 5.66%
native-country : 1.79%
```

In [131]:

```
df[" occupation"].mode()[0]
```

Out[131]:

```
' Prof-specialty'
```

here we have less than 6 percent missing values so we can fill it with mode value

In [132]:

```
df[" workclass"] = df[' workclass'].str.replace('?', 'Private' )
df[' occupation'] = df[' occupation'].str.replace('?', 'Prof-specialty' )
df[' native-country'] = df[' native-country'].str.replace('?', 'United-States' )
```

In [133]:

```
# education Category
df[" education"].replace(['Preschool', '1st-4th', '5th-6th', '7th-8th', '9th','10th', '1
                         inplace = True , regex = True)
df[" education"].replace(['Assoc-voc', 'Assoc-acdm', 'Prof-school', 'Some-college'], 'hi
```

In [134]:

```
# income
df[" salary"] = df[" salary"].replace({'<=50K' : 0 , ">50K" : 1 } , regex = True)
```

In [135]:

```
df.head()
```

Out[135]:

	age	workclass	education	occupation	race	sex	capital-gain	capital-loss	hours-per-week	nat cou
0	3.663562	State-gov	Bachelors	Adm-clerical	White	Male	2174	0	40	Uni St
1	3.912023	Self-emp-not-inc	Bachelors	Exec-managerial	White	Male	0	0	13	Uni St
2	3.637586	Private	HS-grad	Handlers-cleaners	White	Male	0	0	40	Uni St
3	3.970292	Private	school	Handlers-cleaners	Black	Male	0	0	40	Uni St
4	3.332205	Private	Bachelors	Prof-specialty	Black	Female	0	0	40	C

In [136]:

```
for feature in categorical:
    print(f" {feature}  :  {len(df[feature].unique())}")
```

```
 workclass  :  8
 education  :  6
 occupation  :  14
 race  :  5
 sex  :  2
 native-country  :  41
 salary  :  2
```

In [137]:

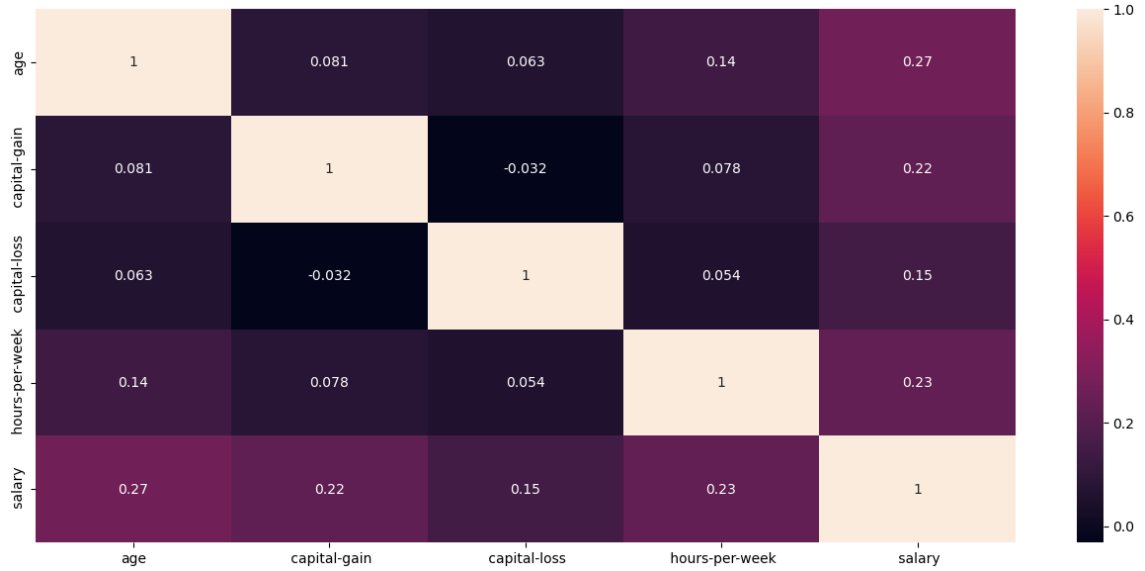
```
df[" education"].unique()
```

Out[137]:

```
array([' Bachelors', ' HS-grad', ' school', ' Masters', ' higher',
       ' Doctorate'], dtype=object)
```

In [138]:

```
plt.figure(figsize = (16 , 7))
sns.heatmap(df.corr(), annot=True);
```



In [139]:

```
from sklearn.preprocessing import  LabelEncoder
```

In [140]:

```
df = df.apply(LabelEncoder().fit_transform)
df .head()
```

Out[140]:

	age	workclass	education	occupation	race	sex	capital-gain	capital-loss	hours-per-week	native-country	sala
0	22	6	0	0	4	1	25	0	39	38	
1	33	5	0	3	4	1	0	0	12	38	
2	21	3	2	5	4	1	0	0	39	38	
3	36	3	5	5	2	1	0	0	39	38	
4	11	3	0	9	2	0	0	0	39	4	

Pre Processing

In [141]:

```
# Preprocess the data for machine learning
target_name = ' salary'
y = df[target_name]
x_train = df.drop(target_name, axis=1)
```

In [142]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_res = sc.fit_transform(x_train)
```

In [143]:

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

In [144]:

```
# Calculate variance inflation factor (VIF) for each feature
vif = pd.DataFrame()
vif["VIF Factor"] = [variance_inflation_factor(x_res, i) for i in range(x_res.shape[1])]
vif["features"] = x_train.columns
```

In [145]:

```
# Display VIF for each feature
print("Variance Inflation Factor (VIF):")
pd.DataFrame(vif)
```

```
Variance Inflation Factor (VIF):
```

Out[145]:

	VIF Factor	features
0	1.031483	age
1	1.008832	workclass
2	1.024136	education
3	1.003528	occupation
4	1.024756	race
5	1.079126	sex
6	1.036486	capital-gain
7	1.015167	capital-loss
8	1.084125	hours-per-week
9	1.016311	native-country

In [146]:

```
x_res.shape
```

Out[146]:

```
(32561, 10)
```

MODEL EVALUATION

1) Decision Tree Classifier

In [147]:

```
X_train, X_test, y_train, y_test = train_test_split(x_res, y, test_size=0.2, random_stat

# Print the shapes of training and testing sets
print("X_train.shape:", X_train.shape)
print("y_train.shape:", y_train.shape)
print("X_test.shape:", X_test.shape)
print("y_test.shape:", y_test.shape)
```

```
X_train.shape: (26048, 10)
y_train.shape: (26048,)
X_test.shape: (6513, 10)
y_test.shape: (6513,)
```

In [148]:

```
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)

# Predict on training and testing sets using Decision Tree Classifier
dt_train_pred = dt.predict(X_train)
dt_test_pred = dt.predict(X_test)
```

In [149]:

```
# Print accuracy scores and confusion matrix for Decision Tree Classifier
print('Train Accuracy:', accuracy_score(y_train, dt_train_pred) * 100)
print('Accuracy Score:', accuracy_score(y_test, dt_test_pred) * 100)
```

```
Train Accuracy: 95.98817567567568
Accuracy Score: 79.57930293259633
```

Confusion Matrix

In [150]:

```
print(confusion_matrix(y_test, dt_test_pred))
```

```
[[4330  621]
 [ 709  853]]
```

In [151]:

```
#Classfication_report
```

In [152]:

```
print(classification_report(y_test, dt_test_pred, digits=4))
```

```
              precision    recall  f1-score   support

           0     0.8593    0.8746    0.8669      4951
           1     0.5787    0.5461    0.5619      1562

    accuracy                         0.7958      6513
   macro avg     0.7190    0.7103    0.7144      6513
weighted avg     0.7920    0.7958    0.7937      6513
```

Precision Score

In [153]:

```
print('Precision score of macro is: ' , round(precision_score(y_test , dt_test_pred , av
print('Precision score of micro is: ' , round(precision_score(y_test , dt_test_pred , av
print('Precision score of weighted is: ' , round(precision_score(y_test , dt_test_pred ,
```

```
Precision score of macro is:  71.9
Precision score of micro is:  79.58
Precision score of weighted is:  79.2
```

Recall Score

In [154]:

```
print('recall_score score of macro is: ' , round(recall_score(y_test , dt_test_pred , av
print('recall_score score of micro is: ' , round(recall_score(y_test , dt_test_pred , av
print('recall_score score of weighted is: ' , round(recall_score(y_test , dt_test_pred ,
```

```
recall_score score of macro is:  71.03
recall_score score of micro is:  79.58
recall_score score of weighted is:  79.58
```

F1 Score

In [155]:

```
print('f1_score score of macro is: ' , round(f1_score(y_test , dt_test_pred , average =
print('f1_score score of micro is: ' , round(f1_score(y_test , dt_test_pred , average =
print('f1_score score of weighted is: ' , round(f1_score(y_test , dt_test_pred , average
```

```
f1_score score of macro is:  71.44
f1_score score of micro is:  79.58
f1_score score of weighted is:  79.37
```

2)Random Forest

In [156]:

```
from sklearn.ensemble import RandomForestClassifier
rf_classifier = RandomForestClassifier(n_estimators = 20, criterion = 'entropy' ,random_
rf_classifier.fit(X_train, y_train)
y_pred = rf_classifier.predict(X_test)
accuracy_score(y_test, y_pred)*100
```

Out[156]:

```
82.84968524489483
```

In [157]:

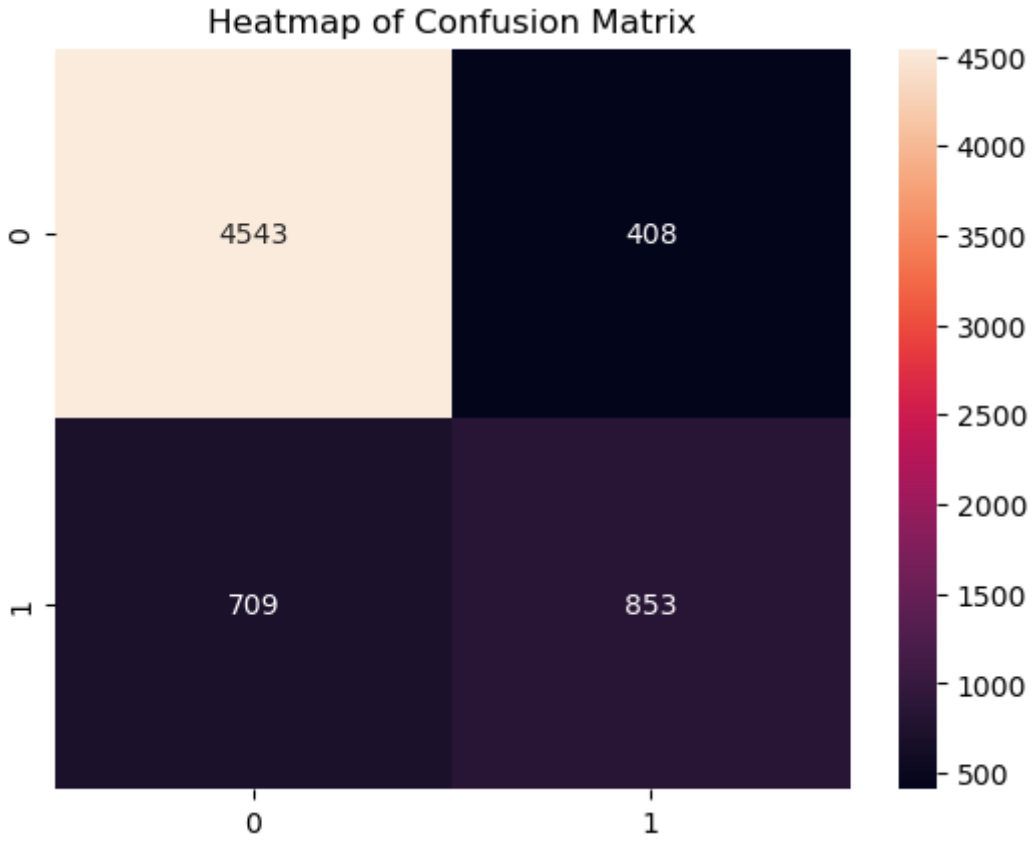
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.92      0.89      4951
           1       0.68      0.55      0.60      1562

    accuracy                           0.83      6513
   macro avg       0.77      0.73      0.75      6513
weighted avg       0.82      0.83      0.82      6513
```

Confusion Matrix

In [158]:

```
cm = confusion_matrix(y_test, y_pred )
plt.title('Heatmap of Confusion Matrix', fontsize = 12)
sns.heatmap(cm, annot = True ,  fmt = "d")
plt.show()
```



3)Support Vector Machine(SVM)

In [159]:

```
# Train Support Vector Classifier (SVC)
svc_model = SVC()
svc_model.fit(X_train, y_train)

# Predict on training and testing sets using SVC
svc_train_pred = svc_model.predict(X_train)
svc_test_pred = svc_model.predict(X_test)
```

In [160]:

```
# Print accuracy scores and confusion matrix for SVC
print('Train Accuracy:', accuracy_score(y_train, svc_train_pred) * 100)
print('Accuracy Score:', accuracy_score(y_test, svc_test_pred) * 100)
```

```
Train Accuracy: 82.8585687960688
Accuracy Score: 82.081989866421
```

Confusion Matrix

In [161]:

```
print(confusion_matrix(y_test, svc_test_pred))
```

```
[[4777  174]
 [ 993  569]]
```

Classification Report

In [66]:

```
print(classification_report(y_test, svc_test_pred, digits=4))
```

```
              precision    recall  f1-score   support

           0     0.8279    0.9649    0.8911      4951
           1     0.7658    0.3643    0.4937      1562

    accuracy                         0.8208      6513
   macro avg     0.7969    0.6646    0.6924      6513
weighted avg     0.8130    0.8208    0.7958      6513
```

Handling Imbalanced data

In [162]:

```
# Check value counts of y
print(y.value_counts())
```

```
0    24720
1     7841
Name:  salary, dtype: int64
```

In [163]:

