

Intelligent Physical Fitness Considering Medical Conditions Project

Report submitted in fulfillment of the requirement for the degree of

Bachelor of Technology

In Information Technology

Under the supervision of

Ms. Ramanpreet Kaur

By

Ankit Chauhan 111414

to



Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

Certificate

This is to certify that project report entitled “Intelligent Physical Fitness Considering Physical Conditions Project”, submitted by Ankit Chauhan in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date:

Supervisor’s Name: Ms. Ramanpreet Kaur

Designation

Acknowledgement

I would like to express my gratitude to all those who gave us the possibility to complete this project. I want to thank the Department of CSE & IT in JUIT for giving us the permission to commence this project in the first instance, to do the necessary research work.

I am deeply indebted to my project guide **Ms. Ramanpreet Kaur**, whose help, stimulating suggestions and encouragement helped me in all the time of research on this project. I feel motivated and encouraged every time I get his encouragement. For her coherent guidance throughout the tenure of the project, I feel fortunate to be taught by her, who gave me her unwavering support.

We are also grateful to **Mr.Amit Singh(CSE Project lab)** for his practical help and guidance.

Table of Contents

| | |
|---|----|
| Certificate | 2 |
| Acknowledgement | 3 |
| Abstract | 7 |
| Problem | 8 |
| Motivation | 9 |
| 1. Introduction | 10 |
| 1.1 Need for the application | 10 |
| 1.2 Scope | 10 |
| 1.3 Android | 11 |
| 1.3.1 Android History | 11 |
| 2 Literature Review | 12 |
| 2.1 Android App with Artificial Intelligence | 12 |
| 2.2 Android apps for Physical Fitness | 12 |
| 2.3 Related Work | 12 |
| 2.3.1 Mobile Based Business Rule Engine | 12 |
| 2.3.2 Android Based Physical Exercise Advisor Application | 15 |
| 3. Tools and Techniques | 18 |
| 3.1 What is Rule Engine? | 18 |
| 3.1.1 Introduction and Background | 18 |
| 3.2 Why use a Rule Engine? | 21 |
| 3.2.1 Advantages of a Rule Engine | 22 |
| 3.2.2 When should you use a Rule Engine? | 23 |
| 3.2.3 When not to use a Rule Engine? | 25 |
| 4. Requirement Analysis | 27 |
| 4.1 Requirement Specification | 27 |
| 4.1.1 Functional Requirements | 27 |
| 4.1.2 Software Requirements | 27 |
| 4.1.3 Hardware Requirements | 27 |
| 5. System Architecture and Design | 28 |

| | |
|--|----|
| 5.1 System architecture | 28 |
| 5.2 System Design | 29 |
| 5.2.1 Use Case Diagram | 29 |
| 6. Graphical User Interface | 31 |
| 7. Android Framework Components | 44 |
| 7.1 AndroidManifest.xml file | 44 |
| 7.2 Activities | 47 |
| 7.3 Intents | 47 |
| 8. Development | 49 |
| 8.1 Reviewing the requirements | 49 |
| 8.2 Structure of an Android project | 49 |
| 8.2.1 Project folder Structure | 49 |
| 8.3 Designing of the layouts | 50 |
| 8.4 Implementing the layouts | 51 |
| 8.5 Maintaining Sessions | 52 |
| 8.6 Flowchart | 53 |
| 8.6 Designing and Implementing the Classes and Functions | 55 |
| 8.6.1 Rule Engine Implementation | 55 |
| 8.6.2 Rule Implementation | 57 |
| 8.6.2.1 Color Codes | 57 |
| 9. Testing | 59 |
| 9.1 Unit Testing | 59 |
| 9.2 Compatibility Testing | 61 |
| 10. Conclusions | 63 |
| 10.1 Future Work | 63 |
| 11. Reference | 64 |

LIST OF FIGURES

| | |
|--|----|
| 2.3.1 High Level Overview of Rule Engine | 13 |
| 3.1.1. Forward Chaining | 21 |
| 5.1 System Architecture | 27 |
| 5.2.1 Use Case Diagram | 29 |
| 6.1 SplashScreen Activity | 30 |
| 6.2 AskManFemale Activity | 31 |
| 6.3AskWeightHeight Activity | 31 |
| 6.4AskDiabetes Activity | 32 |
| 6.5Result Activity | 32 |
| 6.6BodyBuilding Activity | 33 |
| 6.7Normal Activity | 33 |
| 6.8LooseWeight Activity | 34 |
| 6.9 LooseWeight Activity | 35 |
| 6.10 LooseWeight Activity | 35 |
| 6.11Home Activity | 36 |
| 6.12DiabetesSchedule Activity | 36 |
| 6.13ResistanceExerciseSchedule Activity | 37 |
| 6.14Resistance Activity | 37 |
| 6.15AerobicSchedule Activity | 38 |
| 6.16AerobicSchedule Activity | 38 |
| 6.17 AerobicSchedule Activity | 39 |
| 6.18TimeTable Activity | 39 |
| 6.19 TimeTable Activity | 40 |
| 6.20 TimeTable Activity | 40 |
| 6.21 TimeTable Activity | 41 |
| 6.22 MyCalendar Activity | 41 |
| 6.23 CalendarSchedule Activity | 42 |
| 8.6 Flowchart | 52 |
| 8.6.2.1Different type of nodes | 56 |

| | | |
|---------|--|----|
| 8.6.2.2 | Rete tree of the rules applied at AskWeightHeight activity | 56 |
| 8.6.2.3 | Rete tree of the rules applied at AskDiabetes activity | 57 |
| 9.2.1 | Portrait mode | 60 |
| 9.2.2 | Landscape mode | 60 |

List of Tables

| | |
|---------------------|----|
| 3.1.1 Lines of code | 21 |
| 8.6.1 Lines of code | 56 |
| 8.6.1 Lines of code | 56 |
| 8.6.1 BMI Range | 57 |
| 9.1 Unit testing | 61 |

Abstract

Android is a mobile operating system (OS) based on the Linux kernel and currently developed by Google. With a user interface based on direct manipulation. In Android there are Android applications which increase the functioning of a smart phone which is an Android phone. Android is popular with technology companies which require a ready-made, low-cost and customizable operating system for high-tech devices. Android's open nature has encouraged a large community of developers and enthusiasts to use the open-source code as a foundation for community-driven projects, which add new features for advanced users or bring Android to devices which were officially released running other operating systems.

The Android application that I am designing is totally dedicated to fitness. The application is not just a set of pages describing the techniques and routines of fitness just for an ideal person with no physical ailments or any other disease. The application will include physical routines for even those suffering from some general medical problems (diabetes).

There will be two modules, one for fitness and other for diabetic users. The user will be asked to enter some data based upon which the BMI(Body Mass Index) of the user will be calculated and will be further suggested for physical routines.

To accomplish this I'll use is JBoss Drools Rule Engine.

Problem Statement

There are many android applications regarding fitness, diet, body building but there is not a single application with a combination of all these. Moreover the existing applications doesn't have artificial intelligence which will be a part of my application by implementing a rule engine (JBoss Drools Rule Engine). So we need an application which guides the users regarding fitness for normal users as well as the users with general medical problems. The application would provide data to the user to stay fit through a proper physical routine. Users who are having some general medical problems can also take advantage from this application. The general medical problem include diabetes.

Motivation

Android applications increase the functioning of the smart phone by using the features of the phone itself. Android is the most widely used mobile OS and, as of 2014, the highest selling OS overall. Android devices sell more than Windows, iOS, and Mac OS X devices combined, with sales in 2012, 2013 and 2014 close to the installed base of all PCs. As of July 2013 the Google Play store has had over 1 million Android apps published, and over 50 billion apps downloaded.

My vision is to implement an application which is not present in the google app store. This application is much different from what is present there. There are application with artificial intelligence and even applications regarding some general medical ailments but no single application which would contain a combination of both. The things that make the application different is the rule engine and fitness techniques available even for users with general medical problems. My target is the Android market which needs my application as it is not present there.

1. INTRODUCTION

This project is about the developing a physical fitness android application. Nowadays, the gadgets are rolling the world. Many people cannot imagine even one day without their favorite mobile device. We use them for everything: find information, stay connected with our friends and families, find the way around, decide what to do, and many other things. But very often we come to the point when we would like to have an application for particular situation or for certain need, but there is no such one.

Developing of an application usually takes lots of time and needs professional knowledge of software. And as there is no such android application on the Google play store which can be used as a guide to stay in desirable physical condition not only for normal users but also for users with general medical problems.

1.1 Need for the application

I have been looking for android applications dedicated to fitness on the google play store. The number is very huge and moreover effective. One thing that was missing an android application which could act as a guide not only to normal users but also for users with medical problems. This application would provide physical routines.

Another thing which is new in this application is the Rule Engine. This will make the application artificially intelligent as it will act according to your inputs. The Engine used is JBoss Drools Engine.

1.2 Scope

Due to the time limits of this project we must narrow the scope. The scope of this project is to develop an application will cover one medical ailments which is diabetes. The application would contain the two modules fitness and diabetes. All the data that will be available in the application will be from a trusted source and after consulting the concerned doctor.

Out of the scope is the further loading of features in the application and increasing the number medical problems considered.

1.3 Android

Android is one of an Open source platforms. It is created by Google and owned by Open Handset Alliance. It is designed with goal “accelerate innovation in mobile” As such android has taken over a field of mobile innovation. It is definitely free and open platform that differs hardware from software that runs on it. It results for much more devices be running the same application. Also it gives possibility of friendlier environment for developers and consumers. Android it is complete software package for a mobile device. Since the beginning android team offers the developing kit (tool and frameworks) for creating mobile applications quick and easy as possible. In some cases you do not specially need an android phone but you are very welcome to have one. It can work right out of the box, but of course users can customize it for their particular needs. For manufactures it is ready and free solution for their devices. Except specific drivers android community provides everything else to create their devices.

1.3.1 Android history

The actual history of Android starts when Google has had purchased and Android inc. in 2005. But the development did not start immediately. The actual progress on android platform starts when 2007 Open Handsets Alliance has announced the Android as Open Source platform and year later the Android SDK 1.0. In the same 2008 the G1 phone was produced by HTC and was retailed within the T-Mobile carrier. In the next two years came out 4 versions of Android. In 2010 there were at least 60 devices running android and it becomes second after Blackberry the best spread mobile platform.

2. LITERATURE REVIEW

2.1 Android App with artificial intelligence

- AIVC-You can have a conversation with Alice, give her instructions or ask for general terms of information.
- Skyvi- Can text/call friends, find places and tells jokes. Update Facebook and Twitter.
- Evi-Voice or text input, Shopping, news, dining and more based on your location.
- Rule Engine- Rules Engine is a complete rules engine to customize your device behavior automatically.

