

Contents	
Contents.....	1
Introduction & Objective of the Project:	6
Introduction	6
Objective.....	7
System Analysis.....	8
Identification of Need:.....	8
Preliminary Investigation:.....	8
Feasibility Study:	8
Planning and Scheduling.....	8
Gantt chart	8
Tracking Gantt	9
Pert Chart	9
Estimation.....	10
Software REQUIREMENTS AND ANALYSIS	11
Problem Definition	11
Requirements Specification.....	12
Functional Requirement	12
Register User.....	12
Login User.....	12
Add food entry for tracking	12
Add exercise entry for tracking	12
Add weight entry for tracking.....	13
Add To Do	13
Share details in web server.....	13
Add User's new food	13
Create food/ diet chart.....	14
Create exercise chart.....	14
View weekly/monthly report	14
Non-functional Requirements	15
Technical Specification	15
Software Engineering Paradigm applied	15
Data models.....	18
Context Diagram.....	18

0-Level DFD	19
1-Level DFD	20
2-Level DFD	21
Sequence diagrams	22
E-R Diagram.....	22
Class Diagram	25
Activity Diagrams	26
System Design	27
Modularisation details.....	27
FITNESS TRACKER Engine:.....	27
FITNESS TRACKER GUI:	27
FITNESS TRACKER Storage:	28
FITNESS TRACKER web site:	28
Data integrity and constraints	28
Entity integrity	28
Referential Integrity	28
Domain Integrity	28
User Defined Integrity	28
Database design.....	29
User Interface Design	30
Android User Interface Design	30
Web User Interface Desing	40
Test Cases (Unit Test Cases and System Test Cases)	42
Unit Test Cases.....	42
System Test Cases	50
Coding.....	54
Complete Project Coding	54
Android coding.....	54
Web coding.....	67
Database connector: FTDb	75
Data Classes : FTData	82
Comments and Description of Coding segments.....	86
Code Commenting.....	86
Description of coding.....	87
Standardization of the coding.....	88

<i>Code Efficiency</i>	<i>89</i>
<i>Error handling</i>	<i>89</i>
<i>Exceptions Overview</i>	<i>90</i>
<i>Validation checks</i>	<i>90</i>
<i>Testing</i>	<i>93</i>
<i>Testing techniques and testing strategies used</i>	<i>93</i>
<i>Database & Data Integrity Testing.....</i>	<i>93</i>
<i>Functional Testing:.....</i>	<i>93</i>
<i>Regression Testing:</i>	<i>94</i>
<i>User Interface Testing:</i>	<i>94</i>
<i>Performance Profiling:</i>	<i>94</i>
<i>Load Testing:</i>	<i>95</i>
<i>Stress Testing:.....</i>	<i>95</i>
<i>Volume Testing:</i>	<i>96</i>
<i>Security & Access Control Testing:</i>	<i>96</i>
<i>Failover & Recovery Testing:.....</i>	<i>96</i>
<i>Configuration Testing:.....</i>	<i>96</i>
<i>Installation/Deploy & Back out Testing:.....</i>	<i>96</i>
<i>Post Production Testing:.....</i>	<i>97</i>
<i>Unit Testing:</i>	<i>97</i>
<i>Smoke Testing:.....</i>	<i>97</i>
<i>Data Migration Testing:</i>	<i>97</i>
<i>Testing Plan used</i>	<i>98</i>
<i>Test reports for Unit Test Cases and System Test Cases</i>	<i>102</i>
<i>Test reports for Unit Test Cases</i>	<i>102</i>
<i>Test reports for System Test Cases.....</i>	<i>104</i>
<i>Debugging and Code improvement:.....</i>	<i>105</i>
<i>Create a Sample with the Debug Class</i>	<i>106</i>
<i>Using the Trace Class.....</i>	<i>107</i>
<i>Verify That It Works</i>	<i>108</i>
<i>Complete Code Listing</i>	<i>109</i>
<i>Troubleshoot.....</i>	<i>110</i>
<i>System Security measures:.....</i>	<i>111</i>
<i>Database/data security:</i>	<i>111</i>
<i>Creation of User profiles and access rights.....</i>	<i>111</i>

<i>Cost Estimation of the Project along with Cost Estimation Model</i>	<i>111</i>
<i>Estimation of development effort</i>	<i>113</i>
<i>Estimation of development time</i>	<i>113</i>
<i>Reports</i>	<i>113</i>
<i>Future scope and further enhancement of the Project</i>	<i>114</i>
<i>Bibliography.....</i>	<i>114</i>
<i>Website.....</i>	<i>114</i>
<i>Books.....</i>	<i>114</i>
<i>Appendices.....</i>	<i>114</i>
<i>IDE Used:</i>	<i>114</i>
<i>Netbeans 7.3.1.....</i>	<i>114</i>
<i>Eclipse IDE for Android</i>	<i>116</i>
<i>Front End</i>	<i>116</i>
<i>XML.....</i>	<i>116</i>
<i>HTML.....</i>	<i>117</i>
<i>Programming Framework.....</i>	<i>118</i>
<i>Codeigniter</i>	<i>118</i>
<i>ADT.....</i>	<i>118</i>
<i>Database/backend:</i>	<i>120</i>
<i>MySQL</i>	<i>120</i>
<i>SQLite</i>	<i>123</i>
<i>IDE for Database</i>	<i>123</i>
<i>MySQL workbench</i>	<i>123</i>
<i>Programming Language.....</i>	<i>125</i>
<i>Java</i>	<i>125</i>
<i>PHP.....</i>	<i>126</i>
<i>Libraries.....</i>	<i>126</i>
<i>Twitter Bootstrap.....</i>	<i>126</i>
<i>Flexi Auth.....</i>	<i>127</i>
<i>Mahana Messaging Library.....</i>	<i>127</i>
<i>Other technologies</i>	<i>127</i>
<i>REST PROTOCOL.....</i>	<i>128</i>
<i>Languages and Time Zones</i>	<i>130</i>
<i>Security.....</i>	<i>130</i>
<i>Version Control System: GitHub</i>	<i>131</i>

<i>Description.....</i>	<i>131</i>
<i>Limitations and constraints</i>	<i>131</i>
<i>Glossary.</i>	<i>131</i>

Introduction & Objective of the Project:

Introduction

In our busy life it is very hard to track our fitness policy by proper guidance. It is also very difficult to maintain schedule for exercise to our health. Mobile technology or elaborately Internet technology is the main trend among us. We like to simplify our daily life by using modern technology. We can track easily whatever we have done or eaten for our health. We usually go to gym or yoga for body fitness and want to keep track of our work to keep our body fit by the guidance of some trainer. But it is very difficult to track the whole process manually. We can do the same things in different ways. With advent of new technologies we use smart phone, notebooks, tablets etc. We could develop an app for Android to keep tracking this process and which will also be synced with the particular website.

Fitness Tracker will provide an interface to organize our daily exercise or activity chart, exercise description, gym schedule etc. in Android device. And it will also share with the same thing in web server for website interface. It will allow users to add same tracking notes to the web sites. User can add details anytime and keep it for update, and then whenever user comes online the pending updates will be uploaded to the destination sites.

Fitness Tracker will help users to track his fitness details from Indian Sub-continent Context. It will have preloaded database for Indian cuisine and exercises more suitable for Indian people.

Main features of the Fitness Tracker are:

- ❖ Tracking daily exercise
- ❖ Tracking Daily food consumption
- ❖ Tracking Weight
- ❖ Exercise chart & description
- ❖ Diet & food chart
- ❖ Gym schedule etc.
- ❖ Updating the details in Web Sites from mobile device ,
- ❖ Web Sync
- ❖ Reminder/ alarm for Gym Schedule
- ❖ Weekly, Monthly or yearly progress view

The overview of this software is displayed below in the diagram.



Figure: Overview of Fitness Tracker

Objective

- ❖ Single solution for fitness related track in mobile device as well as in web site
- ❖ Keeping track of daily fitness related activities
- ❖ Timely update whenever internet is available
- ❖ Manually update data in internet whenever user want
- ❖ Guide user to reach the goal of his/her fitness
- ❖ To be involved with full-fledged software development
- ❖ To know about new technologies like Android, CodeIgniter, Rest API, etc.

System Analysis

Identification of Need:

From our personal life, we know about the necessity of fitness tracking system. Generally, we used to write regularly about the fitness tracking system and our daily activities or try to memorise it. Now a day, we use smart device and web sites as our daily work tracker. The main problem is the availability of internet; we cannot share our thoughts while we are at such a place. Therefore, we might forget our schedule or plan. I faced this problem many times. I have been facing these problems for many years so I felt that we need a solution that could minimize the effort to memorize all the events or schedules.

Preliminary Investigation:

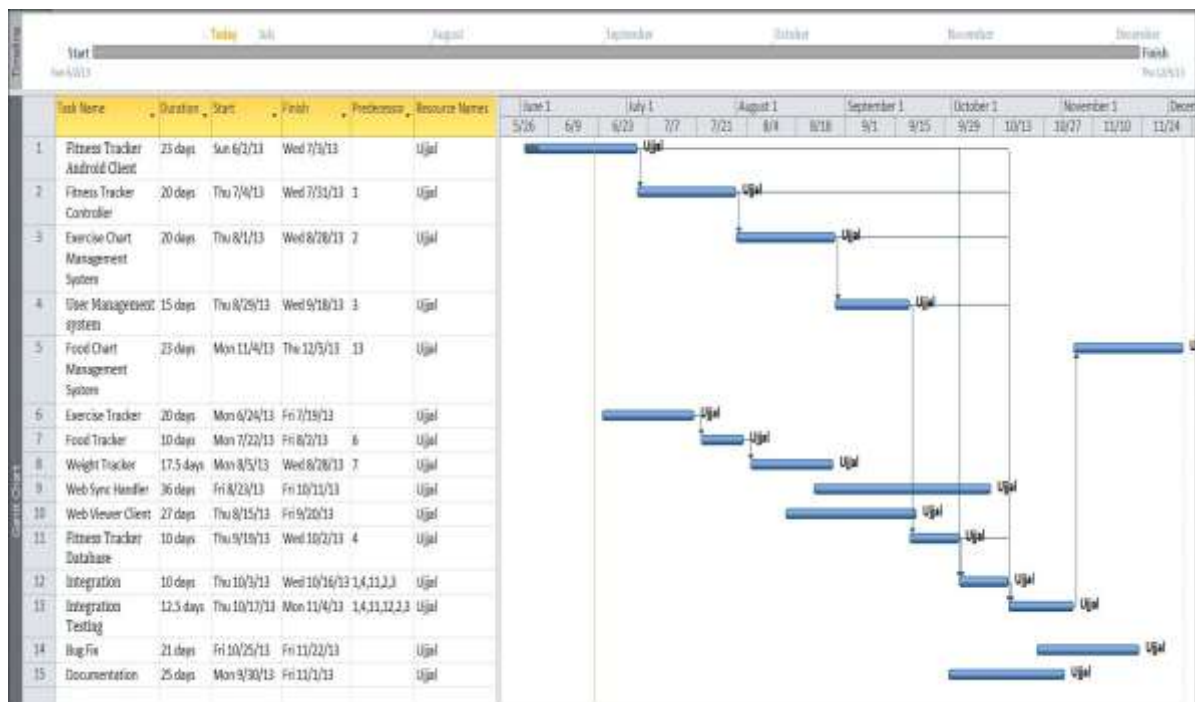
I spoke with many of my friends who use smart phones and are conscious about their health. Most of them face similar kind of problem. I thought a desktop cum web application with an android app could be developed to minimize these problems. I then started gathering opinion of my friends and seniors among whom some are IT professionals. I gathered all the important points including my own opinion and decided to develop Fitness Tracker.

Feasibility Study:

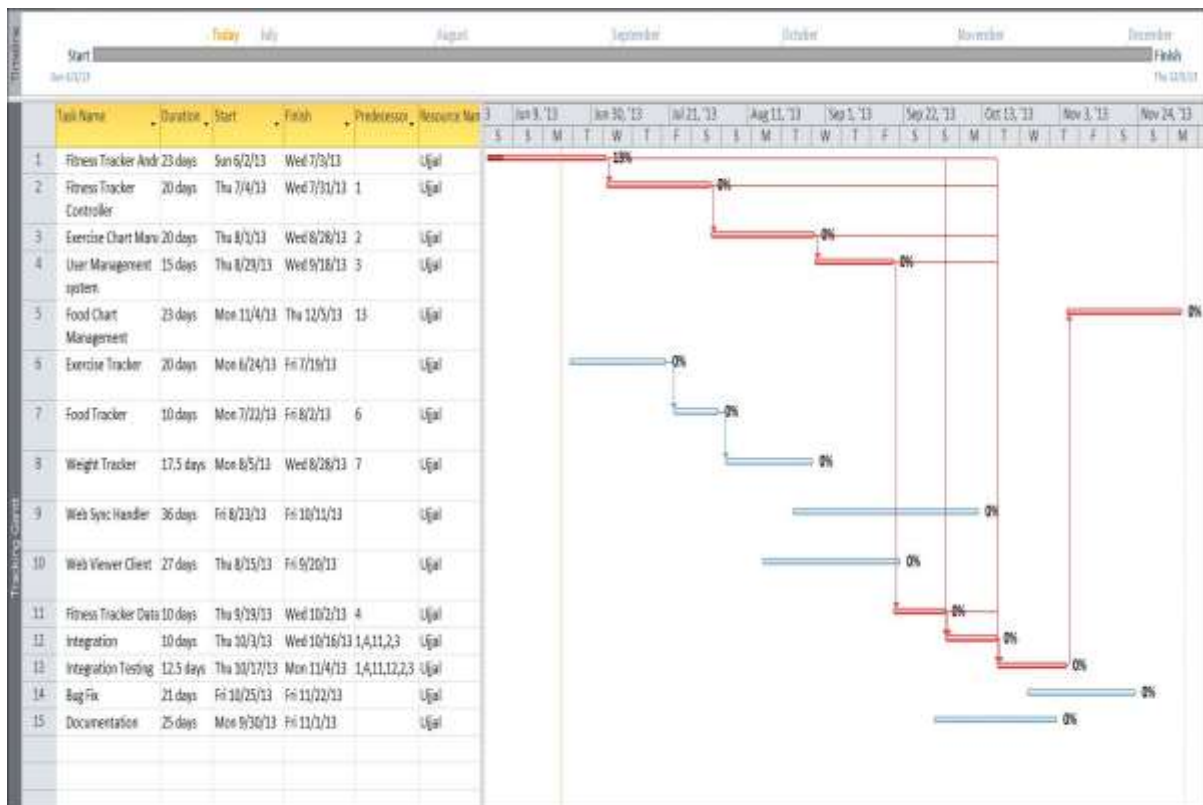
It is an admitted fact that people are becoming more and more addicted to smart mobile technology day by day. People would love an application that would make their memorising effort less. Undoubtedly, it is going to be a popular web cum android application.

Planning and Scheduling

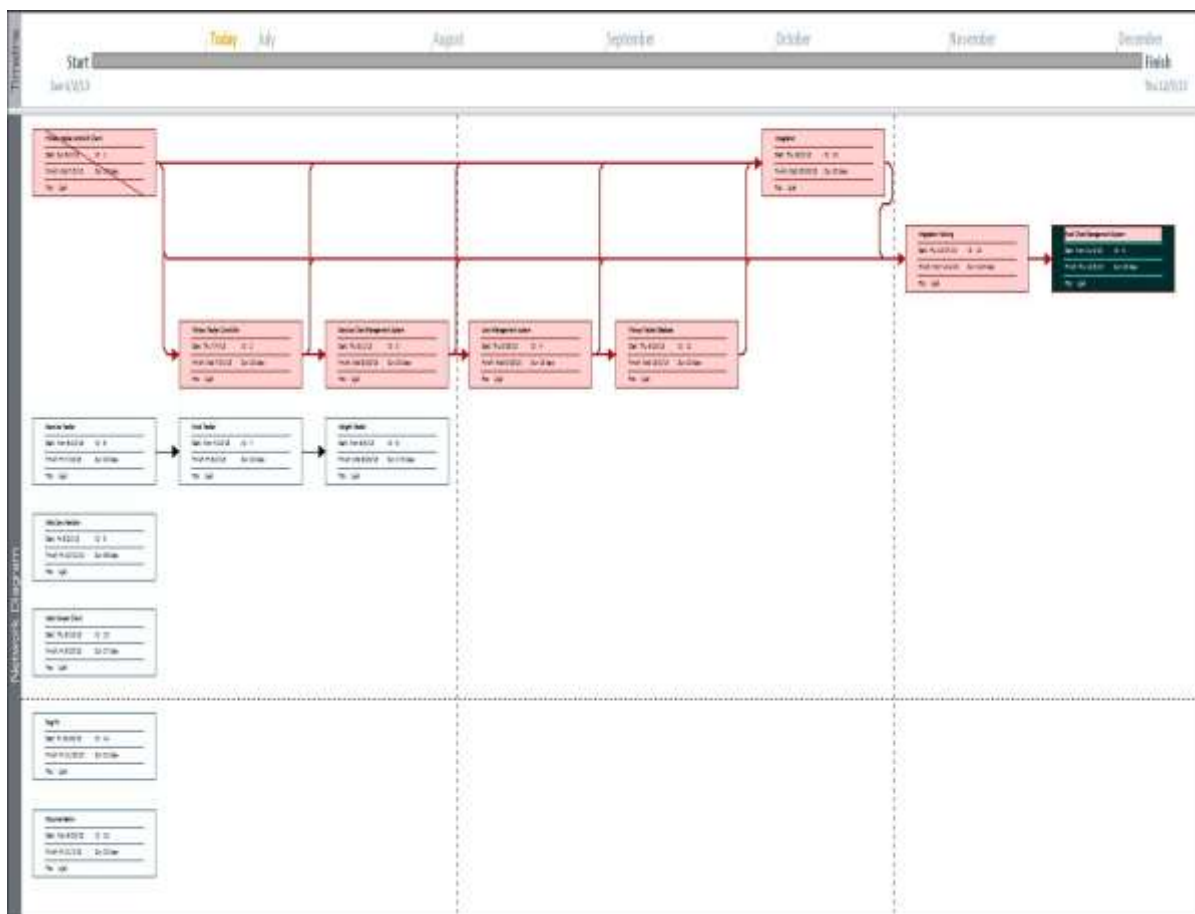
Gantt chart



Tracking Gantt



Pert Chart



Estimation

Timeline	<div> <div>Today</div> <div>August 1</div> <div>September 21</div> <div>November 11</div> </div>					
	Start					Finish
	Sun 6/2/13					Thu 12/5/13
Task Sheet	Task Name	Duration	Start	Finish	Predecessor	Resource Names
	1 Fitness Tracker Android Client	23 days	Sun 6/2/13	Wed 7/3/13		Ujjal
	2 Fitness Tracker Controller	20 days	Thu 7/4/13	Wed 7/31/13	1	Ujjal
	3 Exercise Chart Management System	20 days	Thu 8/1/13	Wed 8/28/13	2	Ujjal
	4 User Management system	15 days	Thu 8/29/13	Wed 9/18/13	3	Ujjal
	5 Food Chart Management System	23 days	Mon 11/4/13	Thu 12/5/13	13	Ujjal
	6 Exercise Tracker	20 days	Mon 6/24/13	Fri 7/19/13		Ujjal
	7 Food Tracker	10 days	Mon 7/22/13	Fri 8/2/13	6	Ujjal
	8 Weight Tracker	17.5 days	Mon 8/5/13	Wed 8/28/13	7	Ujjal
	9 Web Sync Handler	36 days	Fri 8/23/13	Fri 10/11/13		Ujjal
	10 Web Viewer Client	27 days	Thu 8/15/13	Fri 9/20/13		Ujjal
	11 Fitness Tracker Database	10 days	Thu 9/19/13	Wed 10/2/13	4	Ujjal
	12 Integration	10 days	Thu 10/3/13	Wed 10/16/13	1,4,11,2,3	Ujjal
	13 Integration Testing	12.5 days	Thu 10/17/13	Mon 11/4/13	1,4,11,12,2,3	Ujjal
	14 Bug Fix	21 days	Fri 10/25/13	Fri 11/22/13		Ujjal
	15 Documentation	25 days	Mon 9/30/13	Fri 11/1/13		Ujjal

Problem Definition

We used to maintain our trainers guideline without any perfect assumption or calculation. We could not track the accentual progress or digress by the manual calculation. But in the age of digitalization, we are completely dependent on mobile device and internet connection. We are making our daily life easier and faster using the modern technology. So we can track our fitness schedule by use this modern technology. Time to time update regarding fitness via mobile device could make a perfect calculation of health status. The purpose of Fitness Tracker is to store those activities digitally, make schedule, get progress report etc. With an attractive and easy to use Android GUI it will allows us to track down our daily activity. We could also share them with our web account as well whenever we wish. We could also track our activity regarding the fitness via web site also, whenever we have no mobile device available as well.

Requirements Specification

Functional Requirement

Register User

Introduction:

Create account for a new User.

Input:

Relevant User data like user name, current weight, your goal weight, date of birth, gender etc.

Processing:

The Fitness Tracker will create a new user entry.

Output:

The Fitness Tracker will generate a user to reach his goal of fitness.

Login User

Introduction:

Log in as an existing User.

Input:

User will provide user id, password.

Processing:

The **Fitness Tracker** will check the authorization of the particular user.

Output:

The **FITNESS TRACKER** will allow accessing feature to the user if the given data match with the internal information, otherwise denying user.

Add food entry for tracking

Introduction:

User can add new entry.

Input:

User will enter details about Meals (Breakfast, Lunch, and Dinner etc.), Water, and Note etc.

Processing:

The **FITNESS TRACKER** will create a new entry with food value.

Output:

The **FITNESS TRACKER** will save entry in its records and share it in web server.

Add exercise entry for tracking

Introduction:

User can add new entry.

Input:

User will enter details about Exercise (Cardio, Strength etc.), Note etc.

Processing:

The **FITNESS TRACKER** will create a new entry with exercise value.

Output:

The **FITNESS TRACKER** will save entry in its records and share it in web server.

Add weight entry for tracking

Introduction:

User can add his weight on that day.

Input:

User will enter details about weight.

Processing:

The **FITNESS TRACKER** will create a new entry with weight value.

Output:

The **FITNESS TRACKER** will save entry in its records and share it in web server.

Add To Do

Introduction:

The **FITNESS TRACKER** will Store a new task in To Do.

Input:

Relevant task data like task detail, priority etc.

Processing:

Admin will enter the data in the **FITNESS TRACKER** and create a new task entry.

Output:

The **FITNESS TRACKER** will generate a task detail.

Share details in web server

Introduction:

User can manually share details in web server via **FITNESS TRACKER**.

Input:

The user selects the sync option from **FITNESS TRACKER** to upload data in web server.

Processing:

The **FITNESS TRACKER** will upload data in web server.

Output:

The Web site will display the information after sync.

Add User's new food

Introduction:

The user can store new food in **FITNESS TRACKER** database.

Input:

Relevant food data like food name, description, serving size, calories etc.

Processing:

Admin will enter the data in the **FITNESS TRACKER** and create a new food entry.

Output:

The **FITNESS TRACKER** will display the food whenever user want to enter that.

Create food/ diet chart

Introduction:

User can create his food/diet chart.

Input:

User will enter details about Meals (Breakfast, Lunch, Dinner etc.), Water to be taken regularly with help of dietitian or trainer.

Processing:

The **FITNESS TRACKER** will create food/diet chart.

Output:

The **FITNESS TRACKER** will save entry in its records and share it in web server.

Create exercise chart

Introduction:

User can create his exercise chart.

Input:

User will enter details about exercises to be performed regularly with help of dietitian or trainer.

Processing:

The **FITNESS TRACKER** will create exercise chart.

Output:

The **FITNESS TRACKER** will save entry in its records and share it in web server.

View weekly/monthly report

Introduction:

User can view his weekly/monthly.

Input:

User will enter his choice of duration.

Processing:

The **FITNESS TRACKER** will create report of that duration.

Output:

The **FITNESS TRACKER** will display the report in graphical and textual format.

Non-functional Requirements

- ❖ The application will be **self-dependent** and no dependency on other parties required.
- ❖ There will be a digital **backup** and restore system.
- ❖ There will be more **opportunity** to extend the application in various type of device in future.
- ❖ The response time will be low and the system will **response fast**.
- ❖ It will be very **user friendly** and **usable** by any person with minimal technical knowledge.
- ❖ In terms of **security** unauthorized access will be denied and register user will be able to change as necessary.
- ❖ It will be **efficient** as it reduce manual labor and searching.
- ❖ **FITNESS TRACKER** will have user manual and help **documents**.
- ❖ It is designed such a way that it can be **maintained** with minimal effort.

Technical Specification

Front End/ GUI Tools	:	<i>XML</i>
IDE	:	<i>Eclipse</i>
Framework	:	<i>ADT (Android Developer Toolkit) , Code Igniter, Bootstrap</i>
Database	:	<i>SQLite</i>
Database Tool	:	<i>SQLite workbench CE</i>
Operating Systems	:	<i>Windows XP, Windows 7 , Android</i>
Others	:	<i>Android SDK</i>

Software Engineering Paradigm applied

We have followed agile version of Model Driven Development (MDD). As the name implies, AMDD is the agile version of Model Driven Development (MDD). MDD is an approach to software development where extensive models are created before source code is written. A primary example of MDD is the Object Management Group (OMG)'s Model Driven Architecture (MDA) standard. With MDD a serial approach to development is often taken, MDD is quite popular with traditionalists, although as the RUP/EUP shows it is possible to take an iterative approach with MDD. The difference with AMDD is that instead of creating extensive models before writing source code you instead create agile models which are just barely good enough that drive your overall development efforts. AMDD is a critical strategy for scaling agile software development beyond the small, co-located team approach that we saw during the first stage of agile adoption.