```
# Apply SMOTE for handling imbalanced dataset
oversample = SMOTE(k_neighbors=4)
X_res1, y_res1 = oversample.fit_resample(x_res, y)
```

In [164]:

```
# Print shapes of resampled dataset
print("X_res1.shape:", X_res1.shape)
print("y_res1.shape:", y_res1.shape)
```

```
X_res1.shape: (49440, 10)
y_res1.shape: (49440,)
```

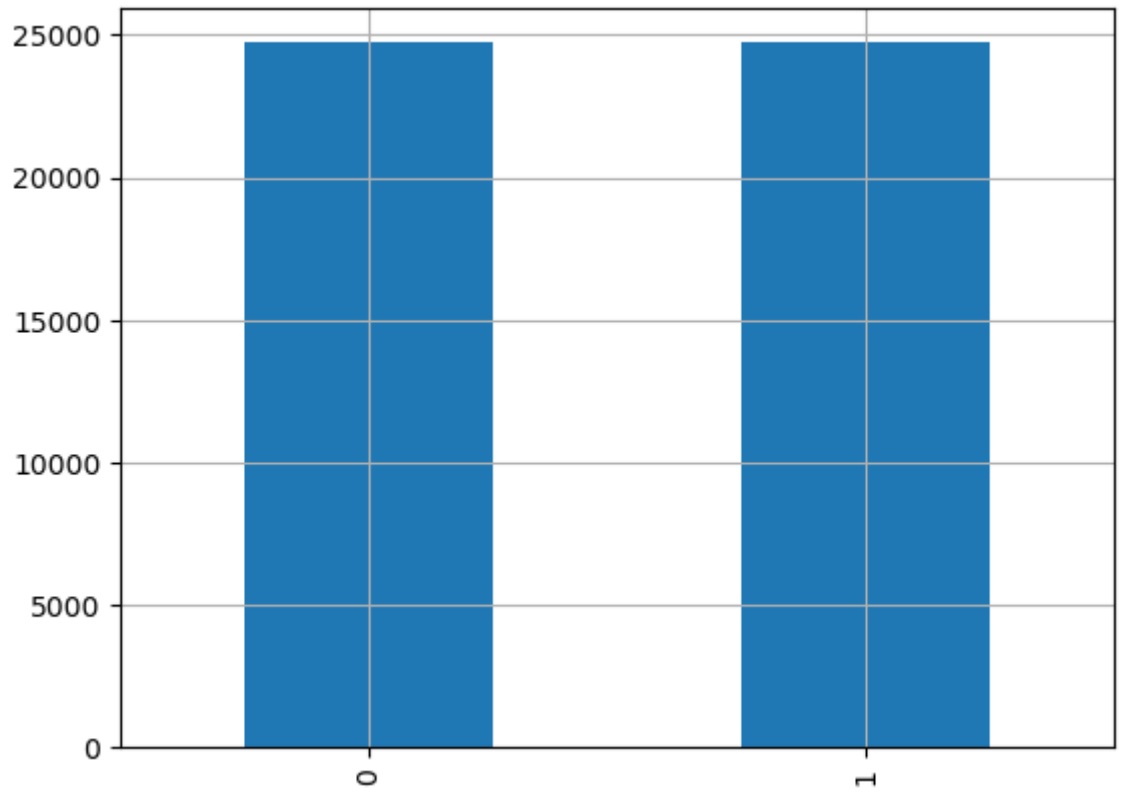
In [165]:

```
# Check value counts of original and resampled y
print("Original y value counts:")
print(y.value_counts())
print("Resampled y value counts:")
print(pd.Series(y_res1).value_counts())
```

```
Original y value counts:
0    24720
1     7841
Name:  salary, dtype: int64
Resampled y value counts:
0    24720
1    24720
Name:  salary, dtype: int64
```

In [166]:

```
# Plot bar chart of resampled y value counts
pd.Series(y_res1).value_counts().plot.bar()
plt.grid()
```



Retraining and Re-evaluation

1) Decision Tree Classifier

In [167]:

```
X_train, X_test, y_train, y_test = train_test_split(X_res1, y_res1, test_size=0.2, rando


print("X_train.shape:", X_train.shape)
print("y_train.shape:", y_train.shape)
print("X_test.shape:", X_test.shape)
print("y_test.shape:", y_test.shape)
```

```
X_train.shape: (39552, 10)
y_train.shape: (39552,)
X_test.shape: (9888, 10)
y_test.shape: (9888,)
```

In [168]:

```
# Decision Tree Classifier
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)

dt_train_pred = dt.predict(X_train)
dt_test_pred = dt.predict(X_test)
```

In [169]:

```
print('Decision Tree Classifier (after SMOTE) Train Accuracy: ', accuracy_score(y_train,
print('Decision Tree Classifier (after SMOTE) Accuracy Score: ', accuracy_score(y_test,
```

```
Decision Tree Classifier (after SMOTE) Train Accuracy:  96.93315129449837
Decision Tree Classifier (after SMOTE) Accuracy Score:  83.59627831715211
```

Confusion Matrix

In [170]:

```
print(confusion_matrix(y_test, dt_test_pred))
```

```
[[4224  723]
 [ 899 4042]]
```

Classfication Report

In [171]:

```
print(classification_report(y_test, dt_test_pred, digits=4))
```

```
              precision    recall  f1-score   support

           0     0.8245    0.8539    0.8389      4947
           1     0.8483    0.8181    0.8329      4941

    accuracy                         0.8360      9888
   macro avg     0.8364    0.8360    0.8359      9888
weighted avg     0.8364    0.8360    0.8359      9888
```

2) Random Forest

In [172]:

```
# Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
rf_classifier = RandomForestClassifier(n_estimators = 20, criterion = 'entropy' ,random_
rf_classifier.fit(X_train, y_train)
y_pred = rf_classifier.predict(X_test)
accuracy_score(y_test, y_pred)*100
```

Out[172]:

```
86.5796925566343
```

Classification Report

In [173]:

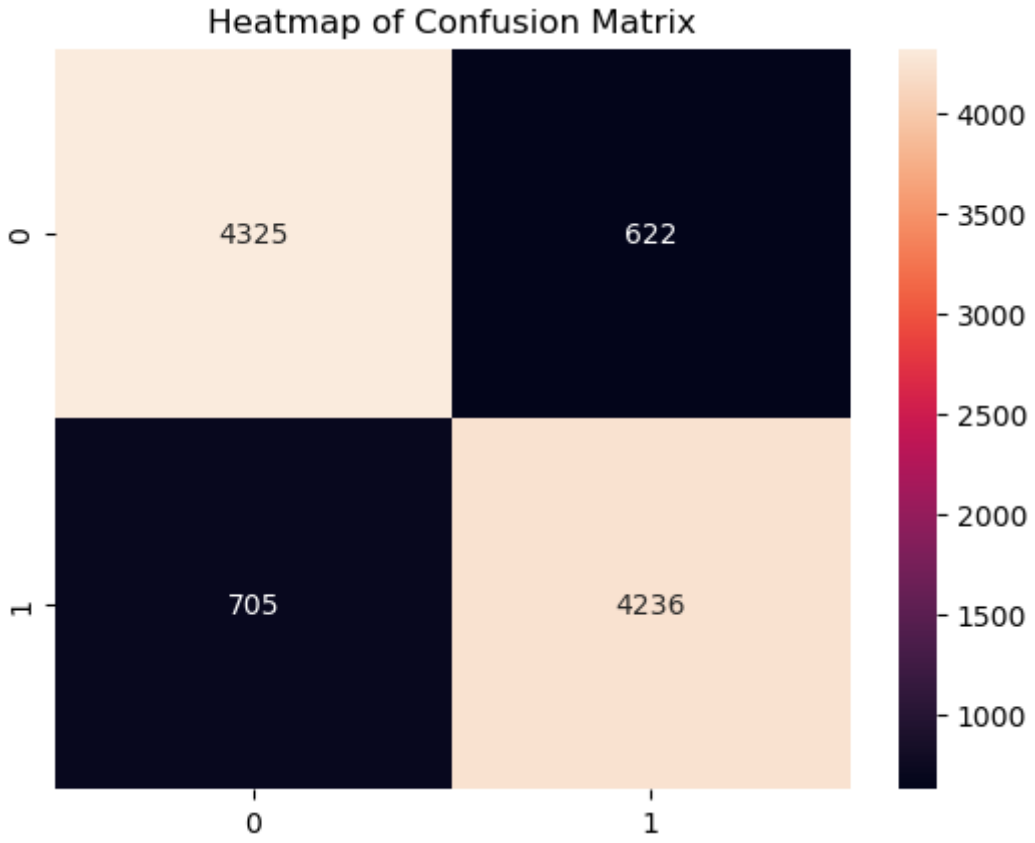
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.87      0.87      4947
           1       0.87      0.86      0.86      4941

    accuracy                           0.87      9888
   macro avg       0.87      0.87      0.87      9888
weighted avg       0.87      0.87      0.87      9888
```

Confusion Report

In [174]:

```
cm = confusion_matrix(y_test, y_pred )
plt.title('Heatmap of Confusion Matrix', fontsize = 12)
sns.heatmap(cm, annot = True ,  fmt = "d")
plt.show()
```



3)SVM

In [175]:

```
# Train Support Vector Classifier (SVC)
svc_model = SVC()
svc_model.fit(X_train, y_train)

# Predict on training and testing sets using SVC
svc_train_pred = svc_model.predict(X_train)
svc_test_pred = svc_model.predict(X_test)
```

In [176]:

```
# Print accuracy scores and confusion matrix for SVC
print('Train Accuracy:', accuracy_score(y_train, svc_train_pred) * 100)
print('Accuracy Score:', accuracy_score(y_test, svc_test_pred) * 100)
```

```
Train Accuracy: 78.85821197411003
Accuracy Score: 78.47896440129449
```

Confusion Report

In [177]:

```
print(confusion_matrix(y_test, svc_test_pred))
```

```
[[3752 1195]
 [ 933 4008]]
```

Classification Report

In [178]:

```
print(classification_report(y_test, svc_test_pred, digits=4))
```

```
              precision    recall  f1-score   support

           0     0.8009    0.7584    0.7791      4947
           1     0.7703    0.8112    0.7902      4941

    accuracy                         0.7848      9888
   macro avg     0.7856    0.7848    0.7846      9888
weighted avg     0.7856    0.7848    0.7846      9888
```

In [ ]:

```

```

Session 2020-2023

Department of Computer Science

Aryabhatta College

University of Delhi

Shape Up Fitness & Nutrition App

Software Engineering Project Report

Project Name: Fitness & Nutrition App
Supervisor: Deepak Sharma
Course: B.Sc (Hons) Computer Science
Semester: IV
Name: Iriventi Bharath Vasishta
Subject: Software Engineering
College Roll No: CSC/20/36
Examination Roll No.: 20059570028

Contents

Acknowledgement

In performing our assignment, we had to take the help and guideline of some respected persons, who deserve our greatest gratitude. The completion of this assignment gives us much Pleasure. We would like to show our gratitude to our professor Mr.Deepak Sharma who introduced us to the Methodology of work, and whose passion for the “underlying structures” had lasting effect and for giving us a good guideline for assignment throughout numerous consultations. We would also like to expand our deepest gratitude to all those who have directly and indirectly guided us in writing this assignment.

Many people, especially our classmates and team members, have made valuable comment suggestions on this proposal which gave us inspiration to improve our assignment. We thank all the people for their help directly and indirectly to complete our assignment.

Certificate

This is to certify that

Iriventi Bharath Vasishta (2 0 0 5 9 5 7 0 0 2 8)

successfully carried out the completion of the project entitled “ShapeUp Fitness & Nutrition App)” under my supervision. The Project has been submitted as per the requirement of the Lab based on Software Engineering of **B.Sc. (H) Computer Science, IV Semester.** successfully carried out the completion of the project.

Submitted by

Iriventi Bharath Vasishta
(20059570028)

Supervisor

Dr. Deepak Sharma

Problem Statement

The workload for a human being in the 21st century is quite stressful and frustrating. For example, like a student who goes to school,studying at home and doing extracurricular activities or a person with a job working 9-5. To obtain the luxury of money, we often risk our health and fitness. People have ruined their lifestyle due to the lack of time and lack of knowledge on how to keep a healthy lifestyle. The application ShapeUp is a simple mobile application. We are aiming towards making a healthy community that is just a fingertip away for the people at home who are busy in their professional lives; providing them with suitable diet plans and exercises. The app offers multiple features such as BMI Index, calorie intake calculator etc catering to various profiles of people.

Process Model

For this project we will select one of the Evolutionary Process models.
The Spiral Model is quite apt for the project. The following points justify our choice of the spiral model:

- The Spiral Model allows evolutionary development of software.
- Scale and the user base of the app would increase over time. As an evolutionary model, the spiral model will allow us to accommodate these expansions.
- People might be requesting some updates & changes to be implemented, and Spiral Model allows to accommodate the change in requirements.
- Allows prototyping at any stage so that the stakeholders can get a better idea about a feature as when implemented.

1.1 Software Requirement Specification

1.1 Overall Description

- Registration Page
- Login
- Selecting Gym trainer
- Select Dietician
- BMI Calculator

1.1.1 Product Function

This product will allow the client/customer to find/interact with professional people (dieticians and trainers). Who can help the customer to find the desired diet and training they need to achieve the body goals they want.

1.1.2 User Characteristics

The users would need to be above the age of 16 to use the app. They would also have to be an android user since the app would be android based.

1.1.3 General Constraints

• The app can be used 24x7 but dieticians and trainers can only be contacted during certain time periods fixed by them.
• The user's device must be connected to the internet in order to browse available trainers and dieticians
• Users with specific allergies are required to specify beforehand.

1.2 External Interface Requirements

1.2.1 User Interfaces

The user will be asked to register the first time they use the app after which they would be required to login to use the app. The user must also have basic knowledge on how to use an android device. They must also be willing to regularly follow the routines provided to them through the app.