2.2 Android apps for physical fitness

- Workout Trainer- provides different type of workout schedules.
- Fitness Buddy- different type of exercises and fitness tips.
- Daily Workouts- Necessary workout to stay fit and challenges.
- Total Fitness- Gives workout routines with proper diet plans.

2.3 Research papers:

2.3.1 Mobile Based Business Rule Engine

Abstract- Business Organization's success depends on agility of organization to cope with the dynamic business environment. Involving a software team frequently for changing the business policies is not a good idea. Such needs can be catered using rule engine. Business rules can be written separately from the application code using rule engine which makes it easy for a business user to change the rules as and when needed without the help of a programmer. Traditionally, rule engines are designed to work on high-end processors, having them on a low-end processor is a big challenge. In this era

we are dealing with palmtops and iPhones, which are going to replace the bulky desktops even for computational purposes. Objective of this paper is to discuss rule engine on mobile platform, the challenges and solutions while implementing rule engine on mobile platform, particularly JRuleEngine on Android as a mobile platform.

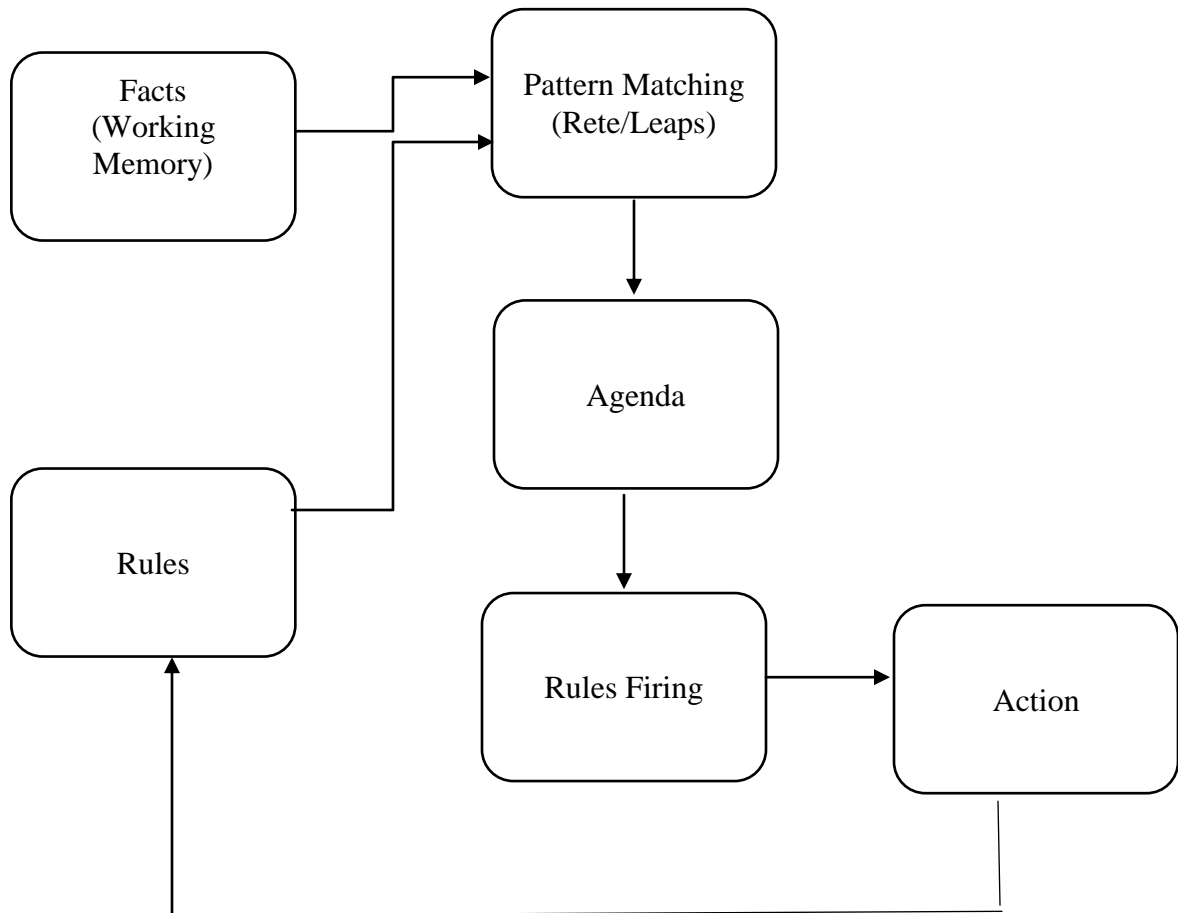


Figure 2.3.1: High Level Overview of Rule Engine

Rule engine basically deals with decision making. Rules are nothing but conditions and actions. They are stored in production memory in a separate file. Inference engine is brain of rule engine. It is a pattern matcher. It matches facts and rules. Matching is done by various matching algorithms like RETE, Leaps.

Facts are stored in working memory. Working memory contains the objects i.e. facts which we need while making a decision. It is possible to add, remove, and modify the facts.

For example, rule is:

<Condition> if age of citizen < 18

<Action> Not eligible for voting

Here fact will be object of 'citizen'. Inference engine matches this fact against given rule and finds out if the condition is satisfied. If system contains large number of rules then there might be a problem when there are multiple rules which are true. Agenda is an important part which determines the order in which the rules should be executed. This is called conflict resolution.

Forward Chaining: It is data driven approach. We arrive at a conclusion using data.

E.g. JBoss Rules

Advantages:

- 1) Speed and Scalability: Being lightweight, it consumes less memory thereby giving high speed and scalability.
- 2) Understandable Rules: Rules can be expressed in easy format. Business user can easily modify them without need of IT intervention.

Disadvantage:

- 1) It does not provide web based tools i.e. a business rules management system for advanced rule authoring, version control, and management.
- 2) Does not support multiple facts of same type i.e. if two or more objects of same type are added to working memory, it only matches the last object added against the rules.
- 3) While defining rules in xml file, the RHS part of 'then' cannot be an object.
 - Developing an Artificial Intelligence Engine- The engine should be such reactive, context specific, flexible, realistic and easy to develop.
 - Android based Physical Exercise Advisor Application- This paper is based on an application which takes inputs for calculating BMI to generate further results.

2.3.2 Android based Physical Exercise Advisor Application

2.3.2.1 Abstract

This paper is about the exercise application using Android as a platform. This is an application of physical exercise that will generate output that will help users to determine the BMI (Body Mass Index) level, the duration of time to exercise, the time to exercise and types of exercise. The user will input the weight, height, their body condition and working hours before the system generate the results. Focuses on developing an Android-based mobile phone prototype to calculate and determine the duration of physical exercise, time to exercise and the types of exercise needed daily by the user.

This is an Android application (prototype) that can calculate and determine the duration of physical exercise time to exercise and the types of exercise needed daily by the user. The factors that will be used in the calculation of the algorithm are BMI (Body Mass Index) of the user, user's body condition and working hours. Rule-based algorithm was used in this application to suggest their exercise schedule. Further and long term study is necessary to see the effectiveness, perceived usefulness and usability of this system. The Artificial intelligent (AI) is used in this system to help users make decisions. This prototype application is able of generating three main outputs which are:

1. Types of exercise
2. Suitable time to exercise and
3. Duration of exercise

2.3.2.2 Requirement

Then, the requirement of this is divided into;software and hardware. For software used for this systemare, ADK, Eclipse and Adobe. For the hardware, theremust have a laptop or PC for the development of theandroid apps. Besides that, the android Smartphone is amust.

2.3.2.3 Analysis

Analysis is the most important to the project development. To make this phase to the target, the research must be done to identify the problem statement and the objective of the project. The particular such as the target user, which in this project is the elderly need to be captured and documented.

2.3.2.4 Design

According to the purpose of system design is to create a physical model of the system that satisfies the logical design requirements that were defined during the system analysis phase. The system must be designed according to the requirements written on the project requirement and analysis.

2.3.2.5 Development

The system is developed by code the program using either Corona SDK Eclipse software or other Software Development Kit for Android application development.

2.3.2.6 Validation

The project's outcome is tested against the requirement and test for error. In this phase, the application should go through testing before deliver it to the end users. The testing of the application is divided into two parts. First testing is to check the error or bug that is in the codes then debug it. Second testing is to ensure the application fulfill the requirements of the users.

2.3.2.7 Deployment

Release the system. For this project, system deployments do not really happen. The deployment only for the presentation purposes.

2.3.2.7 Conclusion

The outcome of this project is a functioning an android based physical exercise system that would recommend each BMI result with different exercise plans, helps user to choose the right type of exercise by their body condition, recommend the right time for users to make the exercise by their free time estimation. It also helps the user to set the duration

time of exercise for the user by their bodycondition and free time. In addition, it can create ExercisePlanner to help user keep track and measure their activitiestowards achieving goals. The system needs to run on androidOS mobile device without any crash.

3. TOOLS AND TECHNIQUE

3. JBoss Drools Rule Engine

3.1. What is a Rule Engine?

3.1.1. Introduction and Background

Artificial Intelligence (A.I.) is a very broad research area that focuses on "Making computers think like people" and includes disciplines such as Neural Networks, Genetic Algorithms, Decision Trees, Frame Systems and Expert Systems. Knowledge representation is the area of A.I. concerned with how knowledge is represented and manipulated. Expert Systems use Knowledge representation to facilitate the codification of knowledge into a knowledge base which can be used for reasoning, i.e., we can process data with this knowledge base to infer conclusions. Expert Systems are also known as Knowledge-based Systems and Knowledge-based Expert Systems and are considered to be "applied artificial intelligence". The process of developing with an Expert System is Knowledge Engineering. EMYCIN was one of the first "shells" for an Expert System, which was created from the MYCIN medical diagnosis Expert System. Whereas early Expert Systems had their logic hard-coded, "shells" separated the logic from the system, providing an easy to use environment for user input. Drools is a Rule Engine that uses the rule-based approach to implement an Expert System and is more correctly classified as a Production Rule System.

The term "Production Rule" originates from formal grammars where it is described as "an abstract structure that describes a formal language precisely, i.e., a set of rules that mathematically delineates a (usually infinite) set of finite-length strings over a (usually finite) alphabet".

Business Rule Management Systems build additional value on top of a general purpose Rule Engine by providing business user focused systems for rule creation, management, deployment, collaboration, analysis and end user tools. Further adding to this value is the fast evolving and popular methodology "Business Rules Approach", which is helping to formalize the role of Rule Engines in the enterprise.

