E – HEALTHIFY STSTEM

1. **PROBLEM STATEMENT**

In everyday busy life there is a need for people to become fit and healthy with each passing day and the food we consume plays a vital role in managing health.

Thus, this E-HEALTHIFY SYSTEM would be great for the people to maintain their “Health & Fitness “.

This E-HEALTHIFY SYSTEM takes person’s age, height, weight, nature of work and provide perfect diet & fitness plan to manage their health. This E-HEALTHIFY system is to help you to get right advice and guidance about “Diet & Fitness” to reach your health & fitness goal. It also provides Diet and Fitness Coach to craft personalized diet & fitness plan for you. These plans include detailed information about calorie intake, water & drinks to be consumed and workouts to perform daily and the entire nutrition. There will be consultation chats and calls with the Diet and Fitness Coach. There will also be monthly and weekly insights for analyzing the performance.

Thus, with this E-HEALTHIFY SYSTEM, you will be regularly motivated and guided to stay on your fitness journey.

1. **AIM AND OBJECTIVE**

* The purpose of this document is to present a detailed information about E-HEALTHIFY SYSTEM. It will explain the purpose and features of the software. This document is intended for users of the software and also potential developers.
* The main objective of this E-HEALTHIFY SYSTEM is to help the user to lose weight, be fitter, gain muscle. It is an Indian digital weight loss and wellness platform that provides fitness services such as calorie tracking, workout coaching and diet plans.

1. **EXISTING /COMPETITIVE SOFTWARE**

* Healthifyme is an already existing application/software with many features.
* To overcome the limitations of current software we are adding these features.

(1) Live exercise sessions

(2) Motivating the users/customers

(3) Cooking recipes according to the calories

(4) Post your performance (self-motivation)

(5) Correcting the RIST tracker bugs

**(D) LIMITATIONS OF EXISTING SOLUTIONS**

The limitations of the existing software is:

1. We cannot post our performances
2. More RIST tracker(steps)problems
3. Customers are not satisfied with the coaches of healthifyme as though they are highly educated and trained (collected from user reviews)

**(E)METHODOLOGY – MODULES & THEIR FUNCTIONALITIES**

1. **Sign up** - In this, new user can create his/her profile by entering their details like user name and password. User name may be their E-mail id or phone number
2. **Login**  – Already registered user can login into application by providing their user name and password correctly. After login he can use access all the features of the application.
3. **Profile**  - Profile contains the details of the user like his/her name, Nature of work, Age, Health details, Blood group, Height, Weight . Health details contains his/her sugar levels ,Blood pressure, cholesterol and any health issues .
4. **Calories intake** - The number of calories taken by the user each session i.e breakfast, lunch , snacks, dinner.
5. **Calories calculation** - The sum of the calories consumed each session by the consumer is the total number of calories taken by the user per day.
6. **Consulting experts** - It includes coaches, nutritionist for providing diet plans and workout plans for the user based on their weight and health conditions.
7. **Calculating daily calorie burn** - To calculate how many calories a person burns each day is calculated by using Harris-Benedict formula .

Harris - Benedict formula : BMR \* average daily activity level. (BMR - Basal Metabolic Rate)

For men : 66 + (6.2 x weight) +(12.7 x height) - (6.76 x age)

For women : 655.1 +(4.35 x weight)+(4.7 x height)-(4.7 x age)

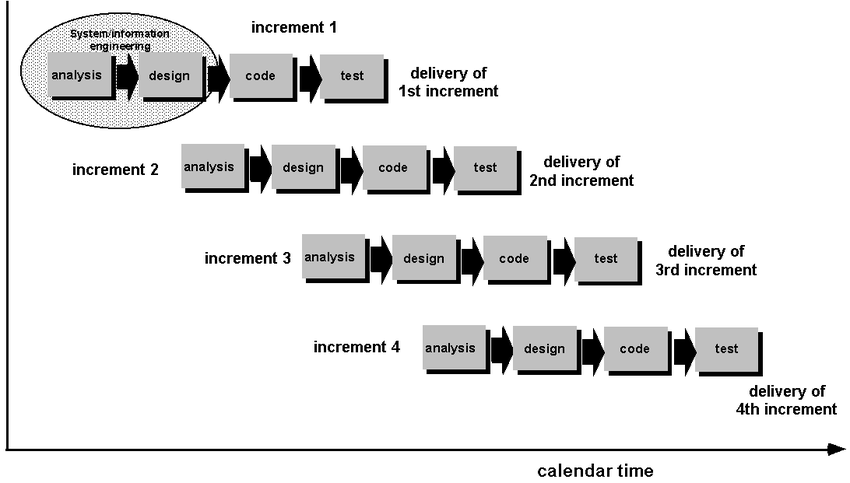
The results of the BMR calculation are then used to multiply against the average daily activity of the person. Points are awarded based on how active a person is .

1. **Post your performance** - can post their workout pictures /videos (like Instagram and face book) for motivating others and themselves. It also includes graphs which indicates their growth.

* **USER FUCTIONS**:
* Provide their proper information about intake.
* Breakfast
* Lunch
* Dinner
* Snacks and any beverages
* Chat & make calls to the Diet and Fitness Coach.
* Can select proper workout and diet plans.
* View his performance on weekly and monthly basis.
* Post their daily workouts.
* **SYSTEM FUNCTIONS:**
* Get all users information and also allow to update.
* Get all calorie information for a given intake.
* View diet plans and workout plans.
* Provide diet and fitness coach for consultation.
* Calculate performance growth for each user on weekly and monthly basis.
* **ADMINISTRATIVE FUNCTIONS:**
* Add different kind of diets.
* Add different kind of workouts and exercises.
* Update user calorie consumption daily.
* Calculate the performance growth of each user.
* Update user personal details (like height, weight) etc. if they are required.

**PROCESS MODEL**

The suitable process model for this project is **Incremental Model**.



Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle. In this model, each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.

**The various phases of incremental model are as follows:**

**1. Requirement analysis:** In the first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.

**2. Design & Development:** In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and development phase.

**3. Testing:** In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behaviour of each task.

**4. Implementation:** Implementation phase enables the coding phase of the development system. It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

**Incremental Model is used in the following scenarios -**

* When the requirements are superior.
* A project has a lengthy development schedule.
* When Software team are not very well skilled or trained.
* When the customer demands a quick release of the product.
* You can develop prioritized requirements first.