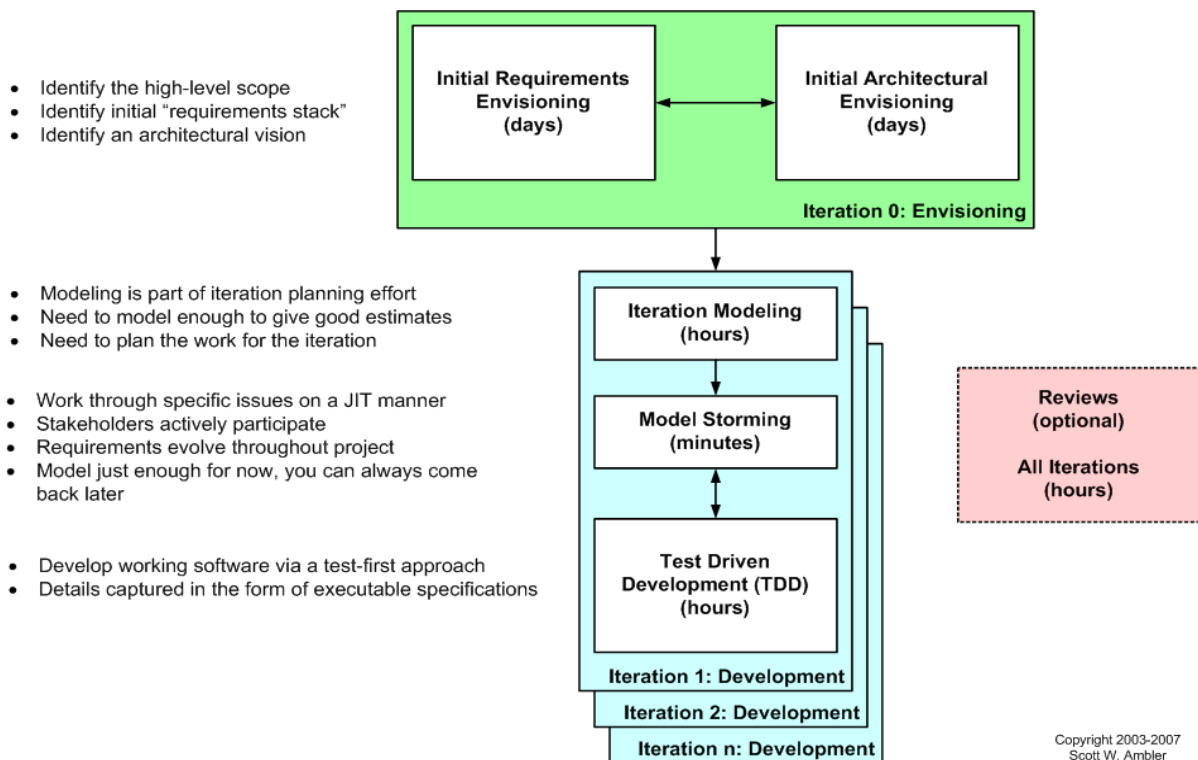


Figure 1: The AMDD lifecycle: Modeling activities throughout the lifecycle of a project

Above Figure depicts a high-level lifecycle for AMDD for the release of a system. First, let's start with how to read the diagram. Each box represents a development activity. The envisioning includes two main sub-activities, initial requirements envisioning and initial architecture envisioning. These are done during iteration 0, iteration being another term for cycle or sprint. "Iteration 0" is a common term for the first iteration before you start into development iterations, which are iterations one and beyond (for that release). The other activities – iteration modeling, model storming, reviews, and implementation – potentially occur during any iteration, including iteration 0. The time indicated in each box represents the length of an average session: perhaps you'll model for a few minutes then code for several hours. I'll discuss timing issues in more detail below..

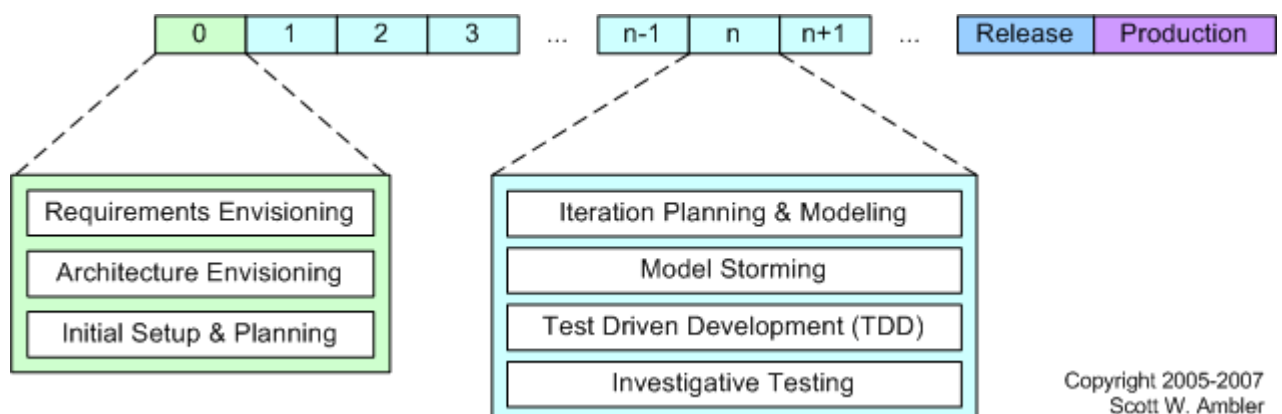


Figure 2AMDD Through the Agile Development Lifecycle.

Above Figure depicts how the AMDD activities fit into the various iterations of the agile software development lifecycle. It's simply another way to show that an agile project begins with some initial modelling and that modelling still occurs in each construction's iteration.

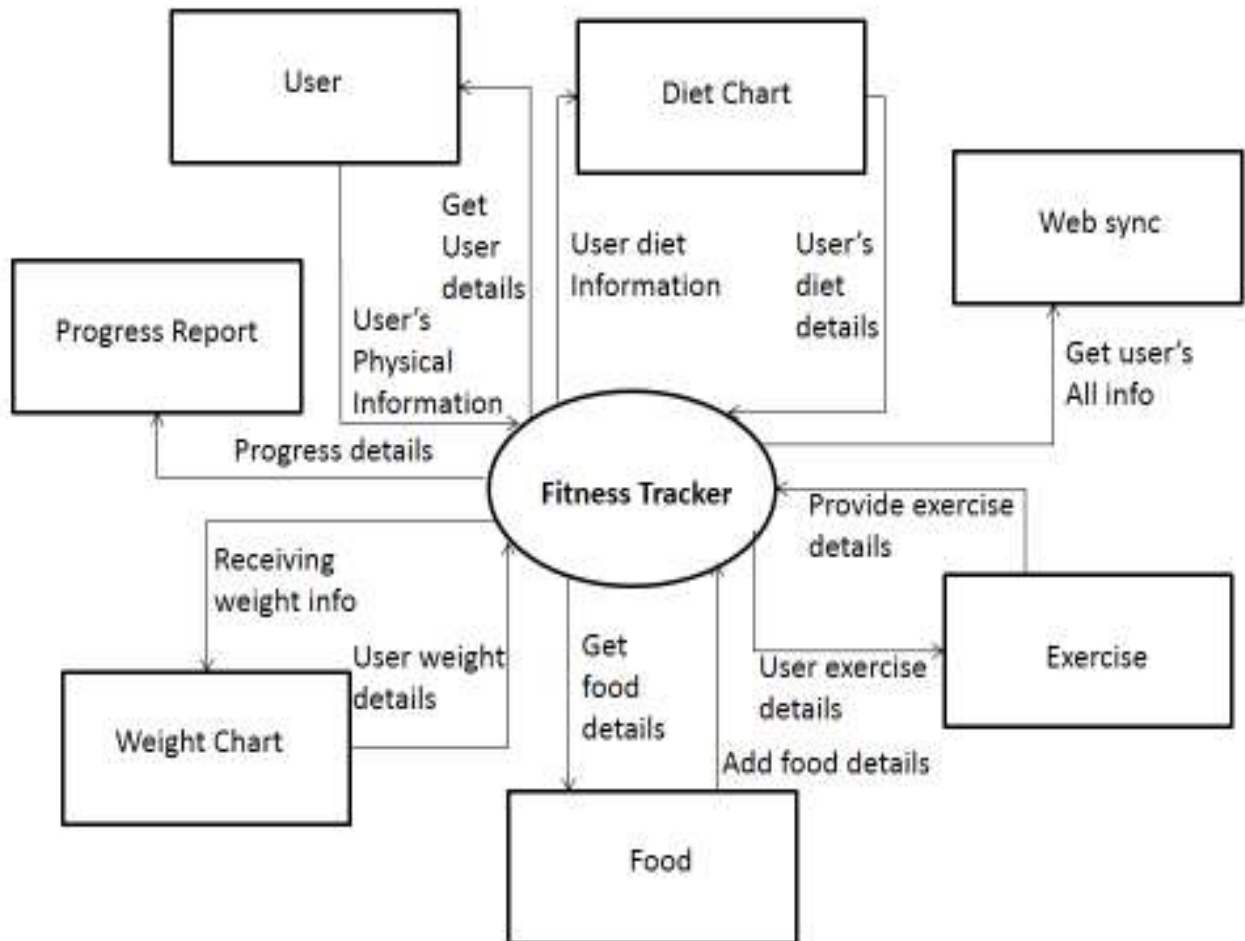


Figure 3:Context Diagram of FTS

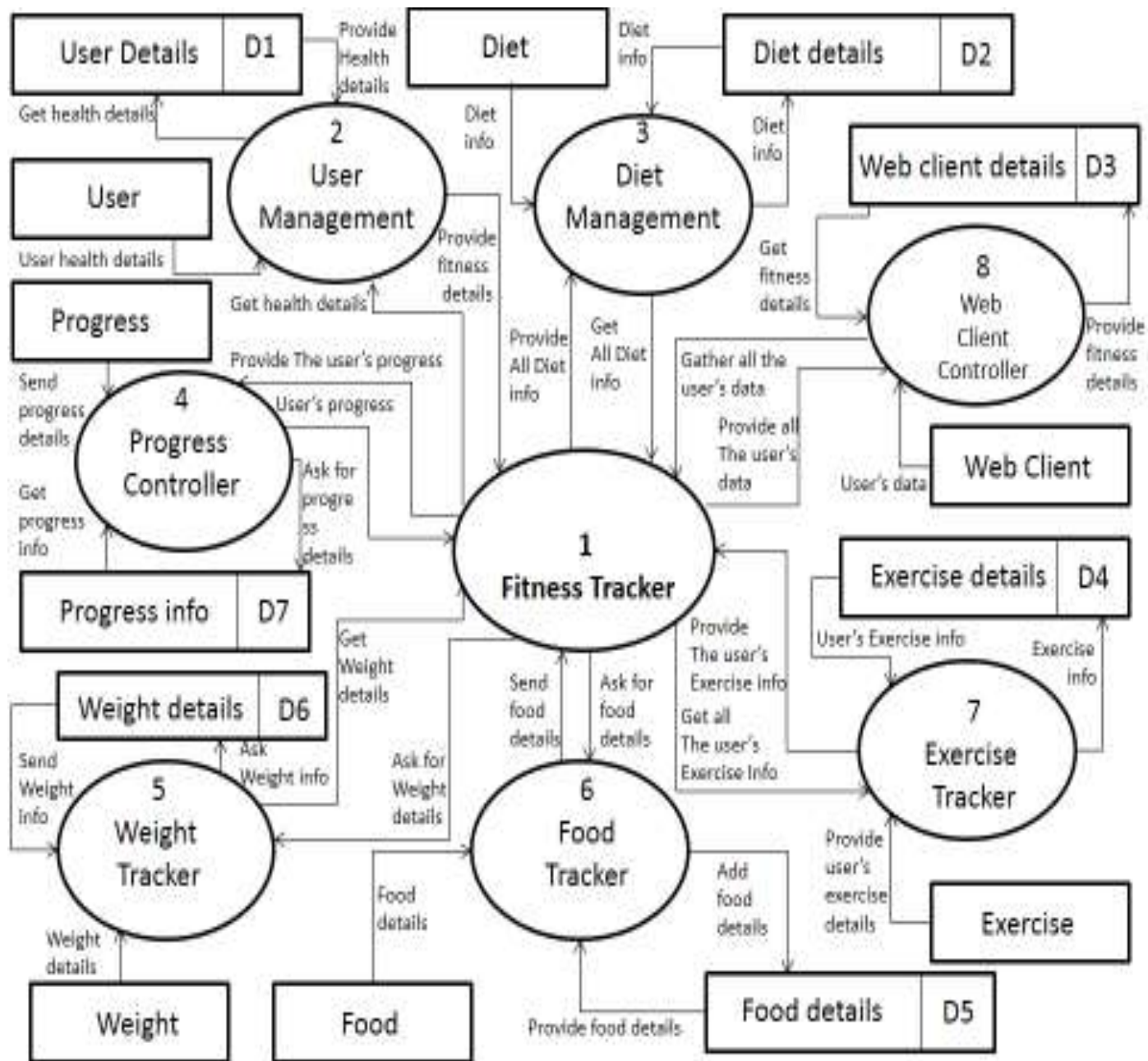


Figure 4:0 level DFD

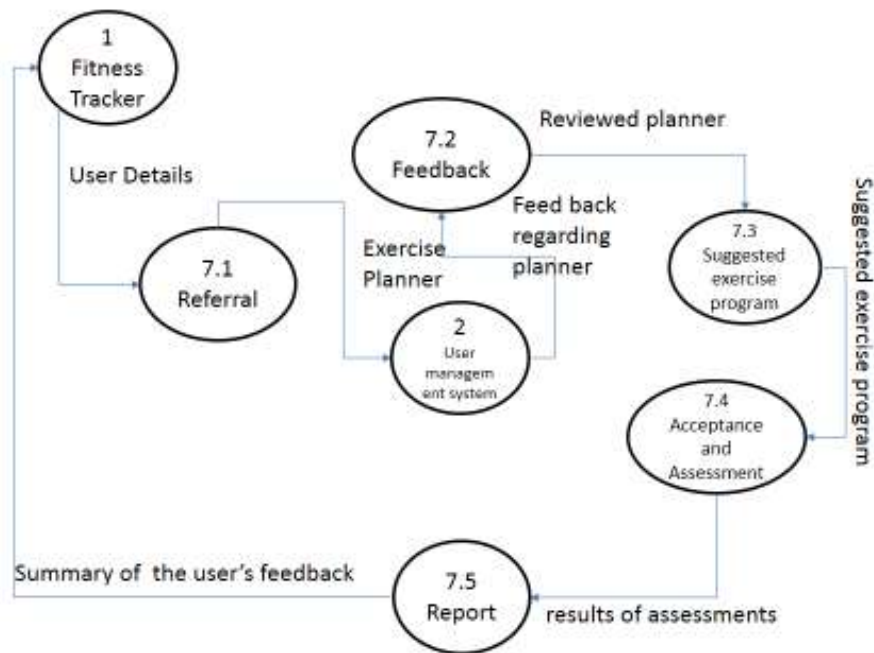


Figure 5:1 level DFD

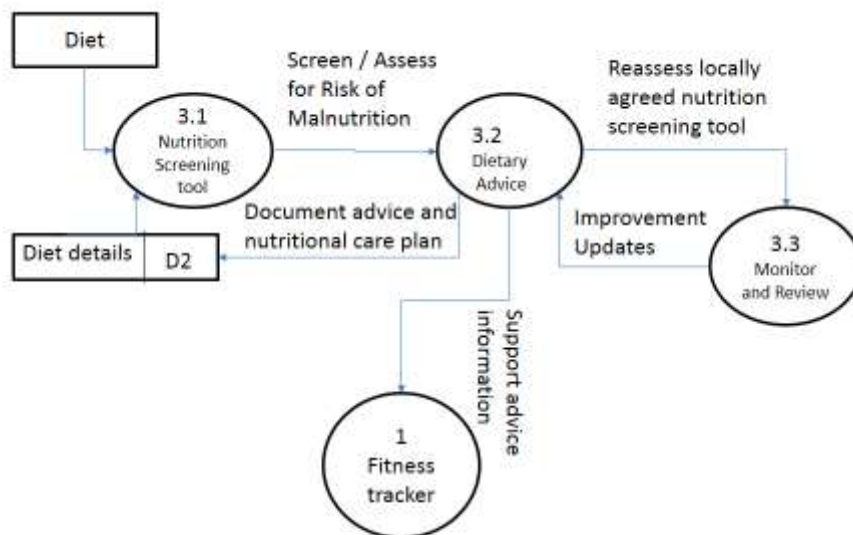


Figure 6:1 level DFD

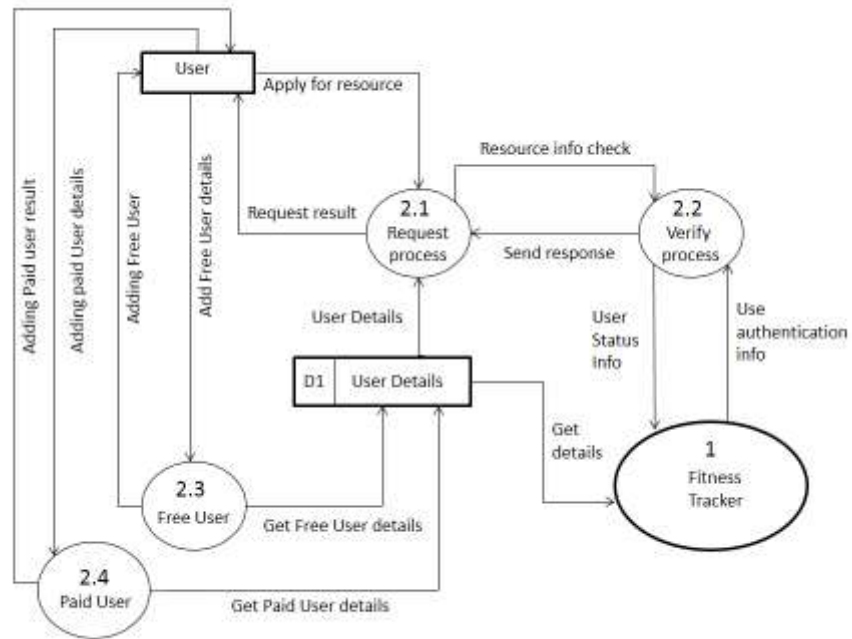


Figure 7:1 level DFD

2-Level DFD

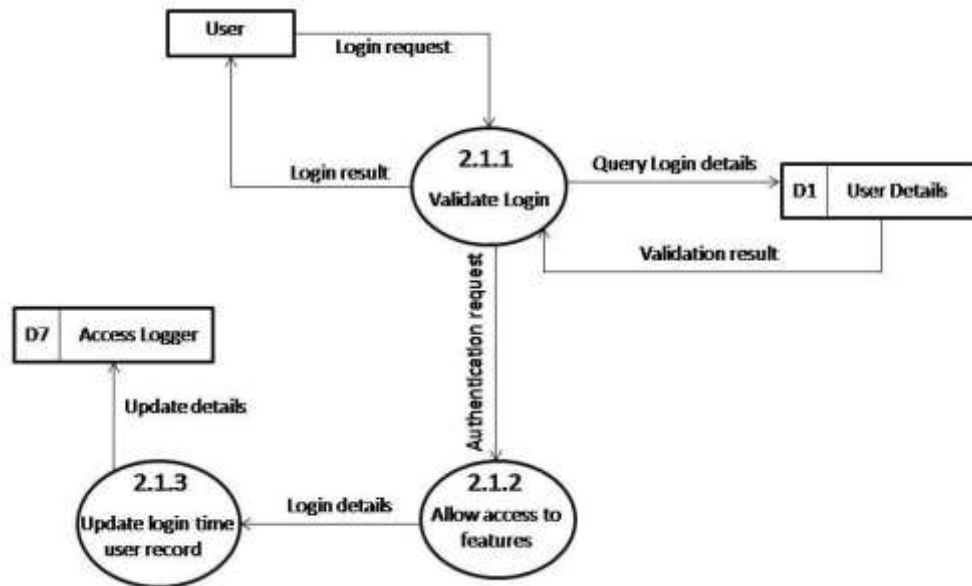
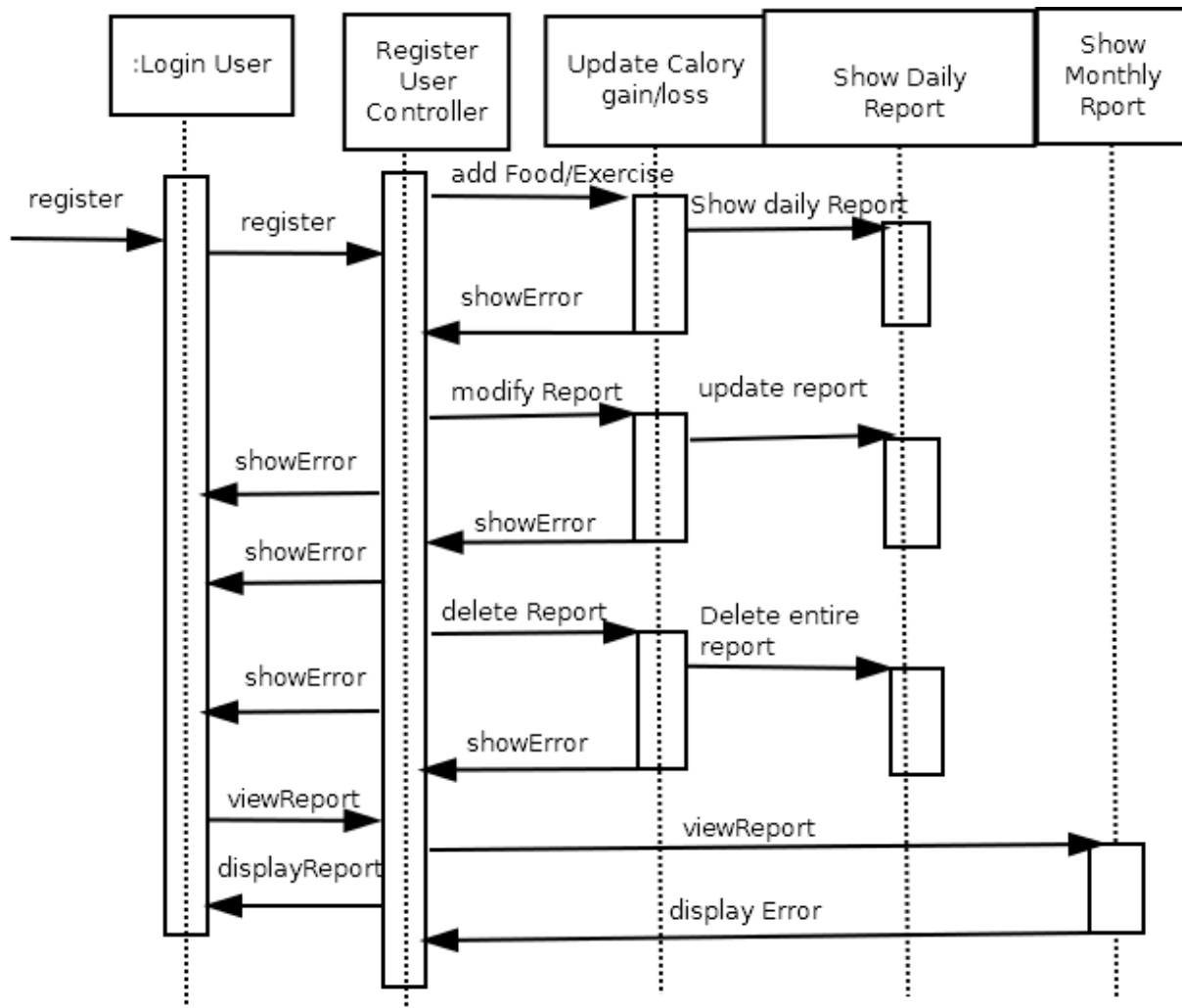


Figure 8:2 level DFD

Sequence diagrams



E-R Diagram

We will design a RDBMS for **FITNESS TRACKER**. The entities and their attributes are listed below. Attributes in Bold letter is the unique key.

Entities	Attributes
User	Id, user Name, trainer, full Name, Image, dob, weight, Exercise Chart, diet Chart, exercises and foods.
Weight Chart	Id, user name, current Weight Value, target Weight Value, Average Weight Value, unit, assigning Date, target Date.
Food Item	Id, calorie Count, protein Count, fat Count, name, preferred Quantity, image.