The term Rule Engine is quite ambiguous in that it can be any system that uses rules, in any form that can be applied to data to produce outcomes. This includes simple systems like form validation and dynamic expression engines. The book "How to Build a Business Rules Engine (2004)" by Malcolm Chisholm exemplifies this ambiguity. The book is actually about how to build and alter a database schema to hold validation rules. The book then shows how to generate VB code from those validation rules to validate data entry. This, while a very valid and useful topic for some, caused quite a surprise to this author, unaware at the time in the subtleties of Rules Engines' differences, who was hoping to find some hidden secrets to help improve the Drools engine. JBoss jBPM uses expressions and delegates in its Decision nodes which control the transitions in a Workflow. At each node it evaluates there is a rule set that dictates the transition to undertake, and so this is also a Rule Engine. While a Production Rule System is a kind of Rule Engine and also an Expert System, the validation and expression evaluation Rule Engines mentioned previously are not Expert Systems.

A Production Rule System is Turing complete, with a focus on knowledge representation to express propositional and first order logic in a concise, non-ambiguous and declarative manner. The brain of a Production Rules System is an Inference Engine that is able to scale to a large number of rules and facts. The Inference Engine matches facts and data against Production Rules - also called Productions or just Rules - to infer conclusions which result in actions. A Production Rule is a two-part structure using First Order Logic for reasoning over knowledge representation.

```
when
```

```
    <conditions>
```

```
then
```

```
    <actions>;
```

Table 3.1.1 Lines of code

The process of matching the new or existing facts against Production Rules is called Pattern Matching, which is performed by the Inference Engine. There are a number of algorithms used for Pattern Matching by Inference Engines including:

- Linear
- Rete
- Treat
- Leaps

Drools implements and extends the Rete algorithm; Leaps used to be provided but was retired as it became unmaintained. The Drools Rete implementation is called ReteOO, signifying that Drools has an enhanced and optimized implementation of the Rete algorithm for object oriented systems. Other Rete based engines also have marketing terms for their proprietary enhancements to Rete, like RetePlus and Rete III. The most common enhancements are covered in "Production Matching for Large Learning Systems (Rete/UL)" (1995) by Robert B. Doorenbos.

The Rules are stored in the Production Memory and the facts that the Inference Engine matches against are kept in the Working Memory. Facts are asserted into the Working Memory where they may then be modified or retracted. A system with a large number of rules and facts may result in many rules being true for the same fact assertion; these rules are said to be in conflict. The Agenda manages the execution order of these conflicting rules using a Conflict Resolution strategy.

There are two methods of execution for a rule system: Forward Chaining and Backward Chaining; systems that implement both are called Hybrid Chaining Systems. As of Drools 5.2 Drools provides seamless hybrid chaining, both forward and backwards. Understanding these two modes of operation is the key to understanding why a Production Rule System is different and how to get the best from it. Forward chaining is "data-driven" and thus reactionary, with facts being asserted into working memory, which results in one or more rules being concurrently true and scheduled for execution by the Agenda. In short, we start with a fact, it propagates and we end in a conclusion.

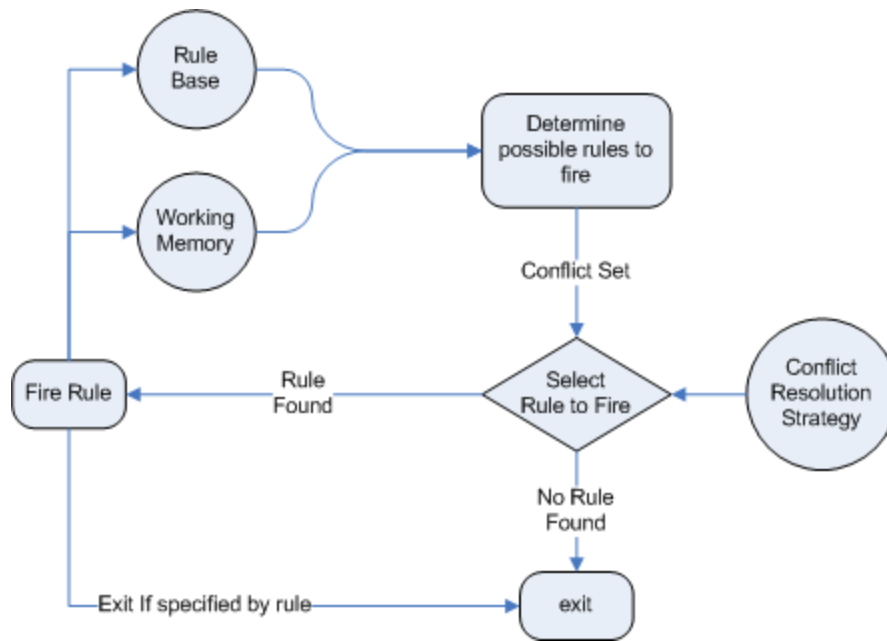


Figure 3.1.1. Forward Chaining

Backward chaining is "goal-driven", meaning that we start with a conclusion which the engine tries to satisfy. If it can't it then searches for conclusions that it can satisfy; these are known as subgoals, that will help satisfy some unknown part of the current goal. It continues this process until either the initial conclusion is proven or there are no more subgoals. Prolog is an example of a Backward Chaining engine. Drools can also do backward chaining, which we refer to as derivation queries.

3.2. Why use a Rule Engine?

Some frequently asked questions:

1. When should you use a rule engine?
2. What advantage does a rule engine have over hand coded "if...then" approaches?
3. Why should you use a rule engine instead of a scripting framework, like BeanShell?

We will attempt to address these questions below.

3.2.1. Advantages of a Rule Engine

- Declarative Programming

Rule engines allow you to say "What to do", not "How to do it".

The key advantage of this point is that using rules can make it easy to express solutions to difficult problems and consequently have those solutions verified. Rules are much easier to read than code.

Rule systems are capable of solving very, very hard problems, providing an explanation of how the solution was arrived at and why each "decision" along the way was made (not so easy with other of AI systems like neural networks or the human brain - I have no idea why I scratched the side of the car).

- Logic and Data Separation

Your data is in your domain objects, the logic is in the rules. This is fundamentally breaking the OO coupling of data and logic, which can be an advantage or a disadvantage depending on your point of view. The upshot is that the logic can be much easier to maintain as there are changes in the future, as the logic is all laid out in rules. This can be especially true if the logic is cross-domain or multi-domain logic. Instead of the logic being spread across many domain objects or controllers, it can all be organized in one or more very distinct rules files.

- Speed and Scalability

The Rete algorithm the Leaps algorithm, and their descendants such as Drools' ReteOO, provide very efficient ways of matching rule patterns to your domain object data. These are especially efficient when you have datasets that change in small portions as the rule engine can remember past matches. These algorithms are battle proven.

- Centralization of Knowledge

By using rules, you create a repository of knowledge (a knowledge base) which is executable. This means it's a single point of truth, for business policy, for instance. Ideally rules are so readable that they can also serve as documentation.

- Tool Integration

Tools such as Eclipse (and in future, Web based user interfaces) provide ways to edit and manage rules and get immediate feedback, validation and content assistance. Auditing and debugging tools are also available.

- Explanation Facility

Rule systems effectively provide an "explanation facility" by being able to log the decisions made by the rule engine along with why the decisions were made.

- Understandable Rules

By creating object models and, optionally, Domain Specific Languages that model your problem domain you can set yourself up to write rules that are very close to natural language. They lend themselves to logic that is understandable to, possibly nontechnical, domain experts as they are expressed in their language, with all the program plumbing, the technical know-how being hidden away in the usual code.

3.2.2. When should you use a Rule Engine?

The shortest answer to this is "when there is no satisfactory traditional programming approach to solve the problem.". Given that short answer, some more explanation is required. The reason why there is no "traditional" approach is possibly one of the following:

- The problem is just too fiddle for traditional code.

The problem may not be complex, but you can't see a non-fragile way of building a solution for it.

- The problem is beyond any obvious algorithmic solution.

It is a complex problem to solve, there are no obvious traditional solutions, or basically the problem isn't fully understood.

- The logic changes often

The logic itself may even be simple but the rules change quite often. In many organizations software releases are few and far between and pluggable rules can help provide the "agility" that is needed and expected in a reasonably safe way.

- Domain experts (or business analysts) are readily available, but are nontechnical.

Domain experts often possess a wealth of knowledge about business rules and processes. They typically are nontechnical, but can be very logical. Rules can allow them to express the logic in their own terms. Of course, they still have to think critically and be capable of logical thinking. Many people in nontechnical positions do not have training in formal logic, so be careful and work with them, as by codifying business knowledge in rules, you will often expose holes in the way the business rules and processes are currently understood.

If rules are a new technology for your project teams, the overhead in getting going must be factored in. It is not a trivial technology, but this document tries to make it easier to understand.

Typically in a modern OO application you would use a rule engine to contain key parts of your business logic, *especially the really messy parts*. This is an inversion of the OO concept of encapsulating all the logic inside your objects. This is not to say that you throw out OO practices, on the contrary in any real world application, business logic is just one part of the application. If you ever notice lots of conditional statements such as "if" and "switch", an overabundance of strategy patterns and other messy logic in your code that just doesn't feel right: that would be a place for rules. If there is some such logic and you keep coming back to fix it, either because you got it wrong, or the logic or your understanding changes: think about using rules. If you are faced with tough problems for which there are no algorithms or patterns: consider using rules.

Rules could be used embedded in your application or perhaps as a service. Often a rule engine works best as "stateful" component, being an integral part of an application. However, there have been successful cases of creating reusable rule services which are stateless.

For your organization it is important to decide about the process you will use for updating rules in systems that are in production. The options are many, but different organizations have different requirements. Frequently, rules maintenance is out of the control of the application vendors or project developers.

3.2.3. When not to use a Rule Engine?

As rule engines are dynamic (dynamic in the sense that the rules can be stored and managed and updated as data), they are often looked at as a solution to the problem of deploying software. (Most IT departments seem to exist for the purpose of preventing software being rolled out.) If this is the reason you wish to use a rule engine, be aware that rule engines work best when you are able to write declarative rules. As an alternative, you can consider data-driven designs (lookup tables), or script processing engines where the scripts are managed in a database and are able to be updated on the fly.

4. Requirement Analysis

The project involved analyzing the design of few applications so as to make the application more user friendly. To do so, it was really important to keep the navigations from one screen to the other well-ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the android version had to be chosen so that it is compatible with most of the Android devices. Hence Android 4.0.0 Icecream Sandwich version was chosen.

4.1 Requirement Specification

4.1.1 Functional Requirements

- The application uses the input to move through activities and help the user to know how to keep his body fit.
- Calculates the BMI value of the user.
- Routine check
- Time table
- Synchronized calendar
- Sessions maintain
- Notification

4.1.2 Software Requirements

For developing the application the following are the Software Requirements: Operating System: Windows 7,8 Language: Android SDK, Tools: Eclipse IDE, Android Plug-in for Eclipse Technologies used: Java, XML. Debugger: Android Dalvik Debug Monitor service

For running the application the following are the Software Requirements: Operating System: Android higher than 2.3.3

4.1.3 Hardware Requirements

For developing the application the following are the Hardware Requirements: Processor: P IV or higher RAM: 256 MB Space on disk: minimum 512MB for running the application: Device: Android version 2.3 and higher Minimum space to execute: 5.0MB.