## Advantages of Incremental Model

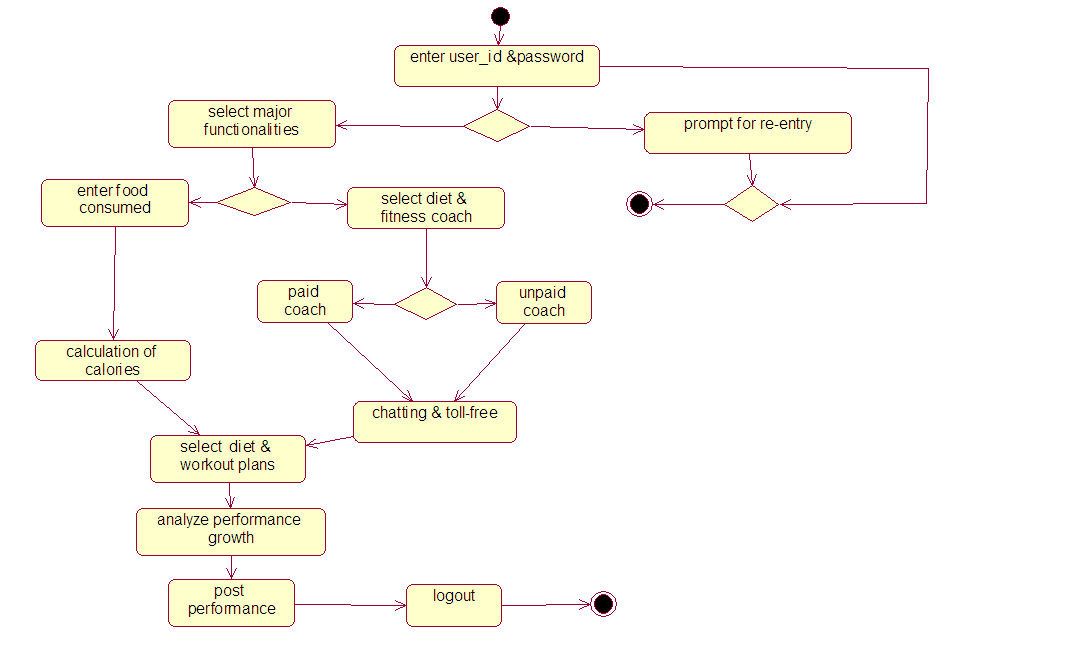
* Errors are easy to be recognized.
* Easier to test and debug
* More flexible.
* Simple to manage risk because it handled during its iteration.
* The Client gets important functionality early.

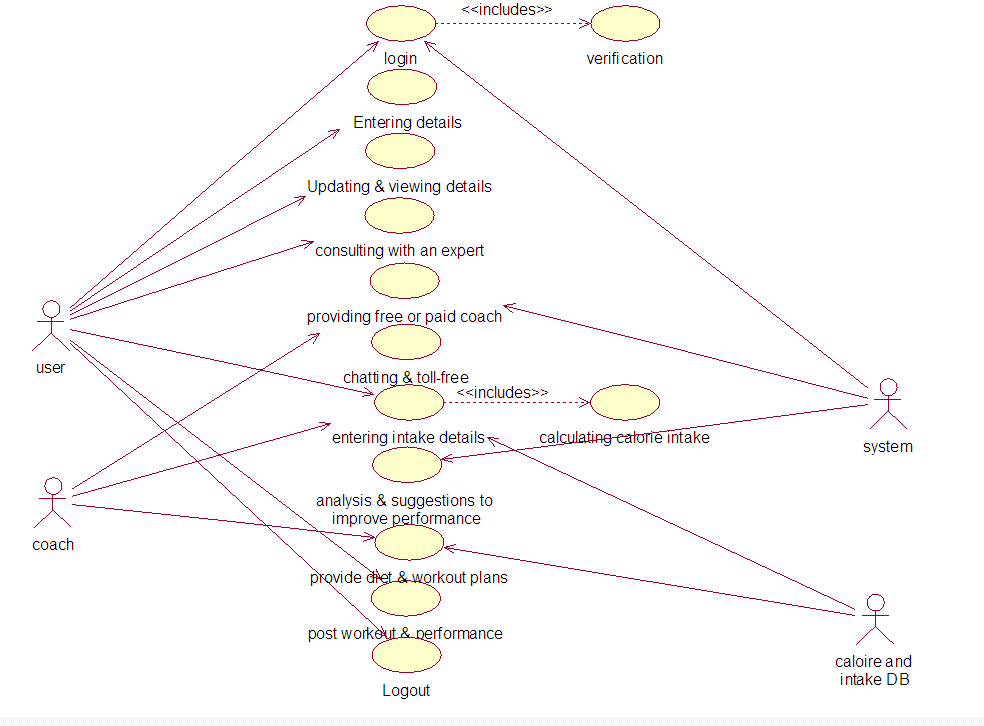
## Disadvantages of Incremental Model

* Need for good planning
* Total Cost is high.
* Well defined module interfaces are needed.

