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.

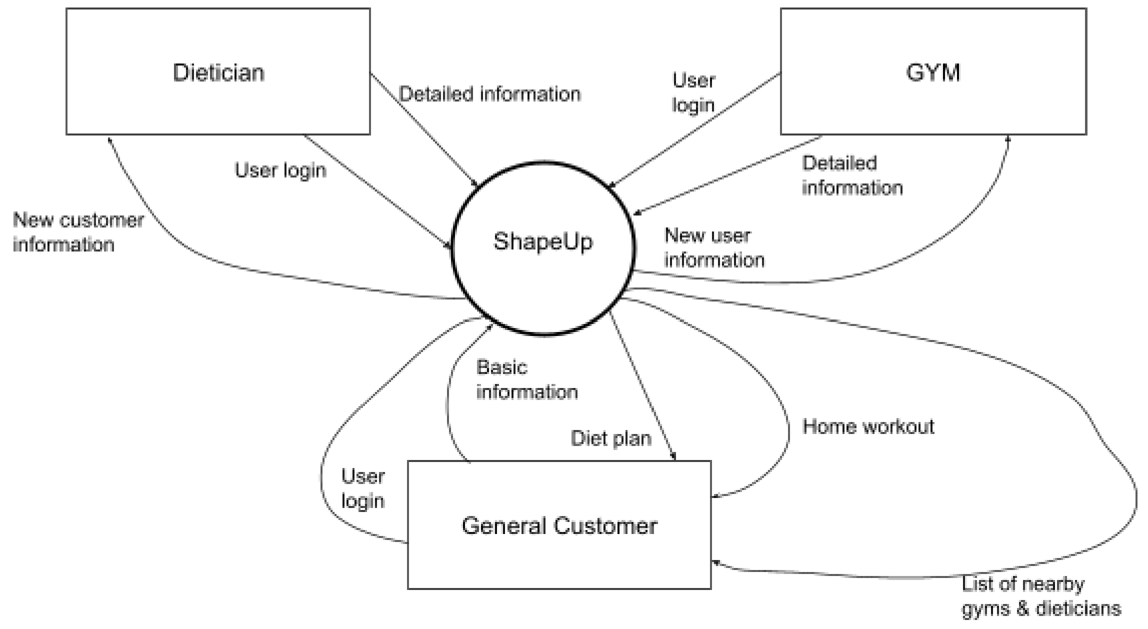
- **User Backup:** The user will not necessarily need to use one particular device to access the app. The user will only need to login from any device with their data being retrieved from our online servers.

1.5 Design Constraints

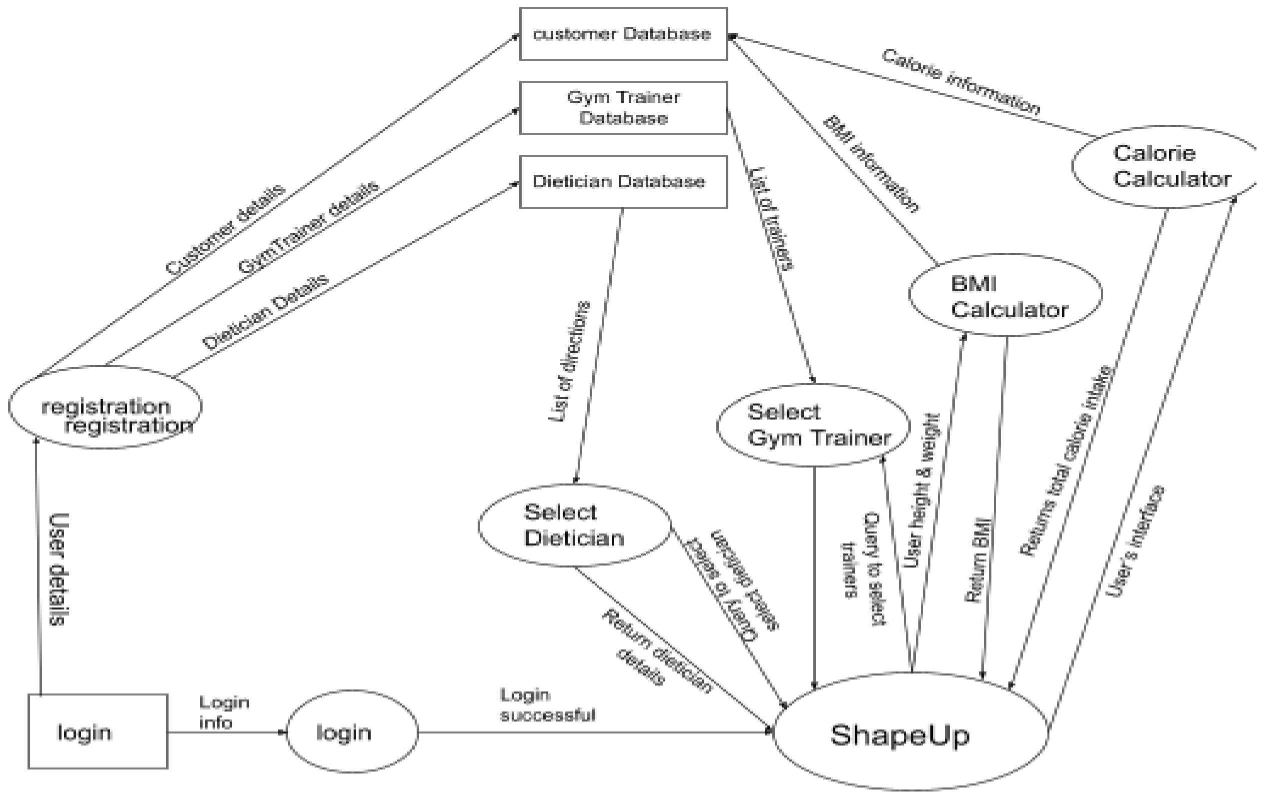
Python doesn’t support proper database management within, user needs to import databse from third party applications like mysql, XML and etc.