Diet Chart	Id, username, trainer Name, breakfast, lunch, dinner, snacks; target Daily Value, assigning Date, target Date.
Exercise Item	Id, name, type, calorie burnt, sets, repetition, Preferred Duration, preferred Weight, Image, video, special Instructions.
Exercise Chart	Id, username, trainer Name, exercises, start Time, duration, Weekly Total Duration, target Daily Value, Assigning Date, target Date.
Trainer	Id, username, image, website, trainees.
Report	Id, date, type, description.

Relationship between Entities:

- ❖ FITNESS TRACKER has User → 1 : N
- ❖ Users add entries → 1 : N

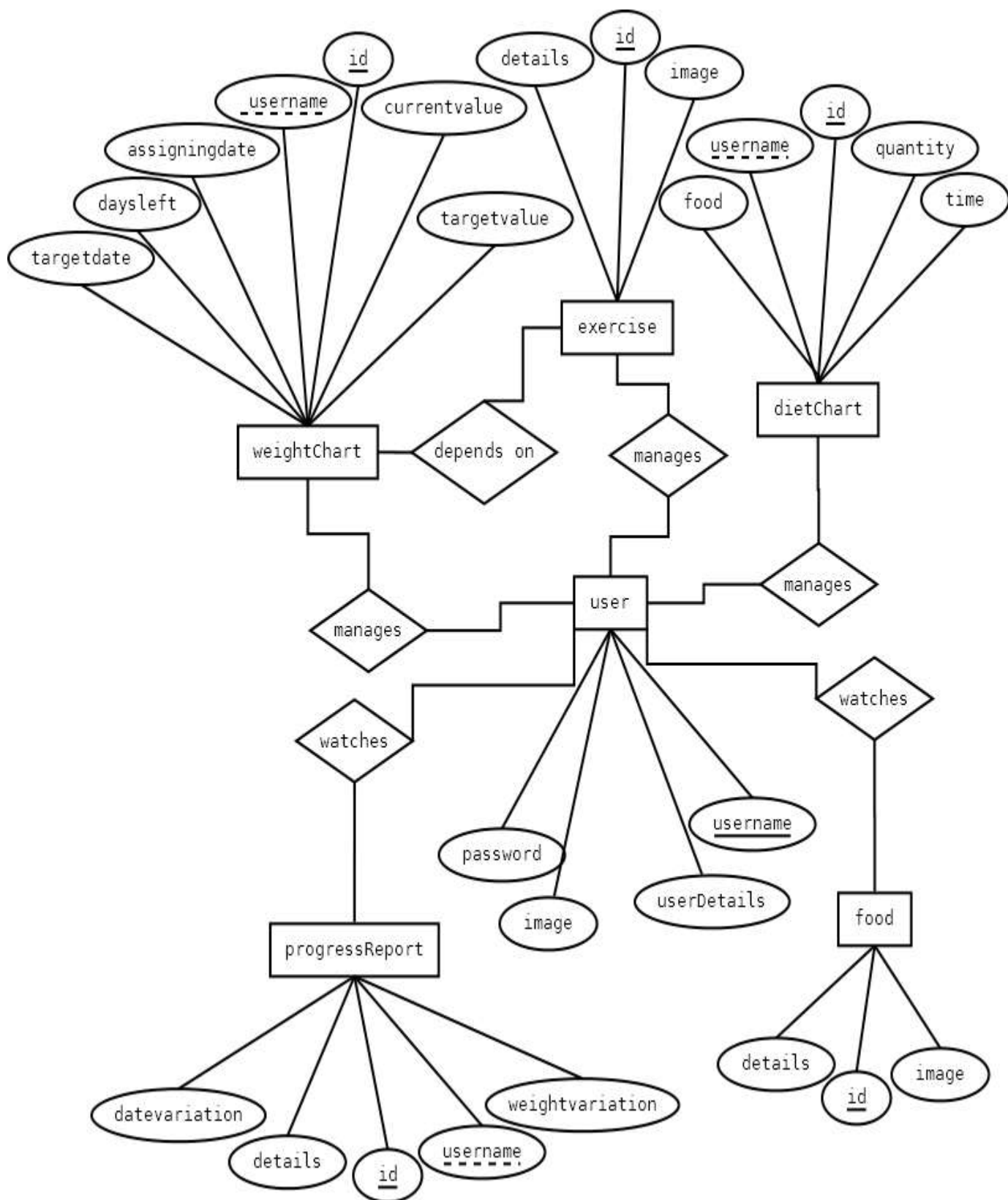
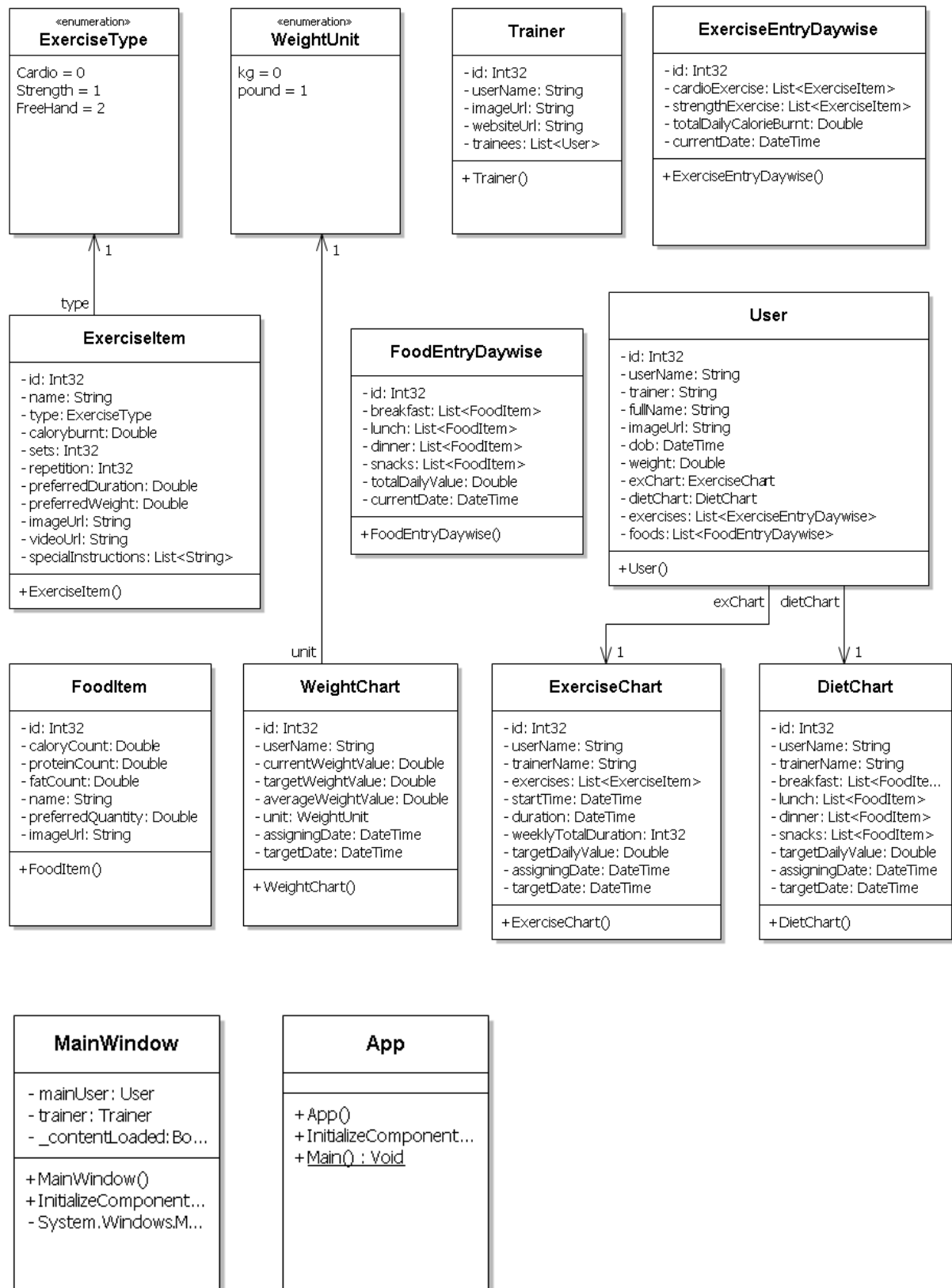
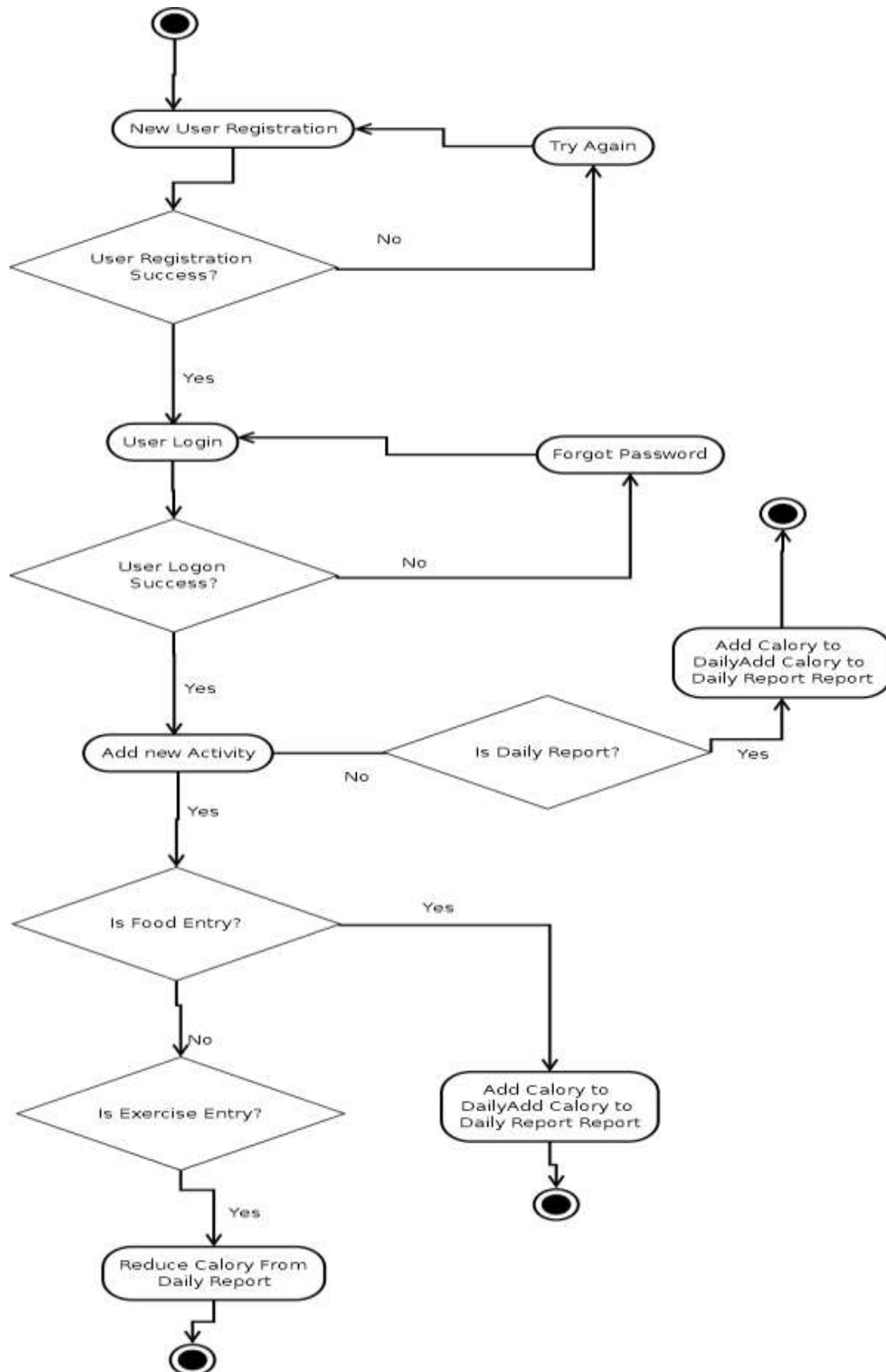


Figure 9:E-R Diagram of FITNESS TRACKER

Class Diagram



Activity Diagrams



System Design

Modularisation details

The picture below represents various modules of **FITNESS TRACKER**. Their detailed description is written below:

FITNESS TRACKER Engine:

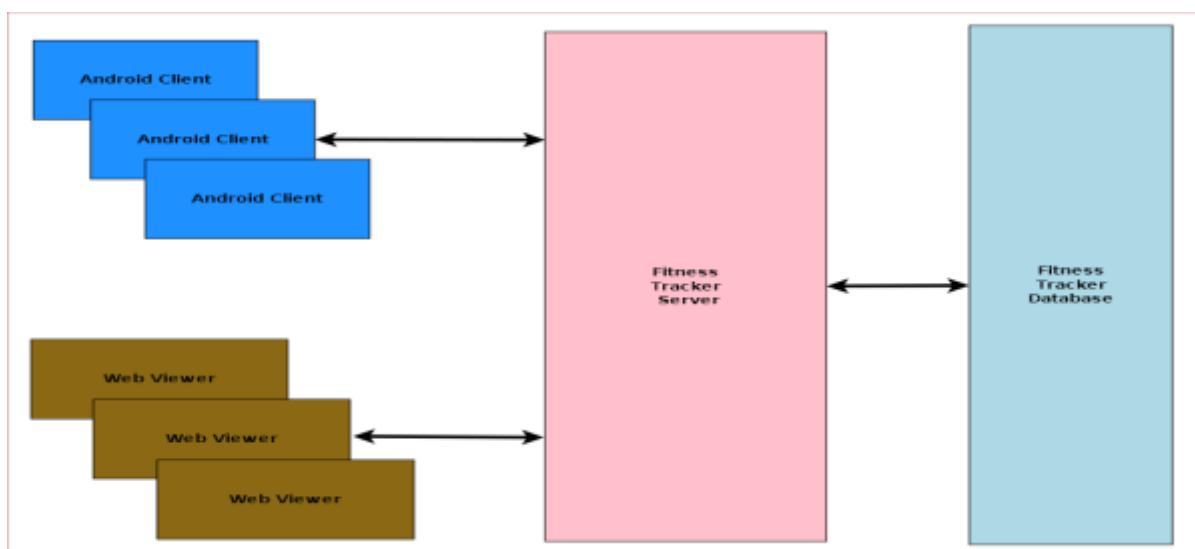
This module handles all the logical parts of **FITNESS TRACKER**. It takes data from user through **FITNESS TRACKER** GUI module and stores them to SQLite database using **FITNESS TRACKER** Storage module. It sends the data to the user's web server account to get the details from web site. This has following components:

1. Food Tracker
2. Exercise Tracker
3. Weight Tracker
4. Exercise Chart Management System
5. Food Chart Management System
6. Web Sync Handler

FITNESS TRACKER GUI:

This part is the place through which user interacts. This module contains all the designs which are visible and intractable by users. User provides input through it and gets the output through it. It is generally created by various tools like Buttons, EditText etc. . This has following components:

1. Android App
2. Web Viewer



FITNESS TRACKER Storage:

In this module all the data are stored. **FITNESS TRACKER** Engine stores data in this module and fetches them for output through this module.

FITNESS TRACKER web site:

This place gets input from the **FITNESS TRACKER** Storage. All the relevant data sent by user to server, which could be accessed by user globally.

Data integrity and constraints

We have used Integrity constraints in **FT** to ensure accuracy and consistency of data in a relational database. Data integrity is handled in a relational database through the concept of referential integrity. There are many types of integrity constraints in **FT** that play a role in referential integrity.

Codd initially defined two sets of constraints but, in his second version of the relational model, he came up with four integrity constraints:

Entity integrity

In **FT** we used various type of primary key and consciously we set the primary key property as not null. The entity integrity constraint states that no primary key value can be null. This is because the primary key value is used to identify individual tuples in a relation. Having null value for the primary key implies that we cannot identify some tuples. This also specifies that there may not be any duplicate entries in primary key column key row.

Referential Integrity

The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations. Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation. It is a rule that maintains consistency among the rows of the two relations.

Domain Integrity

FT has various type of data field with set by default value of Null because if the value is not provided by the user, the value will be set as null. The domain integrity states that every element from a relation should respect the type and restrictions of its corresponding attribute. A type can have a variable length which needs to be respected. Restrictions could be the range of values that the element can have, the default value if none is provided, and if the element can be NULL.

User Defined Integrity

A business rule is a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behaviour of the business.

Database design

Database tables and corresponding keys are shown in tabular form. It shows the tables and its columns. A key in Bold is the primary key.

Screenshots of table structures:

Table: user





Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
 id	VARCHAR(100)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 userid	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 hints	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 passwd	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Table: contact










Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
 contactId	VARCHAR(100)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 name	VARCHAR(145)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 mobile	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'N/A'
 homePhone	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'N/A'
 officePhone	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'N/A'
 email	VARCHAR(75)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'N/A'
 address	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'N/A'
 faxNumber	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'N/A'
 remark	VARCHAR(245)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'N/A'

Table: note





Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
 id	VARCHAR(100)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 note	VARCHAR(10005)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 date	DATETIME	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 sharedWith	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'N/A'

Table: password








Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
 passwordId	VARCHAR(100)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 name	VARCHAR(100)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 password	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 email	VARCHAR(100)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 userId	VARCHAR(100)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'N/A'
 secretQuestion	VARCHAR(200)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'N/A'
 secretAnswer	VARCHAR(200)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'N/A'
 otherInfo	VARCHAR(200)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'N/A'

Table: tasks

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
taskId	VARCHAR(100)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
priority	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
details	VARCHAR(545)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
status	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'N/A'

The database used for this software is called **fitnesstrackerdb**. Database tables and corresponding keys are shown in tabular form. It shows the tables and its columns. A key in **Bold** is the primary key.

Entities	Attributes
User	id, user Name, trainer, full Name, imageUrl, dob, weight, exChart, dietChart, exercises, foods.
WeightChart	id, username, currentWeightValue, targetWeightValue, AverageWeightValue, unit, assigningDate, targetDate.
FoodItem	id, caloryCount, proteinCount, fatCount, name, preferredQuantity, imageUrl.
DietChart	id, username, trainerName, breakfast, lunch, dinner, snacks, target DailyValue, assigningDate, targe Date.
ExerciseItem	id, name, type, caloryburnt, sets, repetition, PreferredDuration, preferredWeight, ImageUrl, videoUrl, specialInstructions.
ExerciseChart	id, username, trainerName, exercises, startTime, duration, WeeklyTotalDuration, targetDailyValue, AssigningDate, targetDate.
FoodEntryDaywise	id, breakfast, lunch, dinner, snacks, totalDailyValue, current Date.
ExerciseEntryDaywise	Id, cardioExercise, strengthExercise, totalDailyCalorieBurnt, currentDate.
Trainer	id, username, imageUrl, websiteUrl, trainees.
Report	id, date, type, description.

User Interface Design

Android User Interface Design

Main Window without Login




Welcome to Fitness Tracker


Please consult with your doctor before making any changes on your food habit

Sign up or Login With Facebook

Sign up with Email/ Phone Number

Login

3G  7:41

 **Fitness Tracker**

Register

Preferred Unit Type

English (lbs, ft, in)

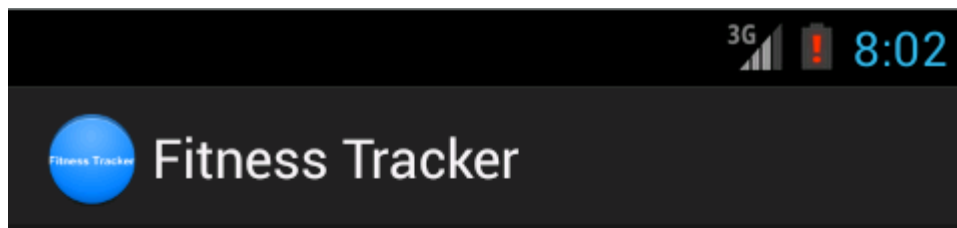
Your current weight

lbs

Your goal weight

lbs

Next



Register

Your sex

Female

Your height

5' 7

Your date of birth


01-01-1981


Daily Activity Level

Active

Previous

Next

3G  8:08

 Fitness Tracker

Register

Choose Username


Password


Confirm Password

Email Address

Previous

Finish

3G  7:26

 **Fitness Tracker**

Email/Phone Login

User Name

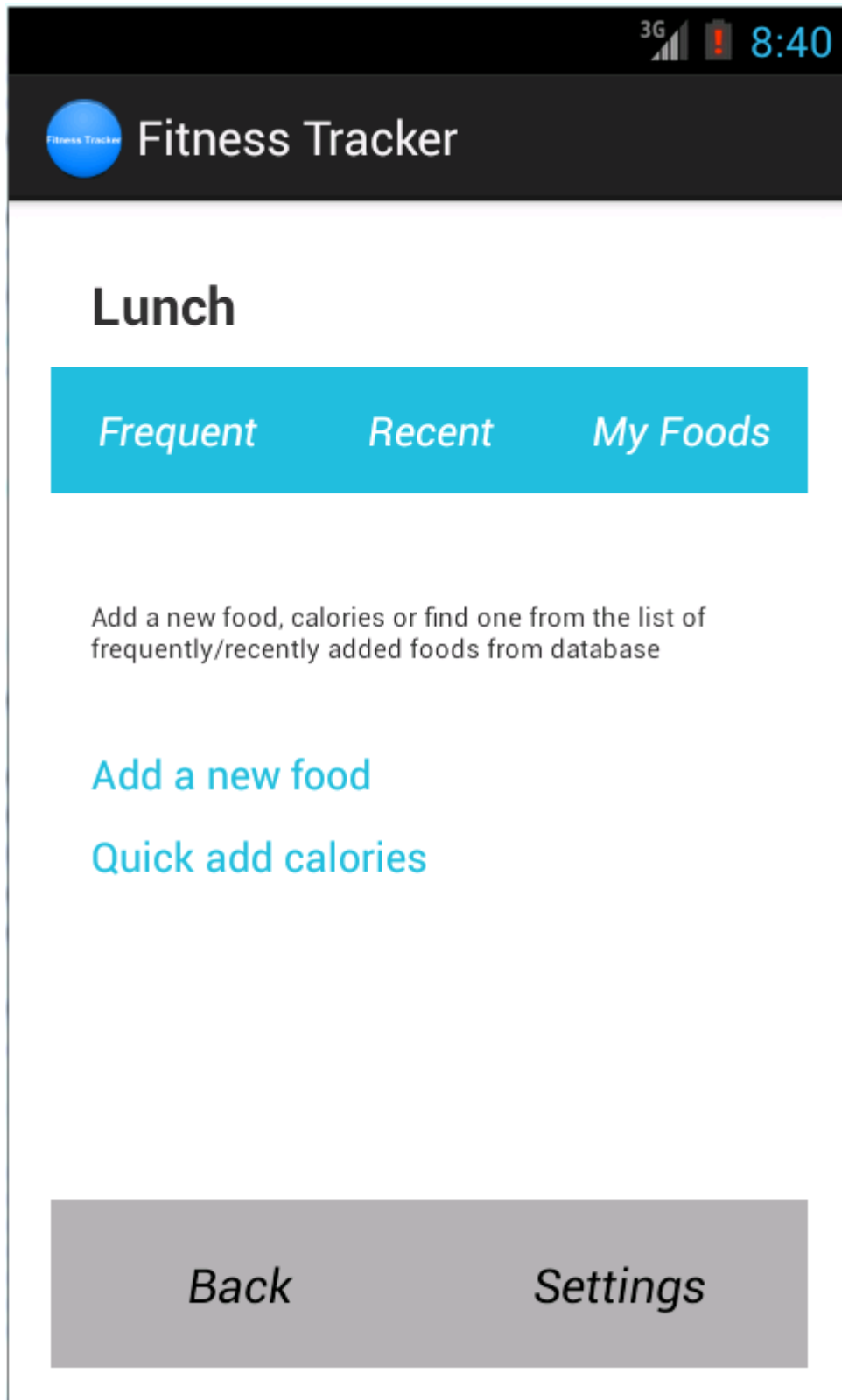
Required

Password


Required


Login

[I forgot my Password](#)



Add exercise details

3G5:49

Fitness Tracker

Exercise

Frequent

Recent

Exercises

Exercise Name

Back push up

Sets Completed

3

Exercise per set

10

Back

Save

3G9:11

Fitness Tracker

sep 24,2013

Daily Progress Report

Goal	Complete	Remaining
1600	1200	400

Caloriy gain through food

1600

Calory loss through exercise


400


Total

1200

Back

Save

3G  8:50

 **Fitness Tracker**

Lunch

Frequent *Recent* *My Foods*

Food Name

Rice Meal

Number of Servings

1

Serving Amount

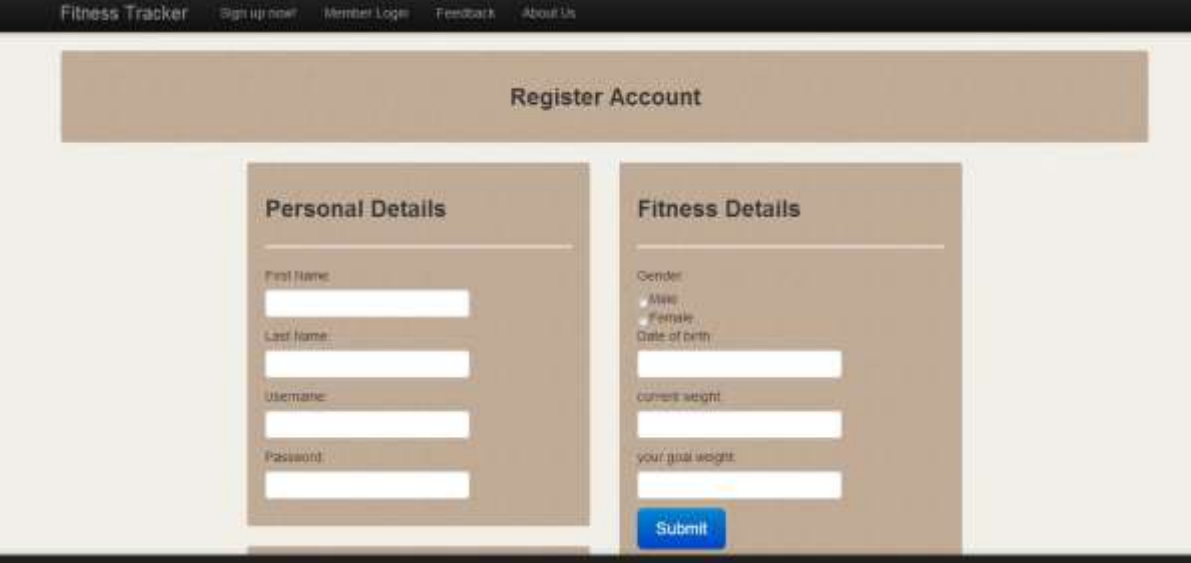
3 Cups

Total Calories

900

Back *Add Entry*

User Registration



The registration form is titled "Register Account" and is part of a "Fitness Tracker" application. It features a navigation bar with links to "Sign up now!", "Member Login", "Feedback", and "About Us". The form is divided into two main sections: "Personal Details" and "Fitness Details".