5. System Architecture and Design

5.1 System architecture

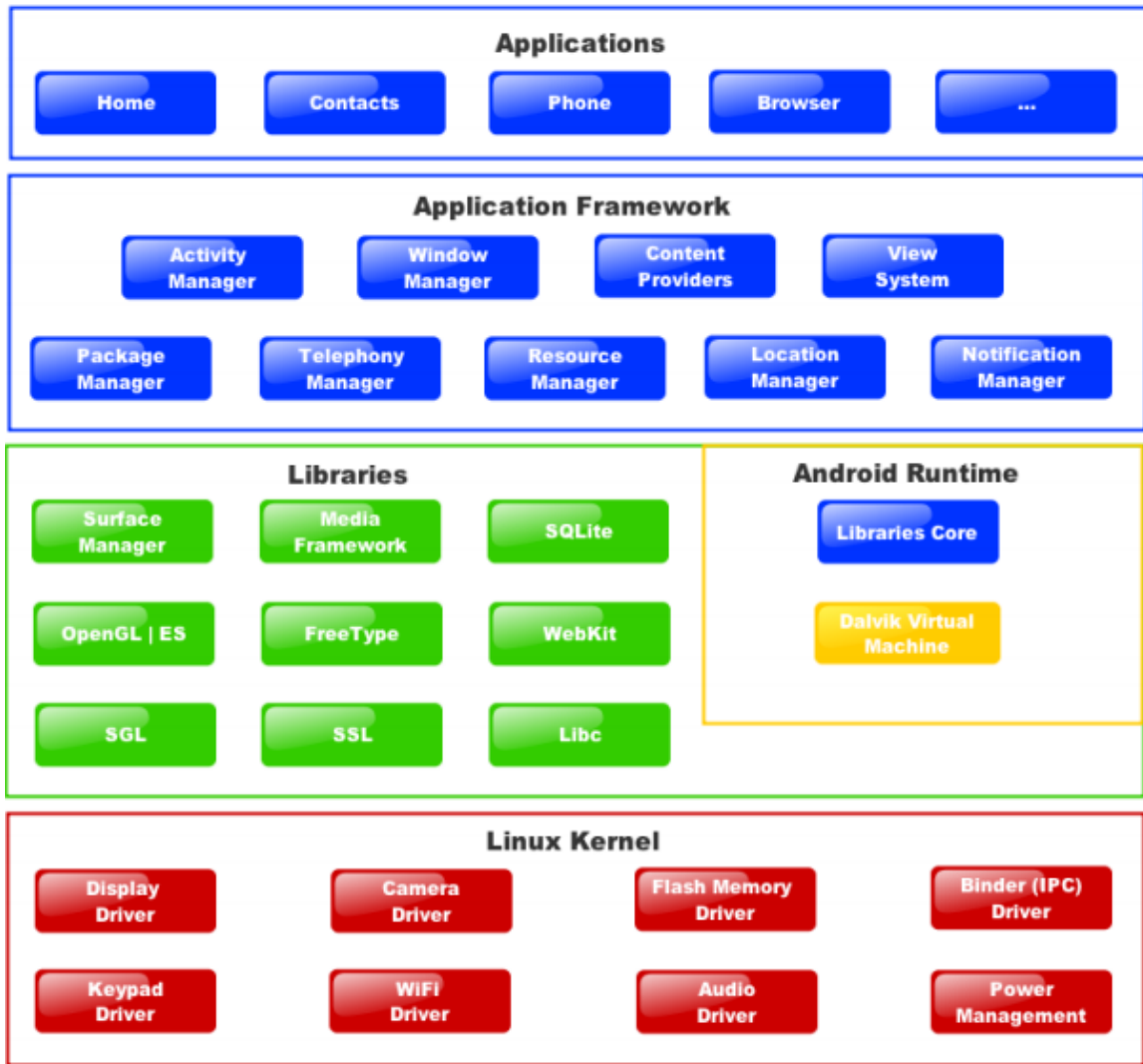


Figure 1 Android Architecture showing the major components of Android OS.

The above figure shows the diagram of Android architecture. Android OS is basically a software stack with various layers with each layer offering different services to the layer above it. The layers include a Linux Kernel which is responsible for interaction with the hardware. Libraries are written in C or C++ and are specific to the hardware of the system.

Android Runtime consists of Dalvik Virtual machine and Core Java libraries responsible for running the applications. It is the Application Framework that our application interacts with.

5.2 System Design

Once the features to implement were decided and the architecture was drafted, a system design was needed to implement the application. This section depicts the design using the Unified Modeling Language. The following were the diagrams that are used to implement and test the application.

- Use Case Diagram

5.2.1 Use Case Diagram

A use case diagram captures the actors and the role they perform in a system.

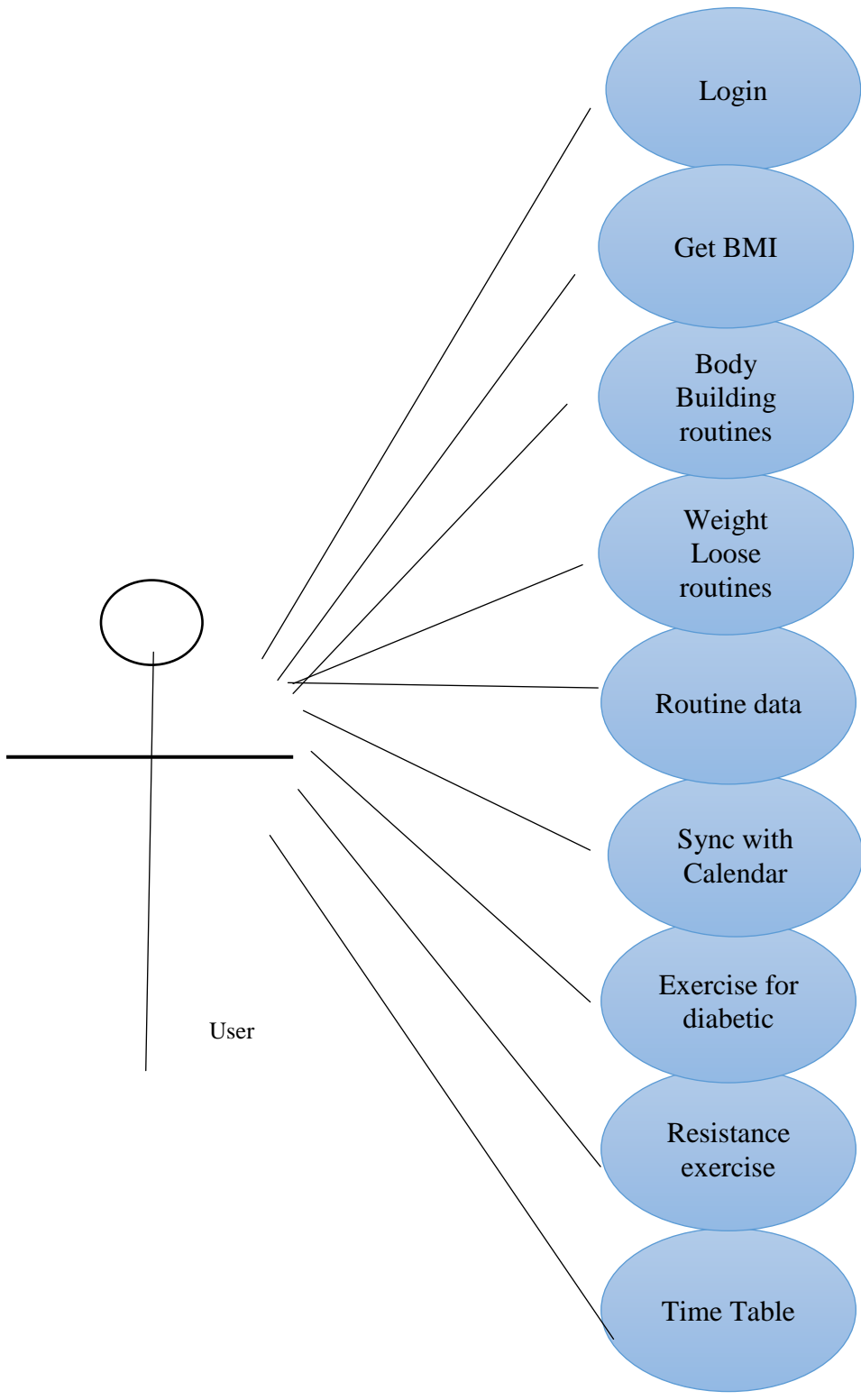


Figure 5.2.1 Use Case Diagram

6. Graphical User Interface

The user interface is kept simple and understandable. The user need not take any additional effort to understand the functionality and navigation in the application. The colors are chosen in such a way that user can easily understand where the input has to be given. Hints are given to help the user in giving the correct input. The following are the main screens and features in this application

SplashScreen Activity

This class is the start page of the application where the user wait for 4 seconds which is the time required by the application to load data and process consistently.



Figure 6.1

AskManFemale Activity

This class asks for the gender of the user.

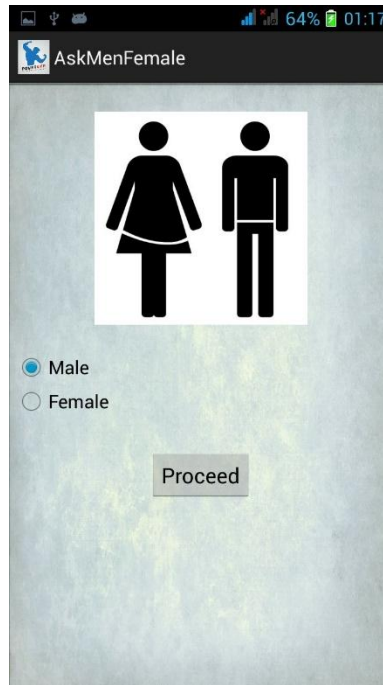


Figure 6.2

AskWeightHeight Activity

This class asks for weight and height of the user.