1.2.2 Hardware Interfaces

In order to use this application, the user must have a device that supports/ is capable of connecting to the internet using any compatible standards (3G, 4G, 5G and Wi-fi etc).

1.2.3 Software Interfaces

The user should have a GPS enabled device, which will help the user to gain accurate information about nearby dietician, gym and trainers. The app also uses a database management system to keep record of registered dieticians and trainers.

1.3 Functional Requirements

Here are the terms user refers to customers

1.3.1 FR 1

Registration
Input: The user will have the option to either register themselves as a customer or a trainer/dietician. Customers will be asked the following: Name, Gender, Age, Address, Email ID and Phone number. The Trainer/Dietician will be asked the following: qualification, address of their practice and their fees.
Process: The details will be stored in their respective databases.
Output: The users will be shown that hey have successfully registered and they will be taken to the login page.

1.3.2 FR 2

Login page
Input: The user has to enter their Username and Password.
Process: The application will verify the user's credentials.
Output: If the credentials match then the user will be logged in. If the credentials do not match then the error message will be shown.

1.3.3 FR 3

BMI Calculator
Input: The app will ask the user to enter their height and body weight.
Process: The application then will calculate and provide the Basic Metabolism Index.
Output: The user would be able to know their optimum fitness state they are in now

1.3.4 FR 4

Calorie Intake Calculator
Input: The user will be asked to set a calorie limit for their every day intake. Then, they would need to input the type and amount of food they had.

Process: The calorie present in the food item that the user choses would be retrieved from a web API
Output: The data retrieved would be then displayed to the user

1.3.5 FR 5

Select gym trainer
Input: The user will be asked to choose for a customized workout routine by a professional gym trainer.
Process: The user will be taken to a list of available gym trainers to connect to and customized workout plans.
Output: The user will be provided with a list of available gym trainers to contact.

1.3.6 FR 6

Select dietician
Input: The user will be asked to enter their diet preferences according to what they want.
Process: The user will be given a list of available dieticians to connect for a customized diet plan
Output: They will be provided with a list of registered dieticians

1.4 Performance Requirement

- **Performance:** The user will be required to follow the workout and diet plan that is provided to them. They will have to stay consistent in order for it to be effective.

- **Safety Requirements:** In the case of damage to the existing database due to unforeseen circumstances, there needs to be a backup to restore the database without delay.

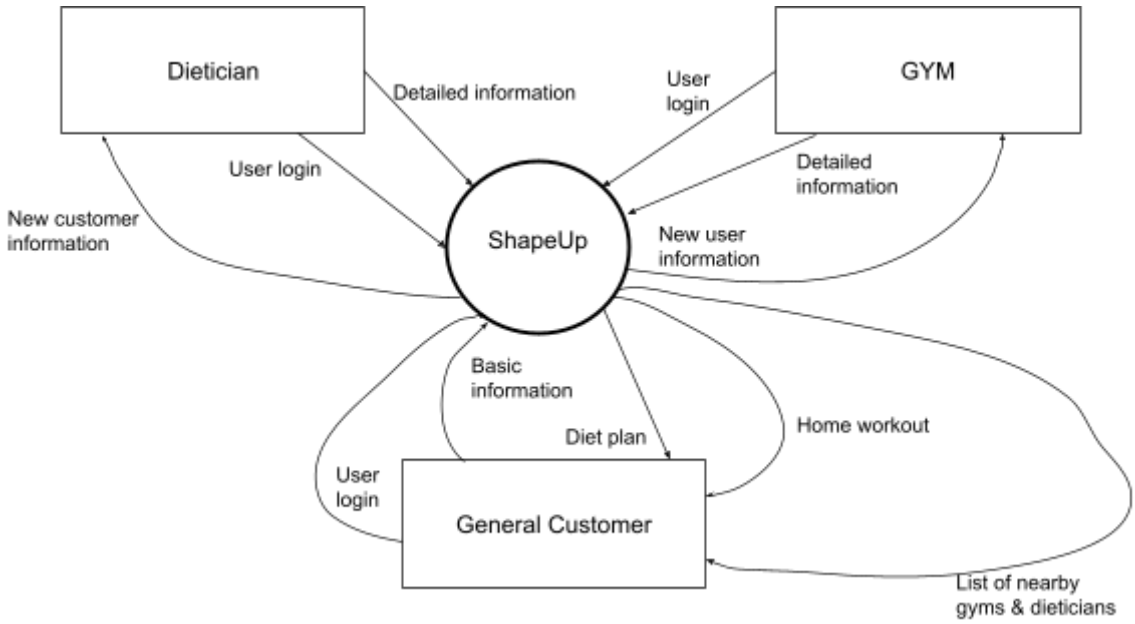
- **User Backup:** The user will not necessarily need to use one particular device to access the app. The user will only need to login from any device with their data being retrieved from our online servers.

1.5 Design Constraints

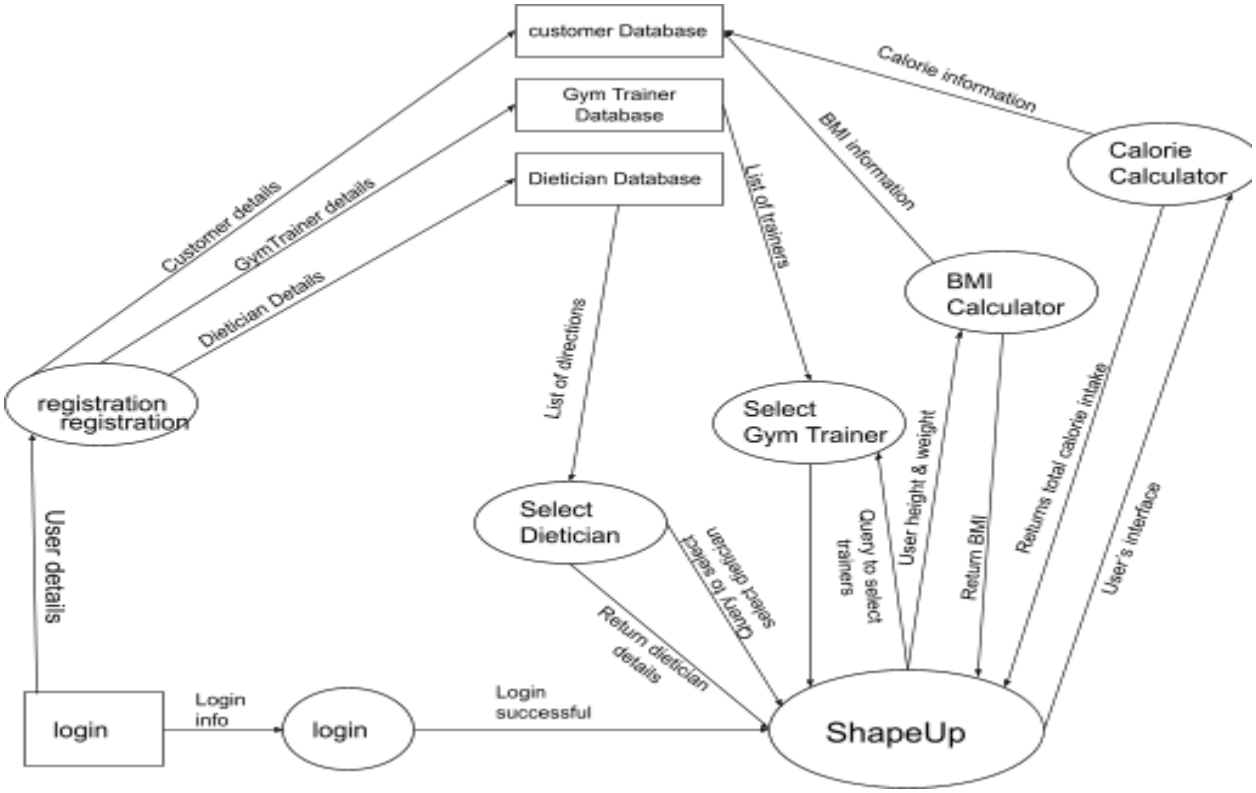
Python doesn't support proper database management within, user needs to import databse from third party applications like mysql, XML and etc.

1.6 Data Flow Diagram

1.6.1 Context Level DFD



1.6.2 Level 1 DFD

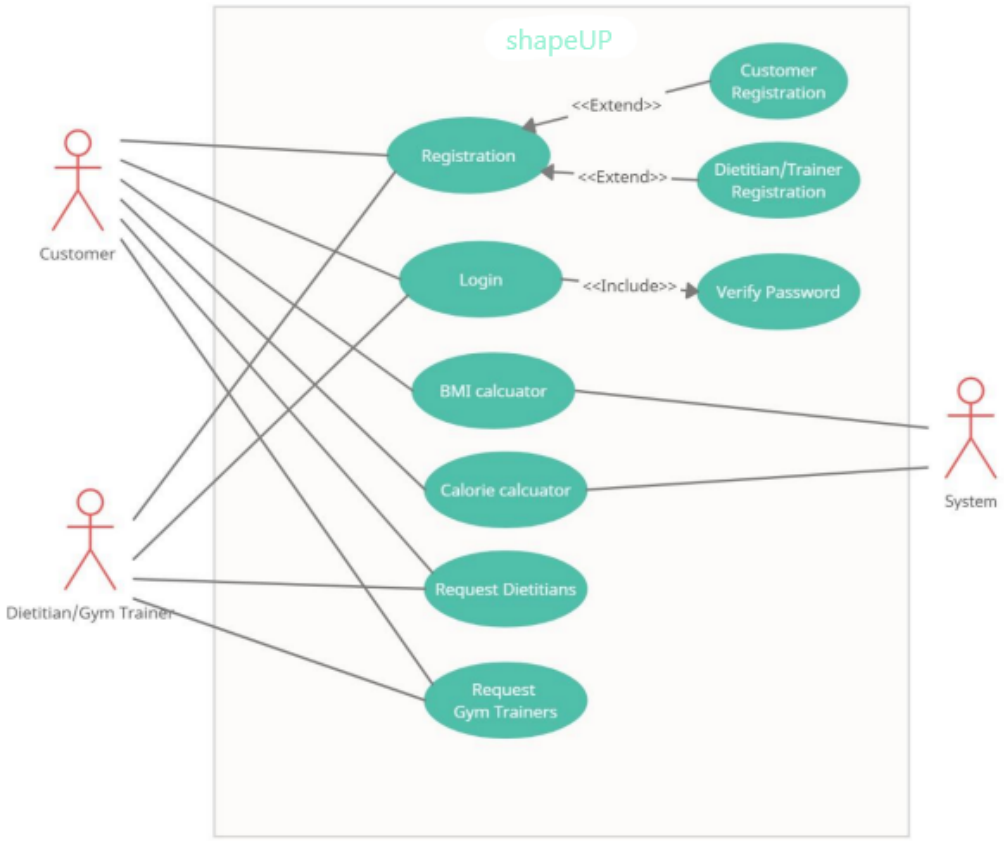


1.7 Data Dictionary

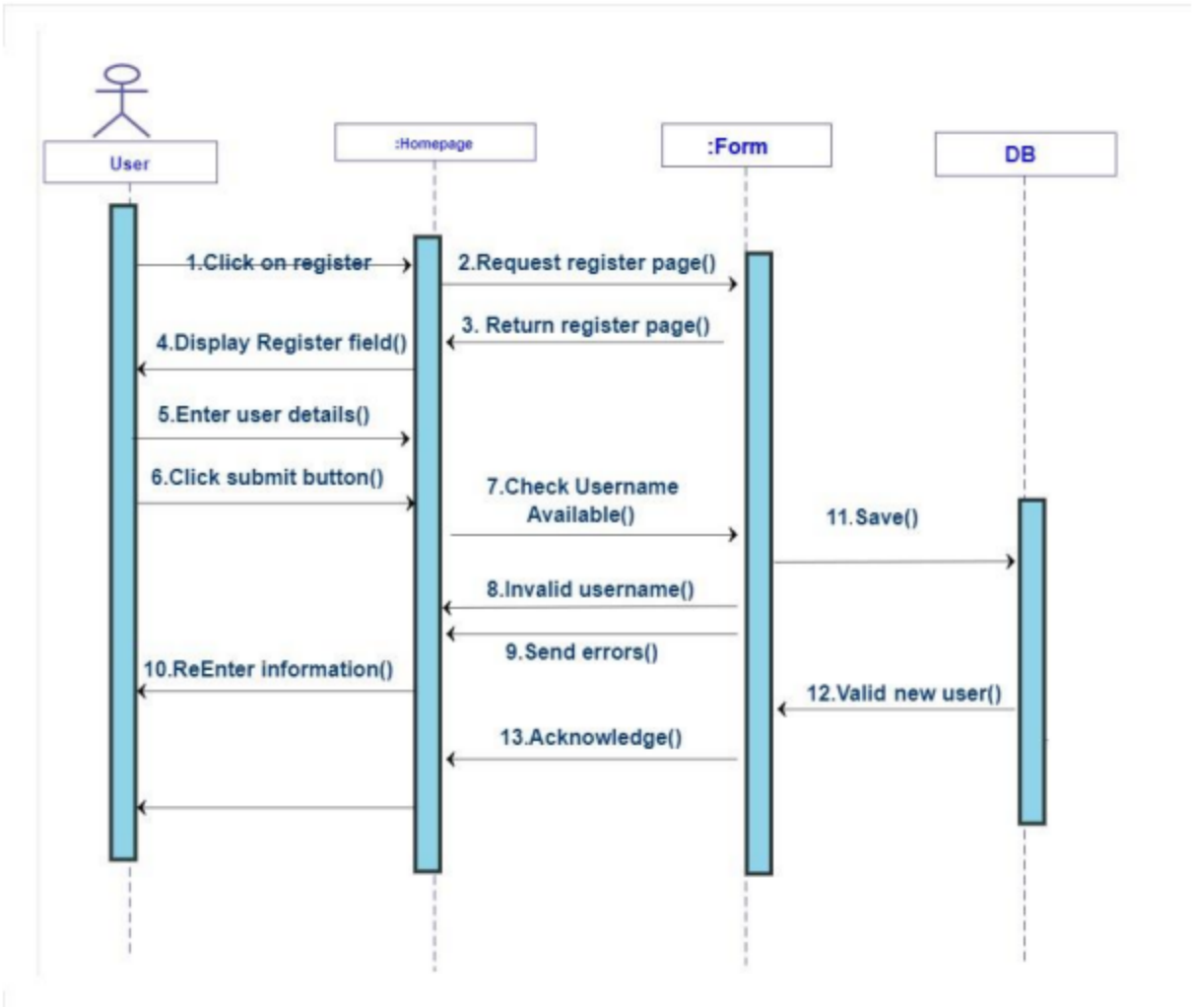
S. No.	Data	Description
1	Trainer’s Details	Name + Qualifications + Address of Practise + Fee
2	Dietician’s Details	Name + Qualifications + Address of Practise + Fee
3	Customer Register	Name + Gender + Age + Address + Email id + Phone Number
4	Customer Login	Username (Name) + Password + Registered Number