1.6 Data Flow Diagram

1.6.1 Context Level DFD



1.6.2 Level 1 DFD

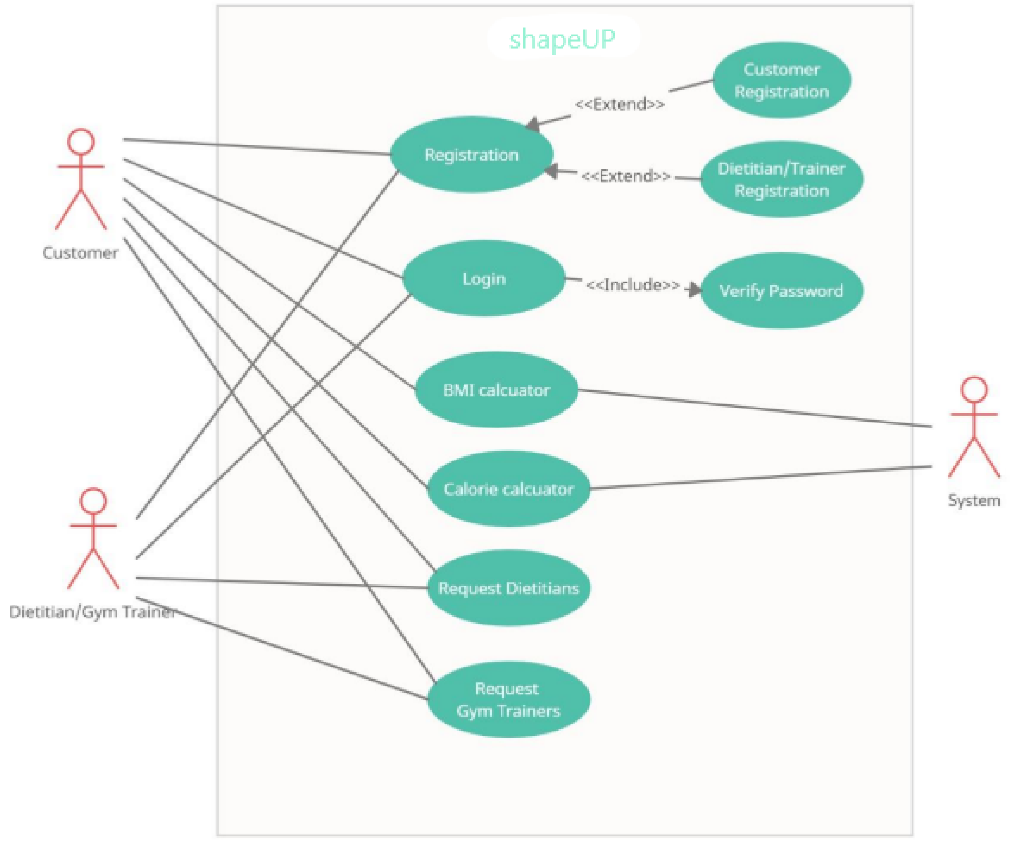


1.7 Data Dictionary

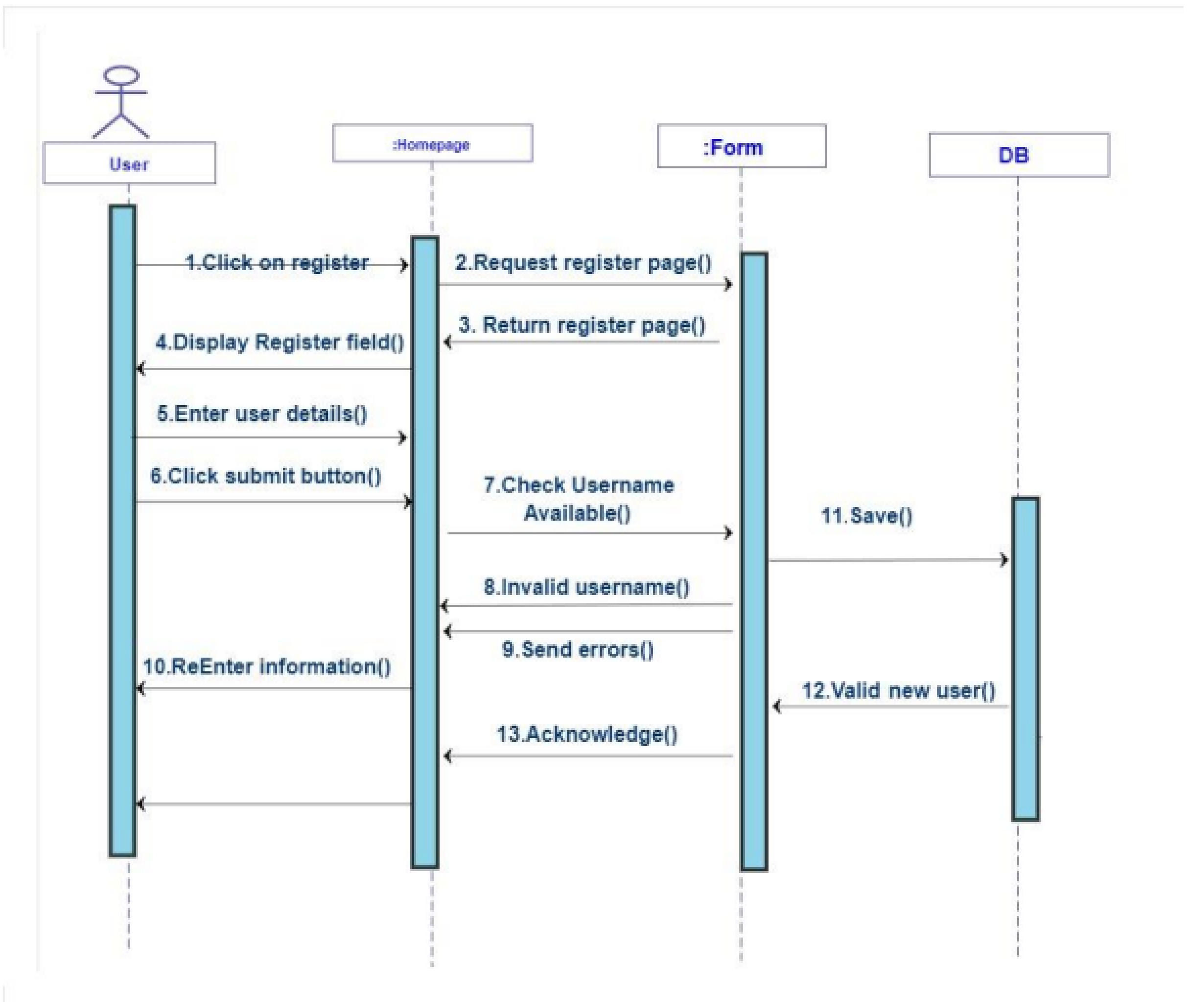
S. No.	Data	Description
1	Trainer’s Details	Name + Qualifications + Address of Practise + Fee
2	Dietician’s Details	Name + Qualifications + Address of Practise + Fee
3	Customer Register	Name + Gender + Age + Address + Email id + Phone Number
4	Customer Login	Username (Name) + Password + Registered Number