Personal Details:

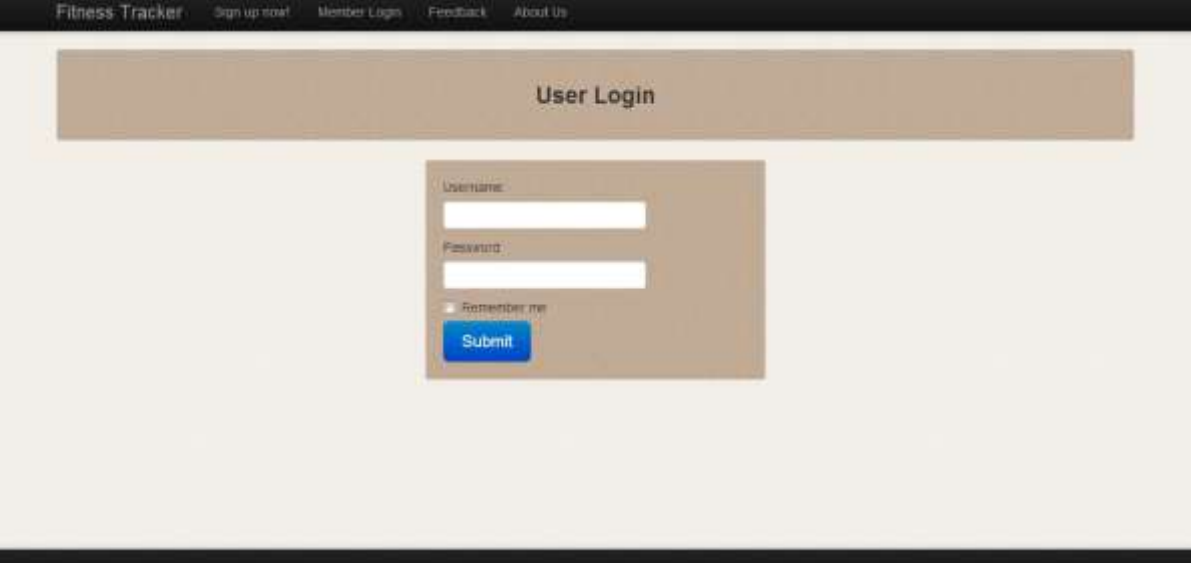
- First Name:
- Last Name:
- Username:
- Password:

Fitness Details:

- Gender: ☐ Male ☐ Female
- Date of birth:
- current weight:
- your goal weight:

A blue "Submit" button is located at the bottom right of the form.

User Login



The login form is titled "User Login" and is part of the same "Fitness Tracker" application. It features the same navigation bar as the registration form. The form is simple, with fields for "Username" and "Password", a "Remember me" checkbox, and a blue "Submit" button.

Username:

Password:

☐ Remember me

Submit

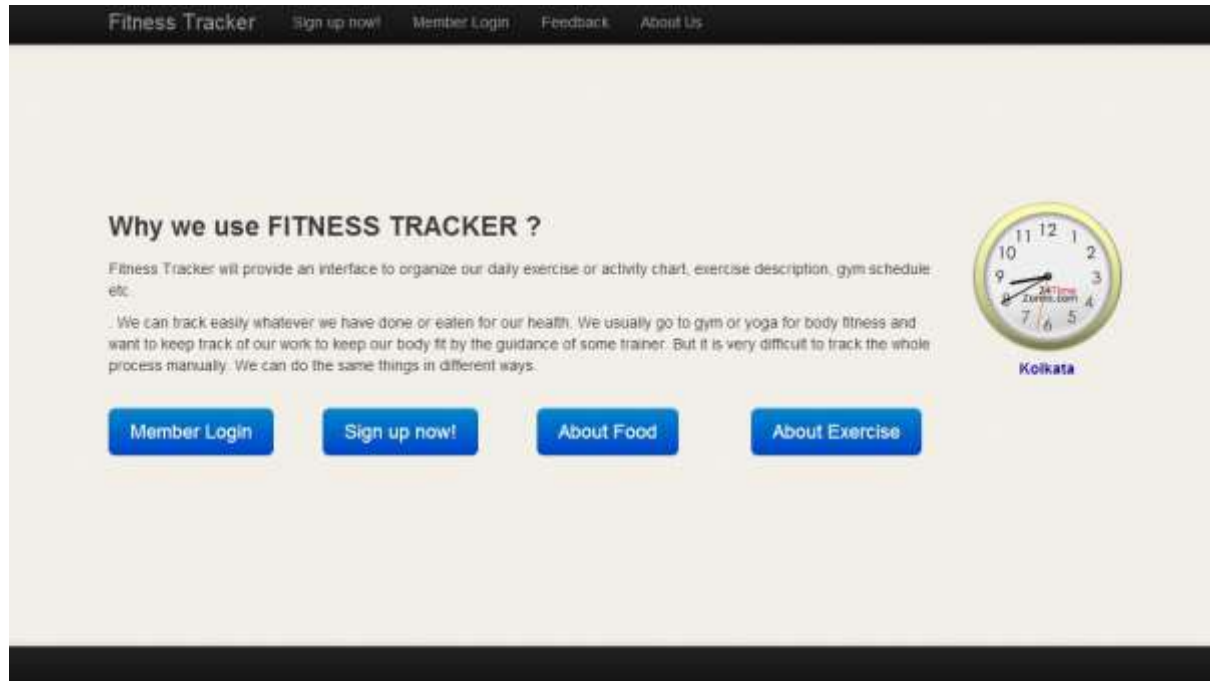
Add Exercise

The screenshot shows the 'Record Activity' page of a web application. The page has a dark header with the title 'Fitness Tracker' and navigation links: 'Sign up how!', 'Member Login', 'Activity', 'Feedback', and 'About Us'. The main content area is titled 'Record Activity' with a subtitle 'Target to lose : 3456 Calory'. Below this, there are three columns. The first column, 'Activity Details', contains a dropdown menu for 'Activity type' with options: 'Meal', 'Meal', 'Light Food', 'Water', 'Exercise', 'Walk', and 'Others'. The 'Walk' option is selected. Below the dropdown is a blue 'Submit' button. The second column, 'Activity type :', shows 'Water' as the selected activity, with 'Quantity : 500 ml' and 'Calory : 24 (v)'. The third column, 'Activity type :', shows 'Walk' as the selected activity, with 'Quantity : 500 ml' and 'Calory : 354 (v)'. The page is styled with a light beige background and dark borders.

Feedback

The screenshot shows a 'Your Valuable Feedback' modal form overlaid on the 'Feedback' page of the Fitness Tracker application. The modal has a white background and a dark border. It contains the following fields: 'Your Full Name' (text input with 'Souman Das'), 'Email' (text input with 'soumandas70@gmail.co'), 'Opinion About' (text input with 'Good site'), and 'Your Opinion' (text area with 'Your site is very nice and useful. Add some forum for interaction between user'). At the bottom right of the modal are 'Close' and 'Submit' buttons. The background page is dark and shows the 'Feedback' title and some text about the fitness tracker.

Home Page



Test Cases (Unit Test Cases and System Test Cases)

Unit Test Cases

Test Case Id	Type	Github ID	Subject	Test Name	Test Description	Step Name	Description	Expected Result
FT-001	Manual	f3563be0a9c431104f52839039e86043cf640cf1	Sign up with FB	Check Successful FB Login for FT	The purpose of this test is to verify that the login through Facebook .com is working or not	Step 1	Insert wrong User Id and Password. And Click on Login Button.	A FB.com window gets opened that asks for username and password of the user's already created fb account

FT-002						Step 2	Insert Wrong User Id and valid Password. And Click on Login Button.	FT will display error message. And Failed to Login.
FT-003						Step 3	Insert Valid User Id and Wrong Password. And Click on Login Button.	FT will display error message. And Failed to Login.
FT-004						Step 4	Insert Nothing in User Id and Password fields. And Click on Login Button.	FT will display error message. And Failed to Login.
FT-005						Step 5	Insert Nothing in User Id and insert Valid Password fields. And Click on Login Button.	FT will display error message. And Failed to Login.
FT-006						Step 6	Insert Nothing in Password and insert Valid User Id fields. And Click on Login Button.	FT will display error message. And Failed to Login.
FT-007						Step 7	Insert Nothing in User Id and insert invalid Password fields. And Click on Login Button.	FT will display error message. And Failed to Login.
FT-008						Step 8	Insert Nothing in Password and insert invalid User Id fields. And Click on Login Button.	FT will display error message. And Failed to Login.
FT-009						Step 9	Insert valid User Id and Password. And Click on Login Button.	Successfully login.

FT-010	M a n u a l	d01197ee4c d3bee92458 74b5937ba7 40019fd131	E:\DEVELOPERS_ ZONE\Git Hub\FitnessTracker\ code	Check Successful Registration for New FT User.	The purpose of this test is to verify that the all new connection could be creating new Account By Registration.	Step 1	Click on Registration link.	New Account creation area is opened.
FT-011						Step 2	Enter existing new User Id, Password, Retype Same password and Hints. And Click on Login Button.	FT will display error message. And Failed to Registration.
FT-012						Step 3	Enter existing new User Id, Password, Retype Same password and Hints. And Click on Login Button.	FT will display error message. And Failed to Registration.
FT-013						Step 4	Enter new User Id, Password, Retype Different password and Hints. And Click on Login Button.	FT will display error message. And Failed to Registration.
FT-014						Step 5	Enter new User Id, Password, Retype Same password and enter nothing in Hints fields. And Click on Login Button.	FT will display error message. And Failed to Registration.

FT-015						Step 6	Enter new User Id, Password, Hints and nothing in Retype password field. And Click on Login Button.	FT will display error message. And Failed to Registration.
FT-016						Step 7	Enter nothing new User Id, Password, Retype password and Hints Fields. And Click on Login Button.	FT will display error message. And Failed to Registration.
FT-017						Step 8	Enter Proper new User Id, Password, Retype Same password and Hints Fields. And Click on Login Button.	Successful Registration is done and this area is closed and come to login area.
FT-018	M a n u a l	f0657bbdf4 7e26ec481f a172b0fa76f 9becb2681	E:\DEVELOPERS_ZONE\GitHub\FitnessTracker\code	Check for successful login through email	The purpose of this test is to check email login process.	Step 1	Write a wrong username and password in the login button	Login failed because of wrong username and password. Display error message.
FT-019						Step 2	Insert wrong username and right password	Error displayed that wrong username is inserted.
FT-020						Step 3	Insert right username and wrong password.	Error displayed that wrong password is inserted.

FT-021						Step 4	Insert nothing in the username password field and click on login.	Display error message. No data inserted.
FT-022						Step 5	Insert nothing in the username field but right data in the password field and click on login.	Display error message. No username inserted.
FT-023						Step 6	Insert nothing in the password field but right data in the username field and click on login.	Display error message. No password inserted.
FT-024						Step 7	Write right username and password and click login	Logs in successfully in the application.
FT-025	Manual	e22bd0e470f145f3db336ed9e28d474d8f4637d7	E:\DEVELOPERS_ZONE\GitHub\FitnessTracker\code	Check Successful insertion of a food details in the database.	The purpose of this test is to check whether the data are getting inserted or not.	Step 1	Keep all the input fields empty and click Add entry	Error message: no data inserted.
FT-026						Step 2	Write alphabetic input in the Number of servings field.	Error message: must insert numeric input on that field.
FT-027						Step 3	Write alphabetic input at calories field.	Error message: must insert numeric input on that field.

Insert right input in the 4 fields and click add entry						Step 4	Insert right entries and click to add entry	Successfully added new food entry and calory entry for that date.
Keep all ther input fields empty and click Add entry	M a n u a l	90330b92328d862892fc77436539081cc2b7f70d	E:\DEVELOPERS_ZONE\Git Hub\Fitne ssTracker\code	Check Successful insertion of a exercise details in the database.	The purpose of this test is to check whether the data are getting inserted or not.	Step 1		Error message: no data inserted.
FT-030						Step 2	Insert alphabetic input in the sets completed field.	Error message: must insert numeric data.
FT-031						Step 3	Insert alphabetic input in the exercise per set field.	Error message: must insert numeric data.
FT-032						Step 4	Insert right entries and click to add entry	All data inserted successfully and calory gain reduced for that exercise on that date.
FT-033	M a n u a l	47fb570f63ffec837a49e235f629c49cc55a70f0	E:\DEVELOPERS_ZONE\Git Hub\Fitne ssTracker\code	Check social site status.	It is to check that we can get and send post or tweet to a particular social site.	Step 1	Click Facebook under Social Site tab. And enter wrong Login id And Password.	Failed to connect with Facebook. And display error message.

FT-034						Step 2	Click Facebook under Social Site tab. And enter correct Login id And Password.	Connected successfully with the Facebook.
FT-035						Step 3	Enter Event in Post field. And Click on post button.	Successfully posted in Facebook.
FT-036						Step 4	Click Twitter under Social Site tab. And enter wrong Login id And Password.	Failed to connect with Twitter. And display error message.
FT-037						Step 5	After logging through fb, try to share your current calory gain statistics	Shared on my fb.com account homepage successfully.
FT-038						Step 6	Share your weight gain/loose progress on fb.	Data of my weight gain/loose progress is shared on my fb wall.
FT-039	M a n u a l	b8a65899fa 408a4d99a8 ee8cabbe2a c0b54226b0	E:\DEVE LOPERS_ ZONE\Git Hub\Fitne ssTracker\ code	Check password change feature	It is to check that whether we can change existing password or not.	Step 1	Insert wrong password in old password field and click change.	Error message: the password can not be changed without inserting right old password.
FT-040						Step 2	Insert right password in old password field and click change.	Password gets changed successfully.

FT-041	M a n u a l	6c53de9297 011864bd6c 07bdcce834 15a76e8bd4	E:\DEVELOPERS_ ZONE\Git Hub\FitnessTracker\ code	Check everything about daily calory gain report.	It is to check that report works properly.	Step 1	Add a food and check the report	The amount of calory for that food gets added and shown in the daily report
FT-042						Step 2	Add an exercise and check the report	The amount of calory for that particular exercise gets reduced from the latest calory.
FT-043						Step 3	Add multiple foods and check the report	The total amount of calory gets added to the report
FT-044						Step 4	Add multiple exercises and check the report	The total amount of calory gets reduced from the total calory amount.
FT-045						Step 5	Check the daily tareget and check the daily progress of calory gain	The total amount gets deducted from the target and shows the remaing amount that is to be gained through eating or lost through exercise.
FT-046						Step 6	Check detection of completion of daily target.	When the total gain reaches thae target, a message is displayed

FT-047						Step 7	Check whether user gets extra calory gain warning or not. Add more calory in the report from the target and click on, add to diary.	Message is shown that the user is gaining more calory than required
FT-048	Manual	90330b92328d862892fc77436539081cc2b7f70d	E:\DEVELOPERS_ZONE\Github\FitnesTracker\code	Check everything about Monthly progress report.	It is to check that monthly weight loss/gain report works properly.	Step 1	Add and deduct calory and add to database daily	You get report that each day, you are gaining more or losing more calory.
FT-049						Step 2	Check your monthly progress report	On gaining more calory you get that your weight has increased
FT-050						Step 3	Check monthly progress report	On gaining less calory, you are notified that your weight is reduced to a certain amount

System Test Cases

Test Case Id	Type	Github ID	Subject	Test Name	Test Description	Step Name	Description	Expected Result
FT-051	Manual	f3563be0a9c431104f52839039e86043cf640cf1	E:\DEVELOPERS_ZONE\Github\FitnesTracker\code	Check Login.	It is to check that Login works properly.	Step 1	Click on Login button after inserting invalid User id and password from FT .	Login failed to FT . And can't able to use the feature.

FT-052						Step 2	Click on Login button after inserting valid User id and password from FT .	Successfully Login to FT . And can able to use the feature.
FT-053	M a n u a l	d01197ee4cd3bee9245874b5937ba740019fd131	E:\DEVELOPERS_ZONE\Git Hub\FitnessTracker\code	Check Successful Registration for New FT User.	The purpose of this test is to verify that the all new connection could be creating new Account By Registration.	Step 1	Click on Registration link.	New Account creation area is opened.
FT-054						Step 2	Click on Registration button after inserting invalid information from FT .	Registration failed to FT . And can't able to use the feature.
FT-055						Step 3	Click on Registration button after inserting valid information from FT .	Registration Successfully done to FT .
FT-056						Step 4	Click on Login button after inserting newly created valid User id and password from FT .	Successfully Login to by new User Id And password FT . And can able to use the feature.
FT-057	M a n u a l	f0657bbdf47e26ec481fa172b0fa76f9becb2681	E:\DEVELOPERS_ZONE\Git Hub\FitnessTracker\code	Check Successful food entry addition along with calory	The purpose of this test is to check Successful food entry addition	Step 1	Add a new food entry	Calory for that corresponding food gets added at the daily report

					along with calory			
FT-058						Step 2	Insert some alphabetic data in the numeric input fields	Error shown and asked to insert right type of data.
FT-059						Step 3	Insert data from some recent foods	The exact calory for that food is added
FT-060						Step 4	Insert data from the existing food database	The exact calory for that food is added
FT-061	M a n u a l	47fb570f63f fec837a49e 235f629c49 cc55a70f0	E:\DEVE LOPERS_ ZONE\Git Hub\Fitne ssTracker\ code	Check social site status and update event.	It is to check that we can get and send post or tweet to a particular social site.	Step 1	Click Facebook under Social Site tab. And enter valid Login id And Password to connect and create event and click on post button.	Successfully connected to Facebook and can post in Facebook wall.
FT-062						Step 2	Select a post and click on delete button to delete post from Facebook wall.	Post Successfully deleted.
FT-063						Step 3	Click Twitter under Social Site tab. And enter valid Login id And Password to connect and create event and click on Tweet button.	Successfully connected to Twitter and can Tweet in Twitter wall.
FT-064						Step 4	Select a Tweet and click on delete button to delete post from Twitter wall.	Tweet Successfully deleted.
FT-065	M a n u	b8a65899fa 408a4d99a8 ee8cabbe2a c0b54226b0	E:\DEVE LOPERS_ ZONE\Git Hub\Fitne	Check addition of exercise info daily	It is to check that exercise	Step 1	Add an exercise with some alphabetic data in a numeric	Error shown: you can not insert alphabetic

	a l		ssTracker\ code		enrty is getting added properly or not.		field	data in numeric fields
FT-066						Step 2	Add a exercise from recently used option	The data gets added successfully and the calory gets deducted
FT-067	M a n u a l	6c53de9297 011864bd6c 07bdcce834 15a76e8bd4	E:\DEVE LOPERS_ ZONE\Git Hub\Fitne ssTracker\ code	Check everything Daily Report.	Check whether daily report of total daily calory gain/loss working or not	Step 1	Add some calory gaining entries, i.e. food entry	Total calory gain is shown in the report
FT-068						Step 2	Add some calory lossing entries, i.e. exercise entry	Total calory loss is shown in the report
FT-069						Step 3	See total gain and loss of calory	If the total amount of calory is more than required, an warning of gaining extra calory is shown.

Activity_login.xml

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin" >

    <TextView
        android:text="Email/Phone Login"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:paddingLeft="10dp"
        android:textSize="20dp"
        android:layout_marginTop="10dp"
        android:layout_marginLeft="5dp"
        android:textStyle="bold"
        />

    <TableRow android:layout_width="fill_parent"
        android:layout_marginTop="10dp"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        >
    </TableRow>
    <TableRow android:layout_width="fill_parent"
        android:layout_marginTop="1dp"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        >
    </TableRow>

    <TextView
        android:text="Username"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:paddingLeft="10dp"
        android:textSize="12dp"
        android:layout_marginTop="10dp"
        android:layout_marginLeft="5dp"
        android:textStyle="italic"
        />

    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="5dp"
        android:hint="required"
        android:background="@android:color/darker_gray"

```

```

    />
    <TextView
        android:text="Password"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:paddingLeft="10dp"
        android:textSize="12dp"
        android:layout_marginTop="10dp"
        android:layout_marginLeft="5dp"
        android:textStyle="italic"
    />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="5dp"
        android:hint="Required"
        android:background="@android:color/darker_gray"
    />

    <TableRow android:layout_width="fill_parent"
        android:layout_marginTop="10dp"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:background="@android:color/transparent">

        <CheckBox
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
        />

        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"

            android:paddingLeft="10dp"
            android:text="Remember"
            android:textSize="15dp"
            android:textStyle="italic" />

    </TableRow>
    <TableRow android:layout_width="fill_parent"
        android:layout_marginTop="10dp"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:background="@android:color/transparent">

        <Button
            android:text="Back"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="#b0b0b0"
            android:padding="10dp"
            android:gravity="center"
            android:textStyle="italic"
        />

```

```

        <Button
            android:text="Save"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="#b0b0b0"
            android:padding="10dp"
            android:gravity="center"
            android:textStyle="italic"
        />
    </TableRow>
</LinearLayout>

```

Activity_Register.xml

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin" >

    <TextView
        android:text="Email/Phone Login"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:paddingLeft="10dp"
        android:textSize="20dp"
        android:layout_marginTop="10dp"
        android:layout_marginLeft="5dp"
        android:textStyle="bold"
    />

    <TableRow android:layout_width="fill_parent"
        android:layout_marginTop="10dp"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
    >
    </TableRow>
    <TableRow android:layout_width="fill_parent"
        android:layout_marginTop="1dp"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
    >
    </TableRow>

    <TextView
        android:text="Choose Username"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:paddingLeft="10dp"
        android:textSize="12dp"
        android:layout_marginTop="10dp"
    >

```



```

        android:layout_marginLeft="5dp"
        android:textStyle="italic"
    />

    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="5dp"
        android:hint=""
        android:background="@android:color/darker_gray"
    />
    <TextView
        android:text="Password"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:paddingLeft="10dp"
        android:textSize="12dp"
        android:layout_marginTop="10dp"
        android:layout_marginLeft="5dp"
        android:textStyle="italic"
    />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="5dp"
        android:hint=""
        android:background="@android:color/darker_gray"
    />
    <TextView
        android:text="Confirm Password"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:paddingLeft="10dp"
        android:textSize="12dp"
        android:layout_marginTop="10dp"
        android:layout_marginLeft="5dp"
        android:textStyle="italic"
    />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="5dp"
        android:hint=""
        android:background="@android:color/darker_gray"
    />
    <TextView
        android:text="Email Address"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:paddingLeft="10dp"
        android:textSize="12dp"
        android:layout_marginTop="10dp"
        android:layout_marginLeft="5dp"
        android:textStyle="italic"
    />
    <EditText

```

```

        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="5dp"
        android:hint=""
        android:background="@android:color/darker_gray"
    />

    <TableRow android:layout_width="fill_parent"
        android:layout_marginTop="10dp"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:background="@android:color/transparent">

</TableRow>
<TableRow android:layout_width="fill_parent"
        android:layout_marginTop="10dp"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:background="@android:color/transparent">

        <Button
            android:text="Back"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="#b0b0b0"
            android:padding="10dp"
            android:gravity="center"
            android:textStyle="italic"
        />
        <Button
            android:text="Finish"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="#b0b0b0"
            android:padding="10dp"
            android:gravity="center"
            android:textStyle="italic"
        />
    </TableRow>
</LinearLayout>

```

Activity_addexercise.xml

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"

    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"

    >

```

```

<TextView
    android:text="Exercise"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:paddingLeft="10dp"
    android:textSize="20dp"
    android:layout_marginTop="10dp"
    android:layout_marginLeft="5dp"
    android:textStyle="bold"
/>
<TableRow android:layout_width="fill_parent"
    android:layout_marginTop="10dp"
    android:layout_height="wrap_content"
    android:gravity="center_vertical"
>