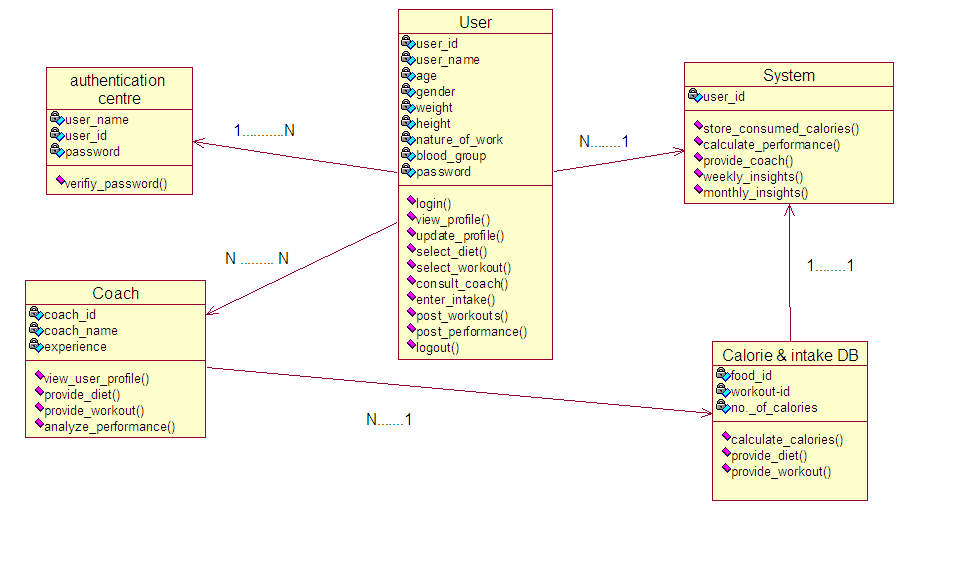
**(F) DESIGN ENGINEERING**

**(1)UML DIAGRAMS:**

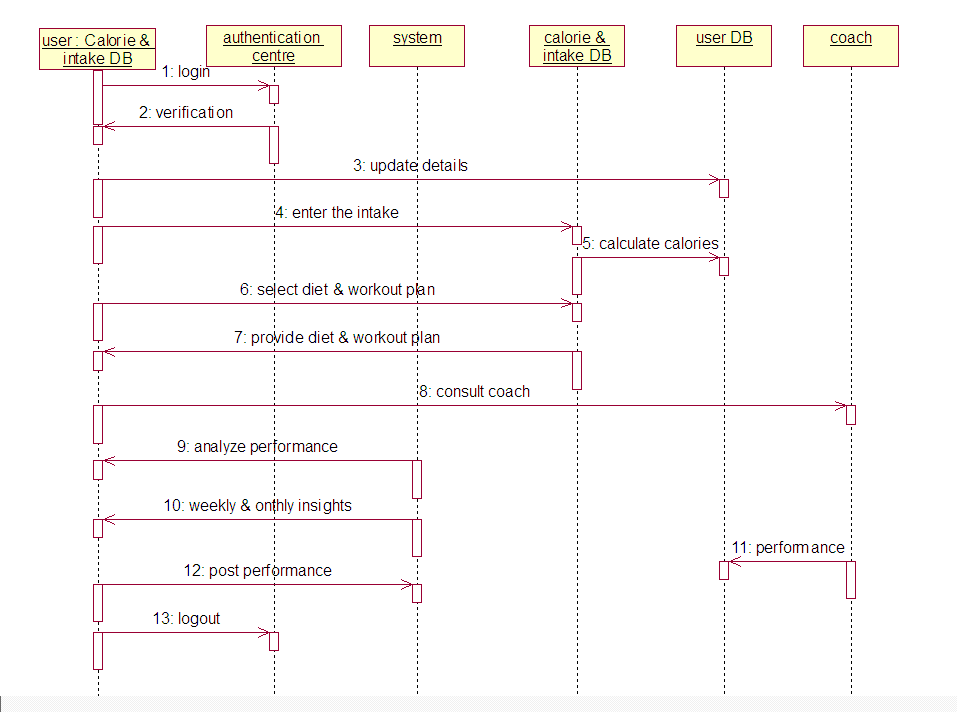
(a) ACTIVITY DIAGRAM:

(b)USECASE DIAGRAM:

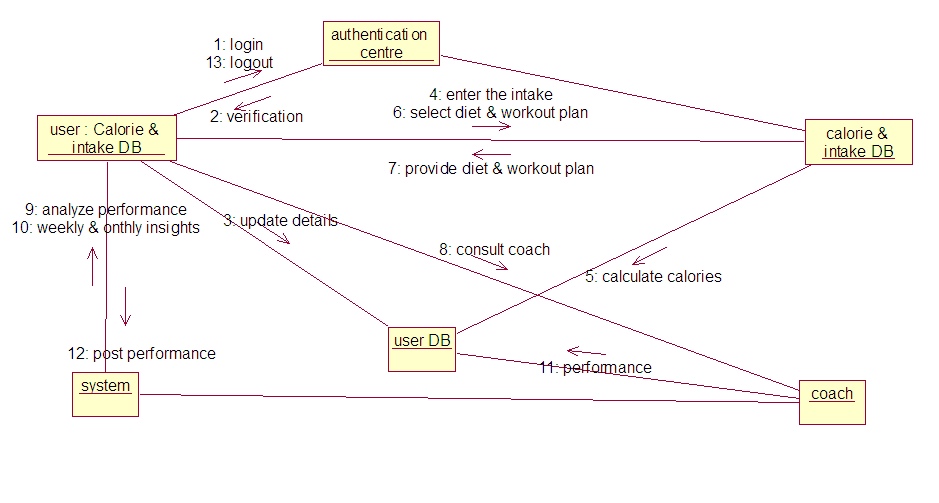
(c)CLASS DIAGRAM:



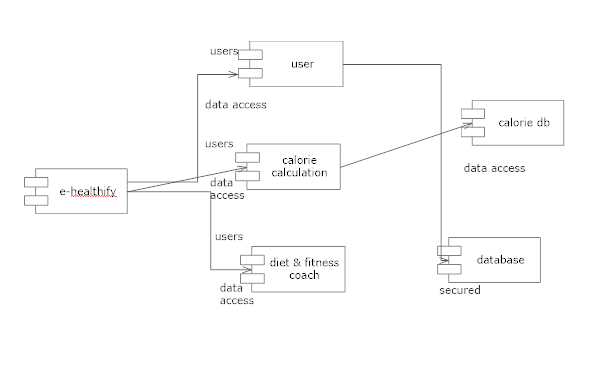
(d)SEQUENCE DIAGRAM:



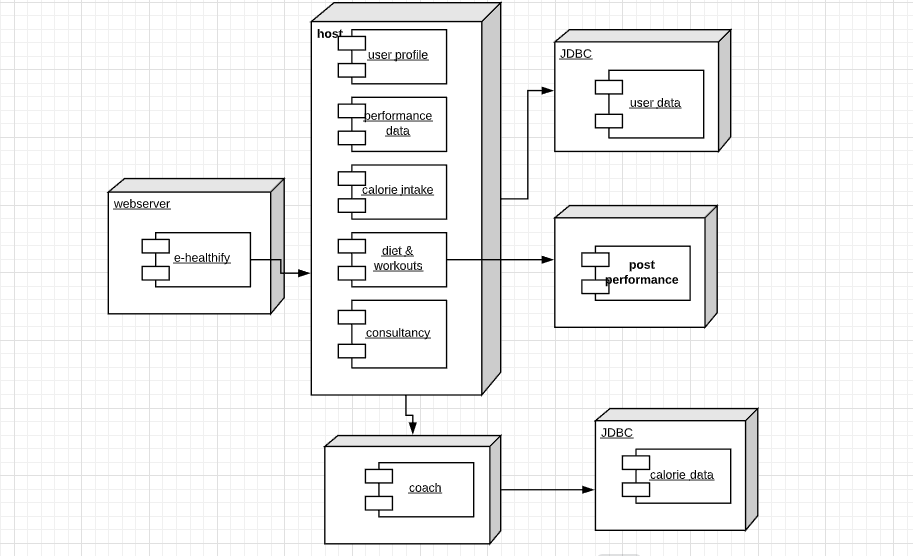
(e)COLLABORATION DIAGRAM:



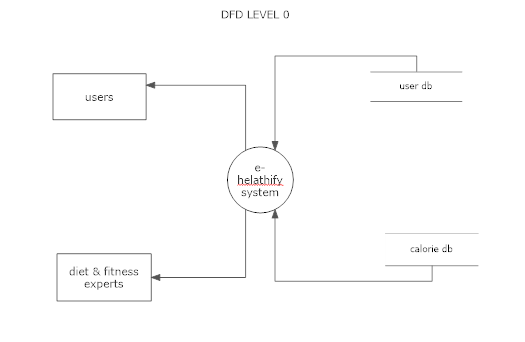
(f) COMPONENT DIAGRAM:

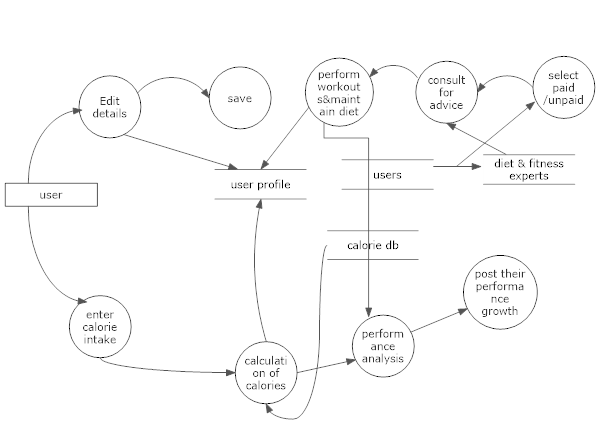


(g) DEPLOYMENT DIAGRAM:

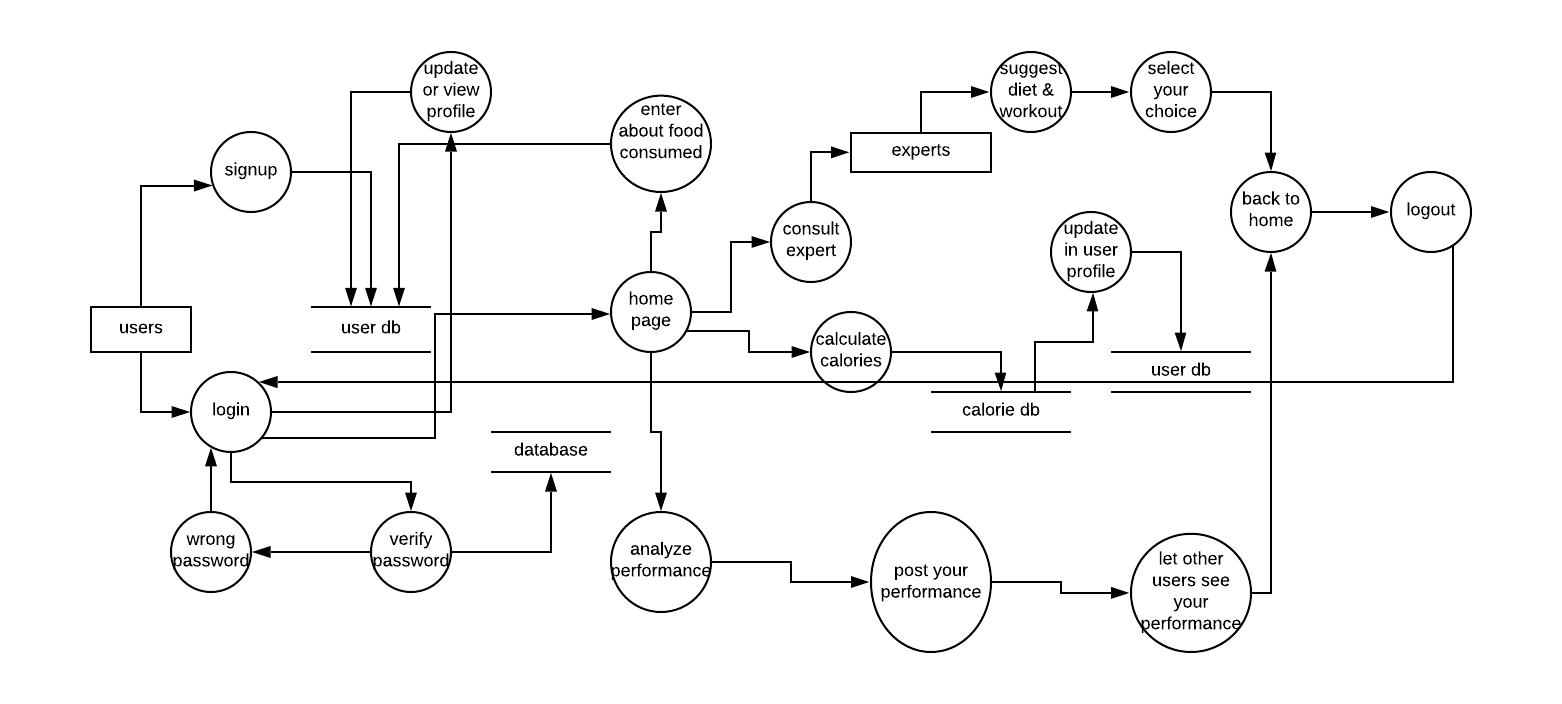


**(ii)DATAFOW DIAGRAMS:**





DFD Level 1



DFD Level 2

**(G)DEVELOPMENT**

**(1) IMPLEMENTATION:**

**User class and characteristics:**

Users of the system should be able to maintain their health and fitness. The information like height, weight, gender, age and their nature of work etc. must be provided by the users. Users get proper guidance about what to consume daily and what exercises do they need to perform to be fit. Calorie information will be calculated based on the food they consumed and everything will be stored in an database. Users are provided with the Diet and Fitness Coach. There are two different kinds of coaches that is paid and unpaid. For consultation users can chat with the coach and they call using the toll-free numbers given. This e-healthify system will have a platform where users can post their workouts and diet daily, like a user interactive blog. All the records of the users are stored in a database and their performances are measured on monthly and weekly basis.

**Communications Protocols and Interfaces:**

1.The system should also use standard protocols for secure transactions (when the users are paying for the paid experts to train them) between the user and the system through the internet.

2. System should provide a required diet and fitness experts to users when required.

3.Sytem should provide the correct diet for the users as required.

4.Users are allowed to communicate with the system while selecting their kind of exercise and workouts.

**Operating Environment:**  
        
The server should have Java installed on the machine, along with Java’s cryptographic packages. The user server runs on a http server, which has JSP enabled. The browsers through which the users access the server should have minimal support for cookies and encrypted transactions.  
  
**Design/Implementation Constraints:**  
  
The system should be designed within 3 months. Only admins can able to access, manage, and modify the application, users, and official details. For adding and removing they had to approach the GP’s admin.  
 **Assumptions and Dependencies:**  
        
1. User side assumptions and dependencies  
                                      – PC (Personal Computer) or workstation with GUI.  
                                      – A web browser with support for cookies and also can be like an application.  
                                      – Working Internet connection.  
2. Server-side assumptions and dependencies  
                                      – A web server with GUI, PHP, MYSQL and an http server installed.

Let us assume that this is a distributed calorie and intake database and it is used in the following application:

* Provide a perfect diet and fitness plan for the particular user which is well suitable to him/her, using information from a database.
* Calculation of calories consumed by the user in a particular day and their performance growth.

We have designed a distributed database, that is practically dispersed for four kind of diet plans like breakfast, lunch, dinner and any kind of snacks and beverages.

**FUNCTIONAL REQUIREMENTS**  
  
**Hardware Interfaces:**  
There are no hardware interfaces to this software system. The only interfaces are through a mobile system.  
  
**Software Interfaces:**  
  
The poll server runs on http server that is enabled to handle server pages. It uses a relational database to keep track of the user data, their workouts, performance, calorie consumption with which it connects through standard database connectivity interfaces. In order to run the setup software, the environment needs to have a JVM running on it.

|  |  |
| --- | --- |
| Software used | Description |
| Operating system | We have chosen Windows XP operating system for its best support. |
| Database | To save the calorie information, users records we have chosen SQL+ database. |
| VB.Net | To implement the project we have chosen Vb.Net language for its more interactive support. |

**User interfaces:**

* Front-end software:     Vb.net version, java netbeans
* Back-end software:       SQL+

**Distributed Database:**

Distributed database implies that a single application should be able to operate transparently on data that is spread across a variety of different databases and connected by a communication network.