1.8 Use Cases and Use Case Diagram

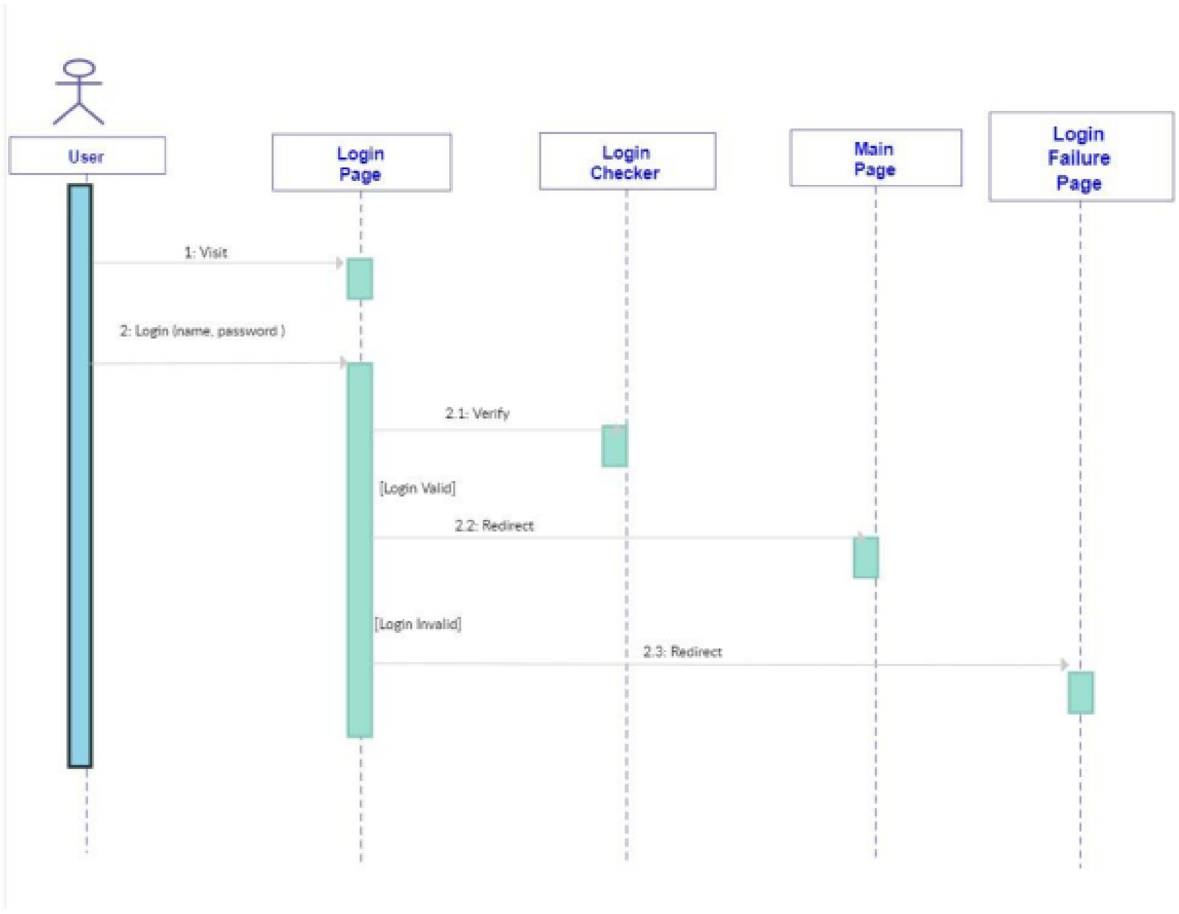
Use Case Diagram



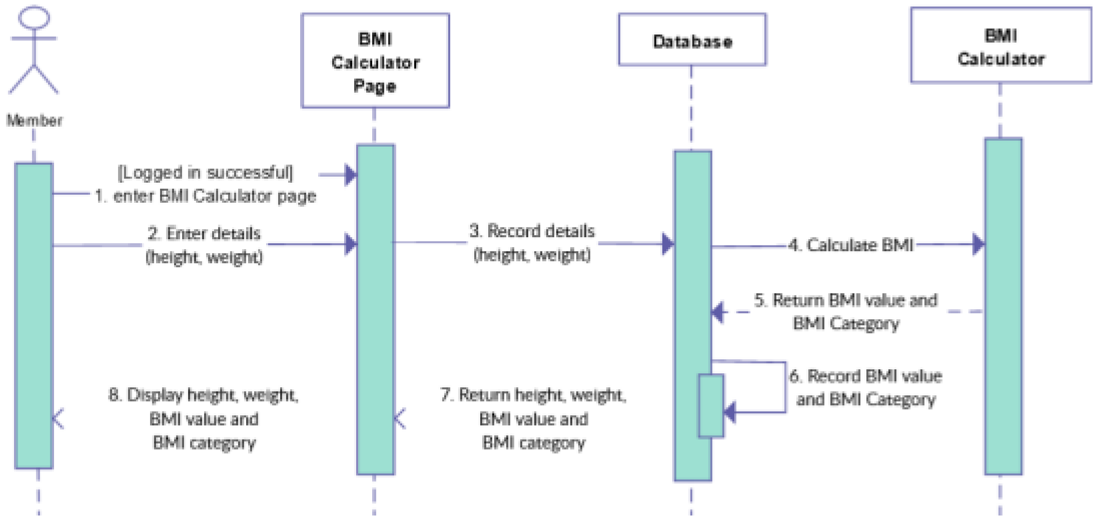
Sequential Diagram



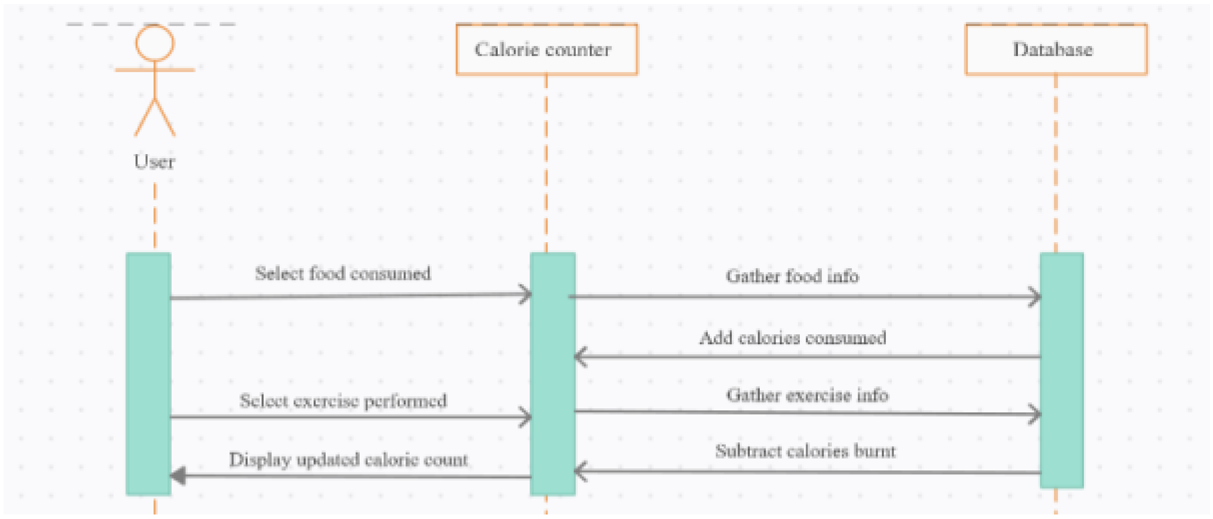
<u>Sequential diagram for registration</u>



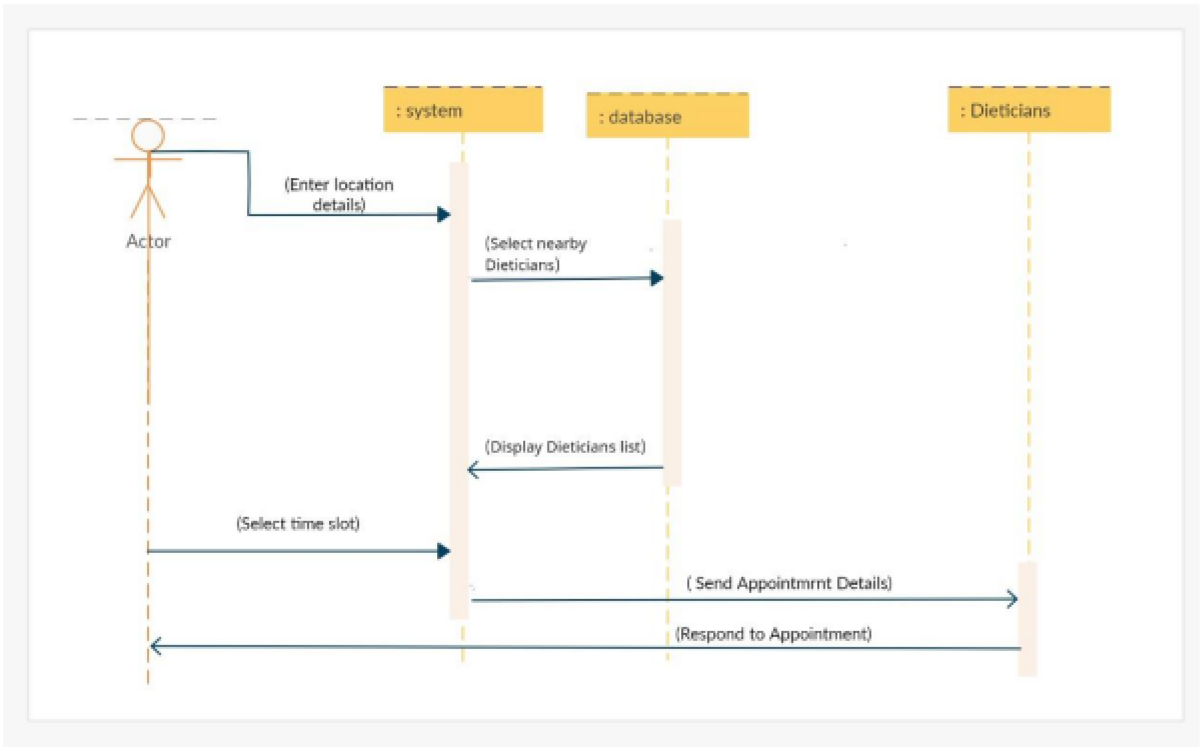
Sequential diagram for login


Member
BMI Calculator Page
Database
BMI Calculator
[Logged in successful]
1. enter BMI Calculator page
2. Enter details (height, weight)
3. Record details (height, weight)
4. Calculate BMI
5. Return BMI value and BMI Category
6. Record BMI value and BMI Category
7. Return height, weight, BMI value and BMI category
8. Display height, weight, BMI value and BMI category