    <Button
        android:textColor="@color/white"
        android:background="@color/Appbackground"
        android:text="Frequent"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"

android:padding="2dp"
android:gravity="center"
android:textStyle="italic"
android:textSize="15dp"
/>
    <Button

        android:textSize="15dp"
        android:textColor="@color/white"
        android:text="Recent"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
    android:background="@color/Appbackground"
    android:padding="2dp"
    android:gravity="center"
    android:textStyle="italic"
/>
    <Button
        android:textSize="15dp"
        android:textColor="@color/white"
        android:text="Exercises"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
    android:background="@color/Appbackground"
    android:padding="2dp"
    android:gravity="center"
    android:textStyle="italic"

/>
</TableRow>
<TableRow android:layout_width="fill_parent"
    android:layout_marginTop="1dp"

```

```

        android:layout_height="wrap_content"
        android:gravity="center_vertical"
    >
</TableRow>

<TextView
    android:text="Exercise Name"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:paddingLeft="10dp"
    android:textSize="12dp"
    android:layout_marginTop="10dp"
    android:layout_marginLeft="5dp"
    android:textStyle="italic"
/>
<EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="5dp"
    android:text="Back push up"
    android:background="@android:color/darker_gray"
/>

<TextView
    android:text="Sets Completed"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:paddingLeft="10dp"
    android:textSize="12dp"
    android:layout_marginTop="10dp"
    android:layout_marginLeft="5dp"
    android:textStyle="italic"
/>

<EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="5dp"
    android:text="3"
    android:background="@android:color/darker_gray"
/>
<TextView
    android:text="Exercise per set"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:paddingLeft="10dp"
    android:textSize="12dp"
    android:layout_marginTop="10dp"
    android:layout_marginLeft="5dp"
    android:textStyle="italic"
/>
<EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="5dp"

```

```

        android:text="10"
        android:background="@android:color/darker_gray"
    />

    <TableRow android:layout_width="fill_parent"
        android:layout_marginTop="30dp"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:background="@android:color/transparent">

        <Button
            android:text="Back"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="#b0b0b0"
            android:padding="10dp"
            android:gravity="center"
            android:textStyle="italic"
        />
        <Button
            android:text="Save"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="#b0b0b0"
            android:padding="10dp"
            android:gravity="center"
            android:textStyle="italic"
        />
    </TableRow>

</LinearLayout>

```

Activity_addfood.xml

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"

    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"

    >

    <TextView
        android:text="Exercise"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"

```

```

        android:paddingLeft="10dp"
        android:textSize="20dp"
        android:layout_marginTop="10dp"
        android:layout_marginLeft="5dp"
        android:textStyle="bold"
    />
    <TableRow android:layout_width="fill_parent"
        android:layout_marginTop="10dp"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
    >

        <Button
            android:textColor="@color/white"
            android:background="@color/Appbackground"
            android:text="Frequent"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"

android:padding="2dp"
android:gravity="center"
android:textStyle="italic"
android:textSize="15dp"
        />
        <Button

            android:textSize="15dp"
            android:textColor="@color/white"
            android:text="Recent"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
android:background="@color/Appbackground"
android:padding="2dp"
android:gravity="center"
android:textStyle="italic"
        />
        <Button
            android:textSize="15dp"
            android:textColor="@color/white"
            android:text="My Foods"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
android:background="@color/Appbackground"
android:padding="2dp"
android:gravity="center"
android:textStyle="italic"

        />
    </TableRow>
    <TableRow android:layout_width="fill_parent"
        android:layout_marginTop="1dp"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
    >
    </TableRow>

```

```

<TextView
    android:text="Food Name"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:paddingLeft="10dp"
    android:textSize="12dp"
    android:layout_marginTop="10dp"
    android:layout_marginLeft="5dp"
    android:textStyle="italic"
/>

<EditText
    android:id="@+id/action_settings"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="5dp"
    android:text=""
    android:background="@android:color/darker_gray"
/>

<TextView
    android:text="Number of Servings"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:paddingLeft="10dp"
    android:textSize="12dp"
    android:layout_marginTop="10dp"
    android:layout_marginLeft="5dp"
    android:textStyle="italic"
/>

<EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="5dp"
    android:text=""
    android:background="@android:color/darker_gray"
/>
<TextView
    android:text="Serving Amount"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:paddingLeft="10dp"
    android:textSize="12dp"
    android:layout_marginTop="10dp"
    android:layout_marginLeft="5dp"
    android:textStyle="italic"
/>
<EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="10dp"

```

```

        android:layout_marginTop="5dp"
        android:text=""
        android:background="@android:color/darker_gray"
    />

    <TableRow android:layout_width="fill_parent"
        android:layout_marginTop="30dp"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:background="@android:color/transparent">

        <Button
            android:text="Back"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="#b0b0b0"
            android:padding="10dp"
            android:gravity="center"
            android:textStyle="italic"
        />
        <Button
            android:id="@+id/addEntryBtn"
            android:text="Save"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="#b0b0b0"
            android:padding="10dp"
            android:gravity="center"
            android:textStyle="italic"
        />
    </TableRow>

</LinearLayout>

```

Activity_main.xml

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/Appbackground"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin" >

    <TextView
        android:text="Welcome to Fitness Tracker"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    >

```



```

        android:paddingLeft="10dp"
        android:textSize="20dp"
        android:textColor="@color/white"
        android:layout_marginTop="30dp"
        android:layout_marginLeft="5dp"
        android:textStyle="bold"
    />
    <TextView
        android:text="Fitness Tracker should be used after consulting your diet
with the doctor."
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:paddingLeft="10dp"
        android:textSize="10dp"
        android:textColor="@color/white"
        android:layout_marginTop="15dp"
        android:layout_marginLeft="5dp"
        android:textStyle="bold"
    />

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@android:color/ho_lo_purple"
            android:layout_marginTop="20dp"
            android:text="Back"
        android:textStyle="italic"
        android:textColor="@color/white" />

    <Button
        android:id="@+id/action_settings"
        android:background="@android:color/ho_lo_purple"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:text="Save"
        android:textStyle="italic"
        android:textColor="@color/white"/>

    <TableRow android:layout_width="fill_parent"
        android:layout_marginTop="10dp"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
    >
    </TableRow>
    <TableRow android:layout_width="fill_parent"
        android:layout_marginTop="1dp"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
    >
    </TableRow>

    <TableRow android:layout_width="fill_parent"
        android:layout_marginTop="10dp"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:background="@android:color/transparent">

    </TableRow>

```

```

        <TableRow android:layout_width="fill_parent"
            android:layout_marginTop="10dp"
            android:layout_height="wrap_content"
            android:gravity="center_vertical"
            android:background="@android:color/transparent">
        </TableRow>

</LinearLayout>

```

Strings.xml

```

<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">File Management System</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <color name="Appbackground">#25bfda</color>
    <color name="white">#ffffff</color>
</resources>

```

Dimens.xml

```

<resources>

    <!-- Default screen margins, per the Android Design guidelines. -->
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>

</resources>

```

AndroidManifest.xml

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.fitnessracker"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="17"
        android:targetSdkVersion="17" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity

```

```

        android:name="com.example.fitnesstracker.MainActivity"
        android:label="@string/app_name" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>

</manifest>

```

Web coding

Controller

```

<?php

class FitnessTracker extends CI_Controller {

    public function __construct() {
        parent::__construct();
        $this->load->model('FitnessTracker_model');
        $this->load->model('jobpost_model');
    }

    public function index() {

        $data['FitnessTracker'] = $this->FitnessTracker_model->get_FitnessTracker();

        $data['title'] = 'FitnessTracker archive';

        $this->load->view('templates/header', $data);
        $this->load->view('FitnessTracker/index', $data);
        $this->load->view('templates/footer');
    }

    public function login_view()
    {

        $this->load->view('templates/header');
        $this->load->view('FitnessTracker/login_view');
        $this->load->view('templates/footer');
    }

    public function activity_view()
    {

        $this->load->view('templates/header');
        $this->load->view('pages/activity');
        $this->load->view('templates/footer');
    }

    public function view($id) {

```

```

    $data['FitnessTracker_item'] = $this->FitnessTracker_model->get_FitnessTracker($id);
    $data['jobpost'] = $this->jobpost_model->get_jobpost();
    if (empty($data['FitnessTracker_item'])) {
        show_404();
    }

    $data['title'] = $data['FitnessTracker_item']['lastName'];

    $this->load->view('templates/header', $data);
    $this->load->view('FitnessTracker/view', $data);
    $this->load->view('templates/footer');
}

public function create() {
    $this->load->helper('form');
    $this->load->library('form_validation');

    $data['title'] = 'Create a FitnessTracker item';

    $this->form_validation->set_rules('lastName', 'lastName', 'required');

    if ($this->form_validation->run() === FALSE) {
        $this->load->view('templates/header', $data);
        $this->load->view('FitnessTracker/create');
        $this->load->view('templates/footer');
    } else {
        $this->FitnessTracker_model->set_FitnessTracker();
        $this->load->view('templates/header');
        $this->load->view('FitnessTracker/success');
        $this->load->view('templates/footer');
    }
}

    public function about_foods()
    {
        echo "foods";
        die();
    }
    public function about()
    {
        $this->load->view('templates/header');
    $this->load->view('pages/about');
    $this->load->view('templates/footer');

    }

}

?>

```

Model

<?php

```
class FitnessTracker_model extends CI_Model {

    public function __construct() {
        $this->load->database();
    }

    public function get_FitnessTracker($id = FALSE) {
        if ($id === FALSE) {
            $query = $this->db->get('FitnessTrackerinfo');
            return $query->result_array();
        }

        $query = $this->db->get_where('FitnessTrackerinfo', array('id' => $id));
        return $query->row_array();
    }

    public function get_FitnessTracker_search($search = FALSE) {

        if ($search === FALSE) {
            $query = $this->db->get('FitnessTrackerinfo');
            return $query->result_array();
        }

        $sql = "SELECT * FROM FitnessTrackerinfo where
        id = ".$search."
        or firstName= ".$search."
        or middleInitial= ".$search."
        or lastName= ".$search."

        ";

        $sql1=mysql_query($sql);
        //return $query->result_array($sql1);

        while ($ar=mysql_fetch_array($sql1))
        {
            echo $ar['id'];
            echo $ar['firstName'];
            echo $ar['middleInitial'];
            echo "<br>";
        }
        die();
    }

    public function set_FitnessTracker() {
        $this->load->helper('url');
```

```

$data = array(
    'firstName' => $this->input->post('firstName'),
    'middleInitial' => $this->input->post('middleInitial'),
    'lastName' => $this->input->post('lastName'),
    'address1' => $this->input->post('address1'),
    'address2' => $this->input->post('address2'),
    'city' => $this->input->post('city'),
    'state' => $this->input->post('state'),
    'region' => $this->input->post('region'),
    'zip' => $this->input->post('zip'),
    'phone1' => $this->input->post('phone1'),
    'phone2' => $this->input->post('phone2'),
    'email1' => $this->input->post('email1'),
    'email2' => $this->input->post('email2'),
    'highestDegree' => $this->input->post('highestDegree'),
    'degreeDetails' => $this->input->post('degreeDetails'),
    'school' => $this->input->post('school'),
    'dateGraduated' => $this->input->post('dateGraduated'),
    'certifications' => $this->input->post('certifications'),
    'certificationNotes' => $this->input->post('certificationNotes'),
    'emrTraining' => $this->input->post('emrTraining'),
    'emrModules' => $this->input->post('emrModules'),
    'trainingLevel' => $this->input->post('trainingLevel'),
    'yearsOfExperience' => $this->input->post('yearsOfExperience'),
    'employmentType' => $this->input->post('employmentType'),
    'employmentTimingType' => $this->input->post('employmentTimingType'),
    'travelChoice' => $this->input->post('travelChoice[]'),
    'paymentType' => $this->input->post('paymentType'),
    'paymentAmount' => $this->input->post('paymentAmount'),
    'preferredTravelRegion' => $this->input->post('$preferredTravelRegion')
);

return $this->db->insert('FitnessTrackerinfo', $data);
}

public function set_FitnessTracker_rest($data) {
    $this->load->helper('url');
    return $this->db->insert('FitnessTrackerinfo', $data);
}

public function delete_FitnessTracker_rest($id) {
    $this->load->helper('url');
    return $this->db->delete('FitnessTrackerinfo', array('id' => $id));
}

public function get_results($search_term='default')
{
    // Use the Active Record class for safer queries.
    $this->db->select('*');
    $this->db->from('FitnessTrackerinfo');

```

```

$this->db->like('firstName',$search_term);
$this->db->or_like('middleInitial',$search_term);
$this->db->or_like('lastName',$search_term);

$this->db->or_like('state',$search_term);

$this->db->or_like('highestDegree',$search_term);

$this->db->or_like('certifications',$search_term);

$this->db->or_like('yearsOfExperience',$search_term);

$query = $this->db->get();
// Return the results.
return $query->result_array();
}
}
?>

```

View

```

<div class="container" >

<div class="row" >
  <div class="span12">
    <div class="hero-unit well">
      <form class="pull-right" action="">

          <div class="input-append">
            <select class="span3" name='hireFor' required>
              <option> ---Select Job--- </option>
              <option> Associates </option>
              <option> Bachelors </option>
              <option> Masters </option>
              <option> Doctorate </option>
              <option> Other </option>
            </select>
            <button class="btn btn-primary" type="button">Invite</button>
          </div>

      </form>
      <h4 align='center'>Welcome to the profile of </h4>
      <h3 align='center'>
<?php
echo $FitnessTracker_item['firstName'] . ' ' . $FitnessTracker_item['middleInitial'] . ' ' .

```

```

$FitnessTracker_item['lastName'];
    ?>
</h3>

</div>
</div>
</div>

<div class="row" >
<div class="span6">

    <div class="well well-red">
        <h3><a id='personalDetails'>Personal Details</a></h3>
        <ul><li><p><?php
            echo '<b>Lives in : </b>' . $FitnessTracker_item['address1'] . ', ' .
$FitnessTracker_item['address2'] . ', ' . $FitnessTracker_item['city'] . ', ' .
$FitnessTracker_item['state'] . ', ' . $FitnessTracker_item['region'] . ', ' . $FitnessTracker_item['zip'];
            ?></p></li>
        </ul>

    </div>

    <div class="well well-red">
        <h3><a id='contactDetails'>Contact Details</a></h3>
        <ul><li><p><?php
            echo '<b>Primary Phone Number : </b>' . $FitnessTracker_item['phone1'];
            ?></p></li>
        </ul>
        <ul><li><p><?php
            echo '<b>Secondary Phone Number : </b>' . $FitnessTracker_item['phone2'];
            ?></p></li>
        </ul>
        <ul><li><p><?php
            echo '<b>Primary Email ID : </b>' . $FitnessTracker_item['email1'];
            ?></p></li>
        </ul>
        <ul><li><p><?php
            echo '<b>Secondary Email ID : </b>' . $FitnessTracker_item['email2'];
            ?></p></li>
        </ul>

```



```

        </ul>
    </div>

    <div class="well well-red">
        <h3><a id='education'>Education</a></h3>
        <ul><li><p><?php
            echo '<b>Highest Degree : </b>' . $FitnessTracker_item['highestDegree'];
        ?></p></li>
        </ul>
        <ul><li><p><?php
            echo '<b>In : </b>' . $FitnessTracker_item['degreeDetails'];
        ?></p> </li>
        </ul>
        <ul><li><p><?php
            echo '<b>From : </b>' . $FitnessTracker_item['school'];
        ?></p></li>
        </ul>
        <ul><li><p><?php
            echo '<b>In Year : </b>' . $FitnessTracker_item['dateGraduated'];
        ?></p></li>
        </ul>

        </ul>
    </div>

```

```

</div>

```

```

<div class="span6">

```

```

    <div class="well well-red">
        <h3><a id='professionalCertifications'>Professional Certifications</a></h3>
        <ul><li><p><?php
            echo '<b>Certification In : </b>' . $FitnessTracker_item['certifications'];
        ?></p></li>
        </ul>
        <ul><li><p><?php
            echo '<b>Description : </b>' . $FitnessTracker_item['certificationNotes'];
        ?></p> </li>
        </ul>

        </ul>
    </div>

```

```

    <div class="well well-red">
        <h3><a id='emrTrainingCertification'>EMR Training/Certification</a></h3>
        <ul class="dl-horizontal"><li><p><?php
            echo '<b>EMR : </b>' . $FitnessTracker_item['emrTraining'];
        ?></p></li>
        </ul>
    </div>

```

```

        <ul class="dl-horizontal"><li><p><?php
            echo '<b>EMR Module : </b>' . $FitnessTracker_item['emrModules'];
        ?></p> </li>
        </ul>
        <ul><li><p><?php
            echo '<b>Level : </b>' . $FitnessTracker_item['trainingLevel'];
        ?></p></li>
        </ul>
        <ul><li><p><?php
            echo '<b>Total Experience : </b>' . $FitnessTracker_item['yearsOfExperience'] .
'<b> years</b>';
        ?></p></li>
        </ul>

    </ul>
</div>

    <div class="well well-red">
    <h3><a id='others'>Others</a></h3>
    <ul class="dl-horizontal"><li><p><?php
        echo '<b>Employment Type : </b>' . $FitnessTracker_item['employmentType'];
    ?></p></li>
    </ul>
    <ul class="dl-horizontal"><li><p><?php
        echo '<b>Employment Timing Type : </b>' .
$FitnessTracker_item['employmentTimingType'];
    ?></p> </li>
    </ul>

    <ul><li><p><?php
        echo '<b>Travel Choice : </b>' . $FitnessTracker_item['travelChoice'];
    ?></p></li>
    </ul>
    <ul><li><p><?php
        echo '<b>Pequired Payment Type : </b>' . $FitnessTracker_item['paymentType'];
    ?></p></li>
    </ul>
    <ul><li><p><?php
        echo '<b>Pequired Payment Amount : </b>' .
$FitnessTracker_item['paymentAmount'];
    ?></p></li>
    </ul>
    <ul><li><p><?php
        echo '<b>Preferred Travel Region : </b>' .
$FitnessTracker_item['preferredTravelRegion'];
    ?></p></li>
    </ul>

</div>

```

```
</div>
</div>
</div>
```

Database connector: FTDb

DbInteraction

```
<?php

if (!defined('BASEPATH'))
    exit('No direct script access allowed');
/**
 * Code Igniter
 *
 * An open source application development framework for PHP 5.1.6 or newer
 *
 * @package      CodeIgniter
 * @author       ExpressionEngine Dev Team
 * @copyright    Copyright (c) 2008 - 2011, EllisLab, Inc.
 * @license      http://codeigniter.com/user_guide/license.html
 * @link         http://codeigniter.com
 * @since       Version 1.0
 * @filesource
 */
// -----

/**
 * Database Utility Class
 *
 * @category     Database
 * @author       ExpressionEngine Dev Team
 * @link         http://codeigniter.com/user_guide/database/
 */
class CI_DB_utility extends CI_DB_forge {

    var $db;
    var $data_cache = array();
```

```

/**
 * Constructor
 *
 * Grabs the CI super object instance so we can access it.
 */
function __construct() {
    // Assign the main database object to $this->db
    $CI = & get_instance();
    $this->db = & $CI->db;

    log_message('debug', "Database Utility Class Initialized");
}

// -----

/**
 * List databases
 *
 * @access    public
 * @return    bool
 */
function list_databases() {
    // Is there a cached result?
    if (isset($this->data_cache['db_names'])) {
        return $this->data_cache['db_names'];
    }

    $query = $this->db->query($this->_list_databases());
    $dbs = array();
    if ($query->num_rows() > 0) {
        foreach ($query->result_array() as $row) {
            $dbs[] = current($row);
        }
    }

    $this->data_cache['db_names'] = $dbs;
    return $this->data_cache['db_names'];
}

// -----

/**
 * Determine if a particular database exists
 *
 * @access    public
 * @param     string
 * @return    boolean
 */
function database_exists($database_name) {

```

```

// Some databases won't have access to the list_databases() function, so
// this is intended to allow them to override with their own functions as
// defined in $driver_utility.php
if (method_exists($this, '_database_exists')) {
    return $this->_database_exists($database_name);
} else {
    return (!in_array($database_name, $this->list_databases())) ? FALSE : TRUE;
}
}

// -----

/**
 * Optimize Table
 *
 * @access public
 * @param string the table name
 * @return bool
 */
function optimize_table($table_name) {
    $sql = $this->_optimize_table($table_name);

    if (is_bool($sql)) {
        show_error('db_must_use_set');
    }

    $query = $this->db->query($sql);
    $res = $query->result_array();

    // Note: Due to a bug in current() that affects some versions
    // of PHP we can not pass function call directly into it
    return current($res);
}

// -----

/**
 * Optimize Database
 *
 * @access public
 * @return array
 */
function optimize_database() {
    $result = array();
    foreach ($this->db->list_tables() as $table_name) {
        $sql = $this->_optimize_table($table_name);

        if (is_bool($sql)) {
            return $sql;
        }
    }
}

```

```

        $query = $this->db->query($sql);

        // Build the result array...
        // Note: Due to a bug in current() that affects some versions
        // of PHP we can not pass function call directly into it
        $res = $query->result_array();
        $res = current($res);
        $key = str_replace($this->db->database . '.', '', current($res));
        $keys = array_keys($res);
        unset($res[$keys[0]]);

        $result[$key] = $res;
    }

    return $result;
}

// -----

/**
 * Repair Table
 *
 * @access    public
 * @param    string    the table name
 * @return    bool
 */
function repair_table($table_name) {
    $sql = $this->_repair_table($table_name);

    if (is_bool($sql)) {
        return $sql;
    }

    $query = $this->db->query($sql);

    // Note: Due to a bug in current() that affects some versions
    // of PHP we can not pass function call directly into it
    $res = $query->result_array();
    return current($res);
}

// -----

/**
 * Generate CSV from a query result object
 *
 * @access    public
 * @param    object    The query result object
 * @param    string    The delimiter - comma by default
 * @param    string    The newline character - \n by default
 * @param    string    The enclosure - double quote by default

```

```

* @return string
*/
function csv_from_result($query, $delim = ",", $newline = "\n", $enclosure = '"') {
    if (!is_object($query) OR !method_exists($query, 'list_fields')) {
        show_error('You must submit a valid result object');
    }

    $out = "";

    // First generate the headings from the table column names
    foreach ($query->list_fields() as $name) {
        $out .= $enclosure . str_replace($enclosure, $enclosure . $enclosure, $name) . $enclosure .
$delim;
    }

    $out = rtrim($out);
    $out .= $newline;

    // Next blast through the result array and build out the rows
    foreach ($query->result_array() as $row) {
        foreach ($row as $item) {
            $out .= $enclosure . str_replace($enclosure, $enclosure . $enclosure, $item) . $enclosure .
$delim;
        }
        $out = rtrim($out);
        $out .= $newline;
    }

    return $out;
}

// -----

/**
 * Generate XML data from a query result object
 *
 * @access public
 * @param object The query result object
 * @param array Any preferences
 * @return string
 */
function xml_from_result($query, $params = array()) {
    if (!is_object($query) OR !method_exists($query, 'list_fields')) {
        show_error('You must submit a valid result object');
    }

    // Set our default values
    foreach (array('root' => 'root', 'element' => 'element', 'newline' => "\n", 'tab' => "\t") as $key =>
$val) {
        if (!isset($params[$key])) {
            $params[$key] = $val;
        }
    }
}

```

```

    }
}

// Create variables for convenience
extract($params);

// Load the xml helper
$CI = & get_instance();
$CI->load->helper('xml');

// Generate the result
$xml = "<{$root}>" . $newline;
foreach ($query->result_array() as $row) {
    $xml .= $tab . "<{$element}>" . $newline;

    foreach ($row as $key => $val) {
        $xml .= $tab . $tab . "<{$key}>" . xml_convert($val) . "</{$key}>" . $newline;
    }
    $xml .= $tab . "</{$element}>" . $newline;
}
$xml .= "</{$root}>" . $newline;

return $xml;
}

// -----

/**
 * Database Backup
 *
 * @access    public
 * @return    void
 */
function backup($params = array()) {
    // If the parameters have not been submitted as an
    // array then we know that it is simply the table
    // name, which is a valid short cut.
    if (is_string($params)) {
        $params = array('tables' => $params);
    }

    // -----
    // Set up our default preferences
    $prefs = array(
        'tables' => array(),
        'ignore' => array(),
        'filename' => "",
        'format' => 'gzip', // gzip, zip, txt
        'add_drop' => TRUE,
        'add_insert' => TRUE,
        'newline' => "\n"
    );
}

```



```

);

// Did the user submit any preferences? If so set them....
if (count($params) > 0) {
    foreach ($prefs as $key => $val) {
        if (isset($params[$key])) {
            $prefs[$key] = $params[$key];
        }
    }
}

// -----
// Are we backing up a complete database or individual tables?
// If no table names were submitted we'll fetch the entire table list
if (count($prefs['tables']) == 0) {
    $prefs['tables'] = $this->db->list_tables();
}

// -----
// Validate the format
if (!in_array($prefs['format'], array('gzip', 'zip', 'txt'), TRUE)) {
    $prefs['format'] = 'txt';
}

// -----
// Is the encoder supported? If not, we'll either issue an
// error or use plain text depending on the debug settings
if (($prefs['format'] == 'gzip' AND !@function_exists('gzencode'))
    OR ($prefs['format'] == 'zip' AND !@function_exists('gzcompress'))) {
    if ($this->db->db_debug) {
        return $this->db->display_error('db_unsupported_compression');
    }

    $prefs['format'] = 'txt';
}

// -----
// Set the filename if not provided - Only needed with Zip files
if ($prefs['filename'] == "" AND $prefs['format'] == 'zip') {
    $prefs['filename'] = (count($prefs['tables']) == 1) ? $prefs['tables'] : $this->db->database;
    $prefs['filename'] .= '_' . date('Y-m-d_H-i', time());
}

// -----
// Was a Gzip file requested?
if ($prefs['format'] == 'gzip') {
    return gzencode($this->_backup($prefs));
}

// -----
// Was a text file requested?