1.8 Use Cases and Use Case Diagram

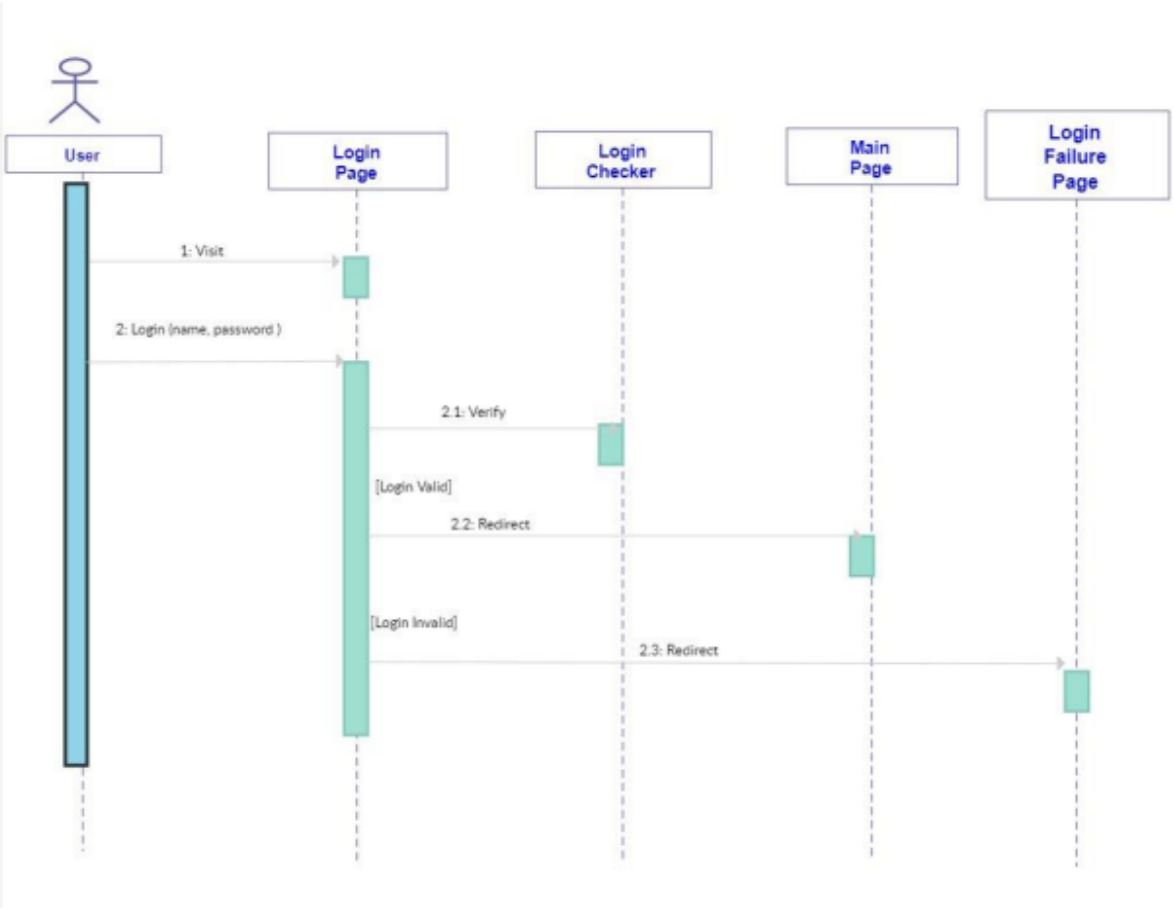
Use Case Diagram



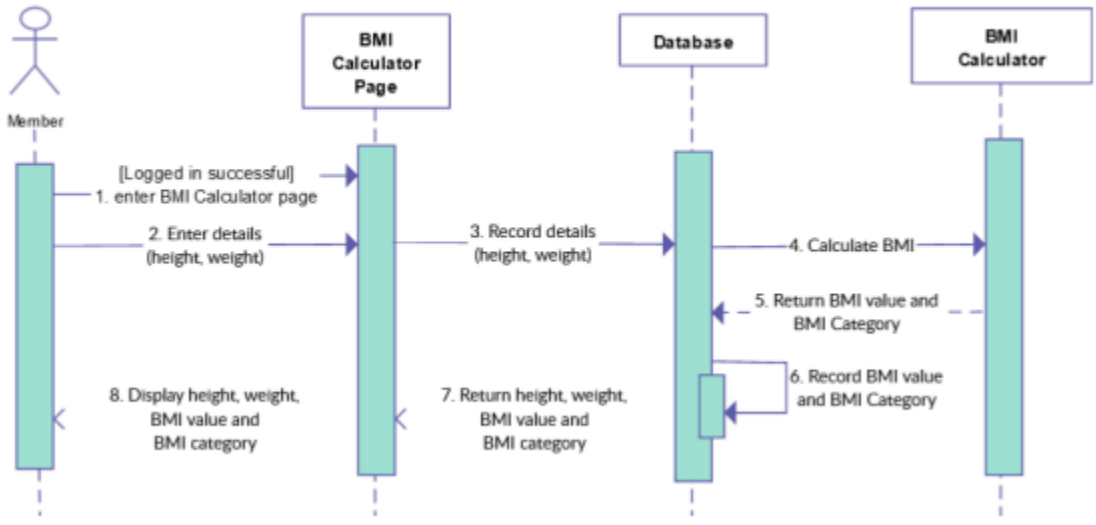
Sequential Diagram



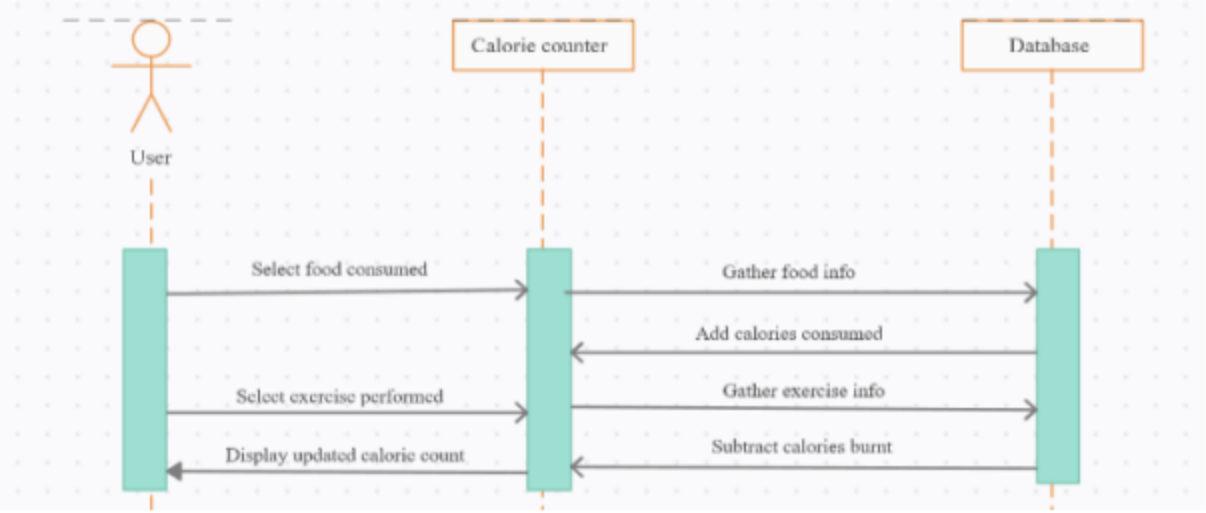
Sequential diagram for registration



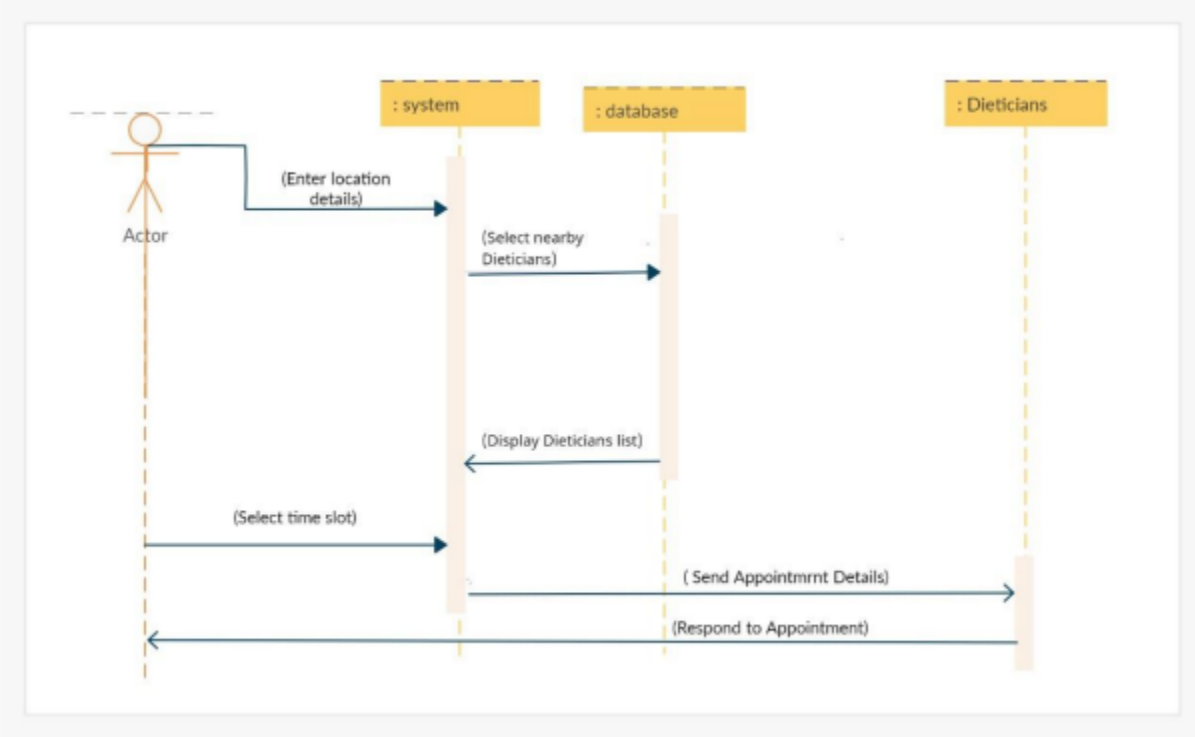
Sequential diagram for login



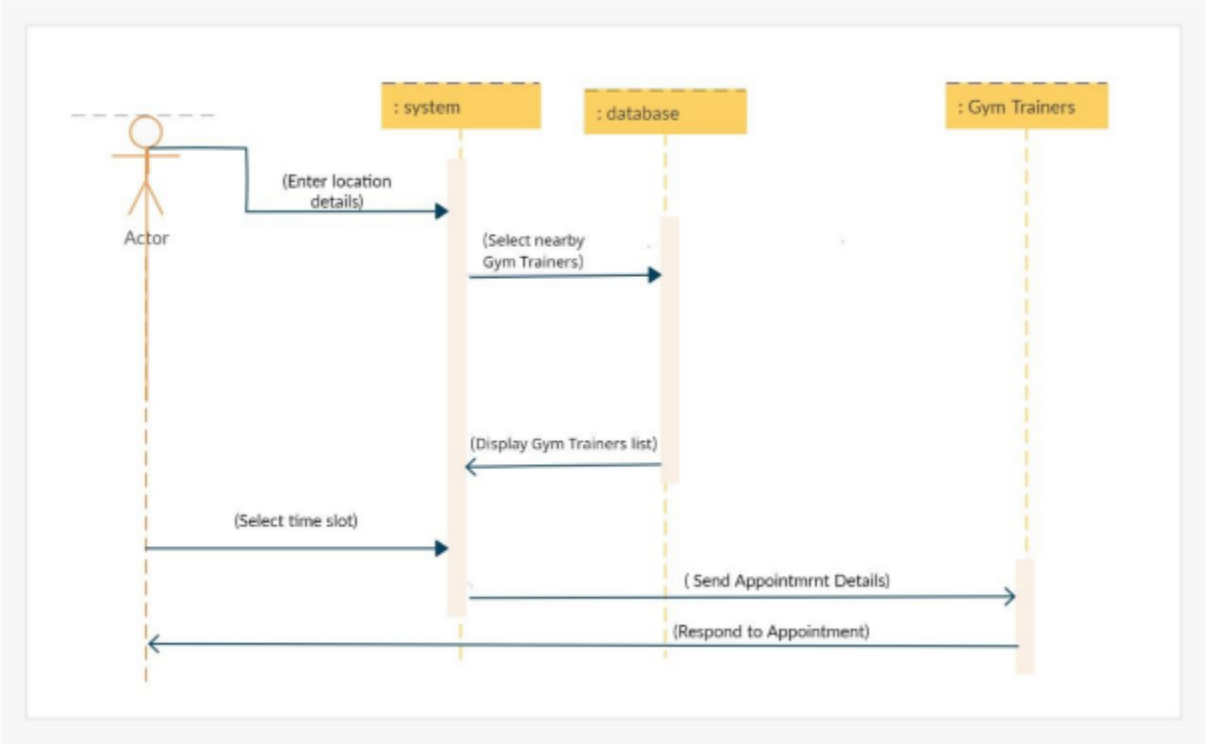
Sequential diagram for BMI calculator



Sequential diagram for calorie calculator



Sequential diagram for dietician request



Sequential diagram for trainers

2. Estimations

2.1 Function Points

The function points are derived using the empirical relationship based on the countable measures of the software's information domain and the qualitative assessments of software complexity. Information domain values are defined in the following manner:

- Number of external inputs (EIs). Each external input originates from a user or is transmitted from another application and provides distinct application-oriented data or control information. Inputs are often used to update internal logical files (ILFs). Inputs should be distinguished from inquiries, which are counted separately.
- Number of external outputs (EOs). Each external output is derived data within the application that provides information to the user. In this context external output refers to reports, screens, error messages, etc. Individual data items within are port are not counted separately
- Number of external inquiries (EQs). An external inquiry is defined as an online input that results in the generation of some immediate

software response in the form of an online output (often retrieved from an ILF).