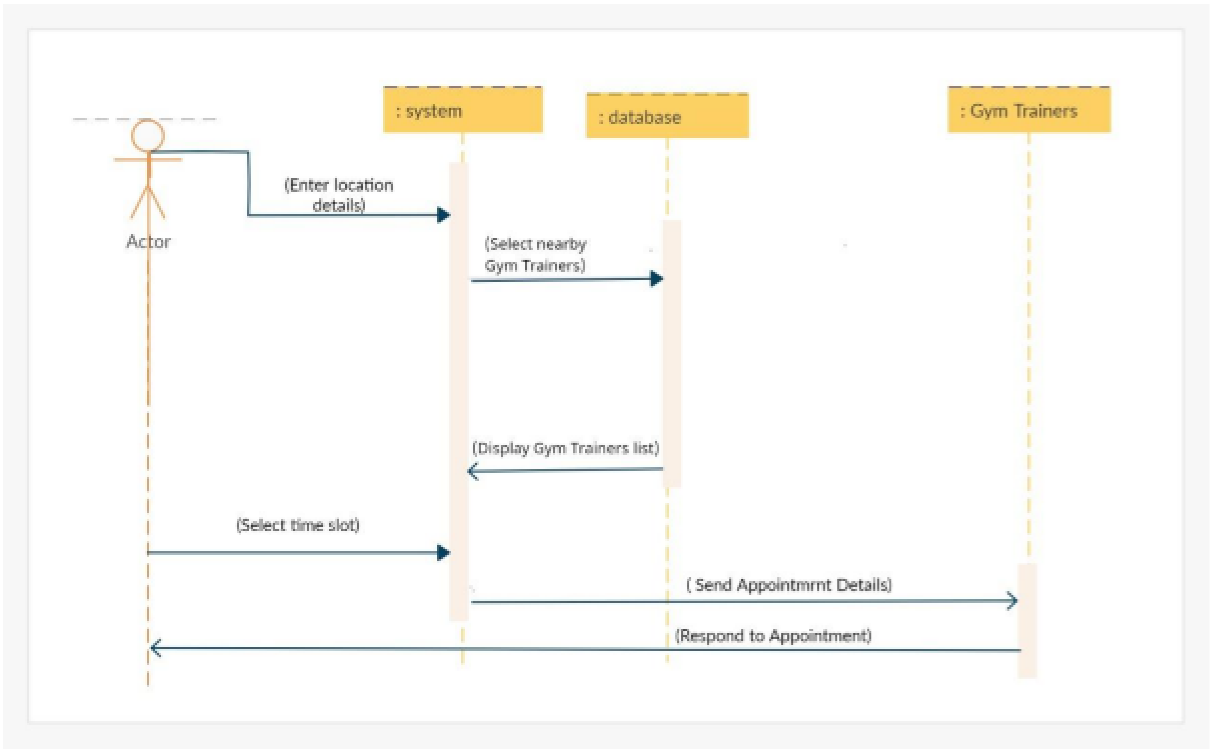
Sequential diagram for BMI calculator



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**

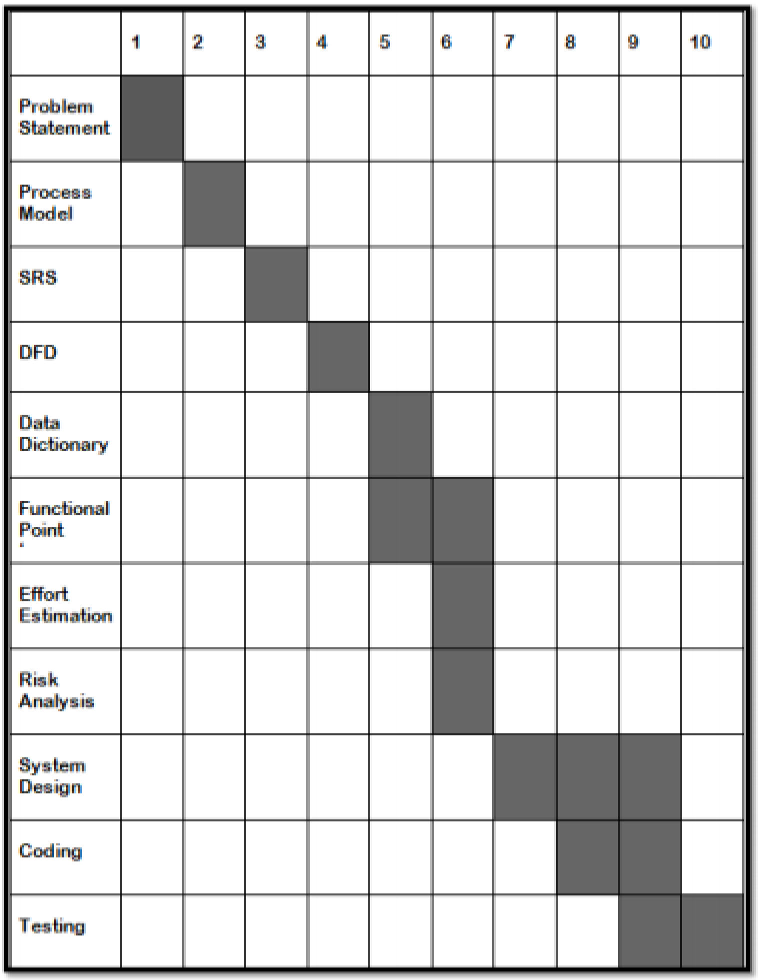
Complexity adjustment factor = Value Adjustment Factor (VAF)
= [0.6 + 0.01 * Sum of F(x)]
= [0.65 + 0.01*41]
= 1.05
Adjusted FP Count = Unadjusted FP count * VAF
= 81 * 1.05
= 85.05
Hence, Adjusted FP count or FP(estimated) = 85 (approx.)

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



	1	2	3	4	5	6	7	8	9	10
Problem Statement	▨									
Process Model		▨								
SRS			▨							
DFD				▨						
Data Dictionary					▨					
Functional Point					▨	▨				
Effort Estimation						▨				
Risk Analysis						▨				
System Design							▨	▨	▨	
Coding								▨	▨	
Testing									▨	▨

