

Hobby Horse - A Social Data Mining Project

A Project Report

Presented to

The Faculty of the College of
Engineering

San Jose State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science in Software Engineering

By

Radhika Wadegaonkar & Sulagna Bal

May 2012

Copyright © 2012

Radhika Wadegaonkar & Sulagna Bal

ALL RIGHTS RESERVED

APPROVED FOR THE COLLEGE OF ENGINEERING

Prof. Magdalini Eirinaki, Project Advisor

Dan Harkey, Director, MS Software Engineering

Dr. Sigurd Meldal, Chair, Computer Engineering Department

ABSTRACT

Hobby Horse (Social Learning) – A Social Data Mining Project

By Radhika Wadegaonkar, Sulagna Bal

Various social networking sites have become a prominent part of life of the current generation. However, apart from the status updates and pictures of the friends across the network, there is not much that one learns from it. In view of this scenario and access of such rich and large user data, we have developed a web-based / mobile social data-mining project named Hobby Horse (Social Learning). This system is for the users who would like to impart their talents related to their hobbies in the form of teaching online (real-time) and for the users who wish to pursue a hobby of their choice at their leisure. Users of the system can have a role of ‘expert hobbyist’, a ‘novice hobbyist’ or both. Users can participate in different learning categories, or can create one dynamically. Hobbyists will be rewarded points and ratings to define their badge-levels or skill-levels based on the feedbacks. The ratings or badge levels will be used to recommend them to each other. They can then redeem their points towards free gifts or coupons. Login to our system has been achieved using either Facebook credentials or HobbyHorse credentials. The mined data from the social networking API(Facebook) is used to send HobbyHorse invitations to Facebook users. This mining is based on various criteria like similar skill sets, interests, similar age group and so on. The system has 3-tier architecture. The business layer is modeled using open-source technologies that implement web-services, thus allowing flexible additions of new custom plugins in the future. The front-end and middle layer logic are driven using the MVC model.

Acknowledgments

The authors are deeply indebted to Professor Dan Harkey, Professor Magdalini Eirinaki, and Dr. Sigurd Meldal for their invaluable comments and assistance in the preparation of this study.

Table of Contents

ABSTRACT.....	vii
Acknowledgments.....	ix
Chapter 1: Project Overview.....	2
1.1 Introduction.....	2
1.1.1 Project goals and objectives	2
1.1.2 Problem and motivation	2
1.1.3 Project application and impact	5
1.2. Proposed Areas of Study and Academic Contribution	6
1.3. Current State of the Art.....	7
1.3.1 Literature survey.....	9
Chapter 2. Project Architecture.....	11
2.1 Introduction.....	11
2.2 Architecture.....	12
2.3 Architecture Subsystems.....	13
2.3.1 HobbyHorse Server Architecture.....	13
2.3.2 External Interfacing & Connectivity.....	14
Chapter 3. Technology Descriptions.....	16
3.1 Introduction.....	16
3.2 Client Technologies	17
3.3 Middle-Tier Technologies.....	18
3.4 Data-Tier Technologies.....	18
Chapter 4. Project Design.....	20
4.1 Client Design.....	20
4.1.1 Login.....	20
4.1.2 Application Flow	21
4.1.3 Use Cases	22
4.2 Middle-Tier Design.....	27
4.2.1 Data Mining	27
4.3 Data-Tier Design.....	29
4.3.1Domain Class Diagram	29
Chapter 5. Project Implementation	31
5.1 Client Implementation.....	31
5.1.1 HobbyHorse Login Page:.....	31
5.1.2 HobbyHorse Register Page:.....	32
5.1.3 HobbyHorse Some Validations:	33
5.1.4 HobbyHorse Welcome Page:.....	34
5.1.5 HobbyHorse Available Lessons:	35

5.1.5 HobbyHorse Forthcoming Lessons:	36
5.1.6 HobbyHorse Start a Lesson (Video):	37
5.2 Middle-Tier Implementation.....	38
5.3 Data-Tier Implementation.....	43
Additional Project Implementation Features	49
5.3.1 Video Calling:.....	49
5.3.2 Badges:.....	51
5.3.3 Ranking:.....	54
5.3.3 Other Data Mining logic	55
Chapter 6. Performance and Benchmarks.....	56
Chapter 7. Deployment, Operations, Maintenance.....	61
7.1 Deployment.....	61
7.2 Operations	61
7.3 Maintenance	62
Chapter 8. Summary, Conclusions, and Recommendations	63
8.1 Summary	63
8.2 Conclusions.....	63
8.3 Recommendations for Further Research.....	63
References	65
Appendices	68
Appendix A. Description of CDROM Contents	68

List of Figures

Figure 1.1: Average age distribution across 19 social networking sites [7].....	3
Figure 1.2: Showing percentage of Internet users throughout the world [8].	3
Figure 1.3. World Internet usage and population statistics [8].	4
Figure 2.1: HobbyHorse System Architecture.	12
Figure 2.2: HobbyHorse Server Architecture.....	14
Figure 2.3: Data Mining Interface.....	15
Figure 4.1: State model of the Login module.....	20
Figure 4.2: Application Flow of HobbyHorse.....	21
Figure 4.3: Use case for user login.....	23
Figure 4.4: Use case for general user	24
Figure 4.5: Use Case for Novice Hobbyist user.....	25
Figure 4.6: Use Case for Expert Hobbyist user.....	26
Figure 4.7: Data-mining in HobbyHorse.....	27
Figure 4.9: Domain Class Diagram of Hobby Horse.....	30
Figure 5.1: Login page.	31
Figure 5.2: Register User page.....	32
Figure 5.3: Required field.	33
Figure 5.4: Authentication failed.....	33
Figure 5.5: Welcome page.....	34
Figure 5.6: Available Lessons page.	35
Figure 5.7: Forthcoming Lessons (Joined Lessons) page.	36
Figure 5.8: Start Lessons page.	37

List of Tables

Table 1.1: Comparison between various existing and popular social networking sites and HobbyHorse.....	7
Table 6.1: Testing methodologies and their purpose.....	56
Table 6.2 : Testing and Bug Reporting tools.....	57
Table 6.3 Entrance and exit criteria for Unit test.....	57

Chapter 1: Project Overview

1.1 Introduction

1.1.1 Project goals and objectives

In this competitive world, people are getting busier and stressful. They are so much occupied with other things in life that they hardly have any time to cultivate their hobbies. To pursue a hobby one might have to take up a hobby class where he / she would have to be present physically. Not only that, he / she would have to take out time from her busy life to attend those classes. However, research shows that most of us are hooked to the internet most of the time. It might be work-related, search-related or study-related. No matter what, most of us do find time to sneak peek into those social networking sites to either while away time or to take a relaxing break from our work. This fact is evident from the facts stated in Figures 1-4 below. Taking this into account, we plan to choose the Web as our medium to build an application for those users who want to pursue a hobby in spite of their busy schedule. We believe, if they would have time for social networking over Facebook or Twitter or LinkedIn, they would definitely find time to play around with HobbyHorse. They can use HobbyHorse as and when they want to.

1.1.2 Problem and motivation

Over the past few years, Social Networking websites have gained prominence in the Internet world. According to a survey conducted by the Pew Internet & American Life Project, 55% of the online American youths aged between 12 and 17 years are users of these Social Networking sites [19]. Not all Social Networking sites target teenagers, for example Match.com. As per Pew Internet and American Life Project:

“Social networking use among internet users ages 50 and older has nearly doubled -- from 22% to 42% over the past year.”

According to a Royal Pingdom's blog, age statistics for 19 different and popular social networking sites (Facebook, LinkedIn, Myspace, Twitter, Slashdot, Reddit, Digg, StumbleUpon, FriendFeed, Last.fm, Friendster, LiveJournal, Hi5, Tagged, Ning, Xanga,

Classmates.com, Bebo) show 25% of the users are aged between 34 and 44 and 3% are aged 65 or older [7].

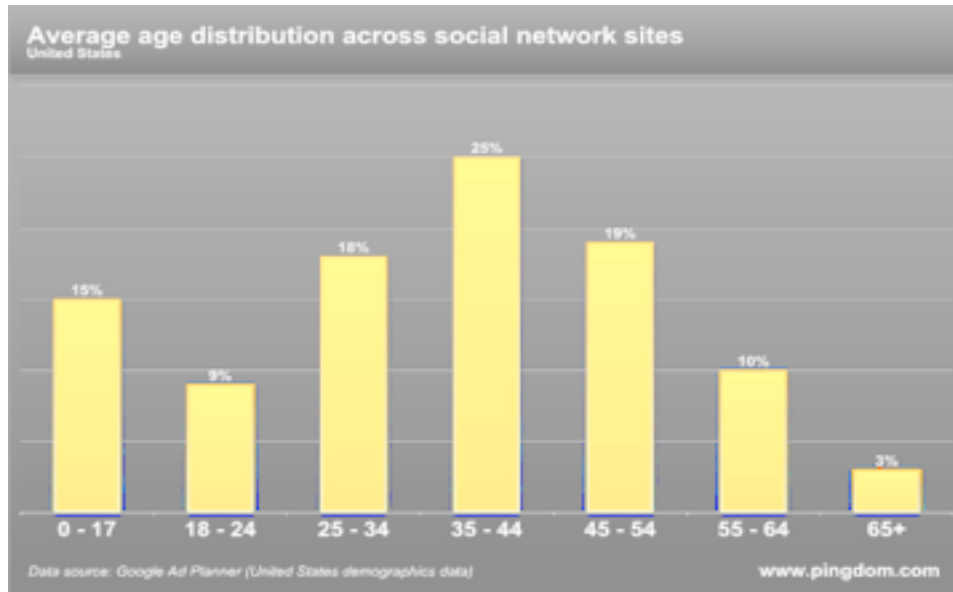


Figure 1.1: Average age distribution across 19 social networking sites [7]

Also it is worth noting that with the advancement of technology and people's eagerness to know the new and keep pace with the time, Internet users are also increasing in number every year throughout the world. We derived few charts from the source [8] that show the increase in the number of Internet users all throughout the world.

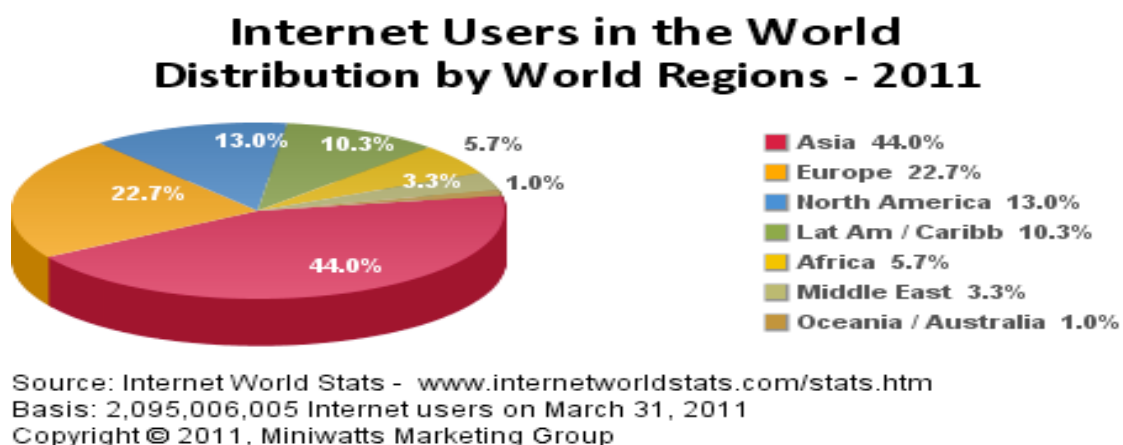


Figure 1.2: Showing percentage of Internet users throughout the world [8].

Below table shows the usage of Internet among the population in various regions of the world. This table based on the data collected on March 31, 2011. The demographic numbers (population) are derived from US Census Bureau and the Internet data usage

number is derived from various other reliable sources like Neilson Online and International Telecommunication Union. It shows a significant growth in the users from 2000 to 2011. It establishes the fact that Internet is ubiquitous and is used by everyone. It gives us enough reason to develop a project that is web based.