- Number of external interface files (EIFs). Each external interface file is logical. Grouping of data that resides external to the application but provides information that may be of use to the application.

Information Domain Value	Count	Simple	Average	Complex	FP Count
External Inputs	4	3	4	6	4*4=16
External Outputs	4	4	5	7	4*5=20
External Inquires	2	3	4	6	2*4=8
Internal Logic Files	3	7	10	15	3*10=30
External Logic Files	1	5	7	10	1*7=7
Total or Unadjusted Function Point Count (UFP)					81

Rating	0	1	2	3	4	5
Degree of influence	*Not present, or no influence i.e. not important or applicable*	*Incidental influence (Simple)*	*Moderate influence (Relatively Simple)*	*Average influence (Average)*	*Significant influence*	*Strong influence throughout absolutely essential*

S. No.	Questions	Scale
1	Does the system require reliable backup and recovery?	4
2	Are specialized data communications required to transfer information to or from the application?	4

3	Are there distributed processing functions?	3
4	Is performance critical?	4
5	Will the system run in an existing, heavily utilized operational environment?	3
6	Does the system require online data entry?	4
7	Does the online data entry require the input transaction to be built over multiple screens or operations?	0
8	Are the ILFs updated online?	4
9	Are the inputs, outputs, files, or inquiries complex?	3
10	Is the internal processing complex?	2
11	Is the code designed to be reusable?	3
12	Are conversion and installation included in the design?	0
13	Is the system designed for multiple installations in different organizations?	3
14	Is the application designed to facilitate change and ease of use by the user?	4
Σ(Fi)		**41**

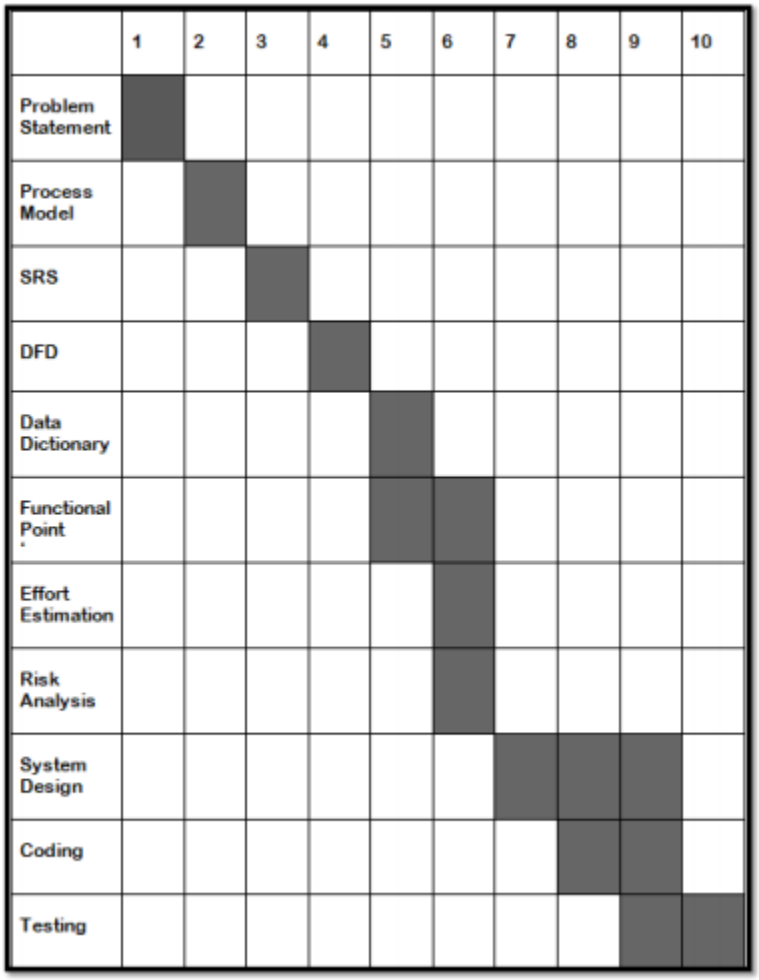
Complexity adjustment factor = Value Adjustment Factor (VAF)
= [0.6 + 0.01 * Sum of F(x)]
= [0.65 + 0.01*41]
= 1.05
Adjusted FP Count = Unadjusted FP count * VAF
= 81 * 1.05
= 85.05
Hence, Adjusted FP count or FP(estimated) = 85 (approx.)

2.2 Efforts

FP (estimated) = 85
We assume that Average Productivity = 2FP per person month Labour Rate = Rs. 25,000 per month
From the above assumptions:
Estimated Effort = FP(estimated)/Average Productivity
85FP/(2 FP per pm) = 42.5 pm
Cost Per FP = Labour Rate / Average Productivity
= Rs. 25,000 pm / (2 FP per pm)
= RS. 12,500 per FP Total Estimated Project Cost
= Estimated Effort * Labour Rate

= 42.5pm * Rs. 25,000 pm
= Rs. 1,062,500
OR
Total Estimated Project Cost
= FP(estimated) * Cost Per FP
= 85 * Rs 12,500 per FP
= Rs. 1,062,500
Thus, the Total Estimated Project Cost is Rs. 1,062,500and the estimated effort is 42.5 person months.

3. Scheduling



4. Risk Management

4.1 Risk Table

After the analysis of our project we found that there are possibilities that the five categories of risk would likely affect the product in the upcoming days. A risk table

provides us with a technique for projection. It will show the probability of the occurrence of those risks and the impact it would have on the project.

S. No.	Risk	Category	Probability	Impact
1	Size estimate may be significantly low	PS	50%	2
2	Larger no. of users as planned	PS	30%	3
3	Technology will not meet expectations	TE	60%	1
4	Staff inexperienced	ST	30%	2
5	End users resist system	BU	50%	2
6	Funding will be lost	CU	40%	1

4.2 RMMM .

RMMM stands for ‘Risk Mitigation, Monitoring, and Management’.
Risk Mitigation is a strategy that is adopted by the software team for avoiding the risk.
Monitoring is a strategy that is adopted by the project manager to monitor factors that will provide an indication of whether there is a highly likely chance for it to become a risk or not.
Management comes into play after the failure of the Mitigation Strategy, when risks become the reality. This’ll help us to reduce the impact of the risk on the project.
In our project there is a risk that lies above the cut-off line. The cut-off line for our project was 50% and above in terms of probability.
The table below shows the mitigation handling strategies for the risks above the cut-off line.

S. No.	Risk	Mitigation
1	Technology will not meet the expectations	Keep upgrading the software from time to time according to the latest technology

4.3 Risk Exposure

The overall risk exposure (RE) is determined using the following relationship “RE= P *C” Where ‘**P**’ is the probability of occurrence for a risk, and ‘**C**’ is the cost to the project if the risk occurs

1. Risk Identification: There will be a lot of staff members that will be new at using the technology.
2. Risk Probability: 60% (likely)
3. Risk Impact: In order to train the staff members, we will be required to hire 2 professionals.

Total cost of the professionals and training would be around 50000Rs.