```

```

    if ($prefs['format'] == 'txt') {
        return $this->_backup($prefs);
    }

    // -----
    // Was a Zip file requested?
    if ($prefs['format'] == 'zip') {
        // If they included the .zip file extension we'll remove it
        if (preg_match("|.+?.zip$|", $prefs['filename'])) {
            $prefs['filename'] = str_replace('.zip', '', $prefs['filename']);
        }

        // Tack on the ".sql" file extension if needed
        if (!preg_match("|.+?.sql$|", $prefs['filename'])) {
            $prefs['filename'] .= '.sql';
        }

        // Load the Zip class and output it

        $CI = & get_instance();
        $CI->load->library('zip');
        $CI->zip->add_data($prefs['filename'], $this->_backup($prefs));
        return $CI->zip->get_zip();
    }
}

/* End of file DB_utility.php */
/* Location: ./system/database/DB_utility.php */

```

■ Data Classes : FTData

WeightChart

```

public class WeightChart
{
    int id;
    string userName;
    double currentWeightValue;
    double targetWeightValue;
    double averageWeightValue;
    WeightUnit unit;
    DateTime assigningDate;
    DateTime targetDate;
}

```

WeightUnit

```

enum WeightUnit
{
    kg,

```

```
}  
    pound
```

FoodItem

```
public class FoodItem  
{  
    int id;  
    double caloryCount;  
    double proteinCount;  
    double fatCount;  
    string name;  
    double preferredQuantity;  
    string imageUrl;  
}
```

DietChart

```
public class DietChart  
{  
    int id;  
    string userName;  
    string trainerName;  
  
    List<FoodItem> breakfast;  
    List<FoodItem> lunch;  
    List<FoodItem> dinner;  
    List<FoodItem> snacks;  
  
    double targetDailyValue;  
  
    DateTime assigningDate;  
    DateTime targetDate;  
}
```

ExerciseType

```
enum ExerciseType  
{  
    Cardio,  
    Strength,  
    FreeHand  
}
```

ExerciseItem

```
public class ExerciseItem  
{  
    int id;  
    string name;  
  
    ExerciseType type;
```

```

    double caloryburnt;
    int sets;
    int repetition;

    double preferredDuration;
    double preferredWeight;

    string imageUrl;
    string videoUrl;
    List<string> specialInstructions;
}

```

ExerciseChart

```

public class ExerciseChart
{
    int id;
    string userName;
    string trainerName;

    List<ExerciseItem> exercises;
    DateTime startTime;
    DateTime duration;

    int weeklyTotalDuration;

    double targetDailyValue;

    DateTime assigningDate;
    DateTime targetDate;
}

```

FoodEntryDaywise

```

public class FoodEntryDaywise
{
    int id;

    List<FoodItem> breakfast;
    List<FoodItem> lunch;
    List<FoodItem> dinner;
    List<FoodItem> snacks;

    double totalDailyValue;

    DateTime currentDate;
}

```

ExerciseEntryDaywise

```

public class ExerciseEntryDaywise
{
    int id;
}

```

```

        List<ExerciseItem> cardioExercise;
        List<ExerciseItem> strengthExercise;

        double totalDailyCalorieBurnt;

        DateTime currentDate;
    }

```

User

```

public class User
{
    int id;
    string userName;
    string trainer;
    string fullName;
    string imageUrl;
    DateTime dob;
    double weight;

    ExerciseChart exChart;
    DietChart dietChart;

    List<ExerciseEntryDaywise> exercises;
    List<FoodEntryDaywise> foods;
}

```

Trainer

```

public class Trainer
{
    int id;
    string userName;
    string imageUrl;
    string websiteUrl;

    List<User> trainees;
}

```

ReportType

```

public enum ReportType
{
    Single,
    Daily,
    Weekly,
    Monthly,
    Quarterly,
    Yearly
}

```

ReportInfo

```
public class ReportInfo
{
    public string id { get; set; }
    public DateTime date { get; set; }
    public ReportType type { get; set; }
    public string description { get; set; }
}
```

Comments and Description of Coding segments

Code Commenting

- All comments have been written in the same language, be grammatically correct, and contain appropriate punctuation.
- Used // or /// but never /* ... */
- Did not “flowerbox” comment blocks.
- Example:
 - // *****
 - // Comment block
 - // *****
- Always Used inline-comments to explain assumptions, known issues, and algorithm insights.
- Never used inline-comments to explain obvious code. Well written code is self documenting.
- Only used comments for bad code to say “fix this code” – otherwise remove, or rewrite the code!
- Included comments using Task-List keyword flags to allow comment-filtering.
- Example:
 - //always close the connection
 - //Note.id = msqlReader.GetString("id");
 - //define the command reference
- Always applied C# comment-blocks (///) to public, protected, and internal declarations.
- Only used C# comment-blocks for documenting the API.
- Included #region and #endregion where possible for whole sections to have a #region-like thing and collapse them.
- Example:

```
#region User

    public static int DoRegisterNewUser(UserInfo NewUser)
    {
        return DoRegisterNewuserindb(NewUser);
    }

#endregion
```

```

    }

    private static int DoRegisterNewuserindb(UserInfo NewUser)
    {
        int returnVal = 0;
        MySql.Data.MySqlClient.MySqlConnection msqConnection =
        OpenDbConnection();

        try
        {
            //define the command reference
            MySql.Data.MySqlClient.MySqlCommand msqCommand = new
            MySql.Data.MySqlClient.MySqlCommand();

            //define the connection used by the command object
            msqCommand.Connection = msqConnection;

            msqCommand.CommandText = "INSERT INTO
            user(id,userid,passwd,hints) " + "VALUES(@id,@userid,@passwd,@hints)";

            msqCommand.Parameters.AddWithValue("@id", NewUser.id);
            msqCommand.Parameters.AddWithValue("@userid", NewUser.userId);
            msqCommand.Parameters.AddWithValue("@passwd", NewUser.pass);
            msqCommand.Parameters.AddWithValue("@hints", NewUser.hints);

            msqCommand.ExecuteNonQuery();

            returnVal = 1;
        }
        catch (Exception er)
        {
            returnVal = 0;
        }
        finally
        {
            //always close the connection
            msqConnection.Close();
        }
        return returnVal;
    }

    #endregion

```

Description of coding

- ❖ Android SDK has been used to develop the application.
- ❖ Object Oriented Programming methodology will be adopted
- ❖ Relational DBMS SQLite will be used to implement & execute SQL query to database for Android device.
- ❖ Relational DBMS MySQL will be used to implement & execute SQL query to database for Web site.
- ❖ Agile Software Development model will be used while developing this software.

Coding style causes the most inconsistency and controversy between developers. Each developer has a preference, and rarely are two the same. However, consistent layout, format, and organization are key to creating maintainable code. The following sections describe the preferred way to implement C# source code in order to create readable, clear, and consistent code that is easy to understand and maintain.

Formatting

- Never declared more than 1 namespace per file.
- Avoided putting multiple classes in a single file.
- Always placed curly braces ({ and }) on a new line.
- Always used curly braces ({ and }) in conditional statements.
- Always used a Tab & Indention size of 4.
- Declared each variable independently – not in the same statement.
- Placed namespace “using” statements together at the top of file. Group .NET namespaces above custom namespaces.
- Grouped internal class implementation by type in the following order:

Member variables.

Constructors & Finalizers.

Nested Enums, Structs, and Classes.

Properties

Methods

- Sequence declarations within type groups based upon access modifier and visibility:

Public

Protected

Internal

Private

- Segregate interface Implementation by using #region statements.
- Append folder-name to namespace for source files within sub-folders.
- Recursively indent all code blocks contained within braces.
- Use white space (CR/LF, Tabs, etc) liberally to separate and organize code.
- Only declare related attribute declarations on a single line, otherwise stack each attribute as a separated declaration.

Example:

```
// Bad!
```

```
[Attribute1, Attribute2, Attribute3]
```

```
public class MyClass
```



```
{...}
```

```
// Good!
```

```
[Attribute1, RelatedAttribute2]
```

```
[Attribute3]
```

```
[Attribute4]
```

```
public class MyClass
```

```
{...}
```

- Place Assembly scope attribute declarations on a separate line.
- Place Type scope attribute declarations on a separate line.
- Place Method scope attribute declarations on a separate line.
- Place Member scope attribute declarations on a separate line.
- Place Parameter attribute declarations inline with the parameter.
- If in doubt, always err on the side of clarity and consistency.

Code Efficiency

We started working on the project keeping in mind that we must develop it in a way that it not only provides a very easy to use GUI but also provide a fast and flexible service to the users. We know that a particular work can be done in more than one ways. We have tried all the options and then chose the one which provides the fastest and most secure performance. First of all, we have used the latest technologies of Microsoft like visual studio 2010 as IDE and WPF as GUI to keep our application's performance few steps ahead. We have studies all the rules of software development life cycle and applied them to keep our application flexible. We have given special attention to the storage related codes. We have avoided all the unnecessary database codes and kept them as short as possible without harming our purpose so that insertion, updating, deletion and fetching of data take place flexibly. You can see the result as a user; our application does all the works very smoothly.

Error handling

The C# language's exception handling features help us to deal with any unexpected or exceptional situations that occur when a program is running. Exception handling uses the try, catch, and finally keywords to try actions that may not succeed, to handle failures when you decide that it is reasonable to do so, and to clean up resources afterward. Exceptions can be generated by the common language runtime (CLR), by the .NET Framework or any third-party libraries, or by application code. Exceptions are created by using the throw keyword.

In many cases, an exception may be thrown not by a method that your code has called directly, but by another method further down in the call stack. When this happens, the CLR will unwind the stack, looking for a method with a catch block for the specific exception type, and it will execute the first such catch block that it finds. If it finds no appropriate catch block anywhere in the call stack, it will terminate the process and display a message to the user.

Exceptions Overview

Exceptions have the following properties:

- Exceptions are types that all ultimately derive from `System.Exception`.

- Use a try block around the statements that might throw exceptions.

- Once an exception occurs in the try block, the flow of control jumps to the first associated exception handler that is present anywhere in the call stack. In C#, the `catch` keyword is used to define an exception handler.

- If no exception handler for a given exception is present, the program stops executing with an error message.

- Do not catch an exception unless you can handle it and leave the application in a known state. If you catch `System.Exception`, rethrow it using the `throw` keyword at the end of the catch block.

- If a catch block defines an exception variable, you can use it to obtain more information about the type of exception that occurred.

- Exceptions can be explicitly generated by a program by using the `throw` keyword.

- Exception objects contain detailed information about the error, such as the state of the call stack and a text description of the error.

- Code in a finally block is executed even if an exception is thrown. Use a finally block to release resources, for example to close any streams or files that were opened in the try block.

- Managed exceptions in the .NET Framework are implemented on top of the Win32 structured exception handling mechanism.

Validation checks

We have performed following data validation checks on available data:

Allowed character checks

Checks that ascertain that only expected characters are present in a field. For example a numeric field may only allow the digits 0-9, the decimal point and perhaps a minus sign or commas. A text field such as a personal name might disallow characters such as `<` and `>`, as they could be evidence of a markup-based security attack. An e-mail address might require exactly one `@` sign and various other structural details. Regular expressions are effective ways of implementing such checks. (See also data type checks below)

Batch totals

Checks for missing records. Numerical fields may be added together for all records in a batch. The batch total is entered and the computer checks that the total is correct, e.g., add the 'Total Cost' field of a number of transactions together.

Cardinality check

Checks that record has a valid number of related records. For example if Contact record classified as a Customer it must have at least one associated Order ($\text{Cardinality} > 0$). If order does not exist for a "customer" record then it must be either changed to "seed" or the order must be created. This type of rule can be complicated by additional conditions. For example if contact record in Payroll database is marked as "former employee", then this record must not have any associated salary payments after the date on which employee left organization ($\text{Cardinality} = 0$).

Check digits

Used for numerical data. An extra digit is added to a number which is calculated from the digits. The computer checks this calculation when data are entered. For example the last digit of an ISBN for a book is a check digit calculated modulus 10.

Consistency checks

Checks fields to ensure data in these fields corresponds, e.g., If Title = "Mr.", then Gender = "M".

Control totals

This is a total done on one or more numeric fields which appears in every record. This is a meaningful total, e.g., add the total payment for a number of Customers.

Cross-system consistency checks

Compares data in different systems to ensure it is consistent, e.g., The address for the customer with the same id is the same in both systems. The data may be represented differently in different systems and may need to be transformed to a common format to be compared, e.g., one system may store customer name in a single Name field as 'Doe, John Q', while another in three different fields: First_Name (John), Last_Name (Doe) and Middle_Name (Quality); to compare the two, the validation engine would have to transform data from the second system to match the data from the first, for example, using SQL:

`Last_Name || ', ' || First_Name || substr(Middle_Name, 1, 1)` would convert the data from the second system to look like the data from the first 'Doe, John Q'

Data type checks

Checks the data type of the input and give an error message if the input data does not match with the chosen data type, e.g., In an input box accepting numeric data, if the letter 'O' was typed instead of the number zero, an error message would appear.

File existence check

Checks that a file with a specified name exists. This check is essential for programs that use file handling.

Format or picture check

Checks that the data is in a specified format (template), e.g., dates have to be in the format DD/MM/YYYY.

Regular expressions should be considered for this type of validation.

Hash totals

This is just a batch total done on one or more numeric fields which appears in every record. This is a meaningless total, e.g., add the Telephone Numbers together for a number of Customers.

Limit check

Unlike range checks, data are checked for one limit only, upper OR lower, e.g., data should not be greater than 2 (≤ 2).

Logic check

Checks that an input does not yield a logical error, e.g., an input value should not be 0 when there will be a number that divides it somewhere in a program.

Presence check

Checks that important data are actually present and have not been missed out, e.g., customers may be required to have their telephone numbers listed.

Range check

Checks that the data lie within a specified range of values, e.g., the month of a person's date of birth should lie between 1 and 12.

Referential integrity

In modern Relational database values in two tables can be linked through foreign key and primary key. If values in the primary key field are not constrained by database internal mechanism,[4] then they should be validated. Validation of the foreign key field checks that referencing table must always refer to a valid row in the referenced table.

Spelling and grammar check

Looks for spelling and grammatical errors.

Uniqueness check

Checks that each value is unique. This can be applied to several fields (i.e. Address, First Name, Last Name).

Table Look Up Check

A table look up check takes the entered data item and compares it to a valid list of entries that are stored in a database table.

Testing

Testing techniques and testing strategies used

FT application will be tested using following strategies to ensure that the application succeeds to complete all the functional and non functional requirements:

Database & Data Integrity Testing

The databases and the database processes should be tested as a subsystem within the **FT** Application. These subsystems should be tested with the target-of-test's User Interface as the interface to the database.

Test Objective:	Ensure that data is stored correctly, audits can be performed, access is controlled
Technique:	SQL queries will be executed in the DB to verify the data content and correctness.
Completion Criteria:	All planned tests have been executed. All defects that have been identified have been resolved All resolutions have been implemented.

Functional Testing:

Function testing focuses on any requirements for test that can be traced directly to use cases or business functions and business rules. The goals of these tests are to verify proper data acceptance, processing, and retrieval, and the appropriate implementation of the business rules. This type of testing is based upon black box techniques; that are verifying the application and its internal processes by interacting with the application via the Graphical User Interface (GUI) and analyzing the output or results. Identified below is an outline of the function testing recommended for **FT**:

Test Objective:	Ensure proper target-of-test functionality, including business process validation.
Technique:	Execute each use case, use-case flow, or function, using valid and invalid data, to verify the following: <ul style="list-style-type: none">• The expected results occur when valid data is used.• The appropriate error or warning messages are displayed when invalid data is used.• Business rules are properly applied.• Black Box end to end testing of configured processes. Manual validation of required and optional fields.
Completion Criteria:	<ul style="list-style-type: none">• All planned tests have been executed.• All defects that have been identified have been resolved• All resolutions have been implemented.

Regression Testing:

Regression testing focuses on software functionality that may have been previously working however through subsequent changes may have been inadvertently impacted. The goals of these tests are to verify that the broader impact of changes has been verified. Identified below is an outline of the regression testing recommended for each application(s)/module(s) of **FT**.

Test Objective:	Ensure that previously passed test cases continue to pass as the new system development is deployed and that surrounding systems that may be impacted by a change are still functioning as expected.
Technique:	<ul style="list-style-type: none">• Execute previous passed testing suites to ensure the following:• The expected results occur when valid data is used.• The appropriate error or warning messages are displayed when invalid data is used.• Each business rule is properly applied.
Completion Criteria:	<ul style="list-style-type: none">• All planned regression tests have been executed.• All identified defects have been resolved.

User Interface Testing:

User Interface (UI) testing verifies a user's interaction with the software. The goal of UI testing is to ensure that the User Interface provides the user with the appropriate access and navigation through the functions of the target-of-test. In addition, UI testing ensures that the objects within the UI function as expected and conform to corporate or industry standards. Most of this testing will have been done during functional testing. The areas of focus will be on design, layout and navigation of the screens.

Test Objective:	UI testing will verify the screens and the layouts and navigation
Technique:	<ul style="list-style-type: none">• Verify the design and layout of the screen.• Identify the integration links.• Test the functioning of the links – that the proper page is displayed and correct messages, pop-ups are shown when they need to be displayed etc• Validation of general navigation
Completion Criteria:	<ul style="list-style-type: none">• All navigation test cases have been executed.• All screens have been verified as per design and layouts• All defects that have been identified have been resolved.

Performance Profiling:

Performance profiling is a performance test in which response times, transaction rates, and other time-sensitive requirements are measured and evaluated. The goal of Performance Profiling is to verify performance requirements have been achieved. Performance profiling is implemented and executed to profile and tune performance behaviours as a function of conditions such as workload or hardware configurations

Test Objective:	The purpose of performance profiling is to ensure the performance of the FT application is up to the desired level.
Technique:	<ul style="list-style-type: none"> • Use a subset of Test Procedures developed for Function and Business Cycle Testing. • Modify data files to increase the number of transactions or the scripts to increase the number of iterations each transaction occurs. • This will be done by using Load Runner or Quick Test Professional (QTP).
Completion Criteria:	<ul style="list-style-type: none"> • Single Transaction or single user: Successful completion of the test scripts without any failures and within the expected or required time allocation per transaction. • Results are recorded and a performance baseline is created for the major logical functions within the scenarios listed above. • All performance defects are reviewed and triaged to an acceptable resolution.

Load Testing:

Load testing is a performance test which subjects the target-of-test to varying workloads to measure and evaluate the performance behaviours and ability of the target-of-test to continue to function properly under these different workloads. The goal of load testing is to determine and ensure that the system functions properly at the expected maximum workload. Additionally, load testing evaluates the performance characteristics, such as response times, transaction rates, and other time sensitive issues.

Test Objective:	The purpose of load testing is to verify performance behaviour time for designated transactions or business cases under varying workload conditions.
Technique:	<ul style="list-style-type: none"> • Use a subset of Test Procedures developed for Function and Business Cycle Testing. • Scripts will be executed to simulate the peak load for 1 hour and report will be generated and analysed. • This will be done using Load Runner.
Completion Criteria:	<ul style="list-style-type: none"> • Multiple transactions or multiple users: Successful completion of the test scripts without any failures and within acceptable time allocation. • Results are recorded and a performance baseline is created for the major business functions within the scenarios listed above. • All load testing defects are reviewed and triaged to an acceptable resolution.

Stress Testing:

Stress testing is a type of performance test implemented and executed to find errors due to low resources or competition for resources. Low memory or disk space may reveal defects in the target-of-test that aren't apparent under normal conditions. Other defects might result from competition for shared resources like database locks or network bandwidth. Stress

testing can also be used to identify the peak workload the target-of-test can handle, which is often beyond the production workload.

Volume Testing:

Volume Testing subjects the target-of-test to large amounts of data to determine if limits are reached that cause the software to fail. Volume Testing also identifies the continuous maximum load or volume the target-of-test can handle for a given period. For example, if the target-of-test is processing a set of database records to generate a report, a Volume Test would use a large test database and check that the software behaved normally and produced the correct report.

Security & Access Control Testing:

Security and Access Control Testing focus on following key areas of security:

- Application-level security, including access to the Data or Business Functions

Application-level security ensures the authentication and authorization of a user. Authentication ensures that the user is a valid user of the system and authorization ensures that the user has the proper privileges to perform specific tasks on desired resources of the system. Testing will be conducted to validate the rules by taking into considerations the various roles applicable for the system.

Failover & Recovery Testing:

Failover and Recovery Testing ensures that the target-of-test can successfully failover and recover from a variety of hardware, software or network malfunctions with undue loss of data or data integrity.

Failover testing ensures that, for those systems that must be kept running, when a failover condition occurs, the alternate or backup systems properly “take over” for the failed system without loss of data or transactions.

Recovery testing is an antagonistic test process in which the application or system is exposed to extreme conditions, or simulated conditions, to cause a failure, such as device Input/Output (I/O) failures or invalid database pointers and keys. Recovery processes are invoked and the application or system is monitored and inspected to verify proper application, or system, and data recovery has been achieved.

Configuration Testing:

Configuration testing verifies the operation of the target-of-test on different software and hardware configurations. In most production environments, the particular hardware specifications for the client workstations, network connections and database servers vary. Client workstations may have different software loaded—for example, applications, drivers, and so on—and at any one time, many different combinations may be active using different resources.

Installation/Deploy & Back out Testing:

Installation testing has two purposes. The first is to ensure that the software can be installed under different conditions—such as a new installation, an upgrade and a complete or custom installation—under normal and abnormal conditions. Abnormal conditions include insufficient disk space, lack of privilege to create directories, and so on. The second purpose is to verify

that, once installed; the software operates correctly and can be backed out successfully. This usually means running a number of the tests that were developed for Function testing before and after the back out.

Post Production Testing:

The purpose of Post production testing is to verify that, once deployed, the software operates correctly. This usually means running a number of the tests that were developed for Function Testing ensuring that no data is changed/modified in production. Typically, the business SME's assist with Post production testing.

Unit Testing:

Unit testing will take place within the construction phase of the project. After application module has been built to meet design specifications, each component (screen, view, interface, conversion program, etc.) will be tested individually to help confirm that it functions properly as an individual unit. Basic performance testing will also be completed during unit test to resolve obvious issues with performance prior to the System Testing.

The resource responsible for development will conduct testing of their module using the unit test conditions defined by the developer based on detailed design documents. The final step of unit test will be a review by the team lead to obtain their signoff on the component test checklist.

Smoke Testing:

Test Objective:	Verifies the major functionality at high level in order to determine if further testing is possible.
Technique:	<ul style="list-style-type: none">• After initial deployment to the test environment validate all critical components of the application prior to proceeding with testing.
Completion Criteria:	<ul style="list-style-type: none">• Navigation through the application at high level is possible, testing can continue.

Data Migration Testing:

This is the process of testing to verify whether or not the data migration (or conversion) has been successfully completed. The testing process will be carried out by running SQL scripts on both the source and destination databases.

The fields which are present in the newdata Model in the Destination DB(s) will be migrated from the existing systemssource DB(s) to Destination DB(s).

Test Objective:	The objective of this test is to verify that data migration is successful from source DB(s) to destination DB(s).
Technique:	<ul style="list-style-type: none"> • The Team is notified before the data migration. • Team runs queries on the source DB and fetches the data. • Data Migration is done. • Mapped data needs to be determined. • Team runs the queries on the Destination DB and fetches the data. • Cross verification of the data is done to see that data fetched from the old database is same as the data fetched from the new database. • Verification of the table structure. • Verification of record counts. • Verification of the data formatting.
Completion Criteria:	<ul style="list-style-type: none"> • Data fetched from the Source DB(s) and Destination DB(s) matches. • The record count in the Source and the Destination databases should be equal. • No data are truncated. • Data formatting is proper (if required at any instance). • All defects that have been identified have been resolved.

Testing Plan used

Testing is a set of activities that can be planned in advance and conducted systematically. During testing, the program to be tested is executed with a set of test cases, and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to.

In a software development project, errors can be injected at any stage during the development. Some requirement errors and design errors are likely to remain undetected. Ultimately, these errors will be reflected in the code. Hence, testing performs a very critical role for quality assurance and for ensuring the reliability of software.

Cause of Testing:

The first test of the system is to see whether it produces the correct output. Following this a variety of other tests are conducted:

Response time: this test is conducted to measure the processing of the software.