WORLD INTERNET USAGE AND POPULATION STATISTICS March 31, 2011						
World Regions	Population (2011 Est.)	Internet Users Dec. 31, 2000	Internet Users Latest Data	Penetration (% Population)	Growth 2000-2011	Users % of Table
Africa	1,037,524,058	4,514,400	118,609,620	11.4 %	2,527.4 %	5.7 %
Asia	3,879,740,877	114,304,000	922,329,554	23.8 %	706.9 %	44.0 %
Europe	816,426,346	105,096,093	476,213,935	58.3 %	353.1 %	22.7 %
North America	347,394,870	108,096,800	272,066,000	78.3 %	151.7 %	13.0 %
Latin America / Carib.	597,283,165	18,068,919	215,939,400	36.2 %	1,037.4 %	10.3 %
WORLD TOTAL	6,930,055,154	360,985,492	2,095,006,005	30.2 %	480.4 %	100.0 %
NOTES: (1) Internet Usage and World Population Statistics are for March 31, 2011. (2) CLICK on each world region name for detailed regional usage information. (3) Demographic (Population) numbers are based on data from the US Census Bureau . Copyright © 2001 - 2011, Miniwatts Marketing Group. All rights reserved worldwide.						

Figure 1.3. World Internet usage and population statistics [8].

1.1.3 Project application and impact

We are predicting a positive impact of this project to the society. It will be extremely beneficial for the people who would like to nurture their hidden talents even when they are running a busy schedule every day. Similarly to how they sneak into Facebook or Twitter during their work hours, they would find Hobby Horse more relaxing and entertaining that would in a matter of seconds refresh their minds and help them work or do other important things better.

Moreover, research shows; it is very important for one to have a hobby to lead a healthy life. Sources [16] and [17] list reasons for pursuing a hobby.

Lastly, according to Benevenuto et al. [18]:

“Video content is becoming a predominant part of user’s daily lives on the Web. By allowing users to generate and distribute their own content to large audiences, the Web has been transformed into a major channel for the delivery of multimedia, leading society to a new multimedia age. Video pervades the Internet and supports new types of interaction among users, including video forums, video chats, video mail, and video blogs.

Additionally, a number of services in the current Web 2.0 are offering video-based functions as alternative to text-based ones, such as video reviews for products (www.amazon.com) and (www.expotv.com) are examples of sites that allow users to post video reviews about products), video ads and video responses [Shannon 2007]. This huge growth of multimedia content in the Web is mostly due to the evolution of the user from content consumer to content creator. As a consequence, several multimedia issues need to be revisited.”

HobbyHorse combines both the above features, that is, it encourages people to pursue a hobby of their choice anytime and anywhere and it enables hobbyists to interact via video for many reasons like learning and teaching hobbies. Therefore, we hope that

HobbyHorse would not only be useful and applicable to people of all ages but simultaneously would definitely make a positive impact on the society.

1.2. Proposed Areas of Study and Academic Contribution

The goal of the project is to make use of the extremely large and rich user data, hook users onto it by giving it a flavor of social interaction and yet make them learn something new and something that is related to their hobbies. The social interaction of users would be implemented using group video calling. We will also make use of advanced data mining techniques such as “Associative Roles”, “Market Basket Analysis” and similar such algorithms with the social data. We will also be implementing the “Ranking” algorithm, which is not exactly a Data Mining algorithm, but an intermediate step in the process. Since we plan to primarily have a web interface for our system, we plan to make it very user-friendly and interactive using the latest UI technologies such as HTML5 and CSS3 and implement the least-click model. On the server side, the logic will be implemented using the MVC model using technologies like PHP. The API layer would be enhanced using Java Web Services with the help of technologies such as Spring MVC and a build tool such as Maven.

PHP: PHP is an object-oriented server-side scripting language. HTML and CSS can be easily embedded within the language. It is designed especially for web-development. It can very well enrich the web-page with respect to the user experience as well as the logistics.

Java & Web-Services: Java is a very powerful object-oriented programming language. It is platform independent and is capable of spawning very complex applications. We are planning to use the REST web-services using the Java technology to enhance our API layer and incorporating an MVC structure using the Spring technologies.

HTML5 & CSS3: HTML5 and CSS3 are the latest technologies in the field of User Experience with respect to web-development. In their basis, HTML will be used to develop the basic forms and basic structure of a page, whereas CSS will be used to define the color schemes, font styles, sizes and so on.

MySQL: It is a relational database system and is majorly known for its scalability capabilities. It is very well known to be compatible with Java with complex query usage.

1.3. Current State of the Art

There are a few portals currently that provide the feature of either learning online or interaction of users with a group video. Examples of learning online are Wikipedia (www.wikipedia.org), W3Schools (www.w3schools.com), YouTube (www.youtube.com) and so on. Examples of social interaction with a group video are Facebook (www.facebook.com), Skype (www.skype.com), Google+ (<https://plus.google.com>), Second Life (secondlife.com) and so on. Other popular social networking portals are LinkedIn (www.linkedin.com), Twitter(www.twitter.com) and MySpace (www.myspace.com) where users can interact for business / job purposes, just share small updates or share photos / updates / videos with friends respectively. However, none of them incorporate both the factors of learning online and social interaction through a group video. They provide means of interaction in different forms. For example, Wikipedia is a great platform to share and contribute information in a particular field. It is a 2-way interaction unlike W3Schools that does not accept any kind of interaction from the users of the system. YouTube is a 1-way interaction in the sense, it allows any users to contribute, but not edit anyone else's work unlike Wikipedia. Facebook and Google+ have the 2-way interaction and they use the video technologies termed as group video calling and hangout respectively. We elaborate on the social networking and popular portals and compare them in what follows:

Table 1.1 : Comparison between various existing and popular social networking sites and HobbyHorse.

Technology	Social Interac tion	Learning (hobby)	Feedback	Group Video Calling	What does the user take back?
facebook.com	Yes	No	Yes, in form of comments and likes.	Yes (termed "video calling")	Information about friends' status updates and pictures
linkedin.com	Yes	No	Yes, in form of comments and likes.	No	Very effectively,

					good jobs!
plus.google.com	Yes	No	Yes, in form of comments and likes.	Yes (termed “hangout”)	Information about friends’ status updates and pictures
secondlife.com	Yes	No	Yes, everything animated	No	Entertainment
www.skype.com	Yes	No	May be	Yes	Getting in touch with friends / family
w3schools.com	No	Yes	Yes, in form of quizzes (but 1-way)	No	Learns a new concept / technology
wikipedia.org	No	Yes	Yes, 2-way feedback as users can edit others’ work too.	No	Learns a new concept / technology
youtube.com	No	Yes, possible	No	No	Entertainment / Learning something new with video tutorials
Hobby Horse	Yes	Yes	Yes, in the form of recommendations, ratings and badges.	Yes	Learning a new hobby of their own choice and at their leisure. They also can redeem coupons for expertise

					level lessons or purchase of related products (like guitar w.r.t guitar lessons)
www.myspace.com	Yes	No	No	No	Information about friends' status updates and pictures
www.twitter.com	Yes	No	No	No	Information about friends' status updates and pictures

1.3.1 Literature survey

Social learning is a concept of learning together in a group with equal expertise or higher expertise people. It is referred as social constructivism by the educational psychologists [1]. It was observed that the students who studied in groups as compared to students who just attended college did far better than the rest. Studying in group gives a sense of competitiveness and quick feedbacks from the fellow mates, and hence encouragement. In our project, social learning refers to learning about something that falls under your liking that is your hobby. We have implemented social learning in a much advanced way, where the novice hobbyists and expert hobbyists can interact with each other using the group video communication. Novice hobbyists who attend a lesson conducted by an expert hobbyist, provide comments or feedback to the expert hobbyists. An expert hobbyist can use these feedbacks to identify loopholes in teaching and improve on them. These feedback or comments are also used for data-mining to rank the expert hobbyists belonging to a particular lesson.

Also, we have implemented data mining techniques to send out suggestions to novice users to join lessons of their interests.. This kind of mining is based on two things:

1. Novice users' interests and skillset.
2. Type of lessons already attended by them.

We have researched on a few data mining algorithms for the same. The most common one we found that proves to be useful to us is called the Aprox algorithm. *“Association rule mining is a common technique for performing market basket analysis. The intent is to gain insight into customers' buying habits and discover groups of products that are commonly purchased together. As an example, an association rule may show that 98% of all customers that purchase frozen pizza also purchase soda”* [21].

Given a set of user profiles U and a set of item sets $I = \{I_1, I_2, \dots, I_k\}$, the support of an item set $I_i \in I$ is defined as $\sigma(I_i) = |\{u \in U : I_i \subseteq u\}| / |U|$ [21].

The simplest form of implementing this was based relying on the frequency of lessons attended by a particular user in a specific order or succession. This is very specific to our project. However, there are multiple libraries available in Java that would implement association rules depending on the factors we feed them. There are basically 2 thresholds that need to be input to these classes, viz; the percentage of frequency (like 20% times, 30% times and so on) and the other factor is the number of elements in the chain of the rule.

Another very basic level algorithm we are following is the “Ranking/Rating” system. This involves nothing more than a mathematical formula that calculates an “average” of the existing ratings given as feedbacks.

Chapter 2. Project Architecture

2.1 Introduction

Figure 4, shows the whole system architecture of the HobbyHorse application. The Client UI is developed using HTML5 and CSS3. We have also used JavaScript and AJAX on the client side for dynamic and fast processing. Between the client and the Database, sits the middle layer. This layer consists of basically four parts, the Data Access layer, the Business layer, the Service Layer and the Presentation Layer. The Data Access Layer directly interacts with our MySQL database. Above this, is the business layer, that makes use of the Data Access (referenced as DAO classes). The business component classes are built over the DAO implementation classes. The data flow within these components is gracefully done using Java Bean classes. The Web Services (The Service Layer) are developed using the Business classes. Different Web Services are built for different domain classes. All these are done using the Java technologies. The Presentation layer in this case is actually divided as MVC architecture for the server side programming. This is done using PHP and the framework CodeIgniter. Different classes for the Models, Controllers and Views are developed for the graceful exchange of data to and from the Web Services. The data format used for this purpose is JSON.

We used MySQL for data persistence & mainly two reasons: a) Open source and b) light weight.

2.2 Architecture

The system architecture as defined in the above section can be clearly depicted in the figure below:

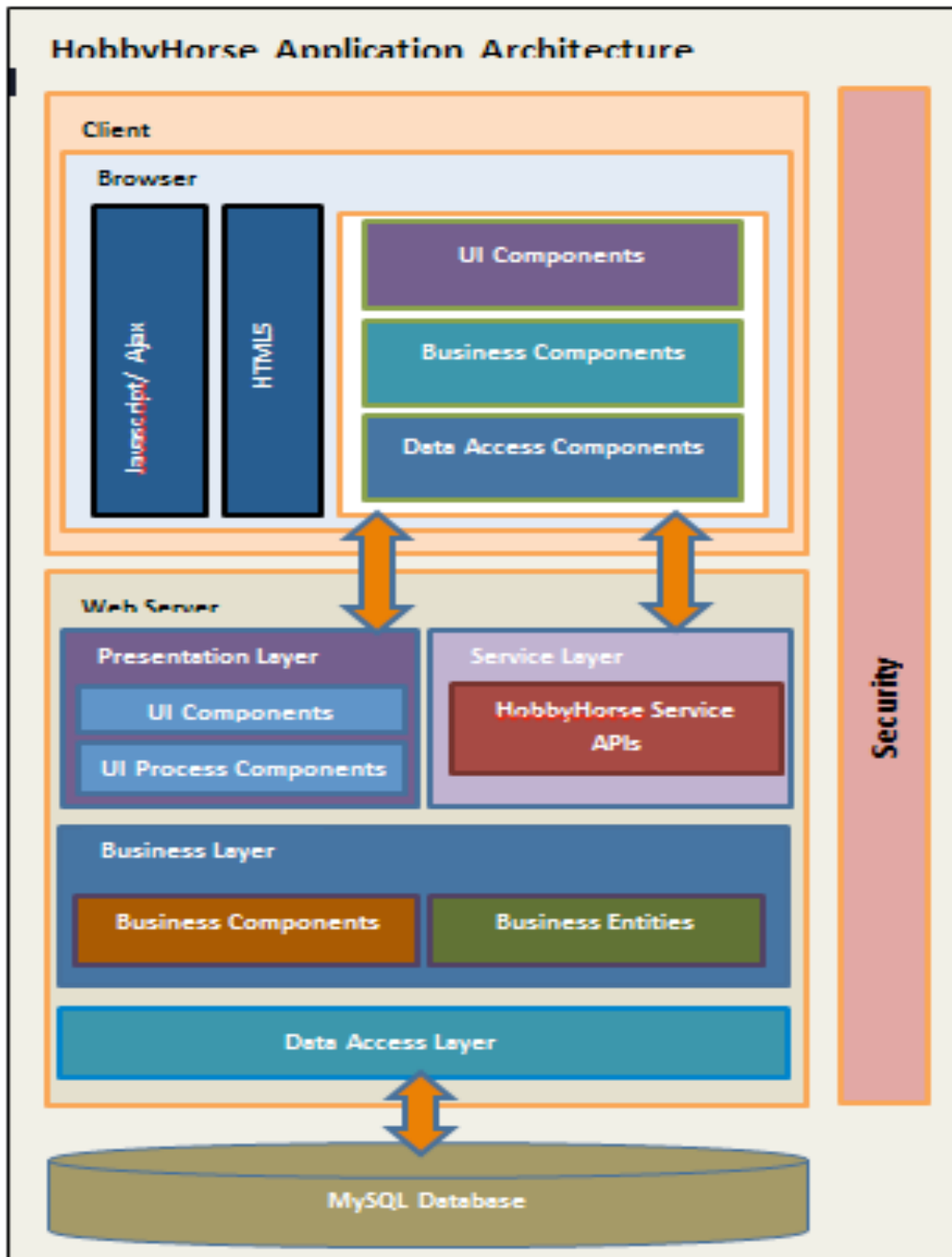


Figure 2.1: HobbyHorse System Architecture.