**NON-FUNCTIONAL REQUIREMENTS**

**Normalisation:**

The basic objective of normalization is to be reduce redundancy which means that information to be stored only once. Storing information several times leads to wastage of storage space and increase in the total size of the data stored.

If a Database is not properly designed it can give rise to modification anomalies. Modification anomalies arise when data is added to, changed or deleted from a database table. Similarly, in traditional databases as well as improperly designed relational databases, data redundancy can be a problem. These can be eliminated by normalizing a database.

Normalization is the process of breaking down a table into smaller tables. So that each table deals with a single theme. There are three different kinds of modifications of anomalies and formulated the first, second and third normal forms (3NF) is considered sufficient for most practical purposes. It should be considered only after a thorough analysis and complete understanding of its implications.

**Safety Requirements:**

If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs the current state of the system.

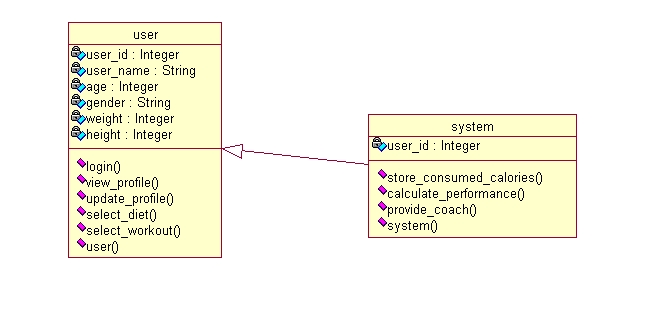
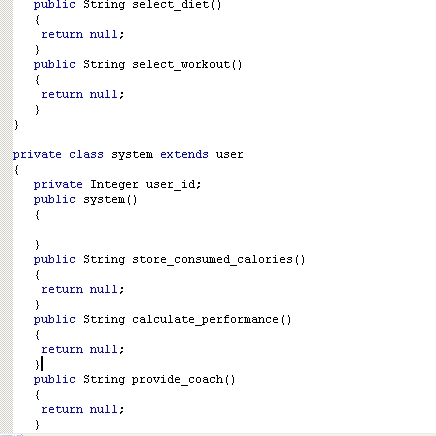
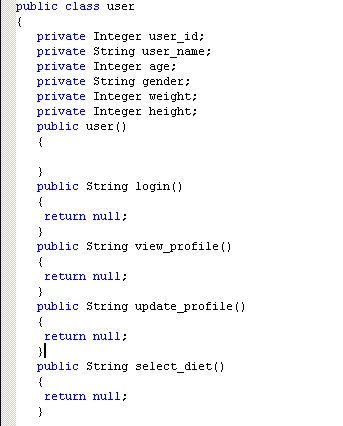
**Security Requirements:**

Security systems need database storage just like many other applications. However, the special requirements of the security market mean that vendors must choose their database partner carefully.

**Software quality attributes:**

* CORRECTNESS:  The calorie measurement based on the consumed food must be correct and correct workout plans must be provided to them.
* AVAILABILITY: The proper diet plans must be available to the users whenever required and diet & fitness coach must be available at every time.
* MAINTAINABILITY: The administrator should maintain correct details about the calorie information of a user.
* USABILITY: The diet and fitness plans should satisfy maximum number of user needs.

**(ii)FORWARD AND REVERSE ENGINEERING**



(H) TESTING

**(i) Unit testing - Test plans & Test cases**

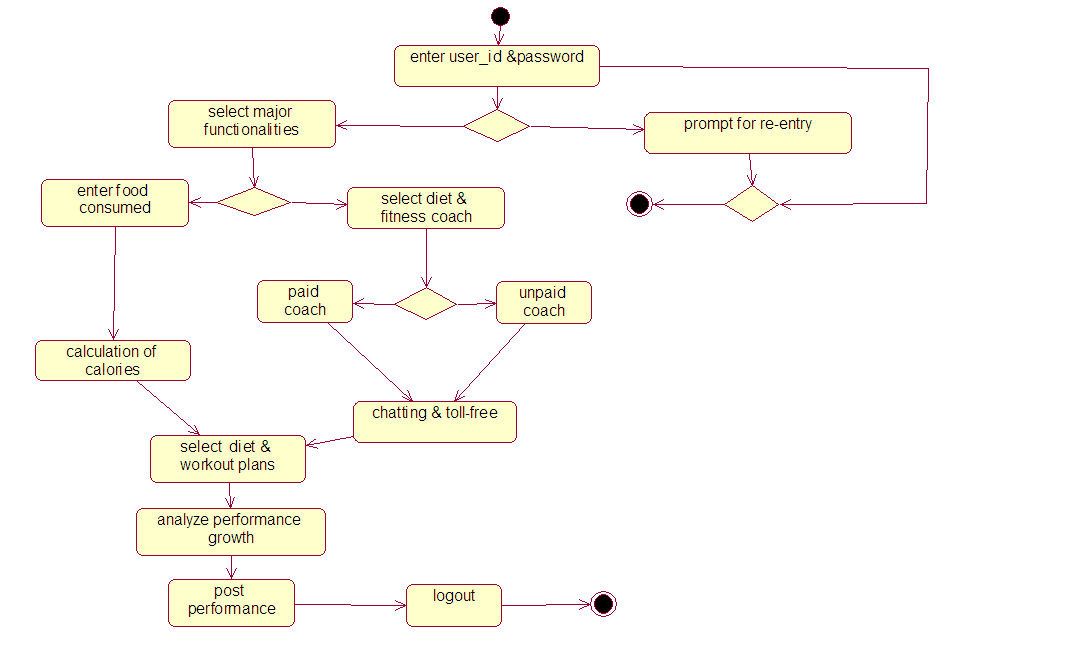
DERIVING THE TEST CASES:

The basis path testing method can be applied to a procedural design or to source  
code. To Illustrate each step in the test-case design method. Note that average, although an extremely simple algorithm, contains compound conditions and loops. The following steps can be applied to derive the basis set:

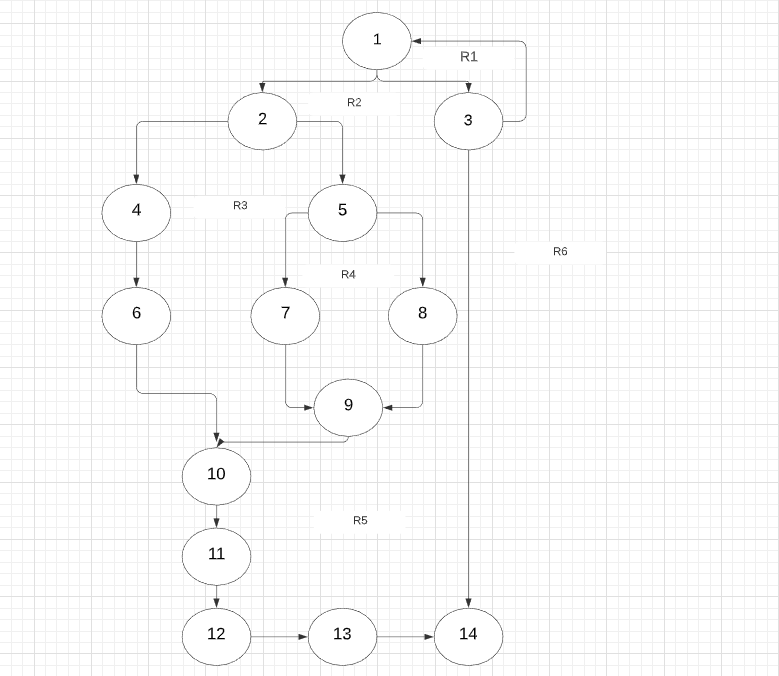
1.Using the design or code as a foundation, draw a corresponding flow graph.