Volume testing: In this test, we create as many data as would normally be used to a variety that the hardware and the software will function correctly.

Stress Testing: The purpose of this is to prove that the candidate system should not malfunction under peak load.

Recovery & Security: A forced system is induced to test a backup recovery procedure.

System testing is verification of the workability of a system. For this purpose the system is used experimentally to ensure that it will run according to its specification and in the way users expect. These are two stages to this:

- ❖ The testing of the individual programs by their programmers called Unit testing and
- ❖ The testing of the overall System testing.

Unit testing includes the following:

- ❖ Test for number of input parameters equal to number of arguments.
- ❖ Test the parameter and argument attributes match.
- ❖ Feasibility and validity checks on input data.
- ❖ Checks for interpretation of symbols correctly.
- ❖ Checks for accurate branching and looping.
- ❖ Checks on logical file addressing and searching.
- ❖ Checks on the capacity of storage areas and buffers.
- ❖ Checks on updating procedure.
- ❖ Checks on contents and layout of printed and displayed output.
- ❖ Checks for batch control totals.
- ❖ Checks on interfacing with other programs, software, database and operating system.
- ❖ Checks on documentation.

System testing includes the following:

- ❖ Interfacing of run within the system.
- ❖ Compilation and continuity of control totals.
- ❖ Capacity of logical file storage areas and the handling of overflow.
- ❖ Error correction procedures including user involvement.
- ❖ User request for amendments and output.
- ❖ Timing of runs and routines for the data volumes to be actually handled.
- ❖ Output preparation and distribution.
- ❖ Audit requirements.
- ❖ Logical physical file housekeeping and control.

The usual procedures in testing are to create data for the initial tests and to use live data for later testing.

The following points should be remembered primarily:

- ❖ Both the artificial and live data should be representative of reality;
- ❖ Live data can often be checked against the previous system's result;
- ❖ If the previous and new system differ, the two sets of result should be reconciled if at all possible;
- ❖ Logical files are usually needed to fully test the programs and routines;
- ❖ Data generating techniques are useful for simulating large volume of input data file records;
- ❖ In the final trial run of the complete routine, asset of input data is passed through to the resultant output and/or file updating stage(s);
- ❖ Test data should include known incorrect data with a view to test the validation and control procedure.

Any engineered product (and most other things) can be tested in one of two ways:

- ❖ Knowing the specified function that a product has been designed to perform, test can be conducted that demonstrate each function is fully operational, at the same time searching for errors in each function;
- ❖ Knowing the internal workings of a product, tests can be conducted to ensure that all internal operation performs according to specification and all internal components have been adequately exercised. The first test approach is called Black-box and the second, White-box testing.

White-box testing:

White-box testing sometimes called glass-box testing, is a test case design to derive test cases. Using white-box testing method, the software engineer can test:

- ❖ Guarantee that all independent paths within a module have been exercised at least one;
- ❖ Exercise all logical decisions on their true and false sides;
- ❖ Execute all loops at there boundaries and within there operational bounds; and
- ❖ Exercise internal data structures to assure there validity.

A reasonable question might be posed at this juncture: “why spend time and energy worrying about (and testing) logical minutes when we might better spend effort ensuring that program requirements have been met?” stated another way, why don't we spend all of our energies on black-box testing? The answer lies in the nature of software defects.

- Logical errors and incorrect assumptions are inversely proportional to the probability that a program path will be executed. Errors tend to creep into our work when we design and implement function, conditions and control that are out of the mainstream. Every processing tends to be well understood (and well scrutinized), while “special case” processing tends to fall into the cracks.

- We often believe that a logical path is not likely to be executed when, in fact, it may be executed in regular basis. The logical flow of a program is sometimes counterintuitive, meaning that our unconscious assumption about flow of control and data may lead us to make design errors that are uncovered only when path testing commences.
- Typographical errors are random. When a program is translated into programming language source code, it is likely that some typing error will occur. Many will be uncovered by syntax checking mechanism, but others will go undetected until testing begins. It is likely that a type will exist on an obscure logical path as on a mainstream path.

Each of these reasons provides an argument for conducting white-box tests. Black-box testing, no matter how thorough, may miss the kinds of errors noted above. As Beizer has stated: “Bugs lurk in corners and congregate at boundaries”. White-box testing is far more likely to uncover them.

Basis Path Testing:

Basis path testing is a **White-box testing** technique first proposed by Tom McCabe. The basis path methods enables the test case designer to derive a logical complexity measure of a procedural design and use this measure as a guide for defining a basic set that are guaranteed to execute every statement in the program at least one time during testing.

Black-box Testing:

It focuses on the functional requirements of the software. That is Black-box testing enables software engineer to derive sets of input conditions that will fully exercise all functional requirements of a program. Black-box testing is not a alternative of white box testing. Rather, it is a complementary approach that is likely to uncover a different class of errors than White-box methods.

Black-box testing attempts to find errors in the following categories.

- ❖ Incorrect or missing function.
- ❖ Interface errors
- ❖ Errors in data structures or external data base access
- ❖ Performance errors
- ❖ Initializations or termination errors

Unlike **White-box** testing, which is performed early in the testing process, **black-box** testing tends to be applied during later stage of testing. Because **black-box** testing purposely

disregards control structure, attention is focused on the information domain. Tests are based on source data from a

previous period so that the result from the new system can be compared with that of the old one.

With those of the previous ones to answer the following questions:

- ❖ How is functional validity testing?
- ❖ What classes of input will make good test cases?
- ❖ Is the system particularly sensitive to certain input values?
- ❖ How are the boundaries of a data class isolated?
- ❖ What data rates and data volume can the system tolerate?
- ❖ What effect will specific combinations of the data have on system operation?

By applying **Black-box** techniques, we derive a set of test cases that satisfy the following criteria:

- ❖ Test cases that reduce, by errors and a designed to achieve reasonableness testing.
- ❖ Test cases that tell us something about the presence or absence of classes of errors, than errors associated only with the specific test at hand.

Test reports for Unit Test Cases and System Test Cases

Test reports for Unit Test Cases

Test Case Id	Comment	Status
FT-001	NA	PASS
FT-002	NA	PASS
FT-003	NA	PASS
FT-004	NA	PASS
FT-005	NA	PASS
FT-006	NA	PASS
FT-007	NA	PASS
FT-008	NA	PASS
FT-009	NA	PASS
FT-010	NA	PASS
FT-011	NA	PASS
FT-012	NA	PASS
FT-013	NA	PASS
FT-014	NA	PASS

FT-015	NA	PASS
FT-016	NA	PASS
FT-017	NA	PASS
FT-018	NA	PASS
FT-019	NA	PASS
FT-020	NA	PASS
FT-021	NA	PASS
FT-022	NA	PASS
FT-023	NA	PASS
FT-024	NA	PASS
FT-025	NA	PASS
FT-026	NA	PASS
FT-027	NA	PASS
FT-028	NA	PASS
FT-029	NA	PASS
FT-030	NA	PASS
FT-031	NA	PASS
FT-032	NA	PASS
FT-033	NA	PASS
FT-034	NA	PASS
FT-035	NA	PASS
FT-036	NA	PASS
FT-037	NA	PASS
FT-038	NA	PASS
FT-039	NA	PASS
FT-040	NA	PASS
FT-041	NA	PASS
FT-042	NA	PASS
FT-043	NA	PASS
FT-044	NA	PASS
FT-045	NA	PASS
FT-046	NA	PASS
FT-047	NA	PASS
FT-048	NA	PASS
FT-049	NA	PASS
FT-050	NA	PASS
FT-051	NA	PASS
FT-052	NA	PASS
FT-053	NA	PASS
FT-054	NA	PASS
FT-055	NA	PASS

FT-056	NA	PASS
FT-057	NA	PASS
FT-058	NA	PASS
FT-059	NA	PASS
FT-060	NA	PASS
FT-061	NA	PASS
FT-062	NA	PASS
FT-063	NA	PASS
FT-064	NA	PASS
FT-065	NA	PASS
FT-066	NA	PASS

Test reports for System Test Cases

Test Case Id	Comment	Status
FT-067	NA	PASS
FT-068	NA	PASS
FT-069	NA	PASS
FT-070	NA	PASS
FT-071	NA	PASS
FT-072	NA	PASS
FT-073	NA	PASS
FT-074	NA	PASS
FT-075	NA	PASS
FT-076	NA	PASS
FT-077	NA	PASS
FT-078	NA	PASS
FT-079	NA	PASS
FT-080	NA	PASS
FT-081	NA	PASS
FT-082	NA	PASS
FT-083	NA	PASS
FT-084	NA	PASS
FT-085	NA	PASS
FT-086	NA	PASS
FT-087	NA	PASS
FT-089	NA	PASS
FT-090	NA	PASS
FT-091	NA	PASS

FT-092	NA	PASS
FT-093	NA	PASS

Debugging and Code improvement:

The steps in the bellow section demonstrate how to create a console application that uses the **Debug** class to provide information about the program execution.

When the program is run, we can use methods of the **Debug** class to produce messages that help we to monitor the program execution sequence, to detect malfunctions, or to provide performance measurement information. By default, the messages that the **Debug** class produces appear in the Output window of the Visual Studio Integrated Development Environment (IDE).

The sample code uses the **WriteLine** method to produce a message that is followed by a line terminator. When we use this method to produce a message, each message appears on a separate line in the Output window.

When we use the **Assert** method of the **Debug** class, the Output window displays a message only if a specified condition evaluates to false. The message also appears in a modal dialog box to the user. The dialog box includes the message, the project name, and the **Debug.Assert** statement number. The dialog box also includes the following three command buttons:

- **Abort:** The application stops running.
- **Retry:** The application enters debug mode.
- **Ignore:** The application proceeds.

The user must click one of these buttons before the application can continue.

We can also direct output from the **Debug** class to destinations other than the Output window. The **Debug** class has a collection named **Listeners** that includes **Listener** objects.

Each **Listener** object monitors **Debug** output and directs the output to a specified target.

Each **Listener** in the **Listener** collection receives any output that the **Debug** class generates. Use the **TextWriterTraceListener** class to define **Listener** objects. We can specify the target for a **TextWriterTraceListener** class through its constructor.

Some possible output targets include the following:

- The Console window by using the **System.Console.Out** property.
- A text (.txt) file by using the **System.IO.File.CreateText("FileName.txt")** statement.

After we create a **TextWriterTraceListener** object, we must add the object to the **Debug.Listeners** collection to receive Debug output.

Create a Sample with the Debug Class

1. Start Visual Studio or Visual C# Express Edition.
2. Create a new Visual C# Console Application project named **conInfo**. Class1 is created in Visual Studio .NET. Program.cs is created in Visual Studio 2005.
3. Add the following namespace at top in Class1 or Program.cs.
`using System.Diagnostics;`
4. To initialize variables to contain information about a product, add the following declaration statements to **Main** method:
5. `string sProdName = "Widget";`
6. `int iUnitQty = 100;`
`double dUnitCost = 1.03;`
7. Specify the message that the class produces as the first input parameter of the **WriteLine** method. Press the CTRL+ALT+O key combination to make sure that the Output window is visible.
`Debug.WriteLine("Debug Information-Product Starting ");`
8. For readability, use the **Indent** method to indent subsequent messages in the Output window:
`Debug.Indent();`
9. To display the content of selected variables, use the **WriteLine** method as follows:
10. `Debug.WriteLine("The product name is " + sProdName);`
11. `Debug.WriteLine("The available units on hand are" + iUnitQty.ToString());`
`Debug.WriteLine("The per unit cost is " + dUnitCost.ToString());`
12. We can also use the **WriteLine** method to display the namespace and the class name for an existent object. For example, the following code displays the **System.Xml.XmlDocument** namespace in the Output window:
13. `System.Xml.XmlDocument oxml = new System.Xml.XmlDocument();`
`Debug.WriteLine(oxml);`
14. To organize the output, we can include a category as an optional, second input parameter of the **WriteLine** method. If we specify a category, the format of the Output window message is "category: message." For example, the first line of the following code displays "Field: The product name is Widget" in the Output window:
15. `Debug.WriteLine("The product name is " + sProdName,"Field");`
16. `Debug.WriteLine("The units on hand are" + iUnitQty,"Field");`
17. `Debug.WriteLine("The per unit cost is" + dUnitCost.ToString(),"Field");`
`Debug.WriteLine("Total Cost is " + (iUnitQty * dUnitCost),"Calc");`
18. The Output window can display messages only if a designated condition evaluates to true by using the **WriteLineIf** method of the **Debug** class. The condition to be evaluated is the first input parameter of the **WriteLineIf** method. The second parameter of **WriteLineIf** is the message that appears only if the condition in the first parameter evaluates to true.
19. `Debug.WriteLineIf(iUnitQty > 50, "This message WILL appear");`
20. `Debug.WriteLineIf(iUnitQty < 50, "This message will NOT appear");`
21. Use the **Assert** method of the **Debug** class so that the Output window displays the message only if a specified condition evaluates to false:
22. `Debug.Assert(dUnitCost > 1, "Message will NOT appear");`

```

23. Debug.Assert(dUnitCost < 1, "Message will appear since dUnitcost < 1 is false");
24. Create the TextWriterTraceListener objects for the Console window (tr1) and for a
    text file named Output.txt (tr2), and then add each object to the Debug
    Listeners collection:
25. TextWriterTraceListener tr1 = new TextWriterTraceListener(System.Console.Out);
26. Debug.Listeners.Add(tr1);
27.
28. TextWriterTraceListener tr2 = new
    TextWriterTraceListener(System.IO.File.CreateText("Output.txt"));
    Debug.Listeners.Add(tr2);
29. For readability, use the Unindent method to remove the indentation for subsequent
    messages that the Debug class generates. When we use the Indent and
    the Unindent methods together, the reader can distinguish the output as group.
30. Debug.Unindent();
    Debug.WriteLine("Debug Information-Product Ending");
31. To make sure that each Listener object receives all its output, call the Flush method
    for the Debug class buffers:
    Debug.Flush();

```

Using the Trace Class

We can also use the **Trace** class to produce messages that monitor the execution of an application. The **Trace** and **Debug** classes share most of the same methods to produce output, including the following:

- **WriteLine**
- **WriteLineIf**
- **Indent**
- **Unindent**
- **Assert**
- **Flush**

We can use the **Trace** and the **Debug** classes separately or together in the same application. In a Debug Solution Configuration project, both **Trace** and **Debug** output are active. The project generates output from both of these classes to all **Listener** objects. However, a Release Solution Configuration project only generates output from a **Trace** class. The Release Solution Configuration project ignores any **Debug** class method invocations.

```

Trace.WriteLine("Trace Information-Product Starting ");
Trace.Indent();

Trace.WriteLine("The product name is "+sProdName);
Trace.WriteLine("The product name is "+sProdName,"Field" );
Trace.WriteLineIf(iUnitQty > 50, "This message WILL appear");
Trace.Assert(dUnitCost > 1, "Message will NOT appear");

Trace.Unindent();
Trace.WriteLine("Trace Information-Product Ending");

Trace.Flush();

```

```
Console.ReadLine();
```

Verify That It Works

1. Make sure that **Debug** is the current solution configuration.
2. If the **Solution Explorer** window is not visible, press the CTRL+ALT+L key combination to display this window.
3. Right-click **conInfo**, and then click **Properties**.
4. In the left pane of the **conInfo** property page, under the **Configuration** folder, make sure that the arrow points to **Debugging**.

Note In Visual C# 2005 and in Visual C# 2005 Express Edition, click **Debug** in the **conInfo** page.

5. Above the **Configuration** folder, in the **Configuration** drop-down list box, click **Active (Debug)** or **Debug**, and then click **OK**. In Visual C# 2005 and in Visual C# 2005 Express Edition, click **Active (Debug)** or **Debug** in the **Configuration** drop-down list box in the **Debug** page, and then click **Save** on the **File** menu.
6. Press CTRL+ALT+O to display the Output window.
7. Press the F5 key to run the code. When the **Assertion Failed** dialog box appears, click **Ignore**.
8. In the Console window, press ENTER. The program should finish, and the Output window should display the output that resembles the following
9. Debug Information-Product Starting
10. The product name is Widget
11. The available units on hand are 100
12. The per unit cost is 1.03
13. System.Xml.XmlDocument
14. Field: The product name is Widget
15. Field: The units on hand are 100
16. Field: The per unit cost is 1.03
17. Calc: Total Cost is 103
18. This message WILL appear
19. ---- DEBUG ASSERTION FAILED ----
20. ---- Assert Short Message ----
21. Message will appear since dUnitcost < 1 is false
22. ---- Assert Long Message ----
- 23.
- 24.
25. at Class1.Main(String[] args) <%Path%>\class1.cs(34)
- 26.
27. The product name is Widget
28. The available units on hand are 100
29. The per unit cost is 1.03
30. Debug Information-Product Ending
31. Trace Information-Product Starting
32. The product name is Widget
33. Field: The product name is Widget
34. This message WILL appear
35. Trace Information-Product Ending

- 36.
37. The Console window and the Output.txt file should display the following output:
38. The product name is Widget
39. The available units on hand are 100
40. The per unit cost is 1.03
41. Debug Information-Product Ending
42. Trace Information-Product Starting
43. The product name is Widget
44. Field: The product name is Widget
45. This message WILL appear
46. Trace Information-Product Ending

Note The Output.txt file is located in the same directory as the conInfo executable (conInfo.exe). Typically, this is the \bin folder where the project source is stored. By default, this is C:\Documents and Settings*User login*\My Documents\Visual Studio Projects\conInfo\bin. In Visual C# 2005 and in Visual C# 2005 Express Edition, the Output.txt file is located in the following folder:

C:\Documents and Settings*User login*\My Documents\Visual Studio
2005\Projects\conInfo\conInfo\bin\Debug

Complete Code Listing

```
using System;
using System.Diagnostics;

class Class1
{
    [STAThread]
    static void Main(string[] args)
    {
        string sProdName = "Widget";
        int iUnitQty = 100;
        double dUnitCost = 1.03;
        Debug.WriteLine("Debug Information-Product Starting ");
        Debug.Indent();
        Debug.WriteLine("The product name is "+sProdName);
        Debug.WriteLine("The available units on hand are"+iUnitQty.ToString());
        Debug.WriteLine("The per unit cost is "+ dUnitCost.ToString());

        System.Xml.XmlDocument oxml = new System.Xml.XmlDocument();
        Debug.WriteLine(oxml);

        Debug.WriteLine("The product name is "+sProdName,"Field");
        Debug.WriteLine("The units on hand are"+iUnitQty,"Field");
        Debug.WriteLine("The per unit cost is"+dUnitCost.ToString(),"Field");
        Debug.WriteLine("Total Cost is "+(iUnitQty * dUnitCost),"Calc");
```

```

Debug.WriteLineIf(iUnitQty > 50, "This message WILL appear");
Debug.WriteLineIf(iUnitQty < 50, "This message will NOT appear");

Debug.Assert(dUnitCost > 1, "Message will NOT appear");
Debug.Assert(dUnitCost < 1, "Message will appear since dUnitcost < 1 is false");

    TextWriterTraceListener tr1 = new TextWriterTraceListener(System.Console.Out);
Debug.Listeners.Add(tr1);

    TextWriterTraceListener tr2 = new
TextWriterTraceListener(System.IO.File.CreateText("Output.txt"));
Debug.Listeners.Add(tr2);

Debug.WriteLine("The product name is "+sProdName);
Debug.WriteLine("The available units on hand are"+iUnitQty);
Debug.WriteLine("The per unit cost is "+dUnitCost);
Debug.Unindent();
Debug.WriteLine("Debug Information-Product Ending");
Debug.Flush();

Trace.WriteLine("Trace Information-Product Starting ");
Trace.Indent();

Trace.WriteLine("The product name is "+sProdName);
Trace.WriteLine("The product name is "+sProdName,"Field" );
Trace.WriteLineIf(iUnitQty > 50, "This message WILL appear");
Trace.Assert(dUnitCost > 1, "Message will NOT appear");

Trace.Unindent();
Trace.WriteLine("Trace Information-Product Ending");

Trace.Flush();

Console.ReadLine();
    }
}

```

Troubleshoot

- If the solution configuration type is **Release**, the **Debug** class output is ignored.
- After we create a **TextWriterTraceListener** class for a particular target, **TextWriterTraceListener** receives output from the **Trace** and the **Debug** classes. This occurs regardless of whether we use the **Add** method of the **Trace** or the **Debug** class to add **TextWriterTraceListener** to the **Listeners** class.
- If we add a **Listeners** object for the same target in the **Trace** and the **Debug** classes, each line of output is duplicated, regardless of whether **Debug** or **Trace** generates the output.

```
TextWriterTraceListener myWriter = new  
TextWriterTraceListener(System.Console.Out);  
Debug.Listeners.Add(myWriter);
```

```
TextWriterTraceListener myCreator = new  
TextWriterTraceListener(System.Console.Out);  
Trace.Listeners.Add(myCreator);
```

System Security measures:

Database/data security:

It encrypts the data stored in the database so that even if someone succeeds to hack the database still not much harm could be done.

The application is authentication based. Without proper username and password no person can access the application.

Creation of User profiles and access rights

The software requires a predefined username and password to login.

It allows a guest login as well which lets a guest user this application with very limited access to the user data.

Cost Estimation of the Project along with Cost Estimation Model

Software development is a highly labor intensive activity. A project of large dimension can easily turn into chaos if proper management controls are not imposed. Therefore the cost/expenditure and the profit gained after implementing the project has to be taken into account. That is we have to consider the cost benefit analysis.

This cost/benefit may be tangible or intangible, direct or indirect, fixed or variable. To build up a large software all the elements required, are estimated to get the development cost of the considering project. When we consider all this requirements we can develop a cost estimation model to find proposed cost of the developing project. And from this model we can track down the expenditure during the course of development.

Now after implementing the project we have to consider gain from it in terms of benefits, that is how much person month can be saved from this project. Therefore we have to consider the total expenditure and the benefit gain from the project once it has been implemented. Here we express the benefits in the terms of person month that is monthly salary of the person concerned for the system, which has to be replaced. Therefore this

cost/benefit analysis report gives us a total picture of how a company gets benefit from candidate system once that has replace a older one.

The project developing components like hardware, personnel, facility and supply cost are also taken into consideration during the cost estimation. Then we identify the cost and benefit of a given system and categories them or analysis. And from that estimated cost we track the expenditure and then calculate the benefits.

In developing the cost estimation of a project we need to consider several cost elements. Among them is hardware, personal, facility, operating and supply cost are noteworthy.

The model for estimating cost is mainly based on the total lines of cop1 delivered. As this is not such code based rather than a plign based project so we estimate the cost on the consideration of how much time it can take in pligning the user interfaces and how many interfaces are required. The cost is then calculated from the total plign hours and as it is a single handed project, so this is the time taken by a single person.

The cost of man-power involved in this project is not considered in this estimation. We should consider the cost when we shall go for any live project. This cost is depending on several factors like skill set, location of country etc. e.g. man-hour cost is around Rs.250.00 to Rs.300.00 in India whereas for USA it varies from US\$60.00 to US\$200.00. Most of the cases, Man (person) power cost is considerable higher than all other costs. Software cost and effort estimation will never be an exact science. Too many variables human, technical, environmental can affect the ultimate cost of the software and effort applied to develop it. To achieve reliable cost and effort estimates, a number of option arise, 1) Base estimates on similar projects that have been already completed; 2) Using relatively simple “decomposition techniques” to generate project cost and effort estimates; 3) Using one or more empirical models for software cost and effort estimation.

We used the basic COCOMO model, which gives an approximate estimate of our **FT** project parameters. The basic COCOMO estimation model is given by the following expressions:

$$\text{Effort} = a_1 * (\text{KLOC})^{a_2} \text{ PM}$$

$$\text{Tdev} = b_1 * (\text{Effort})^{b_2} \text{ months}$$

Where

KLOC is the estimated size of the software product expressed in Kilo Lines of Code a_1 , a_2 , b_1 , b_2 are constants for each category of software products.

Tdev is the estimated time to develop the software, expressed in months.

Effort is the total effort required to develop the software product, expressed in person-month (PM).

Our project is semidetached type, because the development team consists of a mixture of experienced and inexperienced staff like my guide and me. Team members may have limited experience on related system but may be unfamiliar with aspects of the system being developed.

Estimation of development effort

For our Semi-detached class software product **FT**, the formula for estimating the effort based on the code size is shown below:

Semi-detached **FT**: $T_{dev} = 4.0 * (KLOC)^{1.12}$ PM

Estimation of development time

For our Semi-detached class software product **FT**, the formula for estimating the development time based on the effort is given below:

Semi-detached **FT**: $T_{dev} = 3.5 * (Effort)^{0.35}$ months

Assume that the size of a Semi-detached **FT** product has been estimated to be 3,630 lines of source code. Assume that the average salary of software engineer(me) is Rs. 20,000 per month.

Assume that the size of our

The basic COCOMO estimation formula for **FT** semidetached software:

Our Effort $= 3.0 * (3.2)^{1.12}$ PM

$= 11$ PM

Normal Development time $= 2.5 * (11)^{0.35}$ months

$= 6$ months

Cost required to develop the product $= Rs. 6 * 20000$

$= Rs. 130,000$

Reports

- List of calory loss/gain can be generated.
- List of weight improvment can be generated. .
- A list of expected wiight loss/gain report can be generated.