2.3 Architecture Subsystems

2.3.1 *HobbyHorse Server Architecture*

Figure 2.2 depicts the server architecture in detail. The server architecture basically consists of the middle as well as the backend layer. The middle layer itself consists of the Data Access Layer or the DAO classes, the Business Layer, the Service Layer and the Presentation Layer. Some part of the presentation layer however, merges with the front end that is the view part of the MVC server side architecture. This is implemented using PHP & CodeIgniter. To explain more in detail, at the very backend, we have our MySQL instance running. The database tables are mapped with the help of Java Beans at the very next layer of the backend. This acts as the layer for data exchange between the database and the DAO classes. The DAO classes have implementations of the functions that directly query the database and again format the results in Bean classes for simplicity and graceful data handling. Again, this is then given to the Business Layer, where we have written the logical functions that make use of DAO functions. Over this, we have implemented our Web Service classes. All these relations are handled using the spring server architecture, where we have injected our classes to define the relations between each of them. The essential Web Services are defined in the modules (or categories) of Communication, Profile Management, Groups, Login and other general web services. For example, the Login module consists of web service to do complete user authentication. These services take username and password as their parameters. The Communication Module consists of Web Services such as users/{username}.json, participate/join/{lessonId}/{username}.json and so on.

Each of these modules opens to our server side MVC architecture (PHP). That is, we are calling these web services using PHP with the help of the Zend framework and again the model classes handle the data. The model classes then hand over this data to the controllers, which further decide what view to load (completely presentational aspect) based on the given data.

Additionally, at the web services and business logic level, we are also implementing the external API calls for the Data Mining algorithms. And, it is here that we have integrated them into the system so that, from the PHP level all these would seem internal. Similarly, we have also integrated the external video calling libraries at the model level

of the presentational layer. Thus, these video libraries are in PHP as this talks more about the front-end rather than any backend logic.

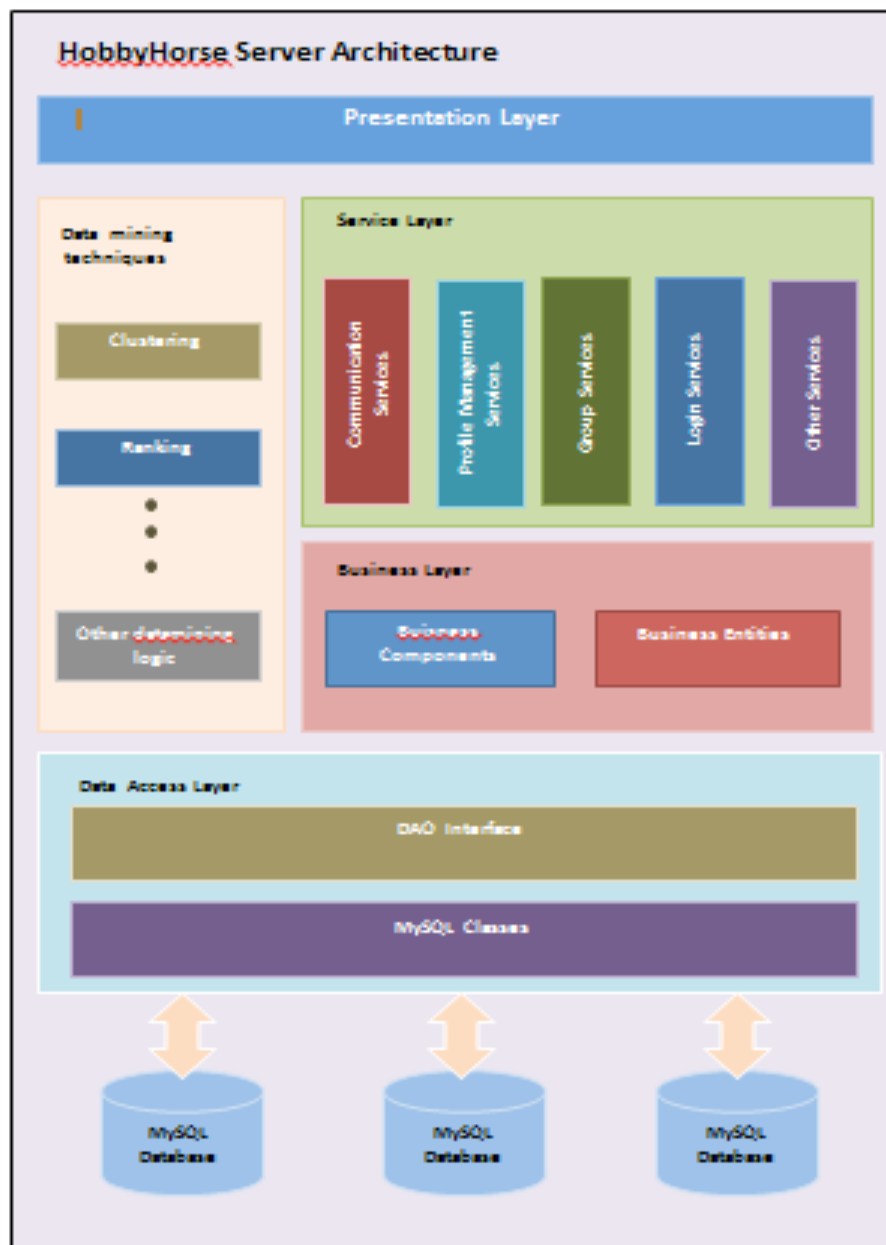


Figure 2.2: HobbyHorse Server Architecture.

2.3.2 External Interfacing & Connectivity

Figure 2.3 shows the block level diagram for the interfacing of the external libraries into our system. For Data Mining techniques such as Association Rules (Recommendations) and Ranking, we are using the available APIs and/ or Web Service Interfaces of the

same. They fit into our system in the middle layer just before the Web Services Layer, that is at the Business Logic Layer.

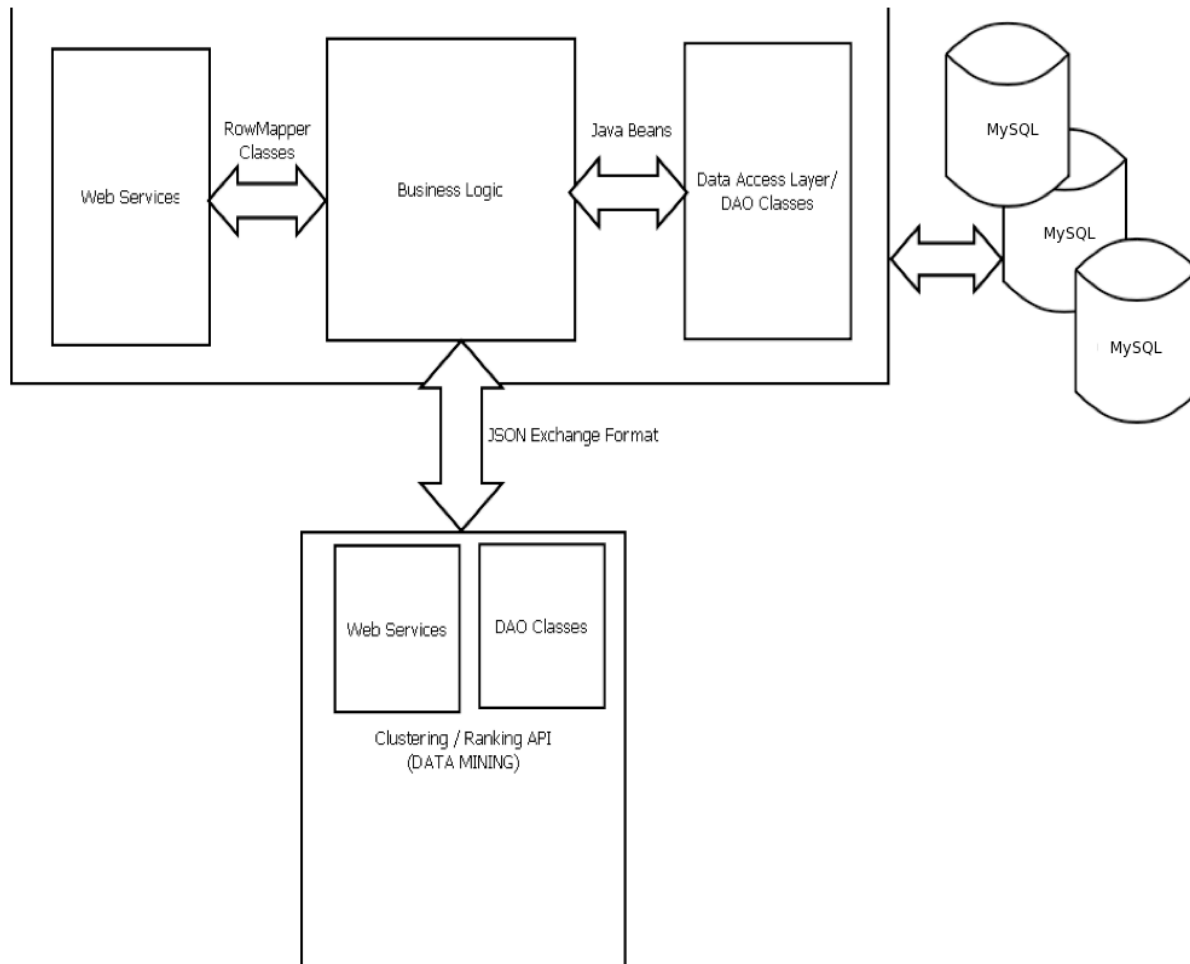


Figure 2.3: Data Mining Interface

Chapter 3. Technology Descriptions

3.1 Introduction

Tools are an important factor while building any kind of projects. It drives the project in a correct path. If wrong tools are used, it may lead to absolute hay wired way of going through the software development lifecycle of the project. To start with the basics, we have used:

- Microsoft Work for Reporting
- Google Gadgets under Google documents for developing the Gantt Chart to display our project schedule and planning.

During documentation, we had to represent our project implementation in a variety of block diagrams and flowcharts. We have used “Dia” which is an open source tool for the same.

We had to do a lot of research about the data mining algorithms and before we ended up choosing the appropriate algorithms, we had to research a lot on the same by reading articles and books. We used the scholar.google.com website for this and it helped us a lot to provide apt data mining books.

Since, we followed the agile development model for our software development; we used the scheduling tool JIRA to track our tasks and the time requirements for the same. This tool also helps us to file bugs if any and assigns it to the appropriate team members. Our major part of project development is communication and transfer of files to each other during the phase. For the communication purposes, we have used Skype as our major tool and also tools like DropBox or simple email for the transfer of files amongst each other.

Standardizing the project development phase is a very important task for the smooth flow of the project phases. It is a very important management aspect of the project as well. We have always tried to maintain the same approach for similar tasks during the project development phases till now.

We have planned to follow the agile development process for the lifecycle development of our software. Hence, we have decided a span of 15 days for each sprint and implement scrum meetings every alternate day to update each other about the development on each team member's side.

We have also maintained the minutes of the meetings after every meet with our project guide and advisor, so as to keep a track of what things have to be changed, done and deleted.

As mentioned earlier, for communication we have created a google group for ourselves and transfer any kind of information only through the group. This helps us to maintain and store data in one place over the Internet and hence is accessible anytime. Also as mentioned earlier again, we have used Dia as the tool for drawing block diagrams and have followed UML standards for the same.