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

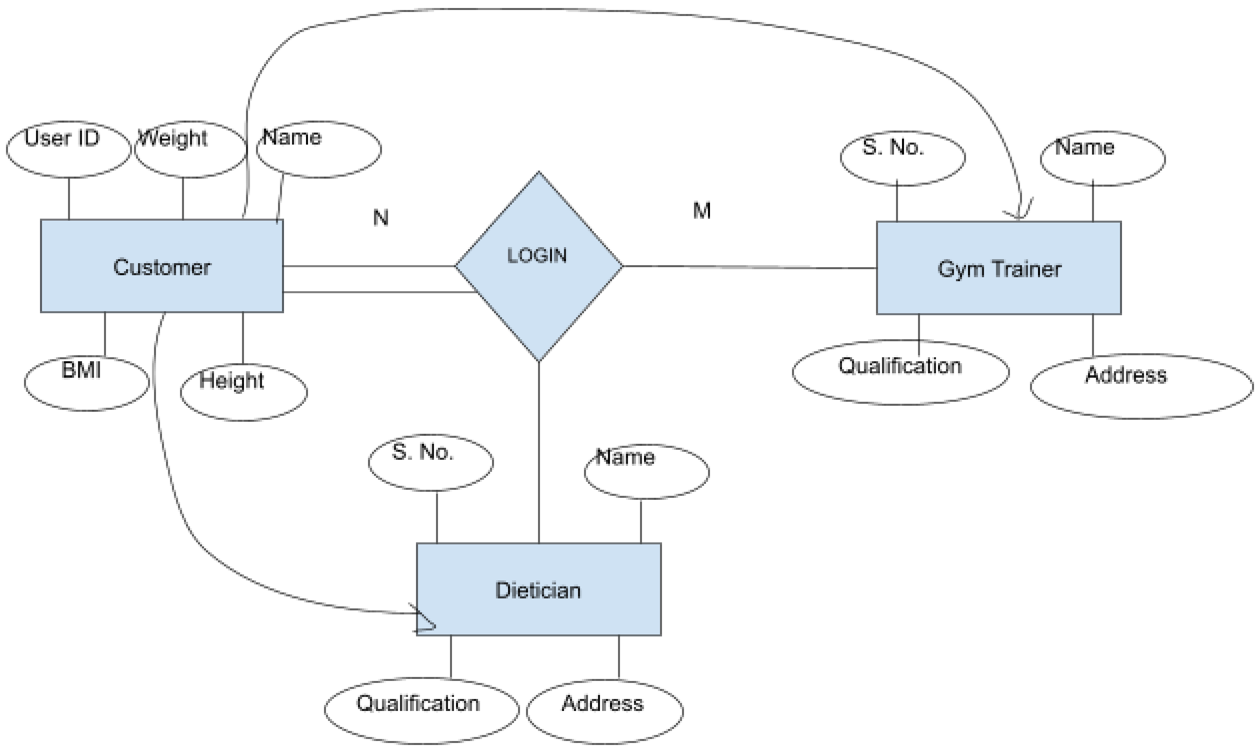
The overall risk exposure (RE) is determined using the following relationship "RE= P *C" Where '**P**' is the probability of occurrence for a risk, and '**C**' is the cost to the project if the risk occurs

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.

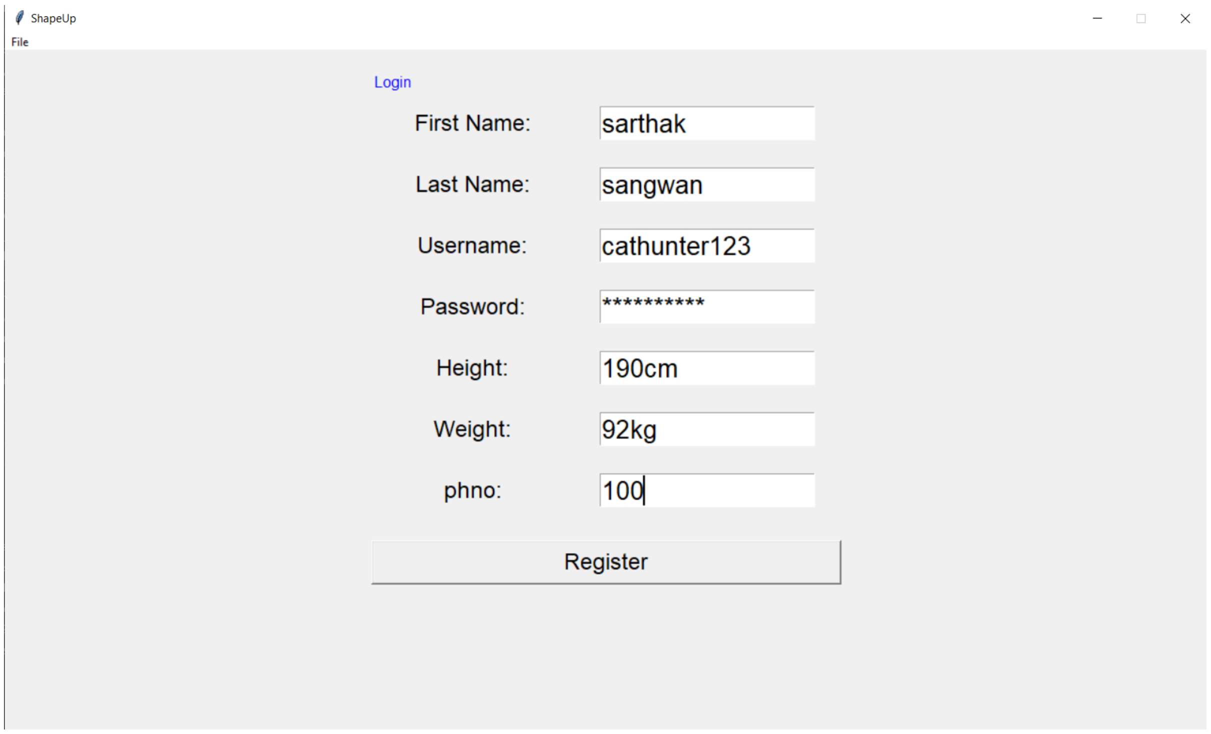
Total cost of the professionals and training would be around 50000Rs.
Risk Exposure: RE = P*C
RE = 0.60* 50,000
= 30,000Rs