A flow graph is created using the symbols and construction rules. Flow graph is created by numbering those PDL statements that will be mapped into corresponding flow graph nodes. The corresponding flow graph is shown in Figure.

(A)FLOW CHART



(B)FLOW GRAPH



2. Determine the cyclomatic complexity of the resultant flow graph.  
The cyclomatic complexity V(G) is determined by applying the algorithms.

Cyclomatic complexity has a foundation in graph theory and provides you with an extremely useful software metric. Complexity is computed in one of three ways:  
1. The number of regions of the flow graph corresponds to the cyclomatic complexity.  
2. Cyclomatic complexity V(G) for a flow graph G is defined as  
V(G) = E - N + 2  
where E is the number of flow graph edges and N is the number of flow graph nodes.  
3. Cyclomatic complexity V(G) for a flow graph G is also defined as  
V(G) = P + 1  
where P is the number of predicate nodes contained in the flow graph G.  
The cyclomatic complexity can be computed using each of the algorithms just noted:  
1. The flow graph has 6 regions.  
2. V(G) = 18 edges - 14 nodes + 2 = 6.  
3. V(G)  = 5 predicate nodes + 1 = 6.  
Therefore, the cyclomatic complexity of the flow graph is 6.  
More important, the value for V(G) provides you with an upper bound for the number of independent paths that form the basis set and, by implication, an upper bound on the number of tests that must be designed and executed to guarantee coverage of all program statements.

3. Determine a basis set of linearly independent paths. The value of V(G) provides the upper bound on the number of linearly independent paths through the program control structure. In the case of procedure average, we expect to specify six paths:  
Path 1: 1 – 3 – 14  
Path 2: 1 – 2 – 4 – 6 – 10 – 11 – 12 – 13 – 14

Path 3: 1 – 2 – 5 – 7 – 9 – 10 – 11 – 12 – 13 – 14

Path 3: 1 – 2 – 5 – 8 – 9 – 10 – 11 – 12 – 13 – 14

4.Prepare test cases that will force execution of each path in the basis set.

Data should be chosen so that conditions at predicate nodes are appropriately set as each path is tested. Each test case is executed and compared to expected results. Once all test cases have been completed, tester can be sure that all statements in the program have been executed at least once. It is important to note that some independent paths cannot be tested in stand-alone fashion. That is the combination of data required to traverse path cannot be achieved in normal flow of the program. In such cases, these paths are tested as part of another path test.

**BLACK BOX TESTING:**

**What is Black Box Testing?**

BLACK BOX TESTING is defined as a testing technique in which functionality of the Application Under Test (AUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on software requirements and specifications. In BlackBox Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.

**[](https://www.guru99.com/images/stories/blackbox.png)**

**Value conditions for signup:**

*class REG(models.Model):*

*username = models.CharField(min\_length=5,max\_length=20)*

*age = models.CharField(min\_length=1,max\_length=2)*

*gender = models.CharField(min\_length=1,max\_length=1)*

*height = models.CharField(min\_length=2,max\_length=10)*

*weight= models.CharField(min\_length=2,max\_length=10)*

*profession = models.TextField(min\_length=5,max\_length=30)*

*mobile = models.CharField(min\_length=10,max\_length=10)*

*password = models.CharField(min\_length=8,max\_length=100)*

**Black box testing using Equivalence class testing method:**

|  |  |  |  |
| --- | --- | --- | --- |
| Characteristic | Invalid | Valid | Invalid |
| Username | <5 | 5-20 | >20 |
| Age | <1 | 1-2 | >2 |
| Gender | <1 | 1 | >1 |
| Height | <2 | 2-10 | >10 |
| Weight | <2 | 2-10 | >10 |
| Profession | <5 | 5-30 | >30 |
| Mobile | <10 | 10 | >10 |
| Password | <8 | 8-100 | >100 |

**Sample test cases:**

* **Invalid:**

Username = sri

Age = 1987

gender = male

height = 5411217218

weight = Shami

profession = mec

mobile = 45323

password = kjhg

**Valid:**

Username = sweety

Age = 19

gender = f

height = 5.4

weight = 59.6

profession = student

mobile = 9898765434

password = asdf45678

**Invalid:**

Username = sykujhyhgftdbvvh200001

Age = 1900

gender = female

height = 5.4987452790076654

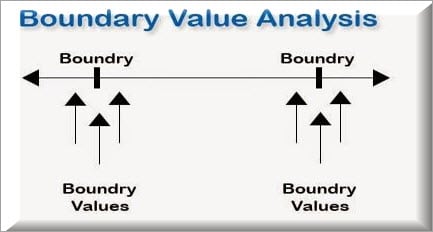
weight = 59.654673456234

profession = stu

mobile = 989

Password = *Any password greater than 100 characters*

**Black box testing using Boundary value analysis method:**



|  |  |  |  |
| --- | --- | --- | --- |
| Characteristic | Invalid  (min-1) | Valid  (min,+min,-max,max) | Invalid  (max+1) |
| Username | 4 | 5,6,19,20 | 21 |
| Age | 0 | 0,1,2,3 | 3 |
| Gender | 0 | 0,1,1,2 | 2 |
| Height | 1 | 1,2,10,11 | 11 |
| Weight | 1 | 1,2,10,11 | 11 |
| Profession | 4 | 4,5,30,31 | 31 |
| Mobile | 9 | 9,10,10,11 | 11 |
| Password | 7 | 7,8,100,101 | 101 |

**Sample test cases:**

* **Invalid:**

Username = sri(value = 3)

Age = 198 (value = 3)

gender = male (value = 4)

height = 5411217218 (value = 10)

weight = Shami (value = 5)

profession = mec (value = 3)

mobile = 45323 (value = 5)

password = kjhg (value = 4)

**Valid:**

Username = sweety (value = 6)

Age = 19 (value = 2)

gender = f (value = 1)

height = 5.4 (value = 2)

weight = 59.6 (value = 3)

profession = student (value = 7)

mobile = 9898765434 (value = 10)

password = asdf45678 (value = 9)

**Invalid:**

Username = sykujhyhgftdbvvh200001 (value = 23)