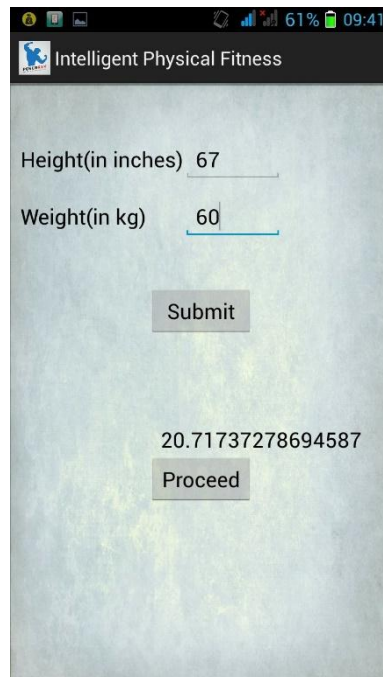


Figure 6.3

AskDiabetes Activity

It asks whether the user is diabetic and the age of the user.

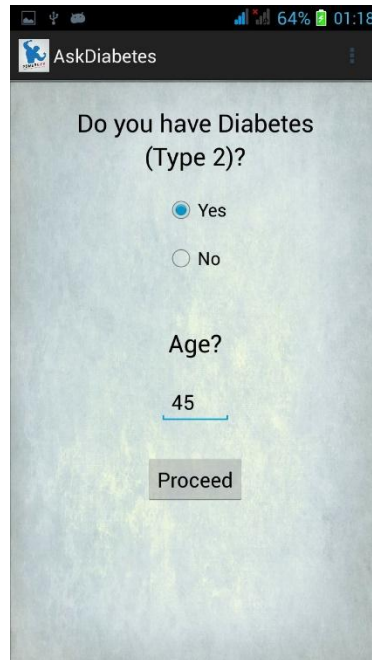


Figure 6.4

Result Activity

This class displays the BMI value determined from height and weight of the user and exercise routine required.

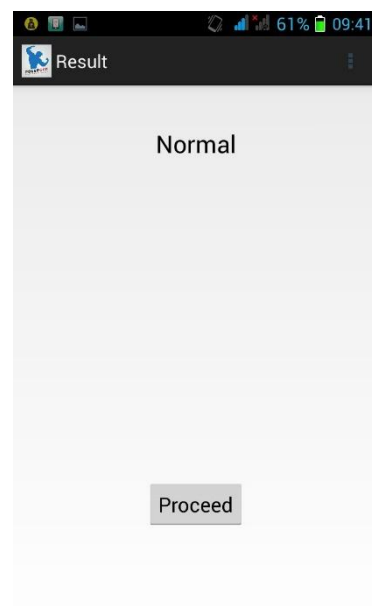


Figure 6.5

BodyBuilding Activity

It displays day wise routine for the user with different buttons of different days for Body Building.

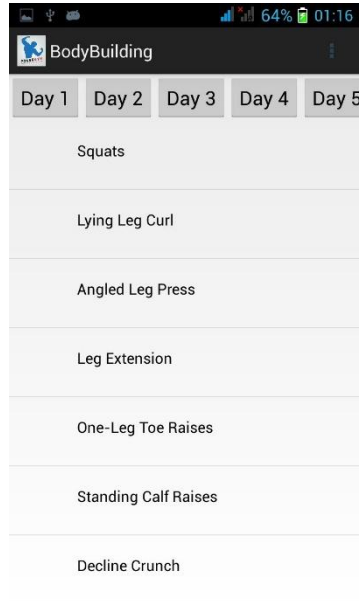


Figure 6.6

Normal Activity

It displays day wise routine for the user with different buttons for different days for Ripping.

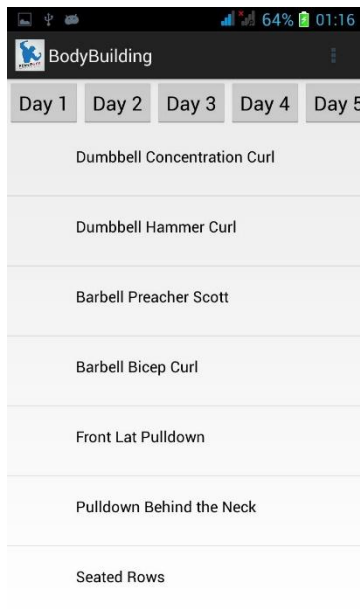


Figure 6.7

LooseWeight Activity

It displays day wise routine for the user with different buttons for different days for Loosing Weight.

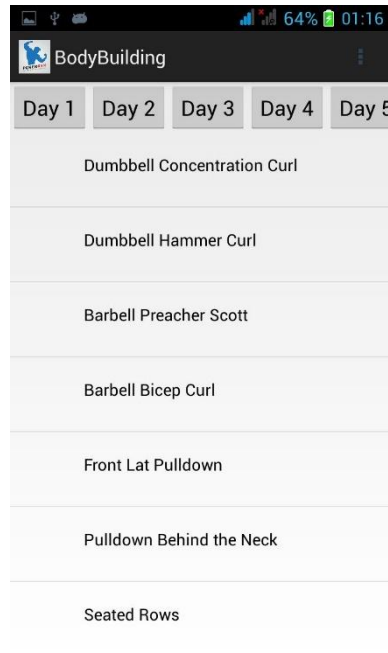


Figure 6.8

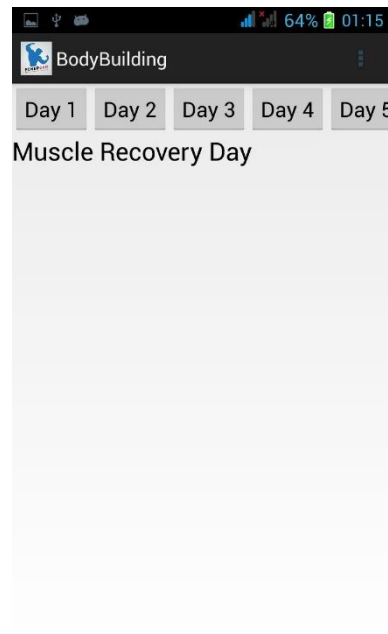


Figure 6.9

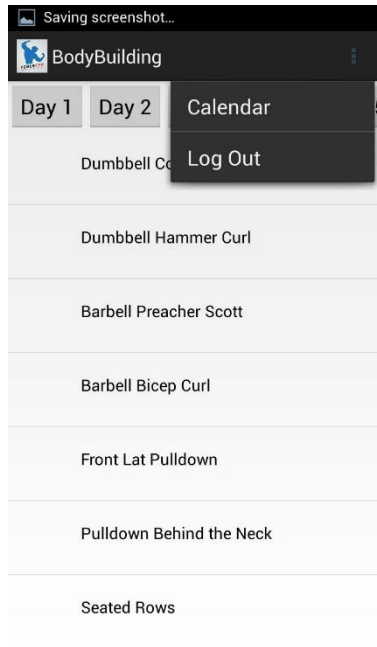


Figure 6.10

Home Activity

It displays image buttons to move to body building or loose weight or normal activity.



Figure 6.11

DiabetesSchedule Activity

It displays the different category of routines for diabetic users.

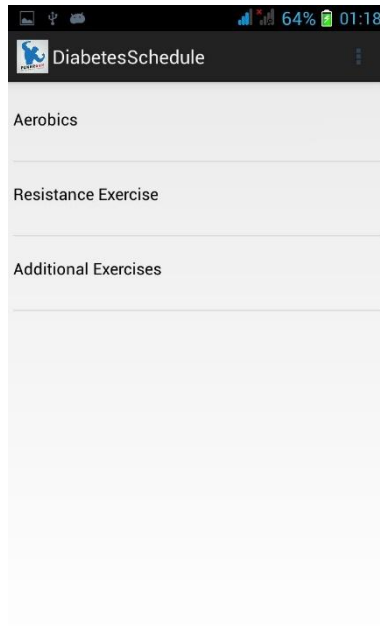


Figure 6.12

ResistanceExerciseSchedule Activity

It displays the list of exercises.

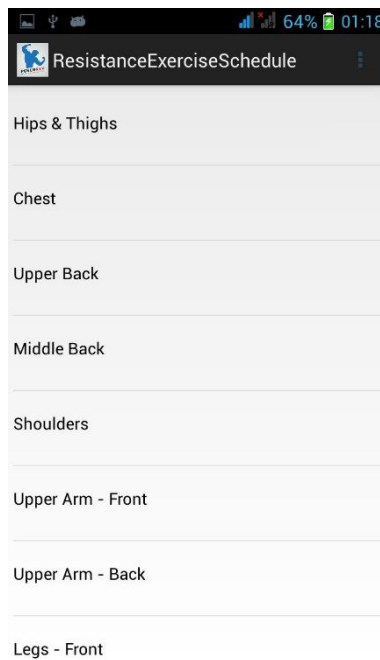


Figure 6.13

Resistance Activity

It displays the list of exercises.

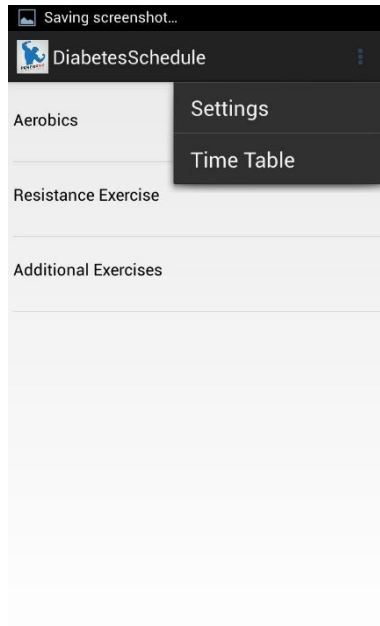


Figure 6.14

AerobicSchedule Activity

It displays the list of exercises.

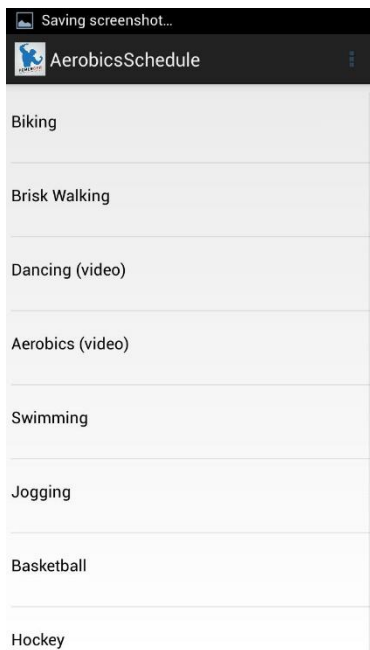
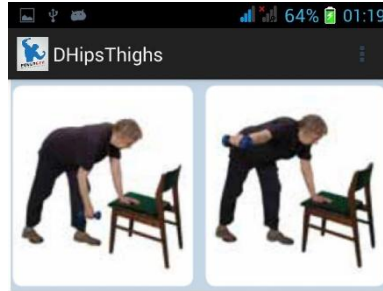


Figure 6.15



Start- Sit at the edge of the chair. Tighten abdominals and keep chest up. Weight is held at shoulder level with palms forward or facing your ears.