5. Design

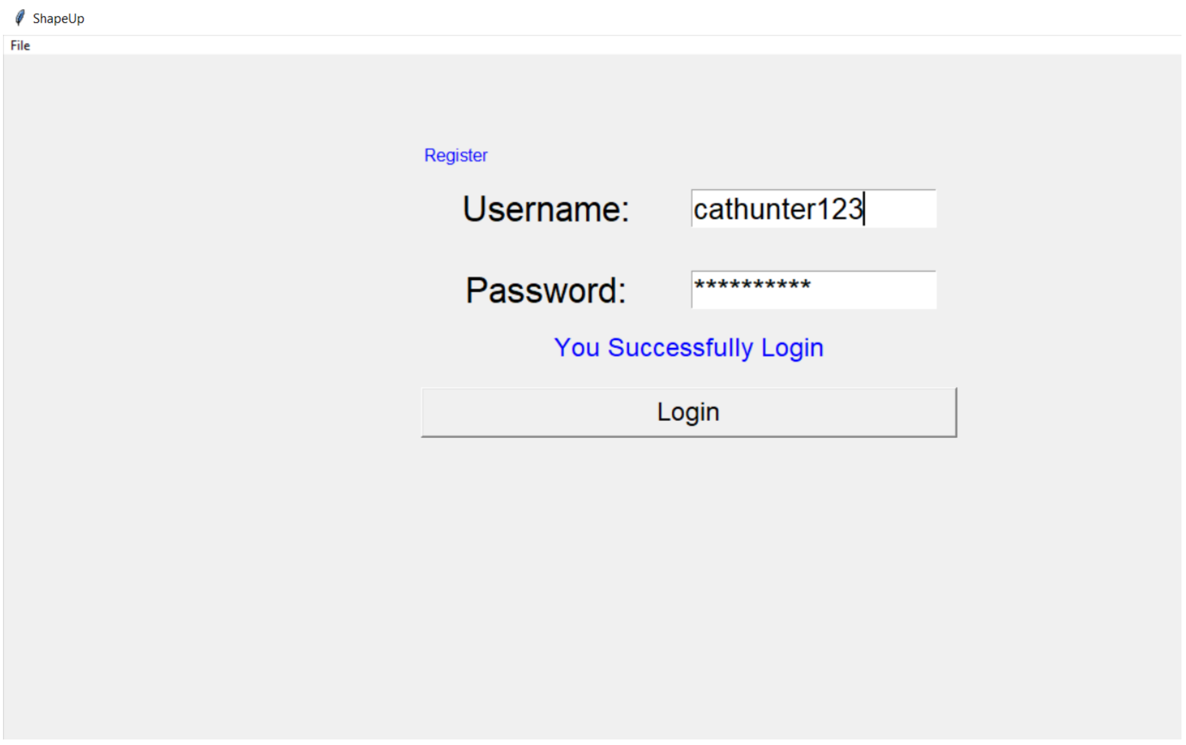
5.1 System Design



5.2 Screen Design



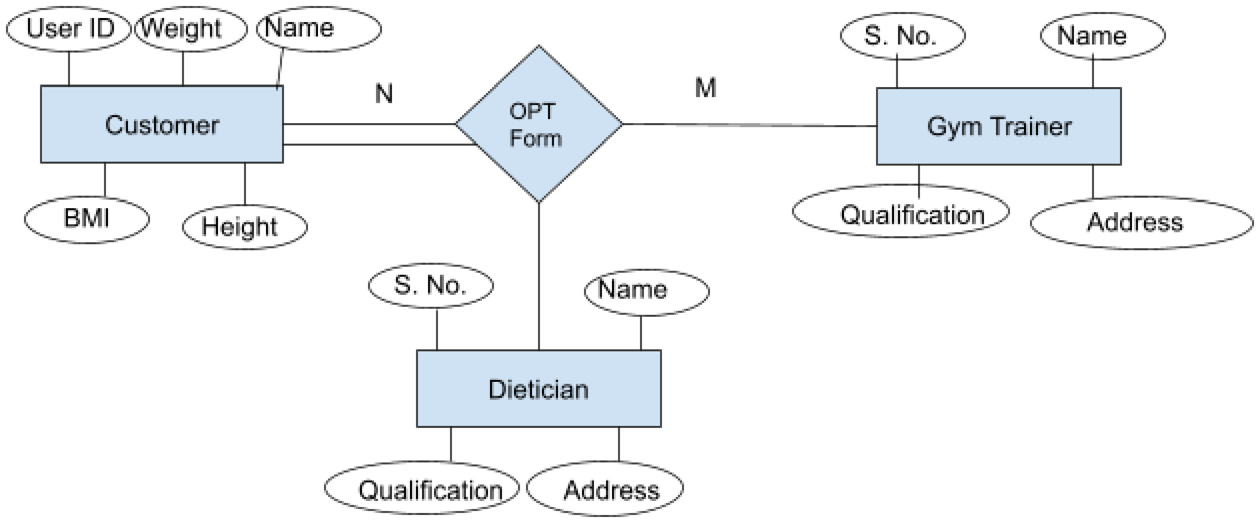
First Name:	sarthak
Last Name:	sangwan
Username:	cathunter123
Password:	**********
Height:	190cm
Weight:	92kg
phno:	100