The most important tool used in our project is "github". We have created a source project on github as a central server repository for our source code. Whenever each of the team members modifies any data, we push it to the git server. So before making any changes next time, each of the team members then pulls the latest changes and then modifies her working copy. Our source code can be accessed on this repository: <https://github.com/rwadegaonkar/hobbyhorse>.

3.2 Client Technologies

With respect to coding, we have used Web 2.0 standard for our website development. The technologies that we have used are:

- HTML5 – This is still not supported in all the browsers and is highly advanced so as to store dynamic data in the browser using SQLite.
- CSS3 – This can almost replace JavaScript and supports complex animation techniques.
- JavaScript – This is majorly used to add dynamicity to the HTML page, such as DHTML and minor animations.
- AJAX – AJAX is majorly used to load the data on the page dynamically, when the entire page need not be completely refreshed.

All these are used according to the way they comply with the standards chosen.

3.3 Middle-Tier Technologies

PHP: It is an Object Oriented language which is highly scalable as a scripting / programming language. We are calling web-services using PHP and displaying the same in HTML on the client side. We have thus used the concept of DTO (Data Transfer Object). Data can be transferred using XML / JSON (preferable) formats.

Zend: It is a PHP framework. We have basically used this framework to call the Webservices in the backend with the HTTP Client.

CodeIgniter Framework: This is also a PHP framework that we have used to incorporate an MVC structure in the middle tier as well.

3.4 Data-Tier Technologies

J2EE / JAVA: We have used J2EE in a complete Object Oriented way with model, controller, DAO and domain classes.

- **Maven:** We have used Maven as the building tool. It is highly advanced in terms of pulling in the dependencies on the fly. It also has a local cache repository so while building it checks in the local repository for the available libraries, if absent it pulls the libraries as defined from the authenticated websites. Apart from these functionalities, Maven provides a seamless way of integrating JUnit tests in the application
- **Apache Tomcat:** Tomcat is the server on which our webapp is deployed.

Spring: We have used Spring3 MVC for laying out our Web-services as well as laying out an MVC structure in the backend. We have also used spring for negotiating our content output for each of the web-services. Currently, we have configured our Spring3 MVC to output XML and JSON formats depending on the extension of the URL.

MySQL: We have used MySQL as our database mainly because it is reasonably scalable and an open-source tool. Depending on the type of prototype we are building, we concluded that MySQL is the safest and best bet with all the simple and complex features it provides.

Additionally, we are using the following APIs

- Clustering API
- Third Party Video Calling software in PHP

All the code is developed using IDEs like:

- Eclipse (Backend)
- NetBeans (Front end and server side logic)

Apart from the above mentioned tools, we have stressed a lot on the testing of our system and hence have planned to use tools like Junit for unit testing.

Chapter 4. Project Design

4.1 Client Design

4.1.1 Login

This is the first domain state that any user will fall under. The diagram below explains the login state in detail. When the user tries to login, he/she can login using their Facebook ids. They can optionally also create a Hobby Horse Id.

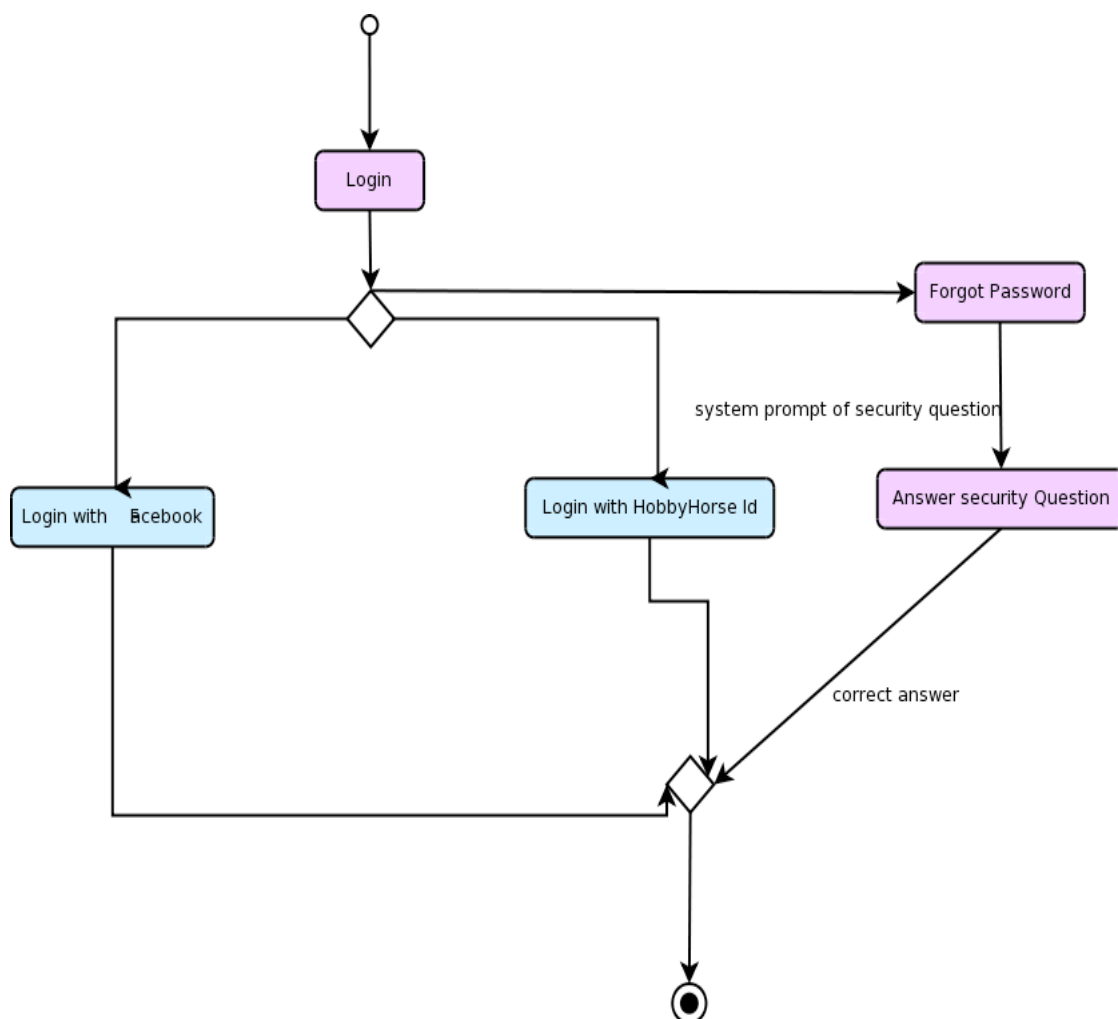


Figure 4.1: State model of the Login module.

4.1.2 Application Flow

Figure 16 is depicted with respect to the application level states. It defines the various situations the expert / novice hobbyists could be in, while using the applications of the system. It defines states considering the different users of the system.

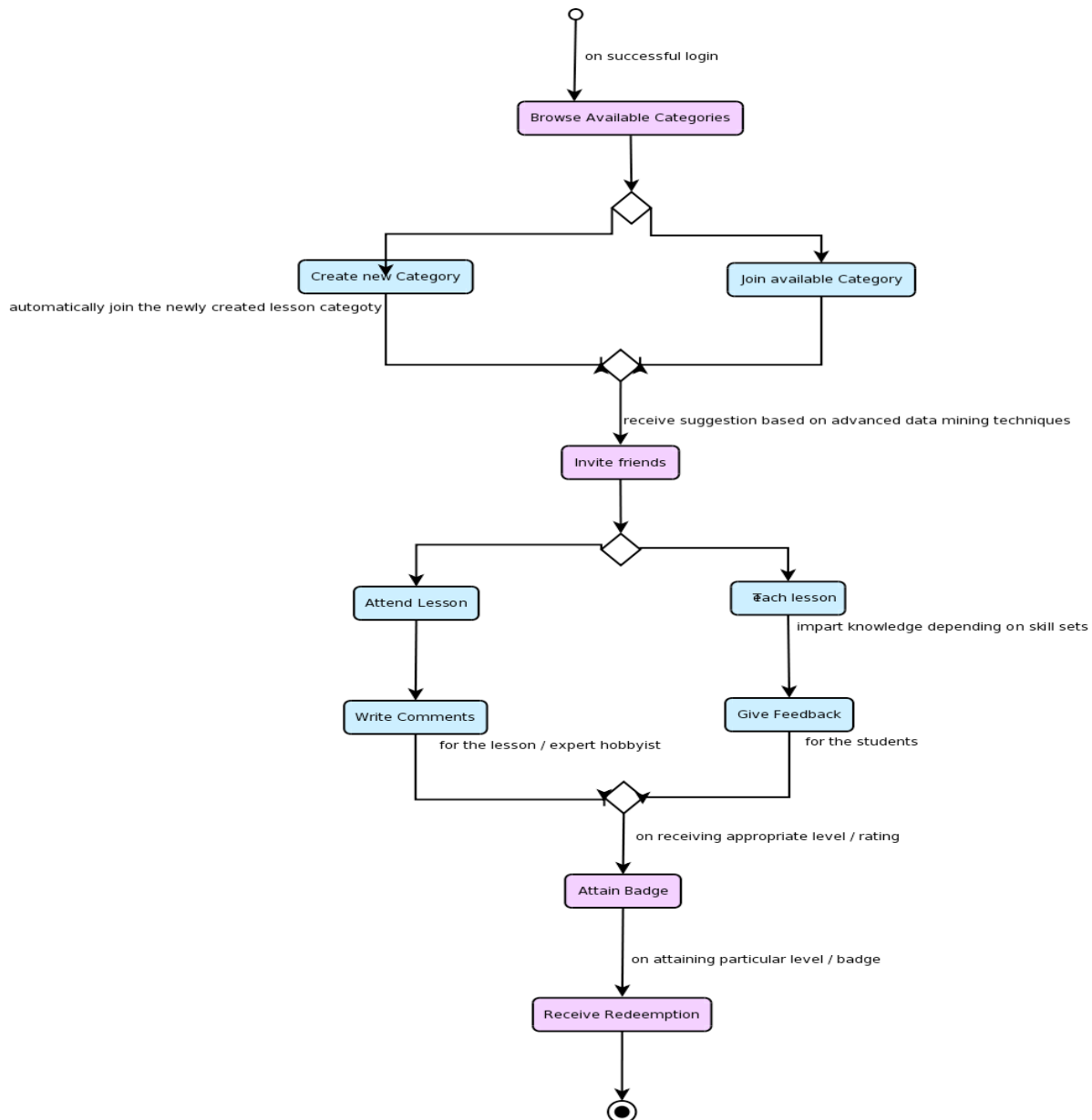


Figure 4.2: Application Flow of HobbyHorse.

Below are shown some of the use cases of the system HobbyHorse.

Section 4.1.3 uses a typical use case for login, Section 4.2.2 shows use case of a general HobbyHorse user that includes both expert hobbyist and novice hobbyist. Since we have identified two types of major users in HobbyHorse (Expert Hobbyist and Novice Hobbyist), therefore separate use cases for these users were necessary. Section 4.2.3 and 4.2.4 shows use cases for these users.

4.1.3 Use Cases

Use Case for User Login

Users of HobbyHorse can login either with their HobbyHorse credentials or using their Facebook/LinkedIn credentials. Presently, we have used only Facebook API. New users also get the option to register with HobbyHorse. Below usecase tells user when open the login page he can do either of the following things:

1. Register as a new user
2. Login using username and password.
 - a. If user logs in with HobbyHorse login credentials, then his login is verified by the HobbyHorse server and then authenticated.
 - b. If user uses Facebook credentials, the login request is routed to Facebook server.
3. User can click “Forgot password” option in case he forgets his password while logging in.