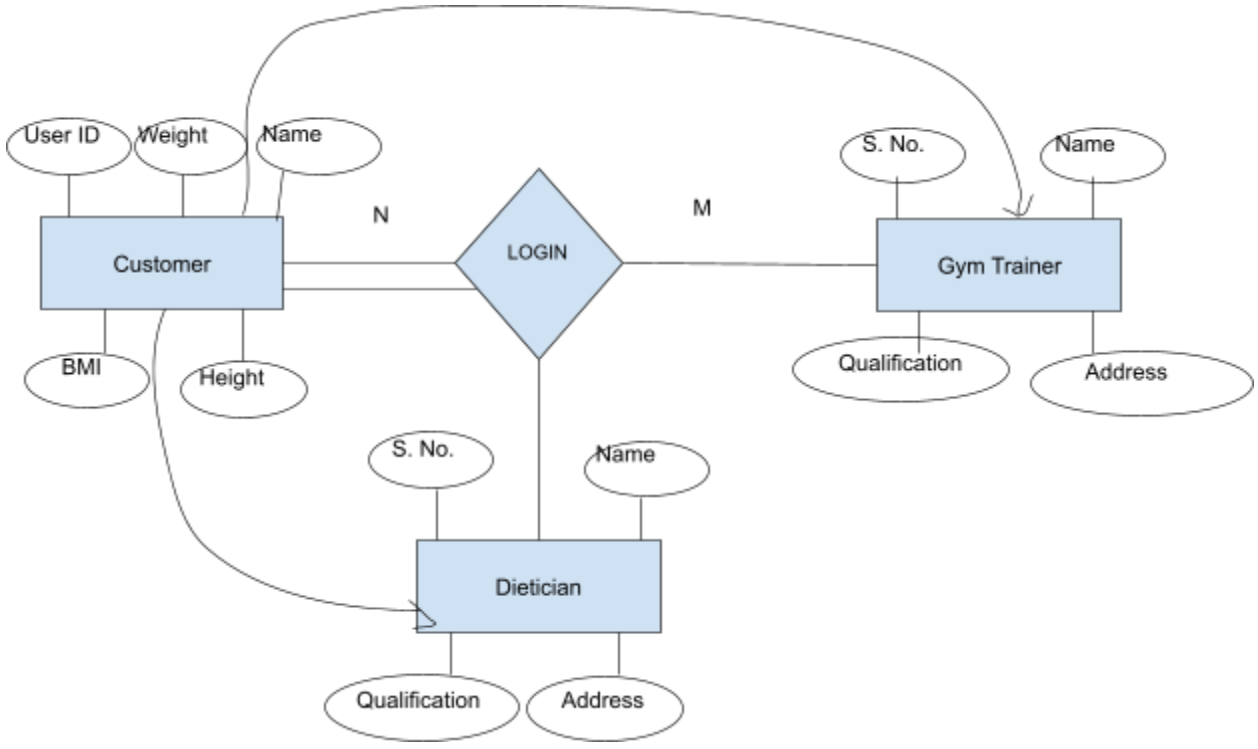
Risk Exposure: RE = P*C

RE = 0.60* 50,000

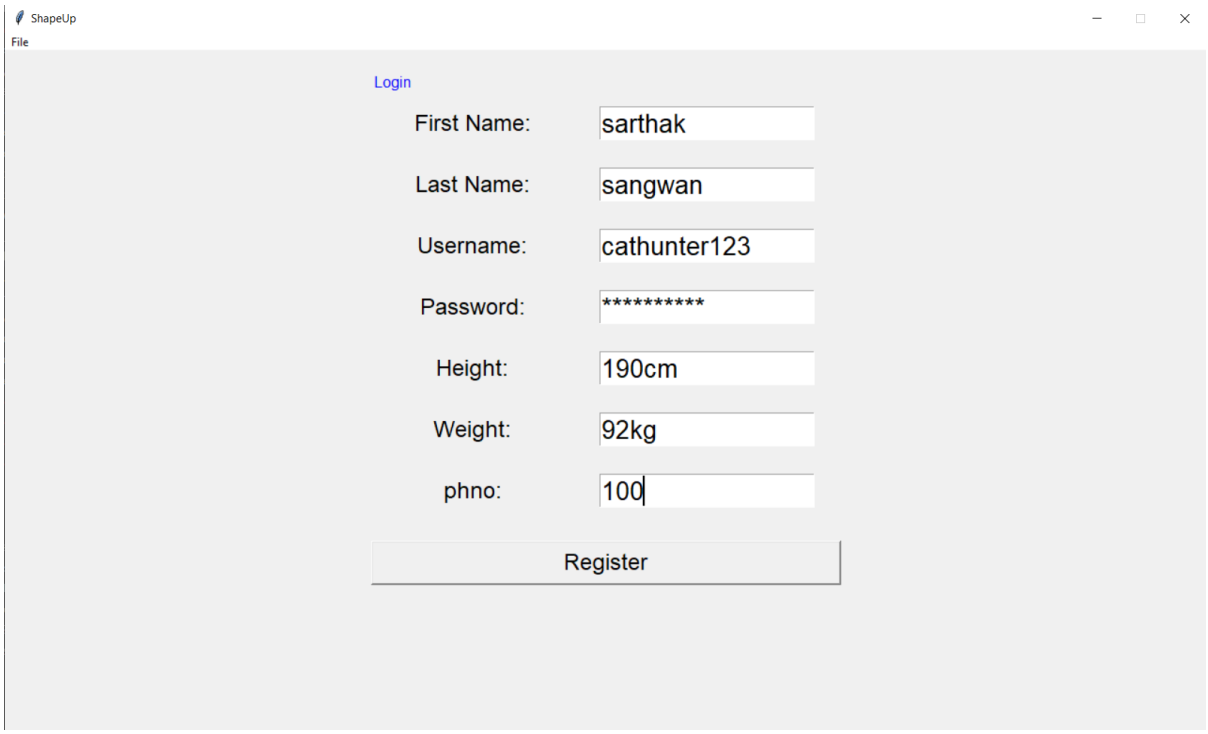
= 30,000Rs

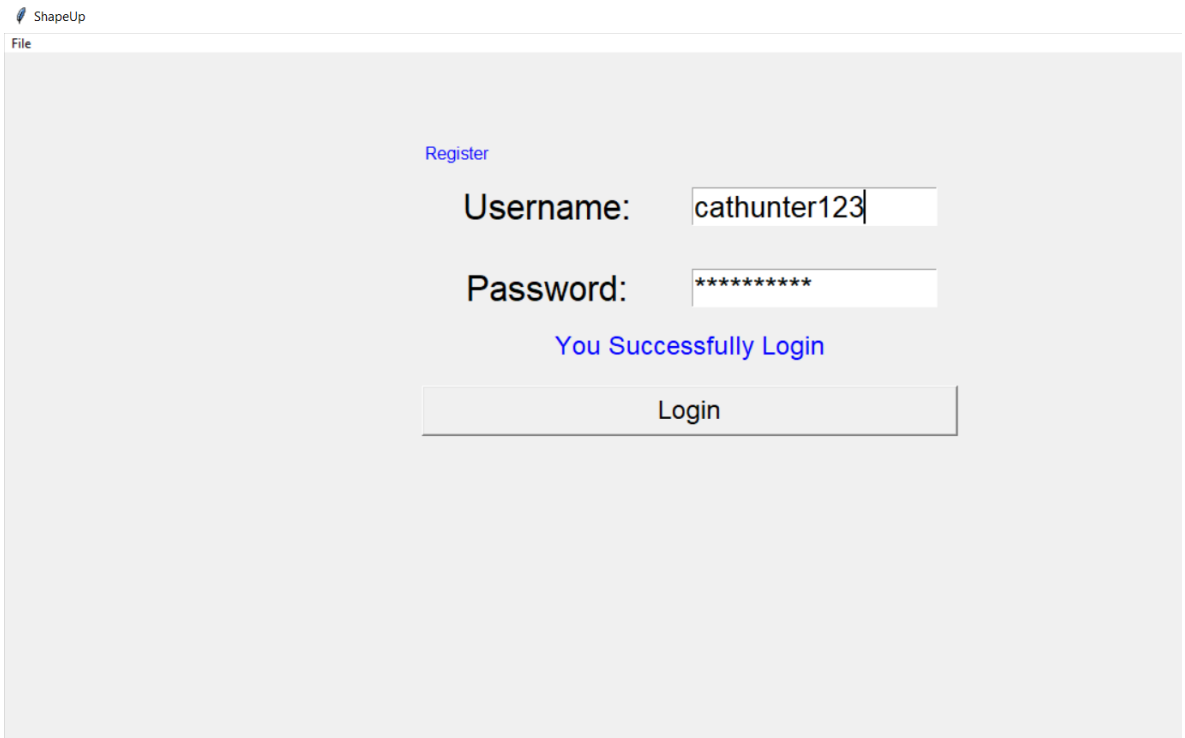
5. Design

5.1 System Design

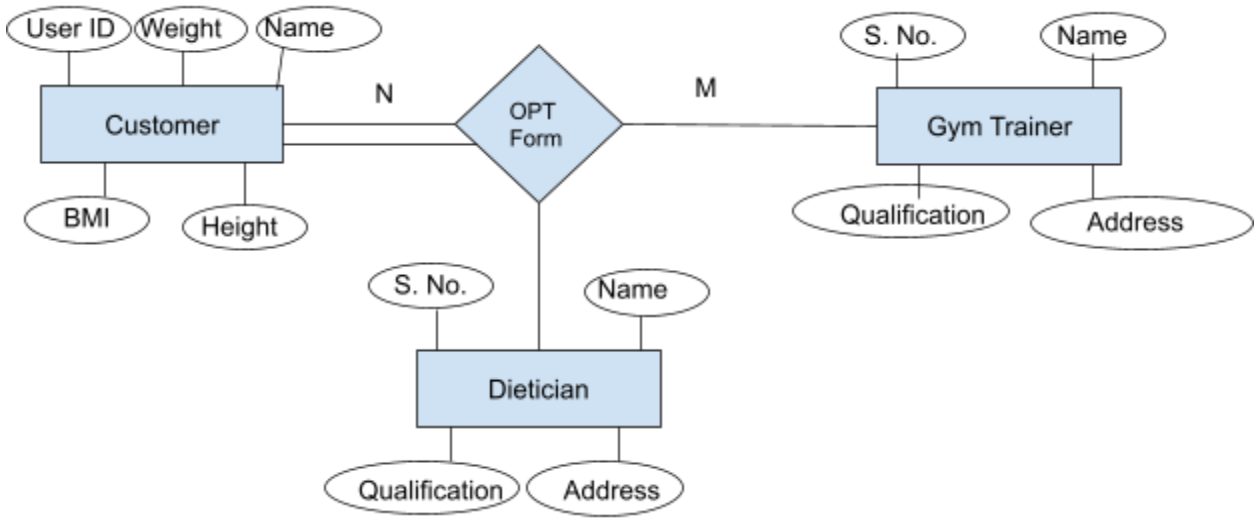


5.2 Screen Design





5.3 ER Diagram



5.4 Database Design

Customer DATABASE

S_no.	First Name	Last Name	Password	Height	Weight	Phone_number
1	sarthak	sangwan	********	190cm	92kg	100
2	bharath	vasishta	****	192cm	89kg	9021213211
3	mohtassim	rahman	*********	200cm	10kg	8329439022
4	nikhil	joshi	**********	100cm	100kg	3234837122
5	Yashvi	Choudhary	********	172cm	40kg	7483490223

Teacher DATABASE

S_no.	Name	Address	Qualifications_any	Charges/Month
1	sarthak	sangwan	Haryana Gold Medalist	60000rs
2	bharath	vasishta	Fastest Man on Earth(x7)	FREE
3	mohtassim	rahman	Olympic Deadlift Gold medalist	8000rs
4	nikhil	joshi	Long jump School Captain	100000rs
5	Yashvi	Choudhary	none	200rs

6. CODING:

```
from tkinter import *
import tkinter.messagebox as tkMessageBox
import sqlite3

root = Tk()
root.title("ShapeUp")

width = 1280
height = 720
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()
x = (screen_width/2) - (width/2)
y = (screen_height/2) - (height/2)
root.geometry("%dx%d+%d+%d" % (width, height, x, y))
root.resizable(0, 0)


#====================================VARIABLES=========================
============

FIRSTNAME = StringVar()
LASTNAME = StringVar()
PASSWORD = StringVar()
USERNAME = StringVar()
HEIGHT = StringVar()
WEIGHT = StringVar()
PHNO = StringVar()
#====================================METHODS===========================
============
def Database():
    global conn, cursor
    conn = sqlite3.connect("db_member.db")
    cursor = conn.cursor()
    cursor.execute("CREATE TABLE IF NOT EXISTS `member` (mem_id INTEGER
PRIMARY KEY AUTOINCREMENT NOT NULL, username TEXT, password TEXT,
firstname TEXT, lastname TEXT)")
```

```
def Exit():
    result = tkMessageBox.askquestion('System', 'Are you sure you want to
exit?', icon="warning")
    if result == 'yes':
        root.destroy()
        exit()


def LoginForm():
    global LoginFrame, lbl_result1
    LoginFrame = Frame(root)
    LoginFrame.pack(side=TOP, pady=80)
    lbl_username = Label(LoginFrame, text="Username:", font=('arial', 25),
bd=18)
    lbl_username.grid(row=1)
    lbl_password = Label(LoginFrame, text="Password:", font=('arial', 25),
bd=18)
    lbl_password.grid(row=2)
    lbl_result1 = Label(LoginFrame, text="", font=('arial', 18))
    lbl_result1.grid(row=3, columnspan=2)
    username = Entry(LoginFrame, font=('arial', 20),
textvariable=USERNAME, width=15)
    username.grid(row=1, column=1)
    password = Entry(LoginFrame, font=('arial', 20),
textvariable=PASSWORD, width=15, show="*")
    password.grid(row=2, column=1)
    btn_login = Button(LoginFrame, text="Login", font=('arial', 18),
width=35, command=Login)
    btn_login.grid(row=4, columnspan=2, pady=20)
    lbl_register = Label(LoginFrame, text="Register", fg="Blue",
font=('arial', 12))
    lbl_register.grid(row=0, sticky=W)
    lbl_register.bind('<Button-1>', ToggleToRegister)

def RegisterForm():
    global RegisterFrame, lbl_result2
    RegisterFrame = Frame(root)
    RegisterFrame.pack(side=TOP, pady=20)
```

```
    lbl_firstname = Label(RegisterFrame, text="First Name:",
font=('arial', 18), bd=18)
    lbl_firstname.grid(row=1)
    lbl_lastname = Label(RegisterFrame, text="Last Name:", font=('arial',
18), bd=18)
    lbl_lastname.grid(row=2)
    lbl_username = Label(RegisterFrame, text="Username:", font=('arial',
18), bd=18)
    lbl_username.grid(row=3)
    lbl_password = Label(RegisterFrame, text="Password:", font=('arial',
18), bd=18)
    lbl_password.grid(row=4)

    lbl_height = Label(RegisterFrame, text="Height:", font=('arial', 18),
bd=18)
    lbl_height.grid(row=5)

    lbl_weight= Label(RegisterFrame, text="Weight:", font=('arial', 18),
bd=18)
    lbl_weight.grid(row=6)

    lbl_phno= Label(RegisterFrame, text="phno:", font=('arial', 18),
bd=18)
    lbl_phno.grid(row=7)

    lbl_result2 = Label(RegisterFrame, text="", font=('arial', 18))
    lbl_result2.grid(row=8, columnspan=2)
    username = Entry(RegisterFrame, font=('arial', 20),
textvariable=USERNAME, width=15)
    username.grid(row=1, column=1)
    password = Entry(RegisterFrame, font=('arial', 20),
textvariable=PASSWORD, width=15, show="*")
    password.grid(row=4, column=1)
    firstname = Entry(RegisterFrame, font=('arial', 20),
textvariable=FIRSTNAME, width=15)
    firstname.grid(row=3, column=1)
    lastname = Entry(RegisterFrame, font=('arial', 20),
textvariable=LASTNAME, width=15)
    lastname.grid(row=2, column=1)
```

```
    height = Entry(RegisterFrame, font=('arial', 20), textvariable=HEIGHT,
width=15)
    height.grid(row=5, column=1)
    weight = Entry(RegisterFrame, font=('arial', 20), textvariable=WEIGHT,
width=15)
    weight.grid(row=6, column=1)
    phno = Entry(RegisterFrame, font=('arial', 20), textvariable=PHNO,
width=15)
    phno.grid(row=7, column=1)

    btn_login = Button(RegisterFrame, text="Register", font=('arial', 18),
width=35, command=Register)
    btn_login.grid(row=8, columnspan=2, pady=20)
    lbl_login = Label(RegisterFrame, text="Login", fg="Blue",
font=('arial', 12))
    lbl_login.grid(row=0, sticky=W)
    lbl_login.bind('<Button-1>', ToggleToLogin)

def ToggleToLogin(event=None):
    RegisterFrame.destroy()
    LoginForm()

def ToggleToRegister(event=None):
    LoginFrame.destroy()
    RegisterForm()

def Register():
    Database()
    if USERNAME.get == "" or PASSWORD.get() == "" or FIRSTNAME.get() == ""
or LASTNAME.get == "":
        lbl_result2.config(text="Please complete the required field!",
fg="orange")
    else:
        cursor.execute("SELECT * FROM `member` WHERE `username` = ?",
(USERNAME.get(),))
        if cursor.fetchone() is not None:
            lbl_result2.config(text="Username is already taken", fg="red")
        else:
```