Finish- Extend one arm overhead until directly over the shoulder. Try not to lean to one side. Pause. Slowly lower to starting position. Alternate arms.

Figure 6.16

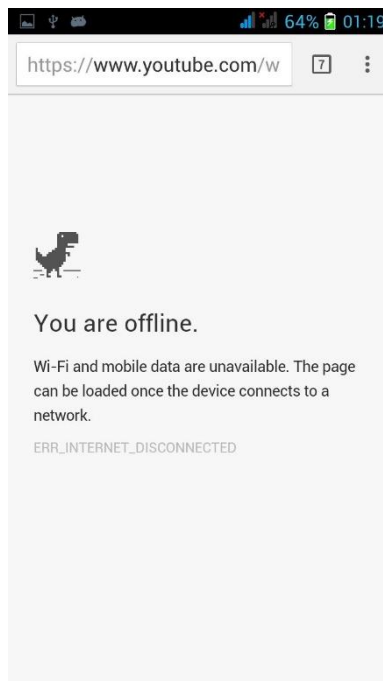
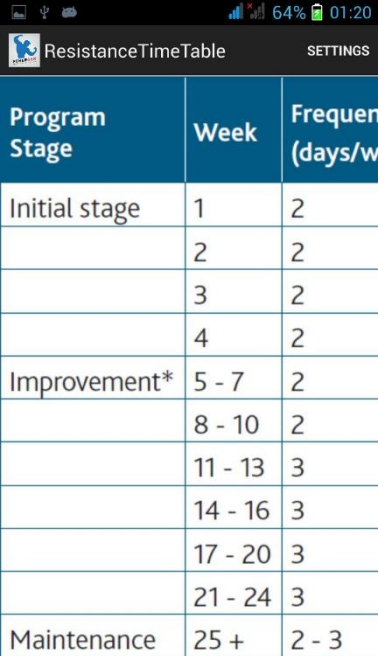


Figure 6.17

TimeTable Activity

It displays a time table for the diabetic patients.

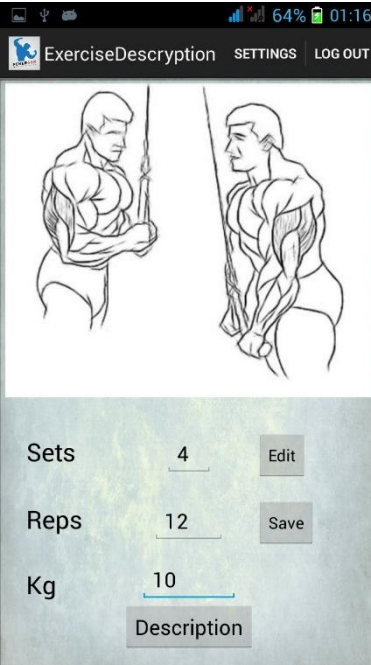


| Program Stage | Week | Frequen (days/w |
|---------------|---------|-----------------|
| Initial stage | 1 | 2 |
| | 2 | 2 |
| | 3 | 2 |
| | 4 | 2 |
| Improvement* | 5 - 7 | 2 |
| | 8 - 10 | 2 |
| | 11 - 13 | 3 |
| | 14 - 16 | 3 |
| | 17 - 20 | 3 |
| Maintenance | 21 - 24 | 3 |
| | 25 + | 2 - 3 |

Figure 6.18

ExerciseDescription Activity

It displays the pic, a link to the video and entries to be save or edited.



Sets

Reps

Kg

Figure 6.19

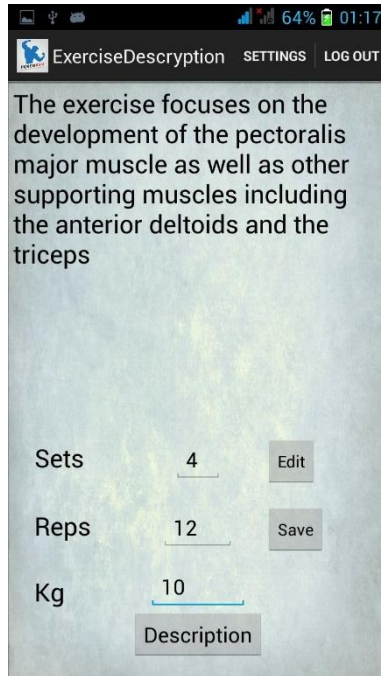


Figure 6.20

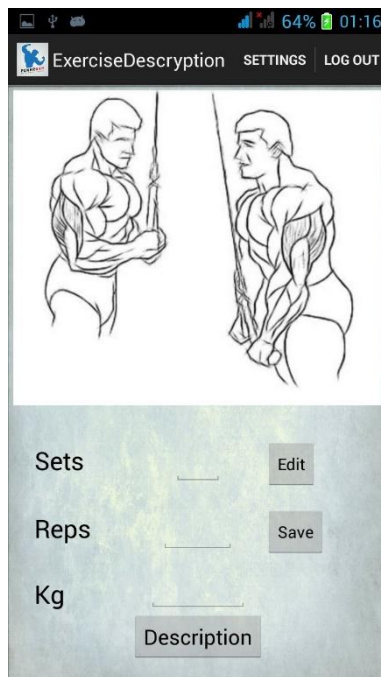


Figure 6.21

MyCalendar Activity

It displays the calendar and contains several button related to calendar functions



Figure 6.22

CalendarSchedule Activity

It displays the list of exercises according to calendar.

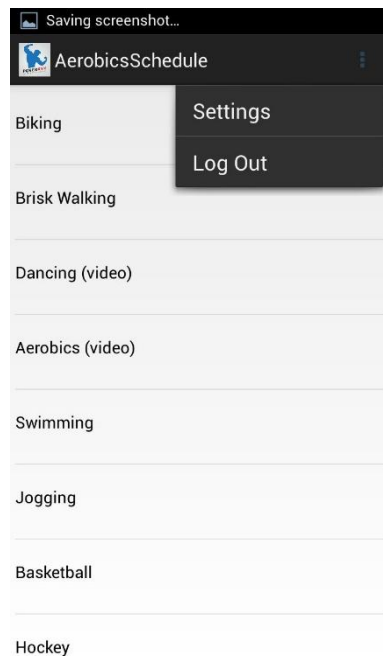


Figure 6.23

Chapter 7 - Android Framework Components

In this section, the various components of an android application such as activity, intent etc are discussed

7.1 AndroidManifest.xml file

Every android application must have an AndroidManifest.xml file in its root directory. This file lists out all the activities, intents, intent-filters, permissions etc the application user. This file is responsible for providing all the information about the application to the Android system. It also has the minimum SDK version needed to run the application. The permissions to access the internet, write onto a calendar etc are also declared in this file.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="mx.dr.drools"
    android:versionCode="16"
    android:versionName="1.6" >

<uses-permission android:name="android.permission.INTERNET" />

<uses-sdk
    android:minSdkVersion="11"
    android:targetSdkVersion="19" />

<application
    android:allowBackup="true"
    android:icon="@drawable/fitnessclublogo232147494931"
    android:label="@string/app_name"
    android:largeHeap="true"
    android:theme="@style/AppTheme" >

<activity
```

```

        android:name=".SplashScreen"
        android:label="@string/app_name" >
<intent-filter>
<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity
        android:name=".AskManFemale"
        android:label="@string/title_activity_ask_men_female" >
<intent-filter>
</intent-filter>
</activity>
<activity
        android:name=".AskWeightHeight"
        android:label="@string/app_name" >
</activity>
<activity
        android:name=".Result"
        android:label="@string/title_activity_result" >
</activity>
<activity
        android:name=".exercises.BodyBuilding"
        android:label="@string/title_activity_body_building" >
</activity>
<activity
        android:name=".exercises.Normal"
        android:label="@string/title_activity_normal" >
</activity>
<activity

```

```

        android:name=".exercises.LooseWeight"
        android:label="@string/title_activity_loose_weight" >
</activity>
<activity
    android:name=".exercises.Home"
    android:label="@string/title_activity_home" >
</activity>
<activity
    android:name=".AskDiabetes"
    android:label="@string/title_activity_ask_diabetes" >
</activity>
<activity
    android:name=".exercises.DiabetesSchedule"
    android:label="@string/title_activity_diabetes_schedule" >
</activity>
<activity
    android:name=".exercises.ResistanceExerciseSchedule"
    android:label="@string/title_activity_resistance_exercise_schedule" >
</activity>
<activity
    android:name=".exercises.Resistance"
    android:label="@string/title_activity_dhips_thighs" >
</activity>
<activity
    android:name=".exercises.AerobicsSchedule"
    android:label="@string/title_activity_aerobics_schedule" >
</activity>
<activity
    android:name=".ResistanceTimeTable"
    android:label="@string/title_activity_resistance_time_table" >
</activity>

```

```
<activity
    android:name=".ExerciseDescription"
    android:label="@string/title_activity_exercise_description" >
</activity>
<activity
    android:name=".ExerciseDescriptionTwo"
    android:label="@string/title_activity_exercise_description" >
</activity>
<activity
    android:name=".ExerciseDescriptionThree"
    android:label="@string/title_activity_exercise_description" >
</activity>
<activity
    android:name=".MyCalendarActivity"
    android:label="@string/title_activity_my_calendar" >
</activity>
<activity
    android:name=".CalendarSchedule"
    android:label="@string/title_activity_calendar_schedule" >
</activity>
</application>