Future scope and further enhancement of the Project

- ❖ To Support Mobile Operating systems for Symbian, Meego, Firefox, Sailfish OS & Windows.
- ❖ To support UNIX / Linux Based Operating systems.
- ❖ To create a Windows/Linux based desktop application and link it up with web/Android application.

Bibliography

Website

- <http://en.wikipedia.org>
- <http://www.codeplex.com/>
- <http://stackoverflow.com/>
- <http://www.codeguru.com/>
- <http://www.w3schools.com>
- www.mysql.org
- <http://twitter.github.io/bootstrap/index.html>
- <http://developer.android.com/training/basics/firstapp/index.html>
- <http://ellislab.com/codeigniter/user-guide/>

Books

- Fundamentals of software engineering by Rajib Mall
- Pro C# 2010 and the .NET 4.0 Platform by Andrew Troselen
- C# Programming by Rob Miles

Appendices

IDE Used:

Netbeans 7.3.1

NetBeans is an open-source project dedicated to providing rock solid software development that address the needs of developers, users and the businesses who rely on NetBeans as a basis for their products; particularly, to enable them to develop these products quickly, efficiently and easily by leveraging the strengths of the Java platform and other relevant industry standards.

Features:

Best Support for Latest Java Technologies

NetBeans IDE provides first-class comprehensive support for the newest Java technologies and latest Java enhancements before other IDEs. It is the first IDE providing support for JDK 7, Java EE 7, and JavaFX 2.

With its constantly improving Java Editor, many rich features and an extensive range of tools, templates and samples, NetBeans IDE sets the standard for developing with cutting edge technologies out of the box.

Fast & Smart Code Editing

An IDE is much more than a text editor. The NetBeans Editor indents lines, matches words and brackets, and highlights source code syntactically and semantically. It also provides code templates, coding tips, and refactoring tools.

The editor supports many languages from Java, C/C++, XML and HTML, to PHP, Groovy, Javadoc, JavaScript and JSP. Because the editor is extensible, you can plug in support for many other languages.

Easy & Efficient Project Management

Keeping a clear overview of large applications, with thousands of folders and files, and millions of lines of code, is a daunting task. NetBeans IDE provides different views of your data, from multiple project windows to helpful tools for setting up your applications and managing them efficiently, letting you drill down into your data quickly and easily, while giving you versioning tools via Subversion, Mercurial, and Git integration out of the box.

Rapid User Interface Development

Design GUIs for Java EE, Java SE, and Java ME applications quickly and smoothly by dragging and positioning GUI components from a palette into the NetBeans Editor.

For Java SE applications, the NetBeans GUI Builder automatically takes care of correct spacing and alignment, while supporting in-place editing, as well. The GUI builder is so intuitive that it has been used to prototype GUIs at customer presentations.

Write Bug Free Code

The cost of buggy code increases the longer it remains unfixed. NetBeans provides static analysis tools, especially integration with the widely used FindBugs tool, for identifying and fixing common problems in Java code. In addition, the NetBeans Debugger lets you place breakpoints in your source code, add field watches, step through your code, run into methods, take snapshots and monitor execution as it occurs.

Eclipse IDE for Android

In [computer programming](#), **Eclipse** is a multi-language [Integrated development environment](#) (IDE) comprising a base [workspace](#) and an extensible [plug-in](#) system for customizing the environment. It is written mostly in [Java](#). It can be used to develop applications in Java and, by means of various plug-ins, other [programming languages](#) including [Ada](#), [C](#), [C++](#), [COBOL](#), [Fortran](#), [Haskell](#), [JavaScript](#), [Lasso](#), [Perl](#), [PHP](#), [Python](#), [R](#), [Ruby](#) (including [Ruby on Rails](#) framework), [Scala](#), [Clojure](#), [Groovy](#), [Scheme](#), and [Erlang](#). It can also be used to develop packages for the software [Mathematica](#). Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

Front End

XML

Extensible Markup Language (XML) is a [markup language](#) that defines a set of rules for encoding documents in a [format](#) that is both [human-readable](#) and [machine-readable](#). It is defined in the XML 1.0 Specification produced by the [W3C](#), and several other related specifications, all free [open standards](#).

The design goals of XML emphasize simplicity, generality, and usability over the [Internet](#). It is a textual data format with strong support via [Unicode](#) for the languages of the world. Although the design of XML

focuses on documents, it is widely used for the representation of arbitrary [data structures](#), for example in [web services](#).

Many [application programming interfaces](#) (APIs) have been developed to aid software developers with processing XML data, and several [schema systems](#) exist to aid in the definition of XML-based languages.

HTML

HyperText Markup Language (HTML) is the main [markup language](#) for creating [web pages](#) and other information that can be displayed in a [web browser](#).

HTML is written in the form of [HTML elements](#) consisting of *tags* enclosed in [angle brackets](#) (like `<html>`), within the web page content. HTML tags most commonly come in pairs like `<h1>` and `</h1>`, although some tags represent *empty elements* and so are unpaired, for example ``. The first tag in a pair is the *start tag*, and the second tag is the *end tag* (they are also called *opening tags* and *closing tags*). In between these tags web designers can add text, further tags, [comments](#) and other types of text-based content.

The purpose of a [web browser](#) is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page.

HTML elements form the building blocks of all [websites](#). HTML allows [images and objects](#) to be embedded and can be used to create [interactive forms](#). It provides a means to create [structured documents](#) by denoting structural [semantics](#) for text such as headings, paragraphs, lists, [links](#), quotes and other items. It can embed [scripts](#) written in languages such as [JavaScript](#) which affect the behavior of HTML web pages.

Codeigniter

CodeIgniter is an [open source](#) rapid development [web application framework](#), for use in building dynamic web sites with [PHP](#). "Its goal is to enable [developers] to develop projects much faster than writing code from scratch, by providing a rich set of libraries for commonly needed tasks, as well as a simple interface and logical structure to access these libraries." The first public version of CodeIgniter was released on February 28, 2006, and the latest stable version 2.1.4 was released July 8, 2013.

CodeIgniter is loosely based on the popular [Model-View-Controller](#) development pattern. While view and controller classes are a necessary part of development under CodeIgniter, models are optional.

ADT

The Android Developer Tools (ADT) plugin for Eclipse provides a professional-grade development environment for building Android apps. It's a full Java IDE with advanced features to help you build, test, debug, and package your Android apps.

Free, open-source, and runs on most major OS platforms.

Features:

Full Java IDE

- Android-specific refactoring, quick fixes, integrated navigation between Java and XML resources.
- Enhanced XML editors for Android XML resources.
- Static analysis tools to catch performance, usability, and correctness problems.
- Build support for complex projects, command-line support for CI through Ant. Includes ProGuard and app-signing.

- Template-based wizard to create standard Android projects and components.

Graphical UI Builders

- Build rich Android UI with drag and drop.
- Visualize your UI on tablets, phones, and other devices. Switch themes, locales, even platform versions instantly, without building.
- Visual refactoring lets you extract layout for inclusion, convert layouts, extract styles.
- Editor support for working with custom UI components.
-

Develop on Hardware Devices

- Use any commercial Android hardware device or multiple devices.
- Deploy your app to connected devices directly from the IDE.
- Live, on-device debugging, testing, and profiling.

Develop on Virtual Devices

- Emulate any device. Use custom screen sizes, keyboards, and other hardware components.
- Advanced hardware emulation, including camera, sensors, multitouch, and telephony.
- Develop and test for broad device compatibility.

Powerful Debugging

- Full Java debugger with on-device debugging and Android-specific tools.
- Built-in memory analysis, performance/CPU profiling, OpenGL ES tracing.
- Graphical tools for debugging and optimizing UI, runtime inspection of UI structure and performance.

- Runtime graphical analysis of your app's network bandwidth usage.

Testing

- Fully instrumentated, scriptable test environment.
- Integrated reports using standard test UI.
- Create and run unit tests on hardware devices or emulator.

Native Development

- Support for compiling and packaging existing code written in C or C++.
- Support for packaging multiple architectures in a single binary, for broad compatibility.

Database/backend:

MySQL



MySQL is the world's most popular open source database software, with over 100 million copies of its software downloaded or distributed throughout its history.

The MySQL Community Edition includes:

- *Pluggable Storage Engine Architecture*
 - *Multiple Storage Engines: InnoDB , MyISAM, NDB (MySQL Cluster),Memory ,Merge , Archive, CSV*
 - *MySQL Replication to improve application performance and scalability*
 - *MySQL Partitioning to improve performance and management of large database applications*
 - *Stored Procedures to improve developer productivity*

Detailed features of mysql

The following list shows the most important properties of MySQL. This section is directed to the reader who already has some knowledge of relational databases. We will use some terminology from the relational database world without defining our terms exactly. On the other hand, the explanations should make it possible for database novices to understand to some extent what we are talking about.

Relational Database System: Like almost all other database systems on the market, MySQL is a relational database system.

Client/Server Architecture: MySQL is a client/server system. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they query data, save changes, etc. The clients can run on the same computer as the server or on another computer (communication via a local network or the Internet).

Almost all of the familiar large database systems (Oracle, Microsoft SQL Server, etc.) are client/server systems. These are in contrast to the file-server systems, which include Microsoft Access, dBase and FoxPro. The decisive drawback to file-server systems is that when run over a network, they become extremely inefficient as the number of users grows.

SQL compatibility: MySQL supports as its database language -- as its name suggests -- SQL (Structured Query Language). SQL is a standardized language for querying and updating data and for the administration of a database. There are several SQL dialects (about as many as there are database systems). MySQL adheres to the current SQL standard (at the moment SQL:2003), although with significant restrictions and a large number of extensions.

Through the configuration setting `sql-mode` you can make the MySQL server behave for the most part compatibly with various database systems. Among these are IBM DB/2 and Oracle. (The setting `sql-mode` changes some of the syntax conventions, and performs no miracles.

SubSELECTs: Since version 4.1, MySQL is capable of processing a query in the form `SELECT * FROM table1 WHERE x IN (SELECT y FROM table2)` (There are also numerous syntax variants for subSELECTs.)

Views: Put simply, views relate to an SQL query that is viewed as a distinct database object and makes possible a particular view of the database. MySQL has supported views since version 5.0.

Stored procedures: Here we are dealing with SQL code that is stored in the database system.

Stored procedures (SPs for short) are generally used to simplify certain steps, such as inserting or deleting a data record. For client programmers this has the advantage that they do not have to process the tables directly, but can rely on SPs. Like views, SPs help in the administration of large database projects. SPs can also increase efficiency. MySQL has supported SPs since version 5.0.

Triggers: Triggers are SQL commands that are automatically executed by the server in certain database operations (INSERT, UPDATE, and DELETE). MySQL has supported

triggers in a limited form from version 5.0, and additional functionality is promised for version 5.1.

Unicode: MySQL has supported all conceivable character sets since version 4.1, including Latin-1, Latin-2, and Unicode (either in the variant UTF8 or UCS2).

User interface: There are a number of convenient user interfaces for administering a MySQL server.

Full-text search: Full-text search simplifies and accelerates the search for words that are located within a text field. If you employ MySQL for storing text (such as in an Internet discussion group), you can use full-text search to implement simply an efficient search function.

Replication: Replication allows the contents of a database to be copied (replicated) onto a number of computers. In practice, this is done for two reasons: to increase protection against system failure (so that if one computer goes down, another can be put into service) and to improve the speed of database queries.

Transactions: In the context of a database system, a transaction means the execution of several database operations as a block. The database system ensures that either all of the operations are correctly executed or none of them. This holds even if in the middle of a transaction there is a power failure, the computer crashes, or some other disaster occurs. Thus, for example, it cannot occur that a sum of money is withdrawn from account A but fails to be deposited in account B due to some type of system error.

Transactions also give programmers the possibility of interrupting a series of already executed commands (a sort of revocation). In many situations this leads to a considerable simplification of the programming process. In spite of popular opinion, MySQL has supported transactions for a long time. One should note here that MySQL can store tables in a variety of formats. The default table format is called MyISAM, and this format does not support transactions. But there are a number of additional formats that do support transactions. The most popular of these is InnoDB, which will be described extensively in this book.

Foreign key constraints: These are rules that ensure that there are no cross references in linked tables that lead to nowhere. MySQL supports foreign key constraints for InnoDB tables.

GIS functions: Since version 4.1, MySQL has supported the storing and processing of two-dimensional geographical data. Thus MySQL is well suited for GIS (geographic information systems) applications.

Programming languages: There are quite a number of APIs (application programming interfaces) and libraries for the development of MySQL applications. For client programming you can use, among others, the languages C, C++, Java, Perl, PHP, Python, and Tcl.

ODBC: MySQL supports the ODBC interface [Connector/ODBC](#). This allows MySQL to be addressed by all the usual programming languages that run under Microsoft Windows (Delphi, Visual Basic, etc.). The ODBC interface can also be implemented under Unix, though that is seldom necessary.

Windows programmers who have migrated to Microsoft's new .NET platform can, if they wish, use the ODBC provider or the .NET interface Connector/.NET.

Platform independence: It is not only client applications that run under a variety of operating systems; MySQL itself (that is, the server) can be executed under a number of operating systems. The most important are Apple Macintosh OS X, Linux, Microsoft Windows, and the countless Unix variants, such as AIX, BSDI, FreeBSD, HP-UX, OpenBSD, Net BSD, SGI Iris, and Sun Solaris.

Speed: MySQL is considered a very fast database program. This speed has been backed up by a large number of benchmark.

SQLite

SQLite is a [relational database management system](#) contained in a small [C](#) programming [library](#). In contrast to other database management systems, SQLite is not a separate process that is accessed from the client application, but an integral part of it.

SQLite is [ACID](#)-compliant and implements most of the [SQL](#) standard, using a dynamically and weakly typed SQL [syntax](#) that does not guarantee the [domain integrity](#).

SQLite is a popular choice as [embedded database](#) for local/client storage in [application software](#) such as [web browsers](#). It is arguably the most widely deployed [database engine](#), as it is used today by several widespread browsers, [operating systems](#), and [embedded systems](#), among others. SQLite has many [bindings](#) to programming languages.

The [source code](#) for SQLite is in the [public domain](#).

IDE for Database

MySQL workbench

MySQL Workbench is a visual database design tool that integrates SQL

development, administration, database design, creation and maintenance into a



single integrated development environment for the MySQL database system. It is the successor to DBDesigner 4 from fabFORCE.net, and replaces the previous package of software, MySQL GUI Tools Bundle. [MySQL Workbench](#) enables a DBA, developer, or data architect to visually design, generate, and manage all types of databases including Web, OLTP, and data warehouse databases. It includes everything a data modeler needs for creating complex ER models, and also delivers key features for performing difficult change management and documentation tasks that normally require much time and effort. MySQL Workbench is available on Windows, Linux and Mac OS.

benefits

- Simplifies database design and maintenance
- Automates time-consuming and error-prone tasks
- Enables data architects to visualize requirements, communicate with stakeholders, and resolve design issues before a major investment of time and resources is made
- Enables model-driven database design—the most efficient methodology for creating valid and well-performing databases—while providing the flexibility to respond to evolving business requirements
- Provides capabilities to forward-engineer physical database designs and reverse-engineer existing databases
- Allows you to import SQL scripts to build models and export models to DDL scripts that can be run at a later time
- Enables you to compare two live databases or a model and a live database, visually see the differences, and perform a synchronization between a model and a live database or vice versa
- Simplifies the documentation of database designs, providing a point-and-click process that delivers documentation in HTML or plain-text format

Tools

The three main tools of MySQL Workbench are:

- SQL Development
- Data Modelling
- Server Administration

Programming Language

Java

Java is a [general-purpose](#), [concurrent](#), [class-based](#), [object-oriented computer programming language](#) that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java applications are typically [compiled](#) to [bytecode](#) ([class file](#)) that can run on any [Java virtual machine](#) (JVM) regardless of [computer architecture](#). Java is, as of 2012, one of the most popular programming languages in use, particularly for client-server web applications, with a reported 10 million users.^{[10][11]} Java was originally developed by [James Gosling](#) at [Sun Microsystems](#) (which has since [merged into Oracle Corporation](#)) and released in 1995 as a core component of Sun Microsystems' [Java platform](#). The language derives much of its [syntax](#) from [C](#) and [C++](#), but it has fewer [low-level](#) facilities than either of them.

The original and [reference implementation](#) Java [compilers](#), virtual machines, and [class libraries](#) were developed by Sun from 1991 and first released in 1995. As of May 2007, in compliance with the specifications of the [Java Community Process](#), Sun relicensed most of its Java technologies under the [GNU General Public License](#). Others have also developed alternative implementations of these Sun technologies, such as the [GNU Compiler for Java](#) (bytecode compiler), [GNU](#)

[Classpath](#) (standard libraries), and [IcedTea](#)-Web (browser plug-in for applets).

PHP

PHP is a [server-side scripting](#) language designed for [web development](#) but also used as a [general-purpose programming language](#). PHP is now installed on more than 244 million [websites](#) and 2.1 million [web servers](#).^[2] Originally created by [Rasmus Lerdorf](#) in 1995, the [reference implementation](#) of PHP is now produced by The PHP Group.^[3] While PHP originally stood for *Personal Home Page*, it now stands for *PHP: Hypertext Preprocessor*, a [recursive acronym](#).

PHP code is [interpreted](#) by a web server with a PHP processor module, which generates the resulting web page: PHP commands can be embedded directly into an [HTML](#) source document rather than calling an external file to process data. It has also evolved to include a [command-line interface](#) capability and can be used in [standalone graphical applications](#).

PHP is [free software](#) released under the [PHP License](#), which is incompatible with the [GNU General Public License](#) (GPL) due to restrictions on the usage of the term *PHP*. PHP can be deployed on most web servers and also as a standalone [shell](#) on almost every [operating system](#) and [platform](#), free of charge.

Libraries

Twitter Bootstrap

Bootstrap is a [free](#) collection of tools for creating [websites](#) and [web applications](#). It contains [HTML](#) and [CSS](#)-based design templates for [typography](#), forms, buttons, navigation and other interface components, as well as optional [JavaScript](#) extensions.

It has been the most popular project in [GitHub](#) and has been used by [NASA](#) and [MSNBC](#) among others.

Flexi Auth

What is flexi auth?

flexi auth is a free open source user authentication/login library for use with the [CodeIgniter](#) 2.0+ framework.

The flexi auth library initially started out as a modified version of the popular [Ion Auth](#) library. As the original library was tweaked with feature after feature being added, the original code base had transformed into a new library all of its own.

For those that have used the Ion Auth library, the general structure of the library may be familiar, but to help anyone wanting to get a running start with using flexi auth, there is an comprehensive user guide and demo detailing covering every function within the library.

flexi auth is designed with modularised features that can be mixed and matched, turned on or off, and can be customised to suit your requirements.

Mahana Messaging Library

A small library to help jump start your internal messaging system, for the CodeIgniter framework

Features:

- *Create new thread*
- *Show thread*
- *Show messages*
- *Show all threads*

Other technologies

Representational State Transfer (REST) is an architectural style that abstracts the architectural elements within a distributed [hypermedia](#) system. REST ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements. REST has emerged as a predominant [web API](#) design model.

Dia for Diagram Drawing & Modeling

Dia is free and open source general-purpose diagramming software, developed as part of the GNOME project's office suite and was originally created by Alexander Larsson. Dia uses a controlled single document interface (CSDI) similar to GIMP and Sodipodi.

Dia has a modular design with several shape packages available for different needs: flowchart, network diagrams, circuit diagrams, and more. It does not restrict symbols and connectors from various categories from being placed together.

Dia is a gtk+ based diagram creation program released under the GPL license.

Dia is inspired by the commercial Windows program 'Visio', though more geared towards informal diagrams for casual use. It can be used to draw many different kinds of diagrams. It currently has special objects to help draw entity relationship diagrams, UML diagrams, flowcharts, network diagrams, and many other diagrams. It is also possible to add support for new shapes by writing simple XML files, using a subset of SVG to draw the shape.

It can load and save diagrams to a custom XML format (gzipped by default, to save space), can export diagrams to a number of formats, including EPS, SVG, XFIG, WMF and PNG, a

nd can print diagrams (including ones that span multiple pages).

Cacoo:: online drawing tool



Cacoo is a diagram creation tool that runs in your web browser. Multiple people can work together on the same diagram in real time. Diagrams can be published directly to websites, wikis, and blogs.

Creating Diagrams

- Elements can be dragged and drop to easily create diagrams.
- Elements can be linked together with connectors.
- Connectors automatically move when elements are repositioned.
- You can use a text box and put text anywhere you like.
- You can upload images from your PC and include them in Diagrams.
- You can take screenshots of your computer from within Cacoo.
- Smart styles can easily be applied to stencils.
- You can have multiple sheets in a diagram and use them as backgrounds or layers.
- When you move the objects on your canvas, they will be snapped at the objects or grids nearby and align automatically.
- Copying, pasting and other functionality of basic drawing software is also built in to Cacoo.
- All actions are stored so there are unlimited levels of undo.
- You can import an image from the other websites by indicating the URL.
- The imported image can be easily trimmed only using your mouse.
- According to your editing status, tips will be shown on the right bottom corner of the canvas.

Collaboration

- You can invite collaborators to work with you in Cacoo.
- Multiple people can edit a diagram in real time.
- There is a chat function in the editor so people can communicate while creating diagrams.
- People can leave comments about the diagrams.
- Each user can set their own user icon.
- When editing with multiple people, users icons appear on selected objects.
- Sharing diagrams become much smoother. Diagrams in the shared folders can be accessible and editable by people who you have shared the folder with.

Sharing Diagrams

- If you keep the diagram private then other users can't see it.
- If you make the diagram URL public, then anyone who knows the URL can see it.
- Publishing a diagram to a blog can be useful in various ways.

- You can place code into blogs to create a slideshow
- Published images always display the most recent version.
- Diagrams can be exported to SVG format (Plus Plan users only) and PNG format. (More formats will be available in the future.)
- Diagrams can be posted to Twitter/Facebook/GoogleBuzz
- Diagrams can be displayed in SVG format for printing. (Plus Plan users only. A few browsers are not supported.)

Managing Diagrams

- Diagrams can be placed into folders.
- Diagrams can be copied.
- Diagrams can be displayed as thumbnails or as a list.

Languages and Time Zones

- All pages and notification e-mails support English and Japanese
- Users can enter text from almost all languages.
- Dates are displayed relative to your local time zone.

Security

- Private diagrams can only be seen by users you select.
- URLs which you do not share can not be found by other users or search engines.
- All editing and management is protected by SSL.
- In order to access information about diagrams a Cacoo ID and password are required.
- User passwords are encrypted on Cacoo's server.

API

- You can access Cacoo using the API.
- The Cacoo API supports OAuth and an API Key.

By using the Cacoo API you are able to interact with Cacoo from other services and applications.

Authorization Methods

There are two ways to access the Cacoo API.

1. API Key

The API key allows you make requests to the Cacoo API. You can make an API key here.

API Key

Append your API key to requests to the API to return data from your account.(Parameter name "apiKey")

Example: <https://cacoo.com/api/v1/diagrams.json?apiKey=abcdefghijklmn>

2. OAuth

OAuth 1.0a is supported as an authorization method for Cacoo. You can register applications here.

You can get your Access Token from the following links.

applications

Access Token:https://cacoo.com/oauth/access_token

Authorize:<https://cacoo.com/oauth/authorize>

Request Token:https://cacoo.com/oauth/request_token

Version Control System: GitHub



GitHub is a web-based hosting service for software development projects that use the Git revision control system. GitHub offers both paid plans for private repositories, and free accounts for open source projects. As of May 2011, GitHub was the most popular open source code repository site. GitHub Inc. was founded in 2008 and is based in San Francisco, California.

Description

The site provides social networking functionality such as feeds, followers and the network graph to display how developers work on their versions of a repository.

GitHub also operates other services: a pastebin-style site called Gist that provides wikis for individual repositories and web pages that can be edited through a Git repository, a slide hosting service called Speaker Deck, and a web analytics platform called Gauges.

As of January 2010, GitHub is operated under the name GitHub, Inc.

The software that runs GitHub was written using Ruby on Rails and Erlang by GitHub, Inc. (previously known as Logical Awesome) developers Chris Wanstrath, PJ Hyett, and Tom Preston-Werner.

Limitations and constraints

According to the terms of service, if an account's bandwidth usage significantly exceeds the average of other GitHub customers, the account's file hosting service may be immediately disabled or throttled until bandwidth consumption is reduced. In addition, while there is no hard limit, the guideline for the maximum size of a repository is one gigabyte.

Glossary.

FT Fitness tracker

Apps Application

FB Facebook

SRS Software Requirement Specification

DFD Data Flow Diagram

ERD Entity Relationship Diagram

GUI Graphical User Interface

UI User Interface

DB Database

API Application Programming Interface

COCOMO Constructive Cost Model

SDK Sweater Development Kit

WPF Windows Presentation Framework

XAML Extensible application Markup Language

IDE Integrated Development Environment

HTML Hyper Text Markup Language

www World Wide Web

DBMS Database Management System

Sync Synchronization

cs C Sharp

ADT Android Development Tool

KLOC Estimated size of the software product expressed in Kilo

Tdev Estimated time to develop the software, expressed in months.

Effort Total effort required to develop the software product, expressed in person-month (PM).

PM Person-month

-----**Thank You**-----