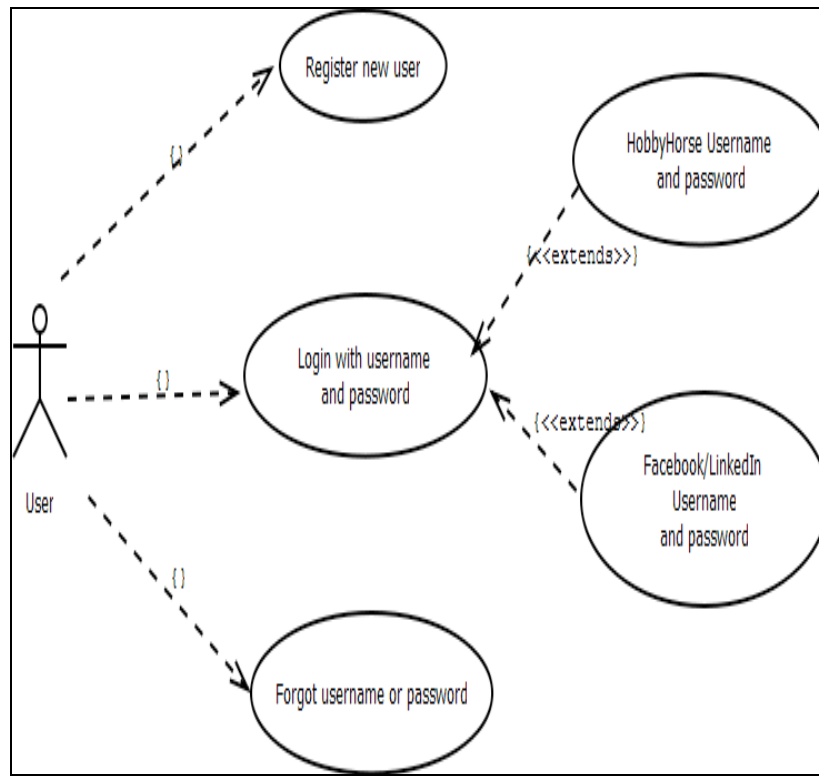


Figure 4.3: Use case for user login

Use Case for General HobbyHorse user

Any HobbyHorse user can do the following:

1. Create/Update profile: While creation of profile is done at the time of registration, user can also update his profile on certain information at any time after successful login.
2. Send Friend request: He can send to other HobbyHorse users.
3. Respond to friend request: He can either accept or reject a friend request that he has received.
4. Delete Profile: If the user does not wish to continue with HobbyHorse, he can send a delete request to the HobbyHorse server. Although practically, user details are never deleted from the server, but a flag is set in the database that determines if the user is active or not.

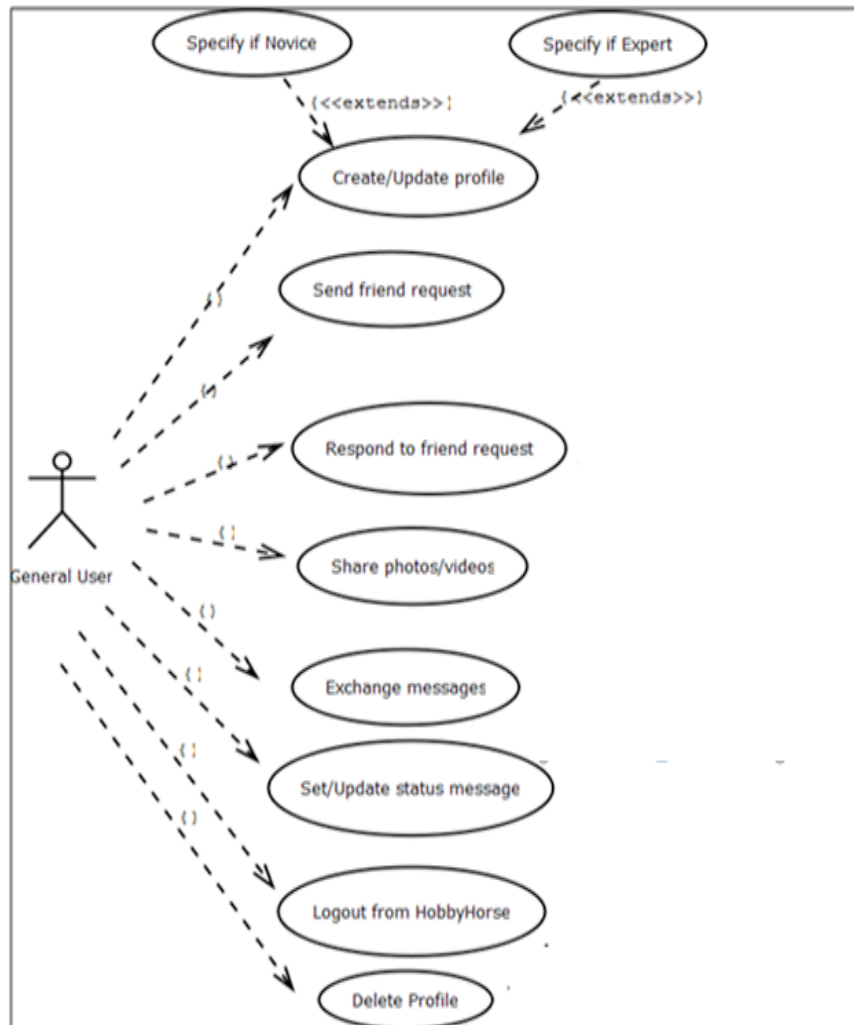


Figure 4.4: Use case for general user

Use Case for Novice Hobbyists

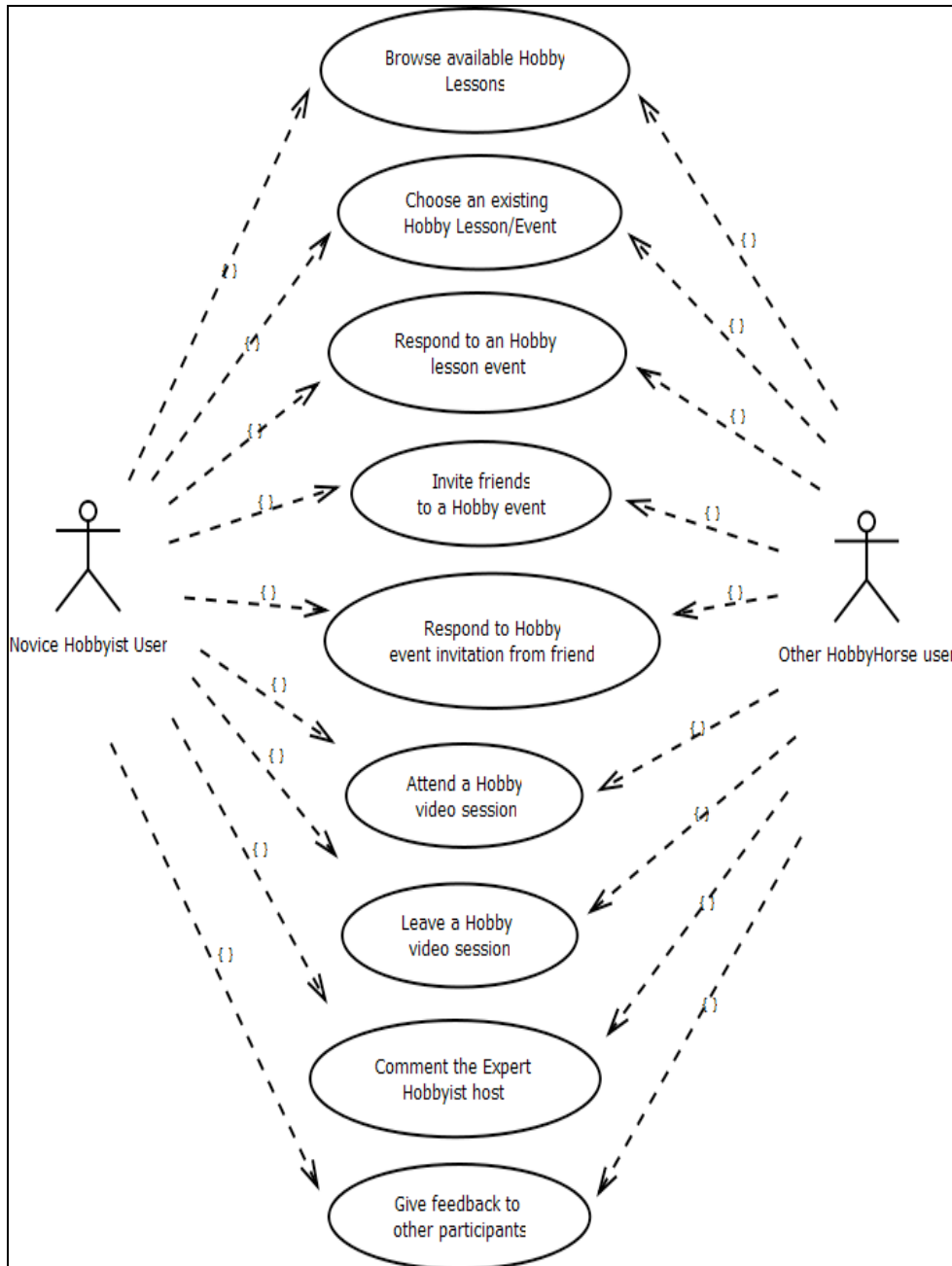


Figure 4.5: Use Case for Novice Hobbyist user.

Use case for Expert Hobbyist user

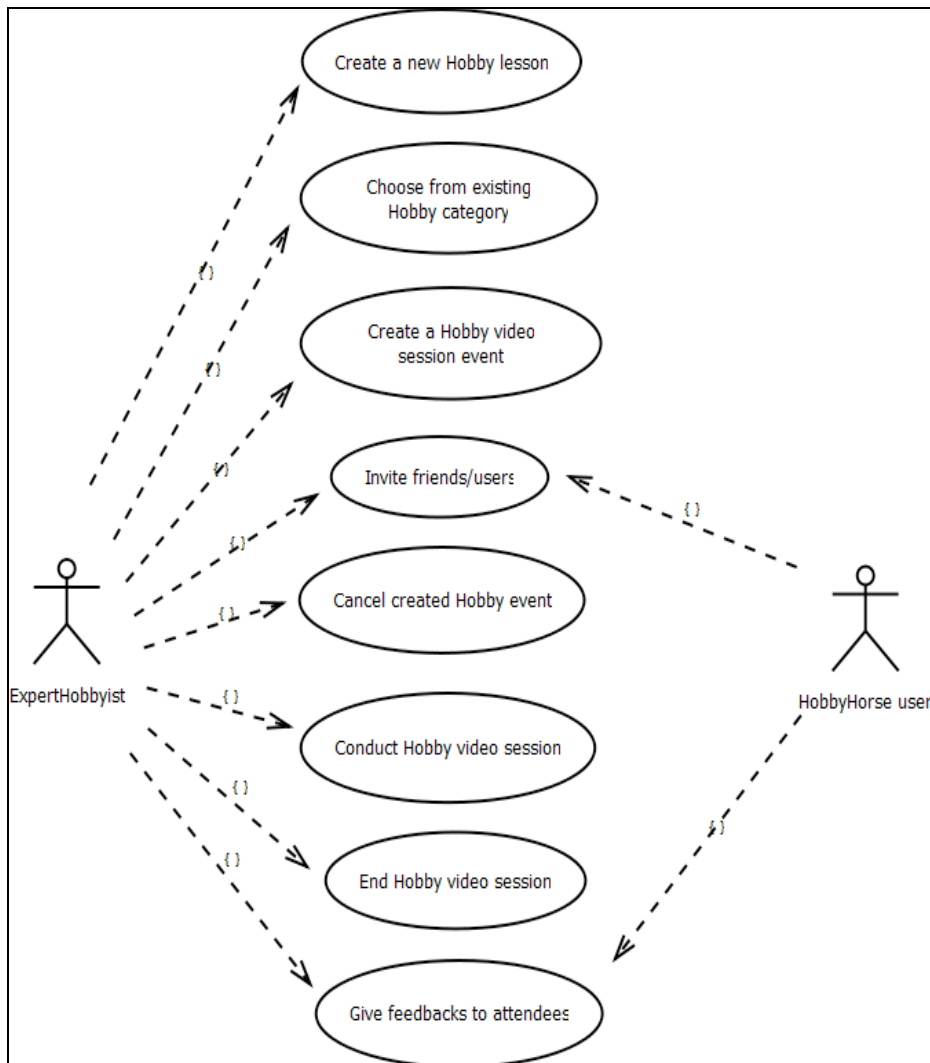


Figure 4.6: Use Case for Expert Hobbyist user.

4.2 Middle-Tier Design

4.2.1 Data Mining

This activity diagram in Figure 4.7 describes the backend domain rather than the application domain. It defines how the user-generated data will be used to implement the data mining techniques like ranking and clustering. It also depicts at what state this can be sent back to the users in the form of suggestions.