</manifest>
```

7.2 Activities

An activity provides a means of interaction to the user. It provides a window where the UI can be designed according to the window. Almost all activities interact with the users. This application has the following activities:

7.3 Intents

The activities are activated through messages called intents. It basically has information that is required by the component. It helps in launching the activity or to do something with the existing activity. An intent passed to `startActivity()` is delivered only to an activity. The intent object holds the name of the component that handles the event, the action to be performed, data on which the operation is performed, the key value pairs to send some additional information. The intents that are used in this application are `ACTION_SEND` and `ACTION_EDIT`.

8. DEVELOPMENT

In this chapter we will design and implement layout with help of XML programming language and Eclipse programming environment. Also we will add Java classes and method to make program match requirements.

8.1 Reviewing the requirements

The main requirement needed is the data which will be displayed to user under different sections and the correctness of this data is also very important. The three sections will be fitness, body building and diet. Data will be place under corresponding categories and available to users as per their needs.

8.2 Structure of an Android project

The structure of android project is mostly the same, but also may differ depending on the project needs and IDE tool. We will describe basic structure while using the ADT. When programmer uses ADT the project structure is generated automatically. Even further, ADT is also generating the ready- made application “Hello word”. The GUI version of ADT is the easiest way to create an Android project but the advance programmer can be also using the set of tools which can be run in terminal session. The terminal tool called “ant” can debug the Android project and create sample structure even if developer uses any other programming tool than Eclipse.

8.2.1 Project folder structure

Basic Android project would have six directories such as: assets, bin, gen, libs, res, src. Also there are some files in project root directory such as: AndroidManifest.xml, licenses, project. Properties and other files.

The most important for the developers are “res” and “src” directories.

“res” directory contains all the current project resources such as: images, layouts, custom strings and other values. Images are stored in different directories depending on their size that application can automatically choose right image depending on the device specifications. Layouts are store in the “layout” folder. Basically layout file example would be an XML file which would specify elements and their position in current view. Also it is possible to code custom strings and colors so the parser can display them in application. It is recommended approach to store them in values directory rather than hard code to the actual code or XML file. It would make easy further development at translating the application to other languages.

The other important directory is “src”. This directory would usually consist of Java files which are adding functionality to the application. Then developer would create classes as separated Java files. If the class is created in GUI ADT environment the tool would generate automatically the statement in “AndroidManifest” file. If other programming environment is used, user must specify any new class activity by hard coding the “AndroidManifest” file.

Android manifest file usually would be placed to the root directory of the project and state required version of android, needed permissions and all activities which are run within the application.

8.3 Designing of the layouts

On the requirements base will be designed four different views. The idea of layout was brought from one of the discussing forums and tested in our project.

First view

This view will consist of a splash screen. The screen will contain a picture which will stay for 3 seconds so as to give the device time to load data from memory. This will be done using sleep function.

The Main View

This view consist of radio boxes, edit text boxes, text view boxes and a button. The radio boxes will help the user enter the user's gender into the device. Similarly labels and edit boxes are used to enter take as input the weight (in kg) and height (in inches) from the user. The button is used to fix the data and move to the next page. On click of this button knowledge will build and selected rules will apply.

8.4 Implementing the layouts

As was stated before, android application uses XML layout for displaying its content.

XML document would consist of several tags with given properties. The parent tag would state type of view which is the main for the document. Also it is allowed to use several views inside of one main view.

Like any other XML tag, this tag is given several properties which will define its identification, style, onclick action, etc. Identification is one of the important part of the tag. By defining the "id" programmer can use it in the Java code. Styles can be hard coded in the statement or linked to separate file which will specify style for this element. Action properties can call the event action which is defined in the code part of the project.

The Main View

The main view of prototype application will have simple look. For the organizing with the help of Relative Layout statement. This statement is parent layout statement for main view XML document. That is why it has to be linked to the <http://schemas.android.com/apk/res/android>. The example of parent statement for XML document would look like this <RelativeLayout

As we can see this statement hosts some properties which are important for displaying of this document. Xmlns:android could define namespace information but in this case it just

defines of unique of the parent tag as "http://schemas.android.com/apk/res/android" does not exist. `Layout_width` tells to the mobile device how to display this view. In this case option "fill_parent" is used so the program will use all the space on horizontal scale. `Layout_height` is the same as width but it uses vertical scale to display the view. `Orientation` is to display view depending on device orientation. It can be vertical or horizontal. In this case it is horizontal so it means that device's screen is used in portrait mode. All the child entries will be entered between the end of definition and closing statement `</RelativeLayout>` .

The child of linear layout can be any view: button, text view, etc. It can also host the other linear layout.

8.5 Maintaining Sessions

A session is a semi-permanent interactive information interchange, also known as a dialogue, a conversation or a meeting, between two or more communicating devices, or between a computer and user. A session is set up or established at a certain point in time, and then torn down at some later point.

In my application I have maintained sessions. By sessions I mean that once a user enters his body measurements then his data is kept till the user logs out. Once the user has logged out all the data is erased. In this I'm not using any memory available and later relocating, rather I'm using shared preferences and it requires just one allocation per string/int.

8.5.1 Shared Preferences

Android provides many ways of storing data of an application. One of this way is called Shared Preferences. Shared Preferences allow you to save and retrieve data in the form of key, value pair.

In order to use shared preferences , you have to call a method `getSharedPreferences()` that returns a `SharedPreferences` instance pointing to the file that contains the values of preferences.

```
SharedPreferences sharedPreferences = getSharedPreferences(MyPREFERENCES);
```

You can save something in the `sharedpreferences` by using `SharedPreferences.Editor` class. You will call the `edit` method of `SharedPreferences` instance and will receive it in an editor object. Its syntax is:

```
Editor editor = sharedPreferences.edit();  
editor.putString("key","value");  
editor.commit();
```

You cannot edit the data that has been stored but you can overwrite the existing shared preferences variable. There are other options too which are deleting the variable or deleting all the variables. You will call the `allClear` or `remove` method of `SharedPreferences` instance and will receive it in an editor object. Its syntax is:

```
Editor editor = sharedPreferences.edit();  
editor.allClear();
```

This way the data of the user is being stored and kept till the user logs out.

8.6Flowchart

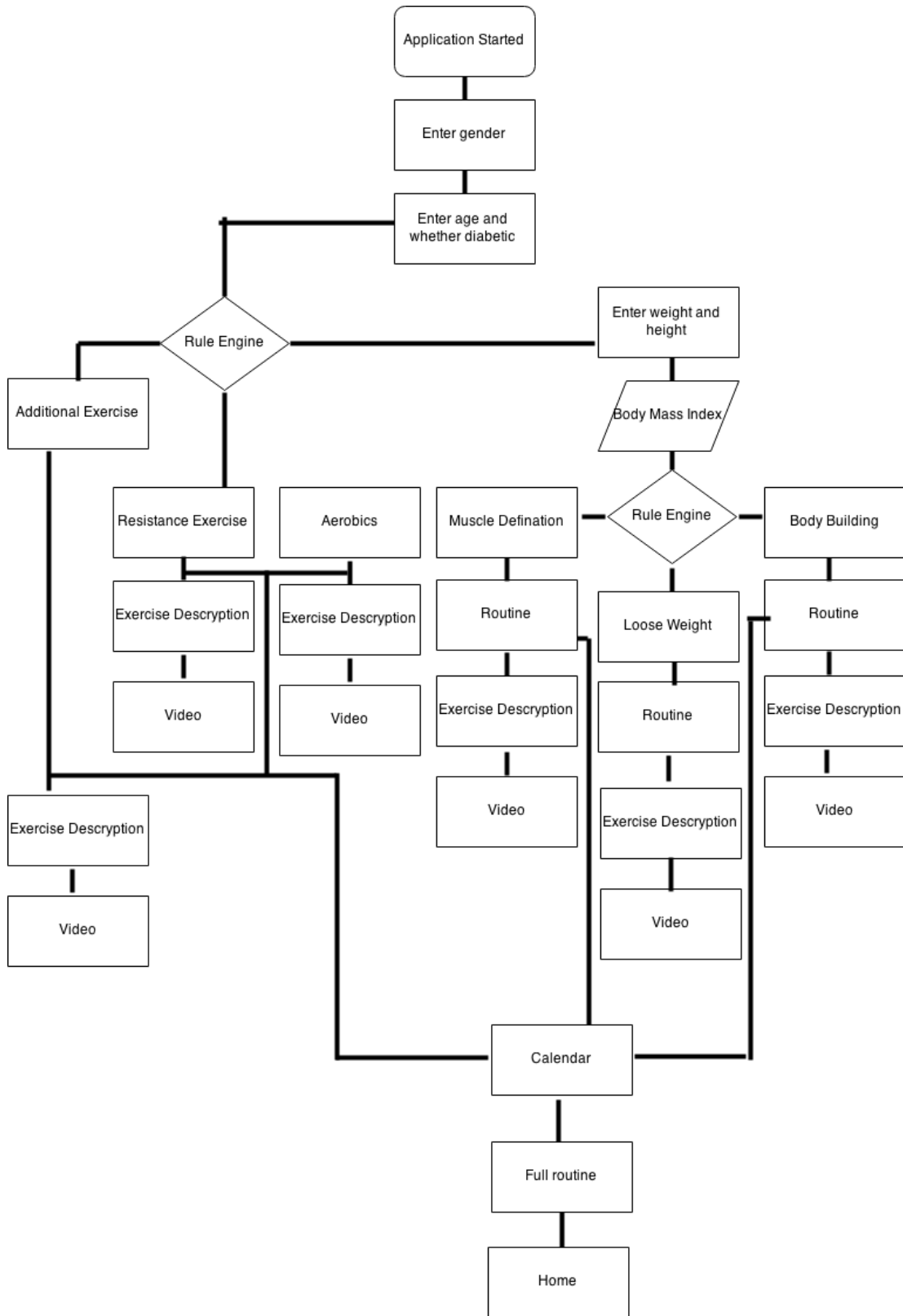


Figure 8.6 Flowchart

8.6 Designing and Implementing the Classes and Functions

Java programming language is used in android development. All the activities are organized in classes and triggered by certain events. Every android application has its starting activity. In the default project it would be main activity class.

The first class is the SplashScreen.java class. It shows a screen for three seconds and then moves to the main screen. I have used a postDelayed() function of the handler class to keep the screen constant for three seconds. Then the next screen is shown, the connecting is done with the help of Intent class through which it moves to the next activity.

8.6.1 Rule Engine Implementation

In the class named AskDetails.java the Rule Engine is implemented. The rule files are stored in the assets folder and will be accessed from there only using the following code.