Age = 1900 (value = 4)

gender = female (value = 6)

height = 5.498745279007665 (value = 16)

weight = 59.654673456234 (value = 14)

profession = stu (value = 3)

mobile = 989 (value = 3)

Password = *Any password greater than 100 characters* (value = 102)

**WHITE BOX TESTING:**

**Python code for signup:**

*1.def sign(request):*

*2.if request.method == 'POST':*

*3. username = request.POST['name']*

*4.password1 = request.POST['pswd']*

*5.password2 = request.POST['confpswd']*

*6.age = request.POST['email']*

*7.gender = request.POST['mobile']*

*8.weight =request.POST['aadhar']*

*9.height =request.POST['address']*

*10.profession =request.POST['address']*

*11.mobile =request.POST['address']*

*12.if password1==password2:*

*13.if REG.objects.filter(username=username).exists():*

*14.messages.info(request,'Username Taken')*

*15.return redirect('sign')*

*16.elif REG.objects.filter(email=email).exists():*

*17. messages.info(request,'Email Taken')*

*18. return redirect('sign')*

*19.else:*

*20. user = REG(username=username, password=password1,age=age, mobile=mobile, height=height, weight=weight, profession=profession)*

*21.user.save();*

*22.print('user created')*

*23.return redirect('login')*

*24.else:*

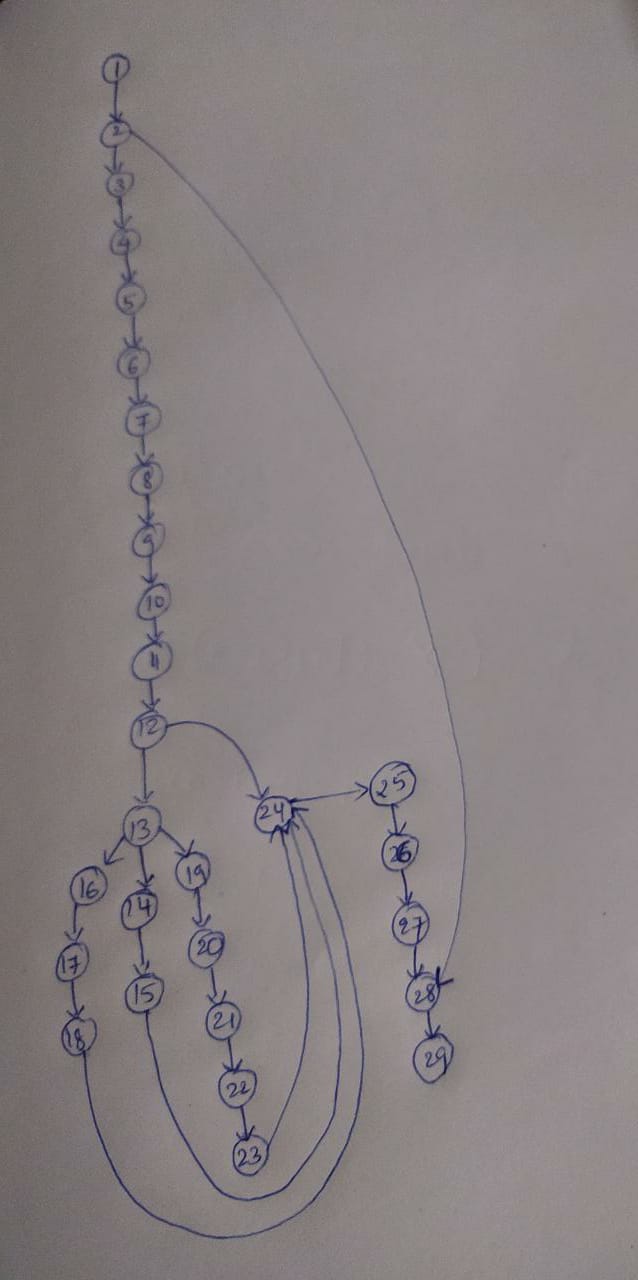
*25.messages.info(request,'password not matching..')*

*26.return redirect('sign')*

*27.return redirect('/')*

*28.else:*

*29.return render(request,'signup.html')*



**CFG For above code**

* Now we need to find the cyclomatic complexity because it is equal to the no.of independent paths
* Cyclomatic complexity = *Predicate nodes+1*

= 4+1 = 5

* Therefore, we have 5 independent paths implies we need to develop 5 testcases to cover all the paths.

**Testcases:**

Testcase 1: 1-2-26-27

Request.method = ’GET’

Testcase 2: 1-2-3-4-5-6-7-8-9-10-22-23-24-25-26-27

Request.method = ‘POST’

Password1=abcd

Password2=abcde

Testcase 3: 1-2-3-4-5-6-7-8-9-10-11-14-15-16-22-23-24-25-26-27

Request.method = ‘POST’

Password1=abcd

Password2=abcd

And if user enters username that already exists in Database.

Testcase 4: 1-2-3-4-5-6-7-8-9-10-11-12-13-22-23-24-25-26-27

Request.method = ‘POST’

Password1=abcd

Password2=abcd

If user enters new username.

And if user enters mobile that already exists in Database.

Testcase 5: 1-2-3-4-5-6-7-8-9-10-11-17-18-19-20-21-22-23-24-25-26-27

Request.method = ‘POST’

Password1=abcd

Password2=abcd

If user enters new username.

If user enters new mobile

**White box testing through Data Flow Testing:**

|  |  |  |
| --- | --- | --- |
| Variable | Defined at | Used at |
| Username | 3,18 | 13,20 |
| password1 | 4 | 12,20 |
| password2 | 5 | 12 |
| Age | 6 | 14,18 |
| Gender | 7 | 20 |
| Height | 9,20 | 20 |
| Weight | 8,20 | 20 |
| Password | 20 | **-** |
| Mobile | 20 | **-** |
| Profession | 10 | **-** |

**White box testing through condition testing/branch testing:**

Condition testing is a test construction method that focuses on exercising the logical conditions in a program module. Errors in conditions can be due to:

* Boolean operator error
* Boolean variable error
* Boolean parenthesis error
* Relational operator error
* Arithmetic expression error

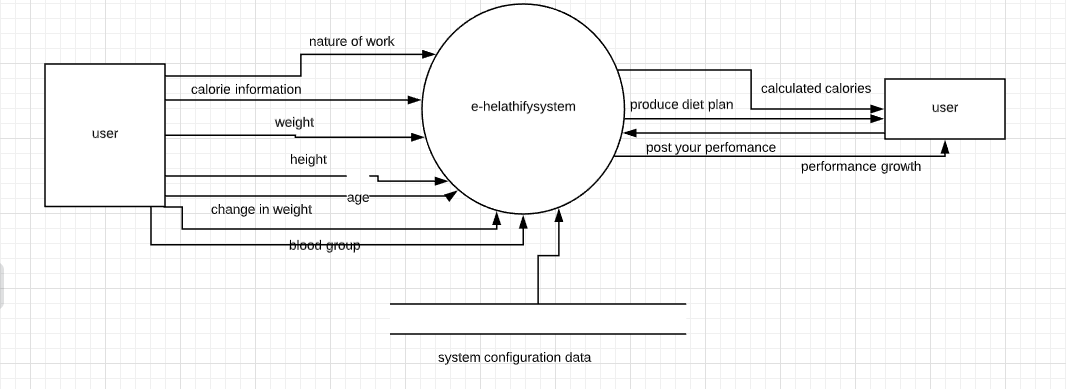
**Definition:** "For a compound condition C, the true and false branches of C and every simple condition in C need to be executed at least once."