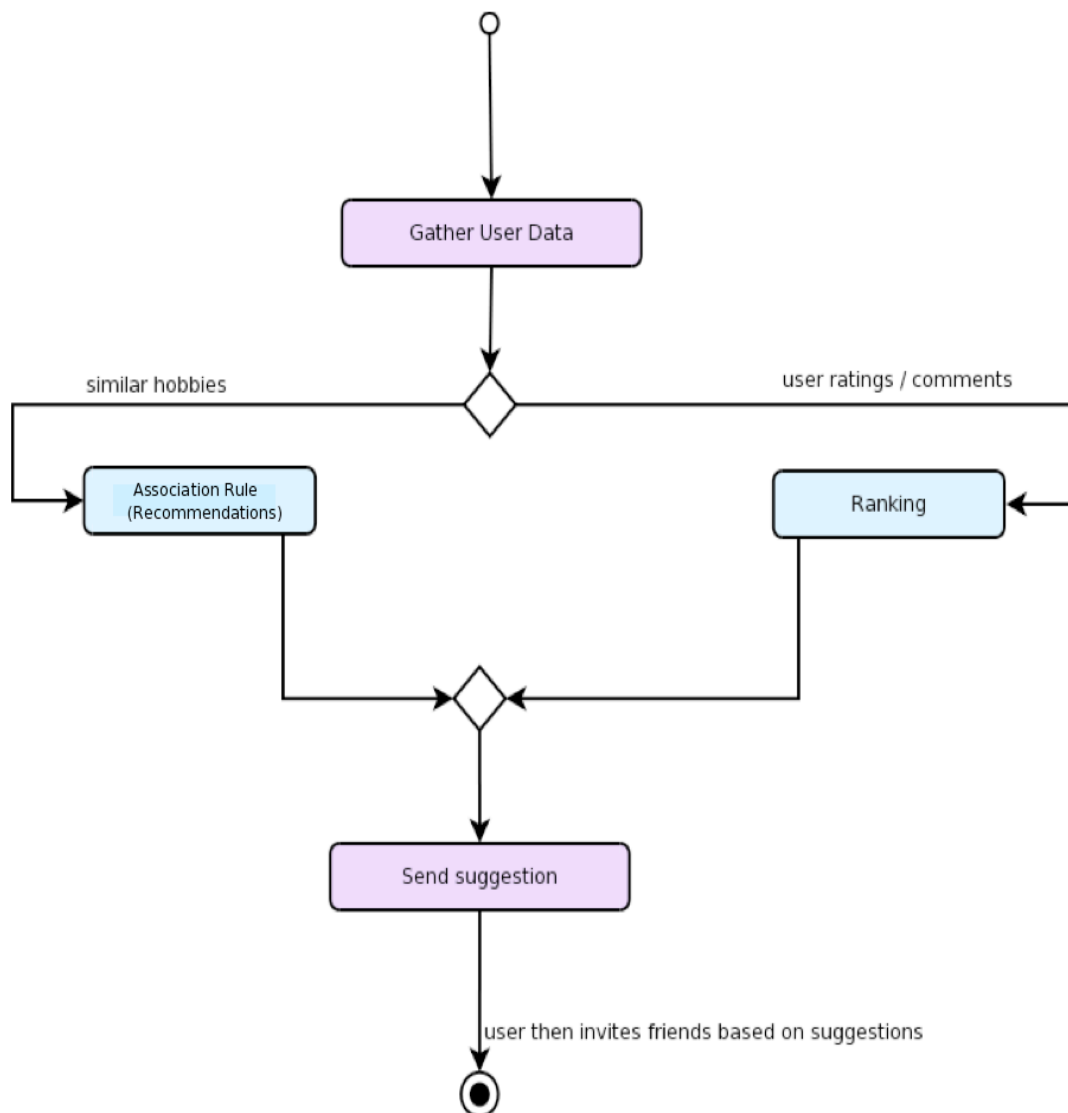


Figure 4.7: Data-mining in HobbyHorse

Detailed Middle-Tier Architecture

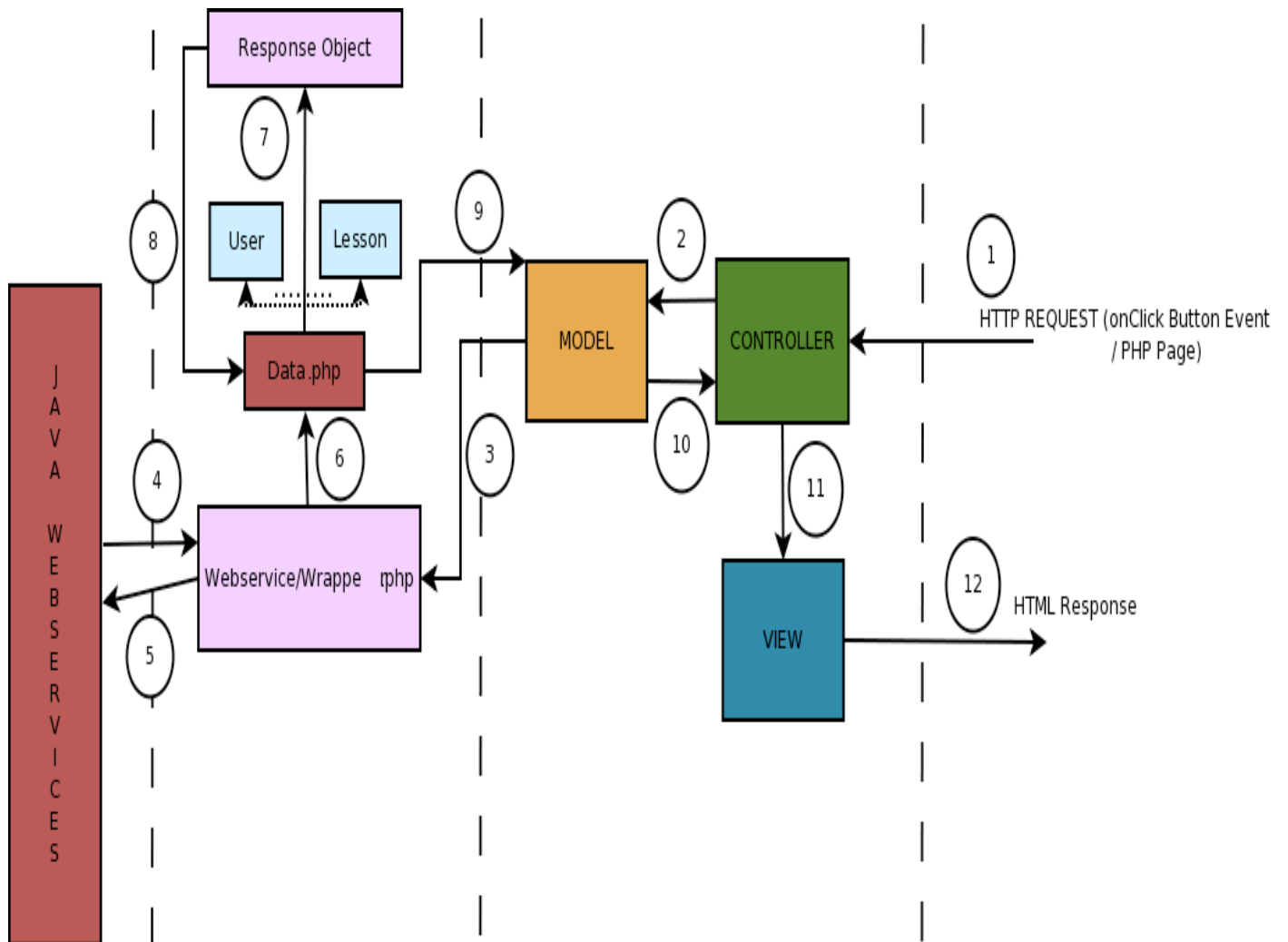


Figure 4.8: Detailed Middle Tier sequence

4.3 Data-Tier Design

4.3.1 Domain Class Diagram

The Figure 4.9 shows the domain level classes of the system. It displays the relations between the different domain classes of the system. For example, how the data-mining concept helps a user in the system, how the feedback, comments and ratings for each user are aggregated and so on. As we can see below, we have identified two main data processing techniques called Ranking and the Association Rules or Recommendations technique. Association Rules technique is based on the patterns of the users and their frequency to join a particular lesson in combination or in successions of others. However, Ranking uses domain level classes such as comments, ratings and so on. The diagram below shows how each of these domain classes are interconnected with a relation and thus forms a complete system called Hobby Horse.

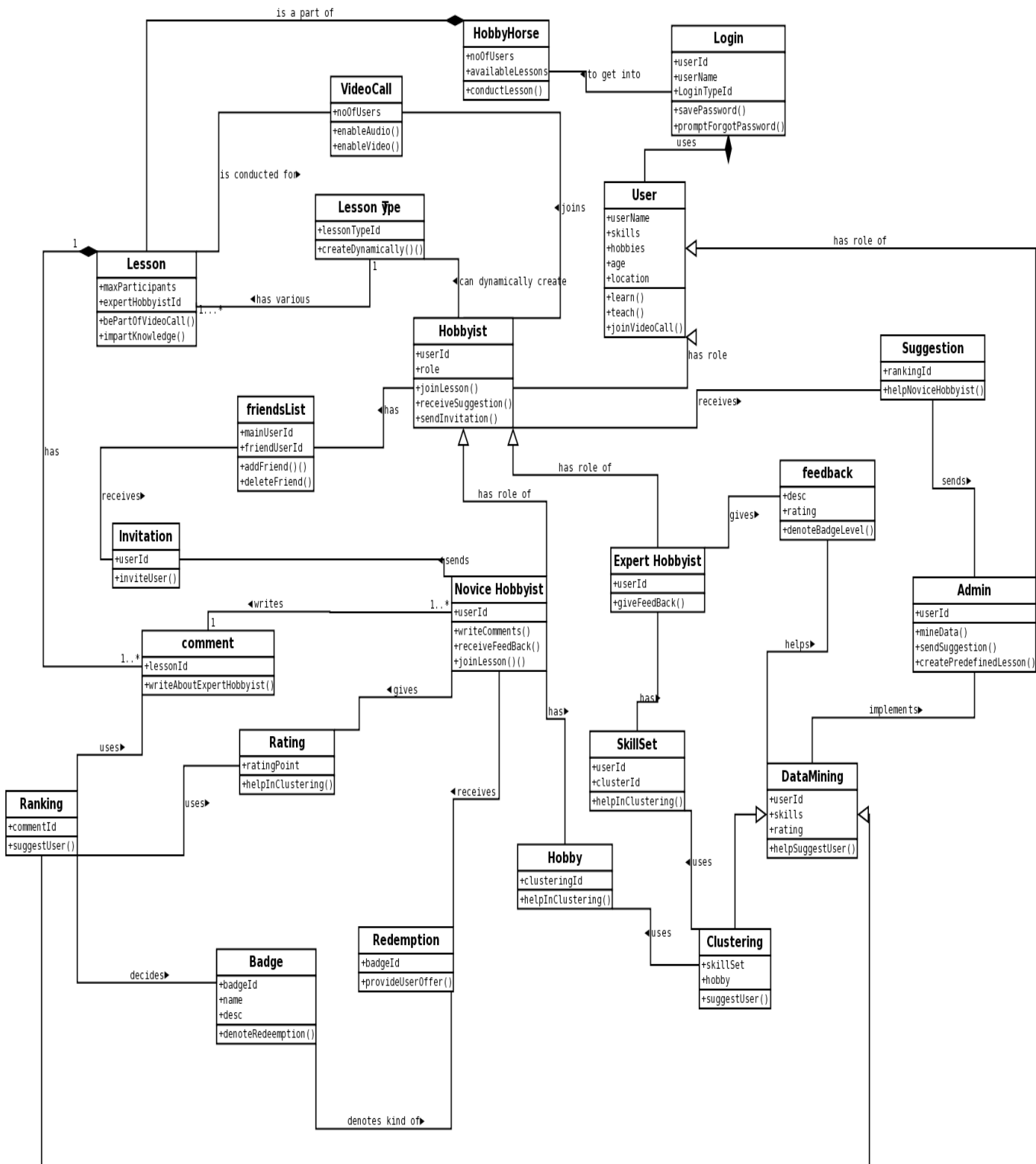


Figure 4.9: Domain Class Diagram of Hobby Horse.

Chapter 5. Project Implementation

5.1 Client Implementation

The basic layout of the HTML5 pages would be as follows:

5.1.1 HobbyHorse Login Page:

This is the basic login page of our application. The user can login through Facebook or through his / her Hobby Horse ID. If none exists or if the user does not want to use his / her Facebook login, he /she can register for one.

HobbyHorse | Learn More...

facebook
Sign up through Facebook →

Login

Username

Password

Log in [Forgot password?](#) [Register](#)

What is Hobbyhorse?

Various social networking sites have become a prominent part of life of the current generation. However, apart from the status updates and pictures of the friends across the network, there is not much that one learns from it. In view of this scenario and access of such rich and large user data, we have developed a web-based / mobile social data-mining project named Hobby Horse (Social Learning). This system is for the users who would like to impart their talents related to their hobbies in the form of teaching online (real-time) and for the users who wish to pursue a hobby of their choice at their leisure.