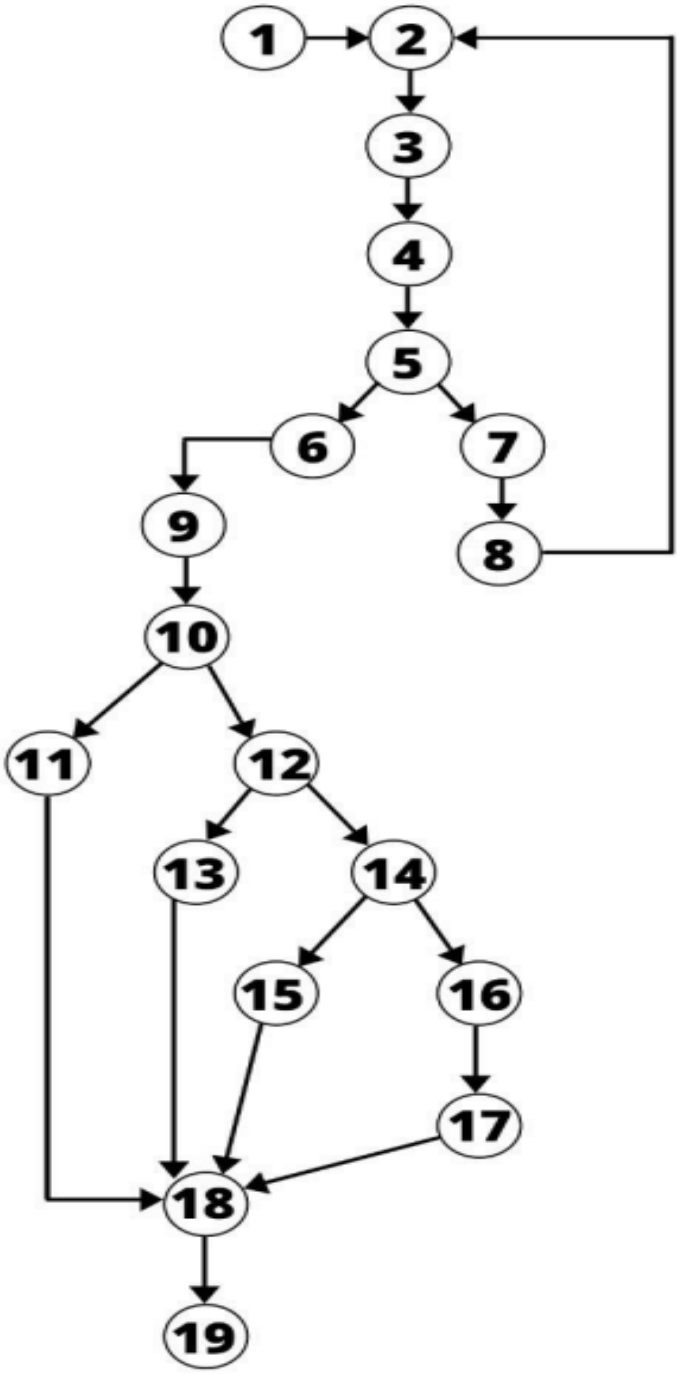
```
            cursor.execute("INSERT INTO `member` (username, password,
firstname, lastname) VALUES(?, ?, ?, ?)", (str(USERNAME.get()),
str(PASSWORD.get()), str(FIRSTNAME.get()), str(LASTNAME.get())))
            conn.commit()
            USERNAME.set("")
            PASSWORD.set("")
            FIRSTNAME.set("")
            LASTNAME.set("")
            lbl_result2.config(text="Successfully Created!", fg="black")
        cursor.close()
        conn.close()
def Login():
    Database()
    if USERNAME.get == "" or PASSWORD.get() == "":
        lbl_result1.config(text="Please complete the required field!",
fg="orange")
    else:
        cursor.execute("SELECT * FROM `member` WHERE `username` = ? and
`password` = ?", (USERNAME.get(), PASSWORD.get()))
        if cursor.fetchone() is not None:
            lbl_result1.config(text="You Successfully Login", fg="blue")
        else:
            lbl_result1.config(text="Invalid Username or password",
fg="red")
LoginForm()

#========================================MENUBAR
WIDGETS==================================
menubar = Menu(root)
filemenu = Menu(menubar, tearoff=0)
filemenu.add_command(label="Exit", command=Exit)
menubar.add_cascade(label="File", menu=filemenu)
root.config(menu=menubar)


#========================================INITIALIZATION==================
================
if __name__ == '__main__':
    root.mainloop()
```

Coding [Put code used for the Flowgraph, DD Path Graph etc of Testing section]

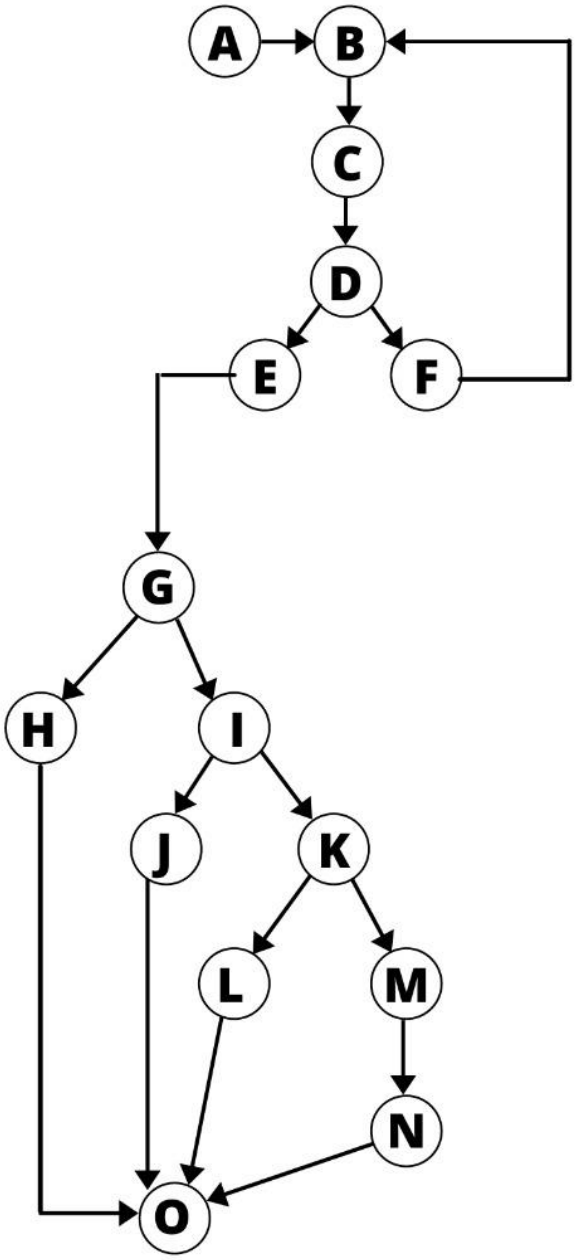
```
def bmi_calc():
  w=h=0 #1
  while(w==0 or h==0): #2
    h = float(input("Enter your height in meters: ")) #3
    w = float(input("Enter your weight in kg: ")) #4
    if(w!=0 and h!=0): #5
        break #6
    else: #7
        print("invalid entries") #8
    bmi = w/(h**2) #9

    if (bmi < 18.5): #10
     status = "UNDERWEIGHT" #11
    elif (bmi >= 18.5 and bmi <24.9): #12
     status = "HEALTHY" #13
    elif (bmi >= 24.9 and bmi < 30): #14
     status = "OVERWEIGHT" #15
    else: #16
     status = "OBESE" #17
    print("Your current BMI is: ", "{0:.2f}".format(bmi)) #18
    print("BMI STATUS: ",status) #19
```

7. Testing

7.1 Flowgraph



7.2 DD Path Graph



7.3 Cyclometric Complexity

CYCLOMATIC COMPLEXITY = E- N + 2P
E= NO. OF EDGES IN PROGRAM FLOW GRAPH
N= NO. OF NODES IN PROGRAM FLOW GRAPH
P= NO. OF CONNECTED COMPONENTS
Here,
E= 22
N= 19
P= 1
Hence, CYCLOMATIC COMPLEXITY = 22 - 19 + 2 = 5
OR
CYCLOMATIC COMPLEXITY = π + 1
π = NO. OF PREDICATE NODES
Here, π = 4
Hence, CYCLOMATIC COMPLEXITY = 4 + 1 = 5

7.3.1 Independent Paths

Path 1 : A-B-C-D-E-G-H-O
Path 2 : A-B-C-D-E-G-I-J-O
Path 3 : A-B-C-D-E-G-I-K-L-O
Path 4 : A-B-C-D-E-G-I-K-M-N-O
*Path 5 : A-B-C-D-F-B-C-D-E-G-H-O

7.4 Test Cases

7.4.1 Test Case 1

Input Value: Height = entered height
Weight = entered weight

When height is equal to the entered height,
weight is equal to the entered weight, **at line 5**.
Then BMI will be calculated on **line 9**.
if the BMI > 30, then status = “OBESE” and then the program will go to line 18 and the output will be printed.

7.4.2 Test Case 2

Input Value: Height = 0
or
Weight = 0

When height=0 or weight =0, at **line 5**.
then at **line 6**, the while loop will continue to execute until both get non zero values as their input. Then the program will go as desired.

7.4.3 Test Case 3

Input Value: Height = entered height
Weight = entered weight

When height is equal to the entered height, weight is equal to the entered weight, at **line 5**.
Then BMI will be calculated on **line 9**.
If the BMI <18.5, then status = “UNDERWEIGHT” and then the program will go to line 18 and the output will be printed.

7.4.4 Test Case 4

Input Value: Height = entered height
Weight = entered weight

When height is equal to the entered height, weight is equal to the entered weight, at **line 5**.
Then BMI will be calculated on **line 9**.
If the BMI >= 18.5 and BMI < 24.9, then status = “HEALTHY” and then the program will go to line 18 and the output will be printed.

7.4.5 Test Case 5

Input Value: Height = entered height
Weight = entered weight

When height is equal to the entered height, weight is equal to the entered weight, at **line 5**.
Then BMI will be calculated on **line 9**.
If the BMI >= 24.9 and BMI < 30, then status = “OVERWEIGHT” and then the program will go to line 18 and the output will be printed.

8. Future Scope

This project can be further updated to have features such as:

- Sleep tracking
- Water Intake check and reminder
- Step count recorder

9. References

Greekforgeeks.
Youtube/ sourceCodester
W3schools

10. Appendix

Python Project: Tetris

Project Name: Tetris
Course teacher: Mona Adlakha
Course: B.Sc (Hons) Computer Science
Semester: III
Name: Md Mohtasim Rahman & Iriventi Bharath Vasishta
College Roll No: CSC/20/39 & CSC/20/36
Examination Roll No.: 20059570032 & 20059570029
Subject: Python

Code:

```
import pygame
import random
import time
from datetime import datetime
"""
10 x 20 square grid
shapes: S, Z, I, O, J, L, T
represented in order by 0 - 6
"""

# initialise pygame with fonts
pygame.font.init()

# GLOBALS VARS
s_width = 800
s_height = 700
play_width = 300  # meaning 300 // 10 = 30 width per block
play_height = 600  # meaning 600 // 20 = 20 height per blo ck
block_size = 30

top_left_x = (s_width - play_width) // 2
top_left_y = s_height - play_height


# SHAPE FORMATS

S = [['.....',
      '.....',
      '..00.',
      '.00..',
      '.....'],
     ['.....',
      '..0..',
      '..00.',
      '...0.',
      '.....']]

Z = [['.....',
      '.....',
```

```
      '.00..',
      '..00.',
      '.....'],
     ['.....',
      '..0..',
      '.00..',
      '.0...',
      '.....']]

I = [['..0..',
      '..0..',
      '..0..',
      '..0..',
      '.....'],
     ['.....',
      '0000.',
      '.....',
      '.....',
      '.....']]

O = [['.....',
      '.....',
      '.00..',
      '.00..',
      '.....']]

J = [['.....',
      '.0...',
      '.000.',
      '.....',
      '.....'],
     ['.....',
      '..00.',
      '..0..',
      '..0..',
      '.....'],
     ['.....',
      '.....',
      '.000.',
      '...0.',
```

```
        '.....'],
       ['.....',
        '..0..',
        '..0..',
        '.00..',
        '.....']]

L = [['.....',
        '...0.',
        '.000.',
        '.....',
        '.....'],
       ['.....',
        '..0..',
        '..0..',
        '..00.',
        '.....'],
       ['.....',
        '.....',
        '.000.',
        '.0...',
        '.....'],
       ['.....',
        '.00..',
        '..0..',
        '..0..',
        '.....']]

T = [['.....',
        '..0..',
        '.000.',
        '.....',
        '.....'],
       ['.....',
        '..0..',
        '..00.',
        '..0..',
        '.....'],
       ['.....',
        '.....',
```

```
      '.000.',
      '..0..',
      '.....'],
     ['.....',
      '..0..',
      '.00..',
      '..0..',
      '.....']]

shapes = [S, Z, I, O, J, L, T]
shape_colors = [(0, 255, 0), (255, 0, 0), (0, 255, 255), (255, 255, 0), (255, 165, 0), (0, 0,
255), (128, 0, 128)]
# index 0 - 6 represent shape


class Piece(object):
    rows = 20  # y
    columns = 10  # x

    def __init__(self, column, row, shape):
        self.x = column
        self.y = row
        self.shape = shape
        self.color = shape_colors[shapes.index(shape)]
        self.rotation = 0  # number from 0-3


def create_grid(locked_positions={}):
    #create the main game grid
    grid = [[(0,0,0) for x in range(10)] for x in range(20)] #each cell is represented by a 0

    for i in range(len(grid)):
        for j in range(len(grid[i])):
            if (j,i) in locked_positions:
                c = locked_positions[(j,i)]
                grid[i][j] = c
    return grid


#return the position on the grid where the shape must be drawn by pygame
```

```
def convert_shape_format(shape):
    positions = []
    format = shape.shape[shape.rotation % len(shape.shape)]

    for i, line in enumerate(format):
        row = list(line)
        for j, column in enumerate(row):
            if column == '0':
                positions.append((shape.x + j, shape.y + i))

    for i, pos in enumerate(positions):
        positions[i] = (pos[0] - 2, pos[1] - 4)

    return positions

#checks weather a given block has reached solid surface
def valid_space(shape, grid):
    accepted_positions = [[(j, i) for j in range(10) if grid[i][j] == (0,0,0)] for i in range(20)]
    accepted_positions = [j for sub in accepted_positions for j in sub]
    formatted = convert_shape_format(shape)

    for pos in formatted:
        if pos not in accepted_positions:
            if pos[1] > -1:
                return False

    return True


def check_lost(positions):
    for pos in positions:
        x, y = pos
        if y < 1:
            return True
    return False


#create a random shape
def get_shape():
    global shapes, shape_colors
```

```
    return Piece(5, 0, random.choice(shapes))


def draw_text_middle(text, size, color, surface):
    font = pygame.font.SysFont('comicsans', size, bold=True)
    label = font.render(text, 1, color)

    surface.blit(label, (top_left_x + play_width/2 - (label.get_width() / 2), top_left_y +
play_height/2 - label.get_height()/2))

#blit the root level grid on the surface
def draw_grid(surface, row, col):
    sx = top_left_x
    sy = top_left_y
    for i in range(row):
        pygame.draw.line(surface, (128,128,128), (sx, sy+ i*30), (sx + play_width, sy + i *
30))  # horizontal lines
        for j in range(col):
            pygame.draw.line(surface, (128,128,128), (sx + j * 30, sy), (sx + j * 30, sy +
play_height))  # vertical lines


def clear_rows(grid, locked):
    # need to see if row is clear the shift every other row above down one

    inc = 0
    for i in range(len(grid)-1,-1,-1):
        row = grid[i]
        if (0, 0, 0) not in row:
            inc += 1
            # add positions to remove from locked
            ind = i
            for j in range(len(row)):
                try:
                    del locked[(j, i)]
                except:
                    continue
    if inc > 0:
        for key in sorted(list(locked), key=lambda x: x[1])[::-1]:
```

```
            x, y = key
            if y < ind:
                newKey = (x, y + inc)
                locked[newKey] = locked.pop(key)

#update the `NEXT` column
def draw_next_shape(shape, surface):
    font = pygame.font.SysFont('comicsans', 30)
    label = font.render('Next Shape', 1, (255,255,255))

    sx = top_left_x + play_width + 50
    sy = top_left_y + play_height/2 - 100
    format = shape.shape[shape.rotation % len(shape.shape)]

    for i, line in enumerate(format):
        row = list(line)
        for j, column in enumerate(row):
            if column == '0':
                pygame.draw.rect(surface, shape.color, (sx + j*30, sy + i*30, 30, 30), 0)

    surface.blit(label, (sx + 10, sy- 30))

#get the previous high score from REGISTER
def get_high_score():
    high_score = 0
    register = open("REGISTER", "a+").readlines()
    for registry in register:
        high_score = max(high_score, float(registry.split(" ")[-2])*100)
    return int(high_score)

#draw the main window
def draw_window(surface):
    surface.fill((0,0,0))
    # Tetris Title
    font = pygame.font.SysFont('comicsans', 60)
    label = font.render('TETRIS', 1, (255,255,255))
    font = pygame.font.SysFont('comicsans', 30)
    high_score = get_high_score()
    high_score_label = font.render('HIGH SCORE', 1, (255,255,255))
    player_score_label = font.render('SCORE', 1, (255,255,255))
```