Username:	cathunter123
Password:	***********

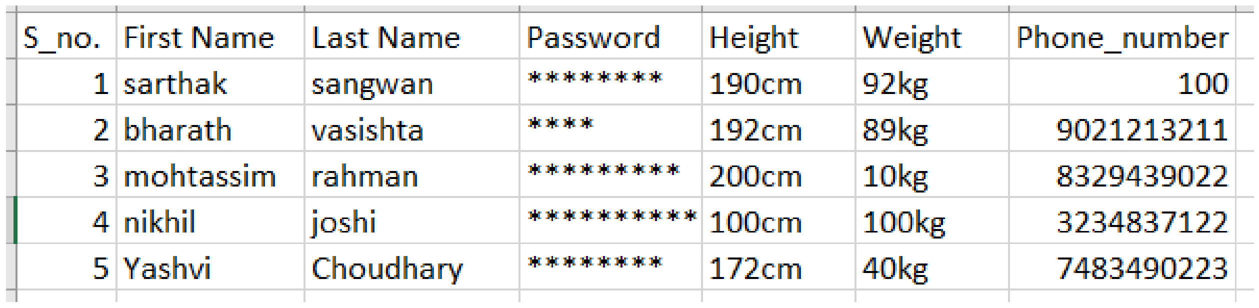
You Successfully Login

5.3 ER Diagram



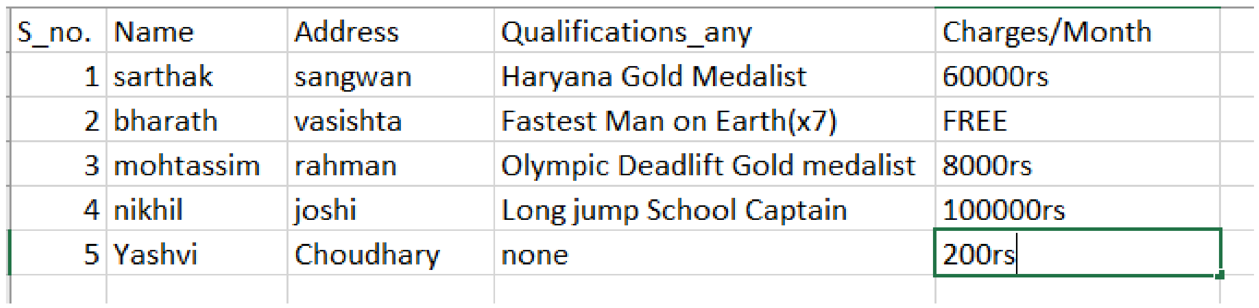
5.4 Database Design

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)")
```

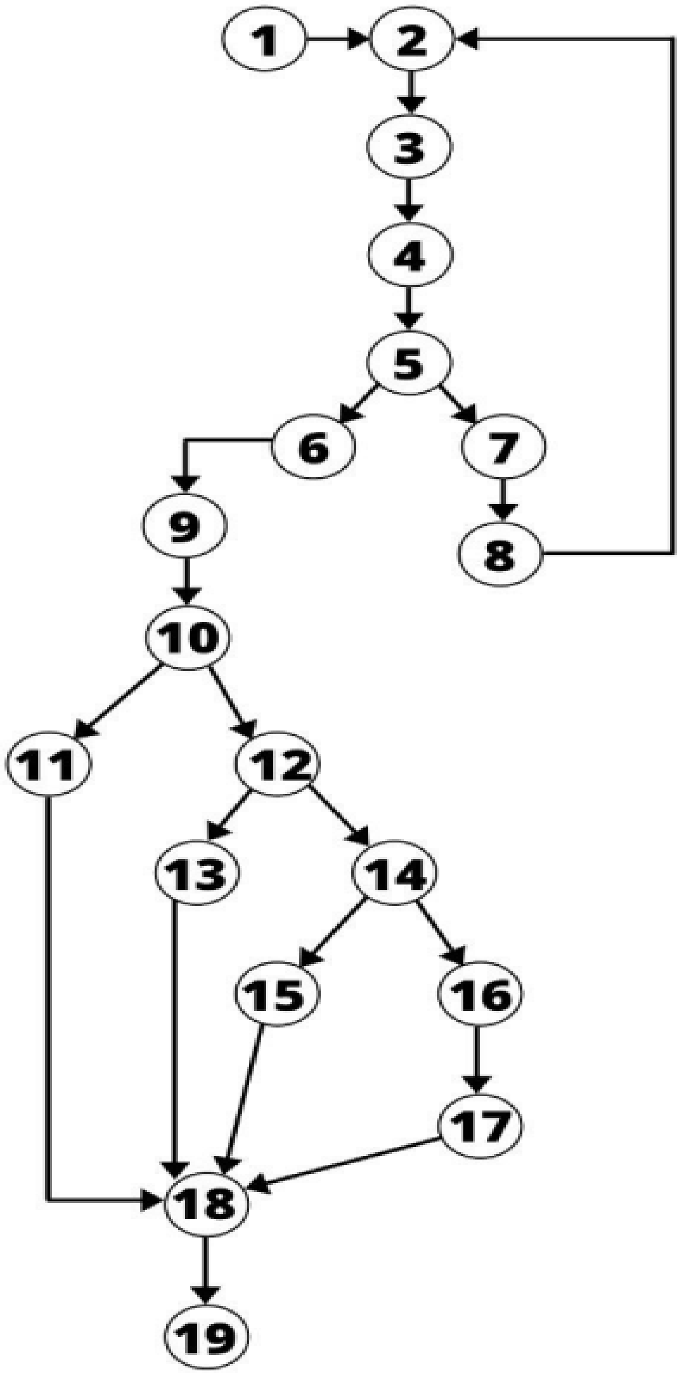
```python
def Exit():
    result = tkMessageBox.askquestion('System', 'Are you sure you want to
exit?', icon="warning")
    if result == 'yes':
        root.destroy()
        exit()


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]

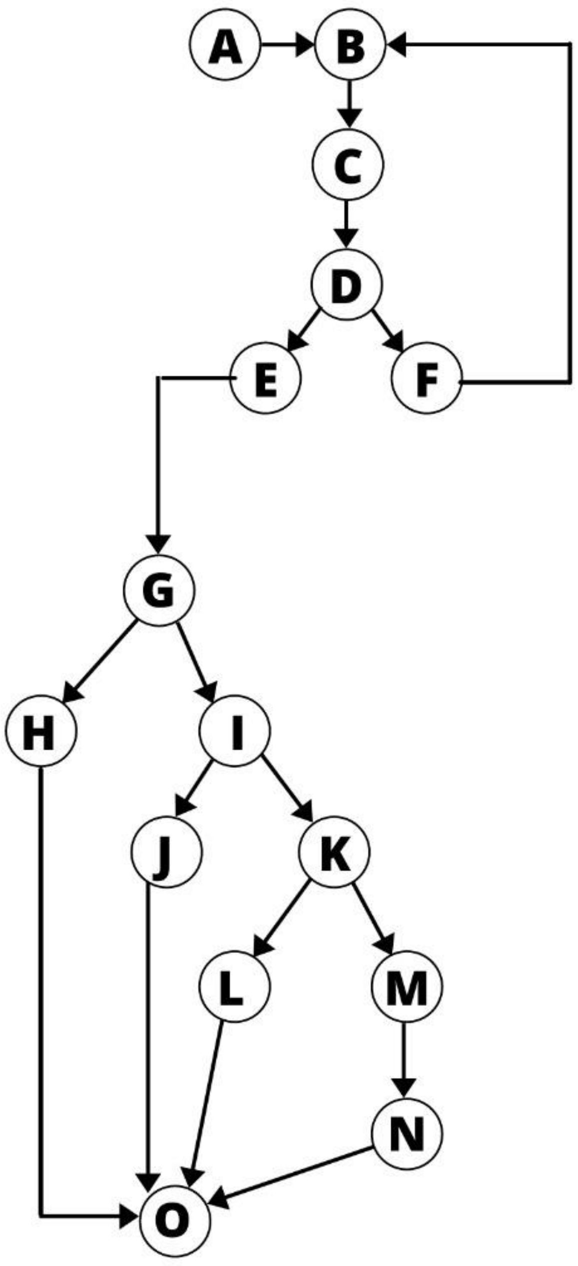
```
def bmi_calc():
  w=h=0 #1
  while(w==0 or h==0): #2
    h = float(input("Enter your height in meters: ")) #3
    w = float(input("Enter your weight in kg: ")) #4
    if(w!=0 and h!=0): #5
        break #6
    else: #7
        print("invalid entries") #8
   bmi = w/(h**2) #9

   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