```
try {  
    DRKnowledgeBuilder.assetsToFiles(this,"Files","drl");  
} catch (IOException e) {  
    e.printStackTrace();  
    return;  
}
```

Table 8.6.1 Lines of code

Next I have used an Async Task class to implement the Rule engine. For long-running or CPU-intensive tasks, there are basically two ways to do this: Java threads, and Android's native AsyncTask.

Neither one is necessarily better than the other, but knowing when to use each call is essential to leveraging the system's performance to your benefit.

Use AsyncTask for:

1. Simple network operations which do not require downloading a lot of data
2. Disk-bound tasks that might take more than a few milliseconds

Use Java threads for:

1. Network operations which involve moderate to large amounts of data (either uploading or downloading)
2. High-CPU tasks which need to be run in the background
3. Any task where you want to control the CPU usage relative to the GUI thread

Example:

```

class Async extends AsyncTask{
    //Run on UI thread
    @Override
    protected void onPreExecute(){}

    //Run in separate thread
    @Override
    protected Z doInBackground(X input){
        //Do task, like getting info from web
        return result;
    }

    //Run on UI thread
    @Override
    protected void onProgressUpdate(Y y){}

    //Run on UI thread
    @Override
    protected void onPostExecute(Z result){
        //Do something with data like update UI
    }
}

Async task = new Async();
task.execute(X input);

```

Table 8.6.1 Lines of code

Thus I have used and Async Task which will run in the background to build knowledge from the rules that are stored in the assets folder. When this background class is loaded a dialog box appears which shows and a spinner saying building knowledge. This dialog box is closed after the building of knowledge is complete.

Rules

First Rule

The first set of rules are related to BMI (Body Mass Index). Based on this data the user will get to know about his physical condition. The set of conditions are mentioned below:

| BMI Range | Category |
|-------------------|----------------------|
| Less than 16.5 | Severely underweight |
| From 16.5 to 18.5 | Underweight |
| From 18.5 to 25 | Normal |
| From 30 to 35 | Overweight |
| From 35 to 40 | Clinically obese |
| Above 40 | Dangerously obese |

Table 8.6.1: BMI Range

To support the rules I have created a bean class named BmiBean. This class contain all the setters and getters methods and used when declaring rules. The bean class is necessary for the declaring of rules otherwise it would become complex.

Second Rule

This rule is checking for whether the patient is diabetic or not and considers the user's age. Based on the data the user will get to know suitable physical exercise routine.

To support the rules I have created a bean class named Diabetes. This class contain all the setters and getters methods and used when declaring rules. The bean class is necessary for the declaring of rules otherwise it would become complex.

8.6.2 Rule Implementation

In the application I have implemented two rules. The first rule is for checking whether the user is diabetic or not, the second helps to find out the BMI of the user and further helps in choosing the right type of physical routine required.

8.6.2.1 Color Codes

A tree is formed from the rules that we apply with their constraints. The tree has different color nodes and each color represents different nodes. Below is a figure which show the different colors and their role.

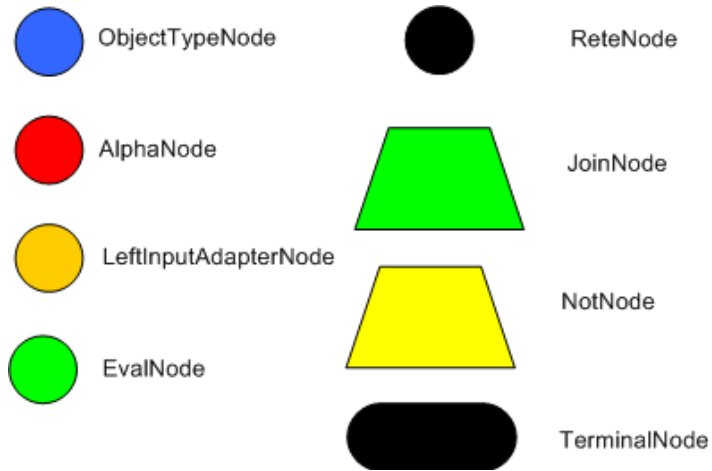


Figure:8.6.2.1 Different type of nodes

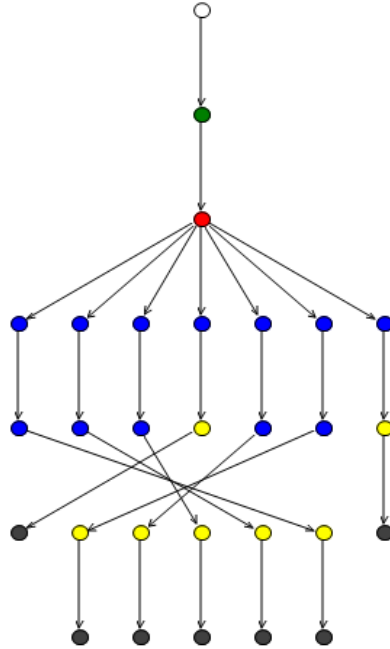


Figure 8.6.2.2 :Rete tree of the rules applied at AskWeightHeight activity

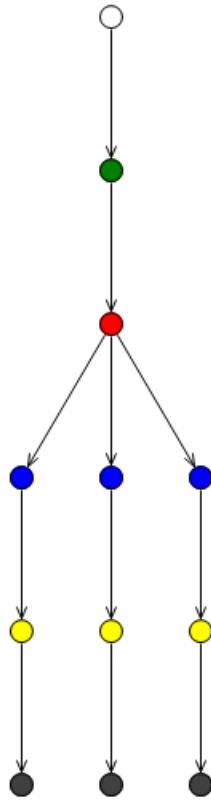


Figure 8.6.2.3 :Rete tree of the rules applied at AskDiabetes activity

9. TESTING

9.1 Unit Testing

In unit testing, various modules have been tested individually. This has been done manually to test if the expected result is actually seen on the screen. The following are test cases with the help of which the application has been tested.

| Sr. No | Test Case Description | Expected Result | Actual Result |
|--------|---------------------------------------|--------------------------------------|---------------|
| 1. | On load of SplashScreen. | Splash screen stays for 4 seconds. | Pass |
| 2. | On load of AskManFemale. | Check box works fine. | Pass |
| 3. | On load of AskDiabetes. | JBoss rule engine is started. | Pass |
| 4. | On load of DiabetesSchedule. | ListAdapter working. | Pass |
| 5. | On load of AerobicSchedule. | ListAdapter working. | Pass |
| 6. | On Click of items. | OnClick listener on items performed. | Pass |
| 7. | On browser open. | Browser opens a youtube.com link. | Pass |
| 8. | On load of ResistanceExerciseSchedule | ListAdapter working. | Pass |
| 9. | On load of ExerciseDescription. | Image and text displayed. | Pass |
| 10. | On load of TimeTable | Image displayed and scrollable. | Pass |
| 11. | On load of Home | Two image button displayed. | Pass |
| 12. | On load of BodyBuilding | Buttons and list displayed. | Pass |
| 13. | On click of day1 button | Visibility of Lists changed | Pass |
| 14. | On click of day2 button | Visibility of Lists changed | Pass |
| 15. | On click of day3 button | Visibility of Lists changed | Pass |
| 16. | On click of day4 button | Visibility of Lists changed | Pass |
| 17. | On click of day5 button | Visibility of Lists changed | Pass |
| 18. | On click of day6 button | Visibility of Lists changed | Pass |
| 19. | On click of day7 button | Visibility of Lists changed | Pass |
| 20. | On click of item of list | Image shown | Pass |
| 21. | On click of edit button | Enable to enter | Pass |

| | | | |
|-----|---------------------------------|---------------------------------------|------|
| | | values. | |
| 22. | On click of save button | Entries are saved. | Pass |
| 23. | On click of description button. | Description is saved. | Pass |
| 24. | On click of action bar button | Menu displayed. | Pass |
| 25. | On click of log out button | Application terminated. | Pass |
| 26. | On click of calendar button. | Calendar activity opened. | Pass |
| 27. | On click of date. | Schedule activity opened. | Pass |
| 28. | On load of Looseweight | LooseWeight activity opened. | Pass |
| 29. | On load of AskWeightHeight | JBoss rule engine is started. | Pass |
| 30. | On load of Result | BMI value is displayed and body type. | Pass |

Table 9.1 Unit testing

9.2 Compatibility Testing

This application was mainly designed for android phones as it helps the users find the physical exercise routines when they are on the move. Generally they try to carry something handy like cell phones with them and not the tablets. Different android phones have different screen sizes and resolution. The application has been tested for its compatibility with different screen sizes on the emulator.

9.2.1 Portrait mode

| Program Stage | Week | Frequency (days/w) |
|---------------|---------|--------------------|
| Initial stage | 1 | 2 |
| | 2 | 2 |
| | 3 | 2 |
| | 4 | 2 |
| Improvement* | 5 - 7 | 2 |
| | 8 - 10 | 2 |
| | 11 - 13 | 3 |
| | 14 - 16 | 3 |
| | 17 - 20 | 3 |
| Maintenance | 21 - 24 | 3 |
| | 25 + | 2 - 3 |

Figure 9.2.1 Portrait mode

9.2.2 Landscape mode

| Program Stage | Week | Frequency (days/week) | Intensity | | Duration (min) |
|---------------|---------|-----------------------|-----------------|-------------|----------------|
| | | | Exertion Level | RPE (10 pt) | |
| Initial stage | 1 | 2 | Light | 2 | 1 x 8 |
| | 2 | 2 | Light | 2 | 1 x 10 |
| | 3 | 2 | Moderate | 3 | 1 x 12 |
| | 4 | 2 | Moderate | 3 | 2 x 8 |
| Improvement* | 5 - 7 | 2 | Moderate | 3 | 2 x 10 |
| | 8 - 10 | 2 | Moderate | 3 | 2 x 12 |
| | 11 - 13 | 3 | Moderate | 3 | 2 x 8 |
| | 14 - 16 | 3 | Somewhat Strong | 4 | 2 x 10 |
| | 17 - 20 | 3 | Somewhat Strong | 4 | 2 x 12 |
| Maintenance | 21 - 24 | 3 | Somewhat Strong | 4 | 2 x 15 |
| | 25 + | 2 - 3 | Moderate Strong | 3 - 4 | 2-3 x 8-15 |

Figure 22 Landscape mode

10. CONCLUSION

My project has resulted into working prototype which can help the user regarding fitness. The project was to create user input interface and implement the JBoss Drools Rule Engine and provide exercises and for users with medical problems.

The application can save the user's data which is entered in the application which will help the application to customize routines for the user. The application can not only be used by one user only but the user can sign out deleting all his data and entering new user's entries.

This type of application is the need of the hour for the young generation who are too busy to take a step towards fitness.

10.1 Future Work

The application can be improved in many ways and can be extended to support more devices like the tablets and iOS devices. Following are some of the possible extensions:

- The application can be extended to provide more promising physical routines.
- The application can be extended to provide routines for users having other general medical problems like asthma and joint problems.
- The application can be extended to provide a diet routine to the user.

11. References

- [1].www.developer.android.com
- [2].www.diabetes.ca
- [3].www.jboss.org
- [4].www.play.google.com
- [5].www.tools.android.com
- [6].www.stackoverflow.com
- [7].www.android.com
- [8].www.wikipedia.com
- [9].www.androidauthority.com
- [10].www.tutorialspoint.com/android
- [11].www.javatpoint.com
- [12].www.vogella.com
- [13].www.leanmusclefitness.com
- [14].www.github.com
- [15].Android based Physical Exercise Advisor Application Zurina Hazemi Syahrul Nizam Junaini Faculty of Computer Science & Information Technology University Malaysia Sarawak 94300 Kota Samarahan, Sarawak, Malaysia.
- [16].Exercise prescription for patients with type 2 diabetes and pre-diabetes:A position statement from Exercise and Sport Science Australia- Matthew D. Hordern, David W. Dunstan, Johannes B. Prins, Michael K. Baker, Maria A. Fiatarone Singh, Jeff S. Coombesb,
- [17].Mobile Based Business Rule Engine-Journal of Computer Applications ISSN: 0974 – 1925,Volume-5, Issue EICA2012-3, February 10, 2012, Minal Bidave, Nilima Kale, Pune. Geetanjali Katare, Suvarna Palde.