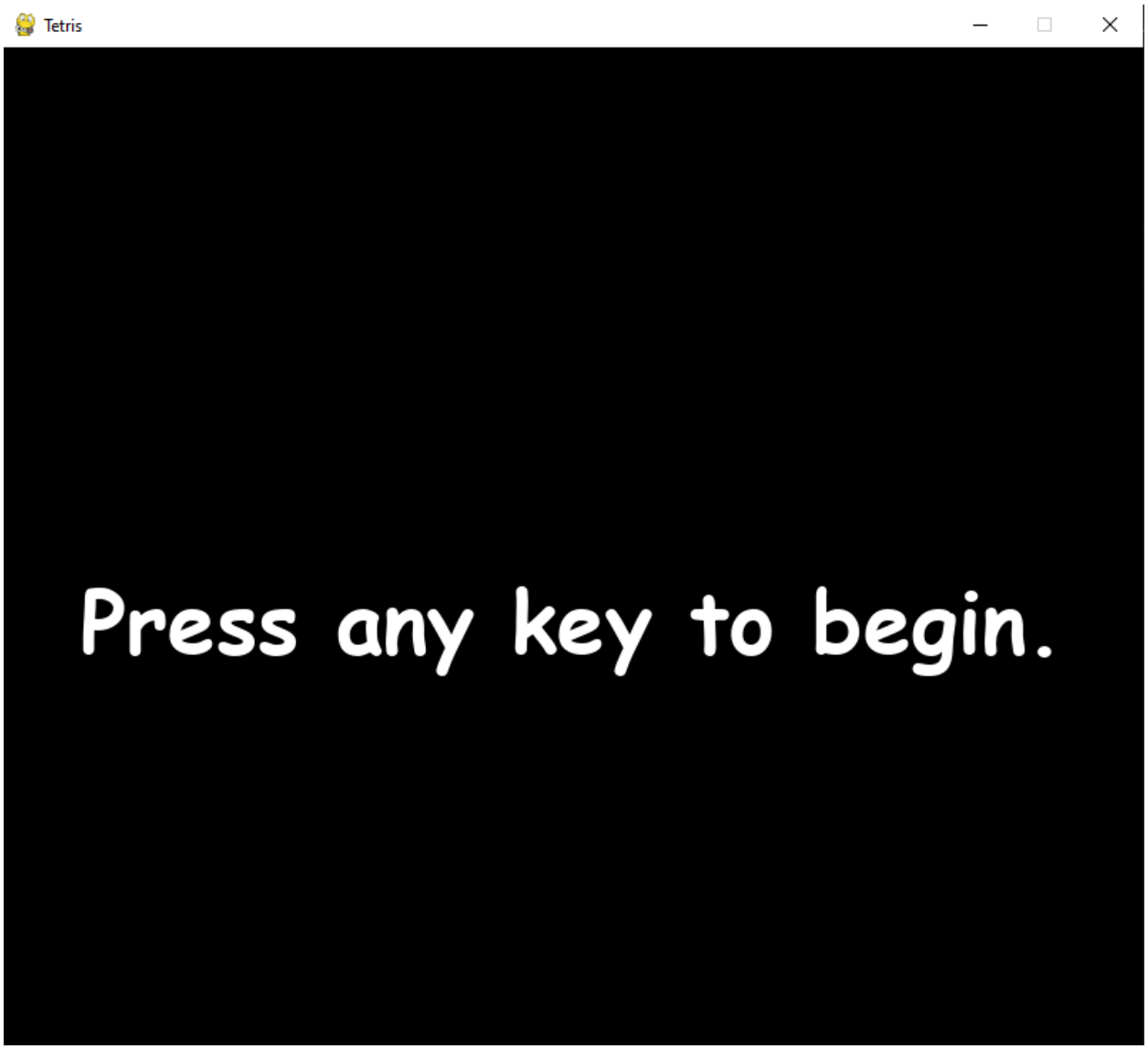
```
    high_score_label_value = font.render(str(high_score), 1, (255,255,255))
    surface.blit(label, (top_left_x + play_width / 2 - (label.get_width() / 2), 30))
    surface.blit(high_score_label, (30, 60))
    surface.blit(player_score_label, (30, 150))
    surface.blit(high_score_label_value, (30, 90))
    for i in range(len(grid)):
        for j in range(len(grid[i])):
            pygame.draw.rect(surface, grid[i][j], (top_left_x + j* 30, top_left_y + i * 30, 30,
30), 0)

    # draw grid and border
    draw_grid(surface, 20, 10)
    pygame.draw.rect(surface, (255, 0, 0), (top_left_x, top_left_y, play_width,
play_height), 5)
    # pygame.display.update()

#update the score each second
def update_score(_begin, surface):
    font = pygame.font.SysFont('comicsans', 30)
    score = font.render(str(round(time.time()-_begin)*100), 1, (255,255,255))
    surface.blit(score, (30, 180))

def main():
    global grid

    locked_positions = {}  # (x,y):(255,0,0)
    grid = create_grid(locked_positions)

    change_piece = False
    run = True
    current_piece = get_shape()
    next_piece = get_shape()
    clock = pygame.time.Clock()
    fall_time = 0
    _begin = time.time()
    while run:
        fall_speed = 0.27

        grid = create_grid(locked_positions)
        fall_time += clock.get_rawtime()
```

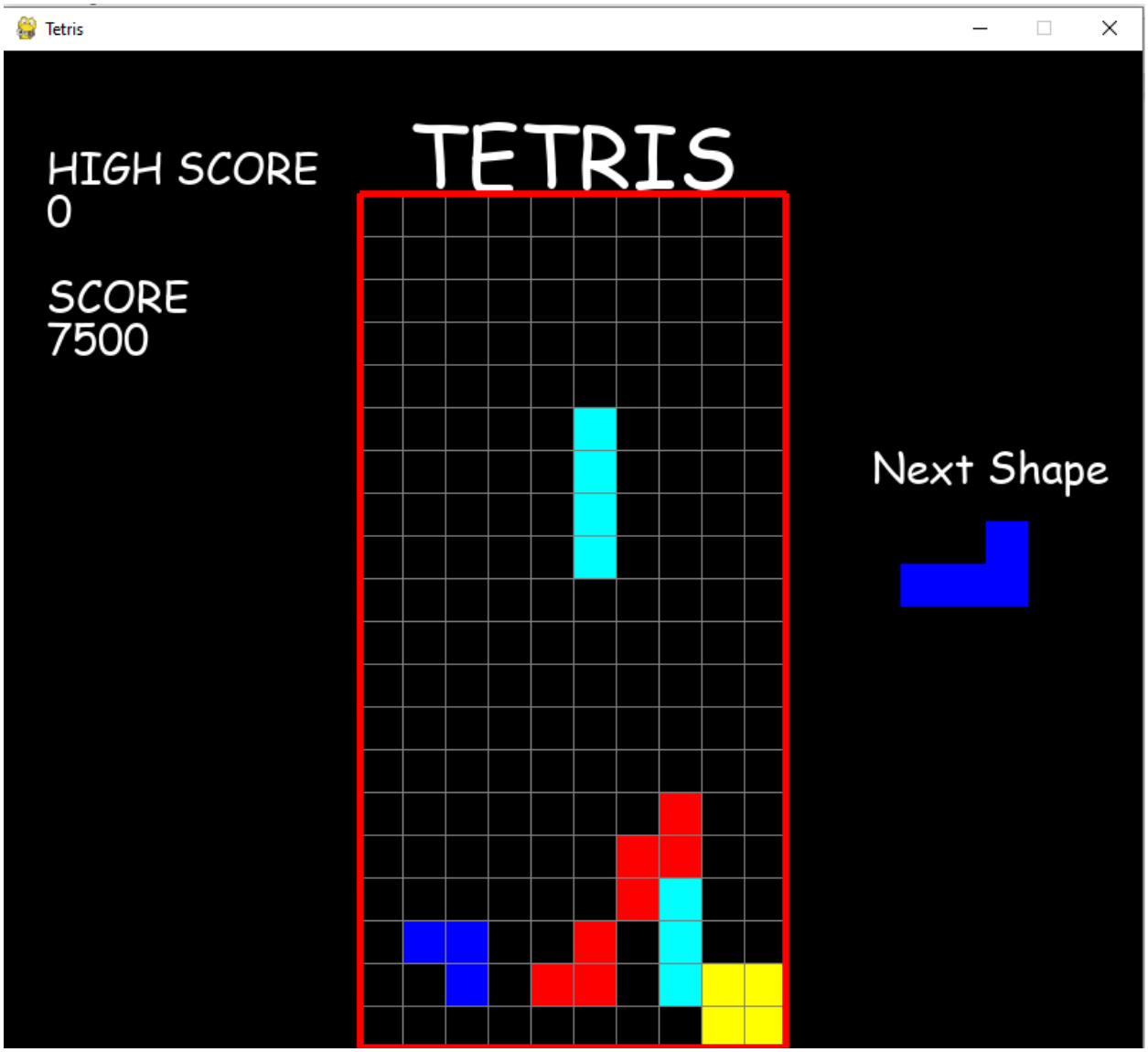
```
        clock.tick()

        # PIECE FALLING CODE
        if fall_time/1000 >= fall_speed:
            fall_time = 0
            current_piece.y += 1
            if not (valid_space(current_piece, grid)) and current_piece.y > 0:
                current_piece.y -= 1
                change_piece = True

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False
                update_registry(_begin)
                pygame.display.quit()
                quit()
                pygame.quit()

            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_LEFT:
                    current_piece.x -= 1
                    if not valid_space(current_piece, grid):
                        current_piece.x += 1

                elif event.key == pygame.K_RIGHT:
                    current_piece.x += 1
                    if not valid_space(current_piece, grid):
                        current_piece.x -= 1
                elif event.key == pygame.K_UP:
                    # rotate shape
                    current_piece.rotation = current_piece.rotation + 1 %
len(current_piece.shape)
                    if not valid_space(current_piece, grid):
                        current_piece.rotation = current_piece.rotation - 1 %
len(current_piece.shape)

                if event.key == pygame.K_DOWN:
                    # move shape down
                    current_piece.y += 1
                    if not valid_space(current_piece, grid):
```

```
                current_piece.y -= 1

            '''if event.key == pygame.K_SPACE:
                while valid_space(current_piece, grid):
                    current_piece.y += 1
                current_piece.y -= 1
                print(convert_shape_format(current_piece))'''  # todo fix

        shape_pos = convert_shape_format(current_piece)

        # add piece to the grid for drawing
        for i in range(len(shape_pos)):
            x, y = shape_pos[i]
            if y > -1:
                grid[y][x] = current_piece.color

        # IF PIECE HIT GROUND
        if change_piece:
            for pos in shape_pos:
                p = (pos[0], pos[1])
                locked_positions[p] = current_piece.color
            current_piece = next_piece
            next_piece = get_shape()
            change_piece = False

            # call four times to check for multiple clear rows
            clear_rows(grid, locked_positions)

        draw_window(win)
        draw_next_shape(next_piece, win)
        update_score(_begin, win)
        pygame.display.update()

        # Check if user lost
        if check_lost(locked_positions):
            run = False
            update_registry(_begin)

    draw_text_middle("You Lost", 40, (255,255,255), win)
    pygame.display.update()
```

```
    pygame.time.delay(2000)

def update_registry(time_start):
    date_time = datetime.now().strftime("%d/%m/%Y %H:%M:%S")
    print(date_time)
    with open("REGISTER", "a") as _r:
        _r.write(f"{date_time}\tPLAYED FOR {round(time.time()-time_start,0)} seconds\n")
        _r.close()

def main_menu():
    run = True
    while run:
        win.fill((0,0,0))
        draw_text_middle('Press any key to begin.', 60, (255, 255, 255), win)
        pygame.display.update()
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False
            if event.type == pygame.KEYDOWN:
                main()
    pygame.quit()


win = pygame.display.set_mode((s_width, s_height))
pygame.display.set_caption('Tetris')

if __name__ == "__main__":
    main_menu()      # start game
```

Output:





tetris.py REGISTER

```
REGISTER
1   22/11/2021 12:13:27 PLAYED FOR 93.0 seconds
2   22/11/2021 12:13:33 PLAYED FOR 4.0 seconds
3
```