Users of the system can have a role of 'expert hobbyist', a 'novice hobbyist' or both. Users can participate in different learning categories, or can create one dynamically. Hobbyists will be rewarded points and ratings to define their badge-levels or skill-levels based on the feedbacks. The ratings or badge levels will be used to recommend them to each other. They can then redeem their points towards free gifts or coupons.

Lesson Categories

Social Learning

Have you ever dreamed about exploring - learning about your hobby? your interest? Have you ever wished, you could play that guitar very well and impress your friends?

It's time to stop thinking and start learning. Yes, you can learn different lessons in vast categories from the people around the globe. Explore the new way of learning. Social Learning!

Contact

- Radhika Wadegaonkar
- Sulagna Bal

radhika.wadegaonkar@gmail.com
sba2050@gmail.com

Copyright © 2012 HobbyHorse - All Rights Reserved

Figure 5.1: Login page.

5.1.2 HobbyHorse Register Page:

As discussed in previous section, the user can register for an account using the following interface:

Create account with hobbyhorse!

First Name:

Last Name:

User Name:

Email:

Security Question:

Answer:

Password:

Confirm password:

DOB:

City:

Skill Sets:

Hobbies:

Image:

Create My Account!

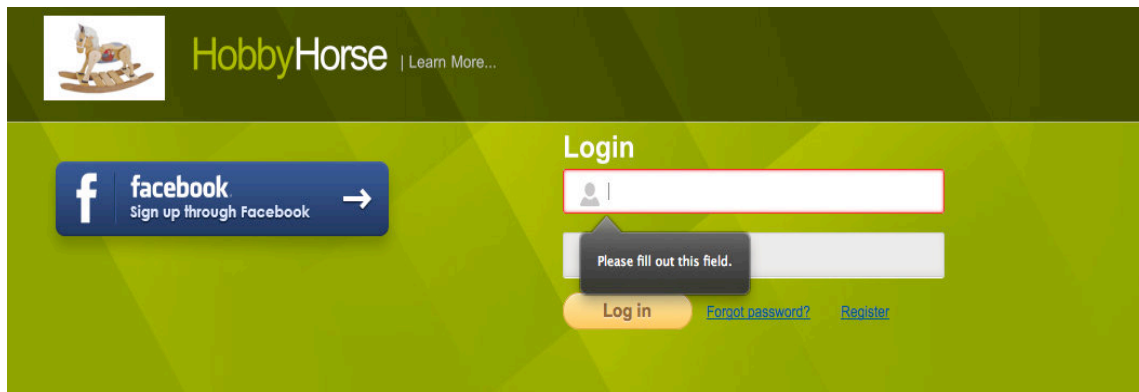
Copyright © 2012 HobbyHorse - All Rights Reserved

radhika.wadegaonkar@gmail.com
sba2050@gmail.com

Figure 5.2: Register User page.

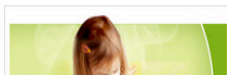
5.1.3 HobbyHorse Some Validations:

Required field Validation:



The screenshot shows the HobbyHorse login page. At the top left is the HobbyHorse logo with a rocking horse icon and the text "HobbyHorse | Learn More...". Below the logo is a Facebook sign-up button that says "facebook Sign up through Facebook" with a right-pointing arrow. To the right of this is the "Login" section. It features a username input field with a red border and a tooltip that says "Please fill out this field." Below the username field is a password input field. At the bottom of the login section are three buttons: "Log in" (orange), "Forgot password?" (blue), and "Register" (blue).

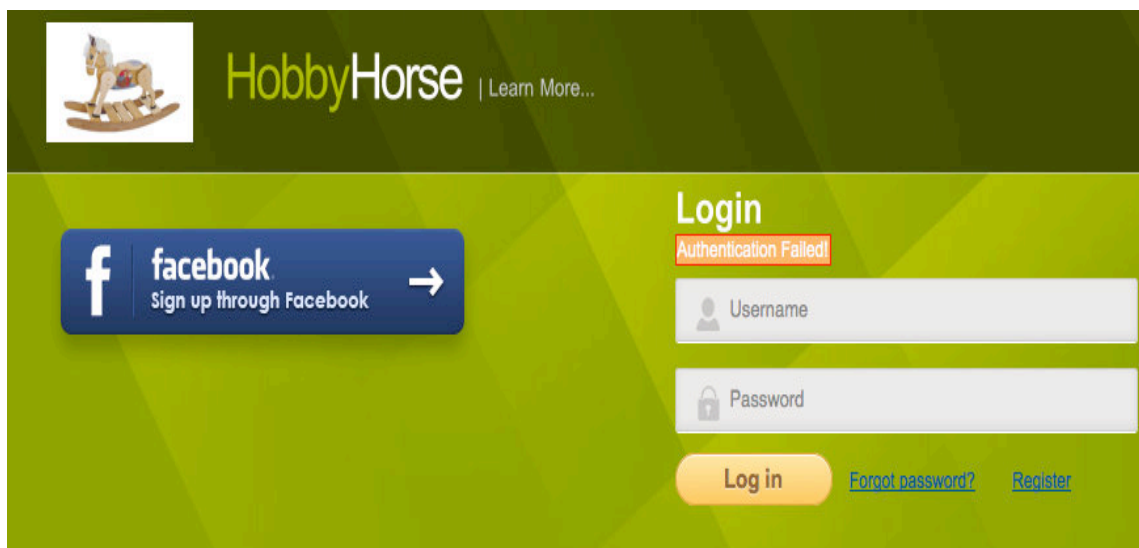
What is Hobbyhorse?



Various social networking sites have become a prominent part of life of the current generation. However, apart from the status updates and pictures of the friends across the network, there is not much that one learns from it. In view of this scenario and access of such rich and large user data, we have developed a web-based / mobile social data-mining project named Hobby Horse (Social Learning). This system is

Figure 5.3: Required field.

Login Failed:



The screenshot shows the HobbyHorse login page after a failed login attempt. The layout is identical to Figure 5.3, but with an "Authentication Failed!" error message in a red box above the username input field. The username and password fields are now labeled "Username" and "Password" respectively. The "Log in" button is orange, and the "Forgot password?" and "Register" buttons are blue.

Figure 5.4: Authentication failed.

5.1.4 HobbyHorse Welcome Page:

This is the welcome page for the user, which will have some lesson suggestions for the user. This is done using data-mining algorithms.

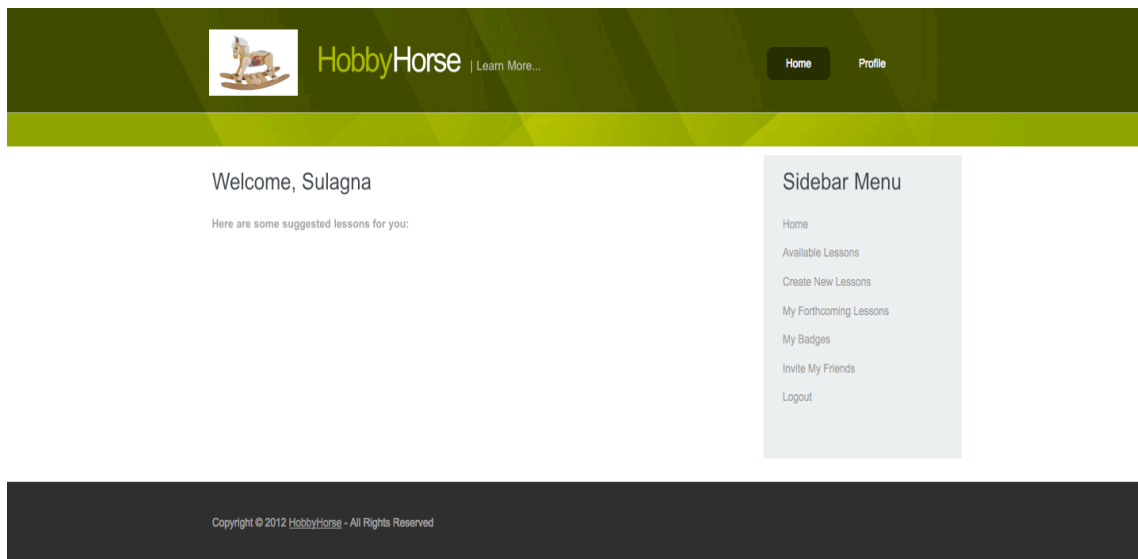
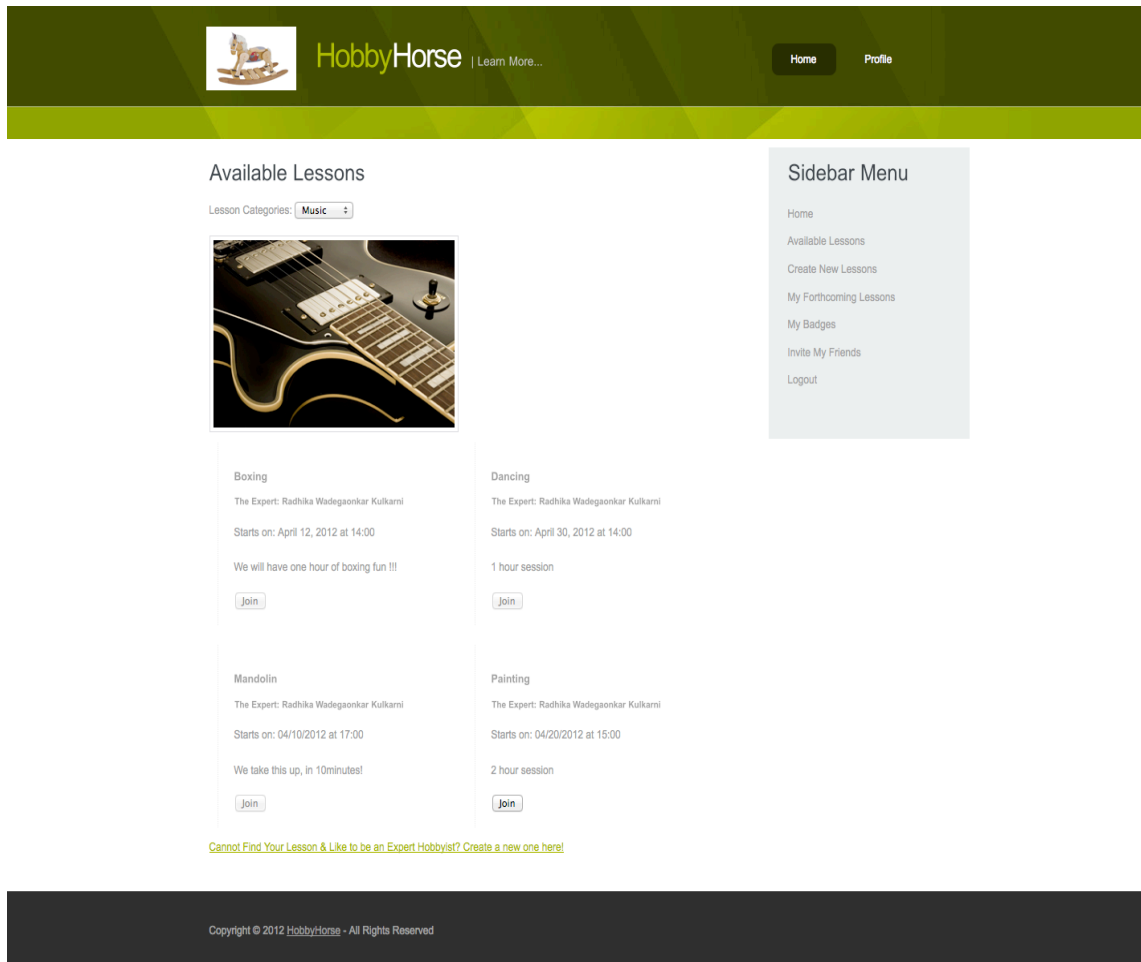


Figure 5.5: Welcome page.

5.1.5 HobbyHorse Available Lessons:

This page shows the list of available lessons that the user can join. Note that the “Join” button is disabled if the user has already joined the lesson or if the lesson has already expired.