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Decision | Possible Outcomes | Test Cases | | | |  |
| 1 | 2 | 3 | 4 | 5 |
| *request.method = = 'POST'* | T | **X** | **X** | **X** | **X** |  |
|  | F |  |  |  |  | **X** |
| *password1= =password2:* | T | **X** | **X** | **X** |  | **X** |
|  | F |  |  |  | **X** |  |
| *REG.objects.filter(username=username).exists():* | T |  |  | **X** |  |  |
|  | F | **X** | **X** |  | **X** | **X** |
| *REG.objects.filter(email=email).exists():* | T |  | **X** |  |  |  |
|  | F | **X** |  | **X** | **X** | **X** |

|  |  |  |
| --- | --- | --- |
| **TEST CASES** | | |
| **Case no.** | **Input values** | **Expected outcomes** |
| **1** | **Request.method=”POST”**  **Password1=abc**  **Password2=abc**  **Username=sweety**  **Mobile=8989754543** | **DISPLAY: User created**   * **Opens login webpage** |
| **2** | **Request.method=”POST”**  **Password1=abcdef**  **Password2=abcdef**  **Username=sweety**  **(Already existed)** | **MESSAGE: Already username taken**   * **Redirects to signup webpage** |
| **3** | **Request.method=”POST”**  **Password1=abc**  **Password2=abc**  **Username=shakthi**  **Mobile=8989754543**  **(Already existed)** | **MESSAGE: Already mobile no taken**   * **Redirects to signup webpage** |
| **4** | **Request.method=”POST”**  **Password1=abc**  **Password2=abcd**  **Username=sweety**  **Mobile=8989754543**  **(Already existed)** | **MESSAGE: password not matching**   * **Redirects to signup webpage** |
| **5** | **Request.method=”GET”**  **Password1=abc**  **Password2=abc**  **Username=sweety**  **Mobile=8989754543**  **(Already existed)** | * **Rediects to home page** |

**(i)Empirical Estimation:**

**(1) FP CALCULATION:**



- Number of user inputs = 5(nature of work, height, weight, age, blood group)

- Number of user outputs = 3(calculated calories, produce diet plan, performance growth)

- Number of user inquiries = 2(change in weight, calorie information)

- Number of files = 1(system configuration data)

- Number of external interfaces = 1 (post your performance)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Measurement parameter | Count |  | simple | average | complex |  | Total average |
| Number of user inputs | 5 | X | 3 | 4 | 7 | = | 20 |
| Number of user outputs | 3 | X | 2 | 6 | 8 | = | 18 |
| Number of user inquiries | 2 | X | 5 | 9 | 11 | = | 18 |
| Number of files | 1 | X | 4 | 10 | 13 | = | 10 |
| Number of external inquiries | 1 | X | 1 | 4 | 10 | = | 4 |
| Total count |  |  |  |  |  |  | 70 |

-To calculate the total FP, the following formula is used:

FP = CT \* (0.65 + 0.01 \* Σ Fi)

Where:

-FP: Total Function Points

-CT: Count Total

-Σ Fi: Total complexity adjustment or Total Weighting Factor value

Assume Weighting Factor is simple:

CT = TOTAL COUNT from above table = (5\*4) + (3\*6) + (2\*9) + (1\*10) + (1\*4) =70

To calculate this, you have to fill the correct number for the following count of each of 5 parameters and select the one of software project complexity levels:

|  |  |  |
| --- | --- | --- |
| **S.NO.** | **General System Characteristics** | **Degree of Influence [value]** |
| 1 | Operational Ease | 3 |
| 2 | Data Communication | 5 |
| 3 | Distributed Functions s | 4 |
| 4 | Performance | 5 |
| 5 | Heavily Used Configuration | 2 |
| 6 | Transaction Rate | 4 |
| 7 | On-line Data Entry | 3 |
| 8 | On-line Update | 2 |
| 9 | End-user Efficiency | 3 |
| 10 | Complex Processing | 4 |
| 11 | Reusability | 5 |
| 12 | Installation Ease | 2 |
| 13 | Multiple Sites | 3 |
| 14 | Facilitates Change | 4 |
| Total complexity adjustment / Total Weighting Factor value (Σ Fi) | | 49 |

Σ Fi =49 from the above table

Therefore,

FP=70x[0.65+(0.01x49)] = 79.8 = 80(approx.)

**(2)LOC ESTIMATION:**

-To calculate the numbers of line of code, the following formula is used.

LOC=LOC\*FP OR LOC=AVC\*FP

where,

AVC is the average number of LOC/FP for a given language

LOC is the numbers of line of code

FP is the total number of functional points

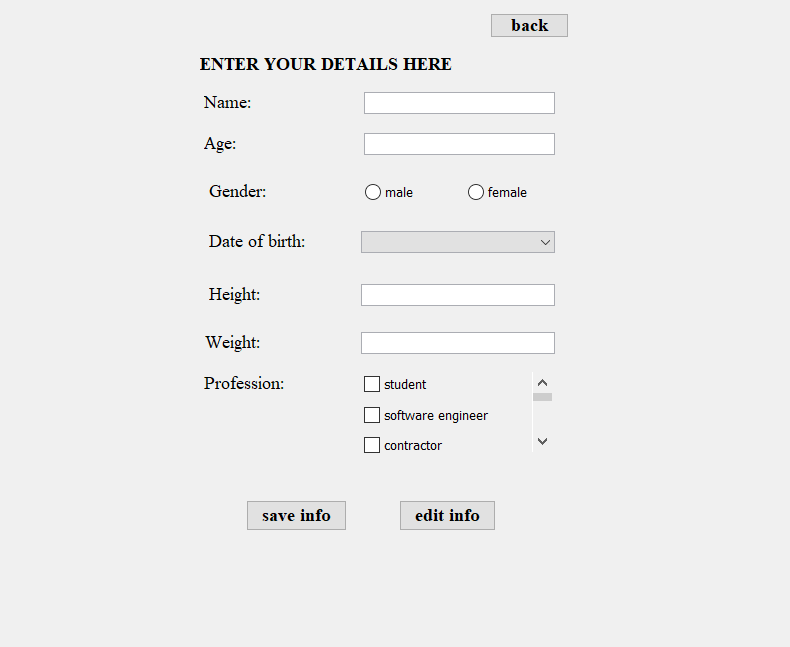
Therefore,

LOC=FP\*37=80\*37=2,960 LOC

**(J) GUI INTERFACE**

**(i)FRONT END**









**(ii)BACKEND**