The screenshot shows the 'Available Lessons' page on the HobbyHorse website. The header includes the HobbyHorse logo and a navigation bar with 'Home' and 'Profile' links. A sidebar menu on the right lists various options like 'Home', 'Available Lessons', 'Create New Lessons', etc. The main content area is titled 'Available Lessons' and shows a dropdown for 'Lesson Categories' set to 'Music'. Below this, there's a large image of a guitar. The lessons are listed in two columns:

Lesson Category	Expert	Starts on	Duration	Join Button
Boxing	The Expert: Radhika Wadegaonkar Kulkarni	April 12, 2012 at 14:00	1 hour session	Join
Dancing	The Expert: Radhika Wadegaonkar Kulkarni	April 30, 2012 at 14:00	1 hour session	Join
Mandolin	The Expert: Radhika Wadegaonkar Kulkarni	04/10/2012 at 17:00	2 hour session	Join
Painting	The Expert: Radhika Wadegaonkar Kulkarni	04/20/2012 at 15:00	2 hour session	Join

At the bottom, there is a link: [Cannot Find Your Lesson & Like to be an Expert Hobbyist? Create a new one here!](#)

Copyright © 2012 HobbyHorse - All Rights Reserved

Figure 5.6: Available Lessons page.

5.1.5 HobbyHorse Forthcoming Lessons:

This shows a list of available lessons, and only when the lesson (based on time) is active, the expert hobbyist and novice hobbyists can start the lesson.

HobbyHorse | Learn More... [Home](#) [Profile](#)

My Forthcoming Lessons

Lesson Name	Expert	Starts on	Duration	Action
Boxing	The Expert: Radhika Wadegaonkar Kulkarni	April 12, 2012 at 14:00	1 hour session	Start Lesson
Mandolin	The Expert: Radhika Wadegaonkar Kulkarni	04/10/2012 at 17:00	10 minutes	Start Lesson
Radhika	The Expert: Radhika Wadegaonkar Kulkarni	April 12, 2012 at 14:00	1 hour session	Start Lesson
Dancing	The Expert: Radhika Wadegaonkar Kulkarni	April 30, 2012 at 14:00	1 hour session	Start Lesson

[Cannot Find Your Lesson & Like to be an Expert Hobbyist? Create a new one here!](#)

Sidebar Menu

- [Home](#)
- [Available Lessons](#)
- [Create New Lessons](#)
- [My Forthcoming Lessons](#)
- [My Badges](#)
- [Invite My Friends](#)
- [Logout](#)

Copyright © 2012 [HobbyHorse](#) - All Rights Reserved

Figure 5.7: Forthcoming Lessons (Joined Lessons) page.

5.1.6 HobbyHorse Start a Lesson (Video):

When the user clicks the start lesson button, he / she will be asked for permission to access the web cam and microphone. Note that multiple videos will be streamed for multiple users of the same lessons. So we implement a multiple video calling lesson.

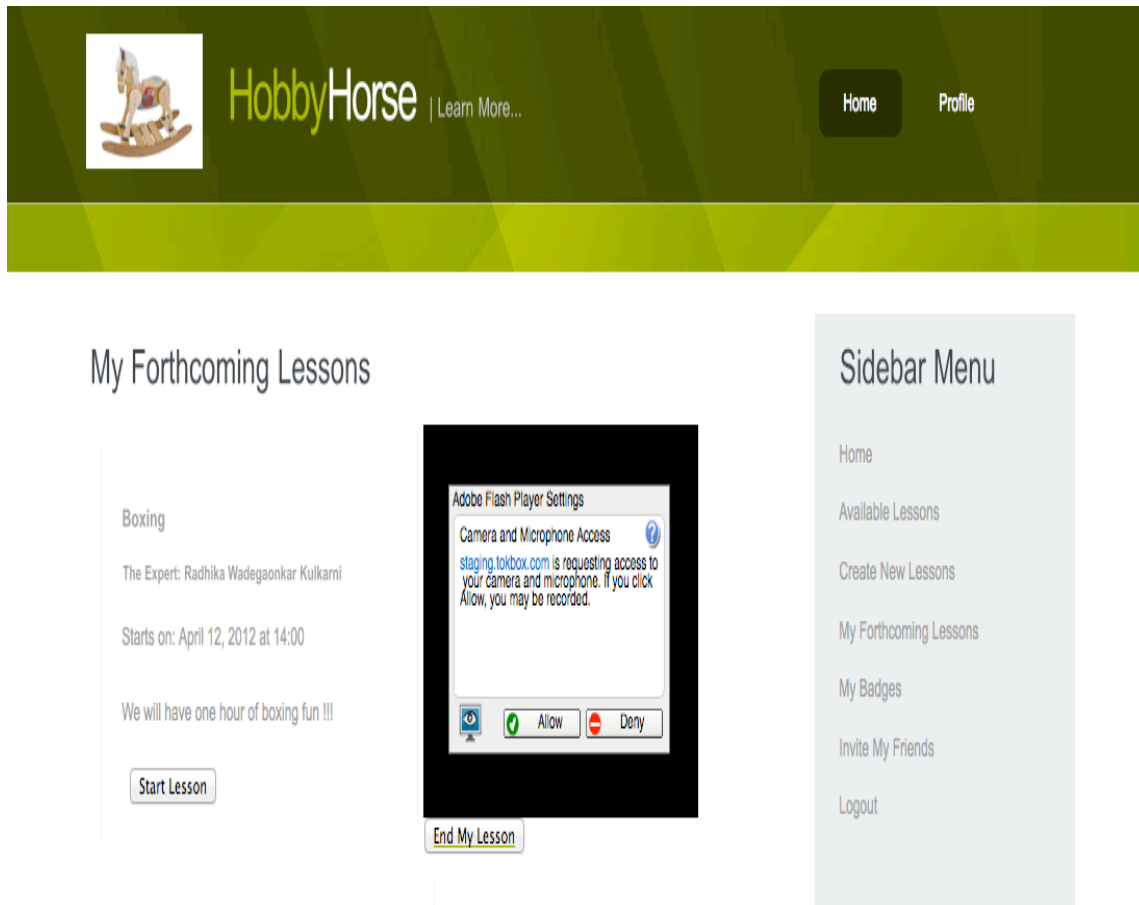


Figure 5.8: Start Lessons page.

5.2 Middle-Tier Implementation

As described in the previous sections, the middle layer uses the technologies such as PHP with its frameworks such as Zend and CodeIgnitor. Zend is used to fetch the web-services using the HTTP Client, whereas, Code-Ignitor lays a complete MVC structure in the middle-tier. The middle-tier thus co-ordinates with the backend web-services (Zend HTTP Client), wraps the objects depending on the response and sends it to the Client (HTML page). The folder structure of the middle-tier is as follows:

At the root level:

```
Rwadegaonkar:hobbyhorse$ ls -l
total 24
drwxr-xr-x  5 sulagnabal admin 170 Apr  7 18:35 Platform
drwxr-xr-x 108 sulagnabal admin 3672 Apr  7 19:22 Zend
drwxr-xr-x 17 sulagnabal admin  578 Apr  7 19:22 application
-rw-r--r--  1 sulagnabal admin 6491 Apr  3 00:00 index.php
-rw-r--r--  1 sulagnabal admin 2496 Nov 22 09:14 license.txt
drwxr-xr-x 10 sulagnabal admin  340 Apr  7 19:22 system
```

Platform: This is the main folder that interacts with the data-tier through the web-services, consumes the response, converts it to the desired output and sends back to the HTML page. The internal structure for this folder:

```
rwadegaonkar:Platform$ ls -la
total 8
drwxr-xr-x  5 sulagnabal admin 170 Apr  7 18:35 .
drwxr-xr-x 11 sulagnabal admin 374 Apr  7 19:22 ..
drwxr-xr-x  5 sulagnabal admin 170 Apr  7 18:56 Data
-rw-r--r--  1 sulagnabal admin 443 Apr  7 18:35 Data.php
drwxr-xr-x  3 sulagnabal admin 102 Apr  7 19:05 Webservices
```

The Webservices folder consists of the main Wrapper.php class, that makes use of the Zend HTTP Client to request the web-service and returns the expected JSON object response. Before this, it sets the CRUD HTTP parameters using the Zend HTTP Client.

The Data folder consists of all the response objects such as Lesson.php, Lessons.php that have their respective getters and setters defined in them.

The Data.php file just parses the incoming object into an array. These files are used by the model classes.

Zend: This is the library folder for the Zend framework.

application: This consists of the controllers, models and views for the project. Views incorporate JavaScript, AJAX and CSS along with HTML.

```
rwadegaonkar:application$ ls -l
total 16
drwxr-xr-x 17 sulagnabal admin 578 Apr 7 19:22 .
drwxr-xr-x 11 sulagnabal admin 374 Apr 7 19:22 ..
-rw-r--r-- 1 sulagnabal admin 13 Nov 22 08:52 .htaccess
drwxr-xr-x 4 sulagnabal admin 136 Apr 7 19:22 cache
drwxr-xr-x 16 sulagnabal admin 544 Apr 7 19:22 config
drwxr-xr-x 5 sulagnabal admin 170 Apr 7 19:22 controllers
drwxr-xr-x 3 sulagnabal admin 102 Nov 22 08:52 core
drwxr-xr-x 7 sulagnabal admin 238 Apr 7 19:22 errors
drwxr-xr-x 3 sulagnabal admin 102 Nov 22 08:52 helpers
drwxr-xr-x 3 sulagnabal admin 102 Nov 22 08:52 hooks
-rw-r--r-- 1 sulagnabal admin 114 Nov 22 08:52 index.html
drwxr-xr-x 3 sulagnabal admin 102 Nov 22 08:52 language
drwxr-xr-x 3 sulagnabal admin 102 Nov 22 08:52 libraries
drwxr-xr-x 3 sulagnabal admin 102 Nov 22 08:52 logs
drwxr-xr-x 4 sulagnabal admin 136 Apr 7 19:22 models
drwxr-xr-x 3 sulagnabal admin 102 Nov 22 08:52 third_party
```

drwxr-xr-x 4 sulagnabal admin 136 Apr 7 19:22 views

The main folders are explained below. The rest of the folders are libraries required by Code-Ignitor.

controllers: The controller classes, just load the model classes and the required functions in them.

models: The model classes call the Wrapper classes (defined earlier) to make the actual HTTP request, and return the response back to the controller.

views: This has all the html content PHP files and it just parses the response obtained from the controller class

config: The main file in this folder is the routes.php, which has all the definitions for the flow of the controller->view mapping.

system: The system folder consists of all the library classes needed for Code-Ignitor.

We will discuss the same example as that of displaying lessons for the users to browse from. For the same screenshot as shown in the client implementation, the following code is implemented in the middle layer:

The Lesson Controller calls the Lesson Model Class:

`/** applications/controllers/Lesson.php**/s`

```
class Lesson extends CI_Controller
{
    /**
     * constructor of Lesson controller class
     */

    public function __construct()
    {
        parent::__construct();
        $this->load->model(Lesson_Model,'lesson_model');
    }

    public function index()
    {
        $data = $this->user_model->getLessons();
        echo "<pre>";
        print_r($data->lessons);
    }
}
```

This controller, as we can see, in its default function calls the model function of loading all the available lessons. The model is as below:

```
/** applications/models/Lesson_Model.php */

require_once('Platform/Webservices/Wrapper.php');
require_once('Platform/Data.php');

class User_Model extends CI_Model
{
    public function __construct()
    {
        parent::__construct();
    }

    public function getLessons()
    {
        $myWrapper = new Platform_Webservices_Wrapper();
        $jsonObj = $myWrapper->request('lessons');
        return Platform_Data::getDataObject($jsonObj);
    }
}
```

The model class has a function “*getLessons()*” which calls the Platform libraries that we have defined. It calls the Wrapper class, which makes a call to the Web-service using Zend libraries of HTTP Client. The response is then sent to the Data.php file, which is as below:

```
<?php
/** Platform/Data.php */
require_once('Platform/Data/Lessons.php');
require_once('Platform/Data/Users.php');

/**
 *
 */
Class Platform_Data {

    /**
     * @return
     */
    protected $_data = array();

    public function __construct($data)
    {
        /** Checks which object has been set in the response */
        if(isset($data->users))
        {
```

```

        $this->_data[] = new Platform_Data_Users($data->users);
    }
    /** Accordingly sets the response object in the respective object type */
    else if(isset($data->lessons))
    {
        $this->_data[] = new Platform_Data_Lessons($data->lessons);
    }
    /** Code omitted for brevity */
    .
    .
    .
    .
    else
    {
        throw new Exception("Error: Wrong response");
    }
}

/**
 * @param Zend_Http_Response $response
 * @return json string
 * @throws
 */

    /** Some exception handling */
    public static function getDataObject($data = null)
    {
        if(!is_object($data))
            throw new Exception("Error: data - {$data} is not the object");

        return new self($data);
    }

    /** Customizing the PHP magic functions to return the correct getter and setter
    functions*/
    public function __get($name)
    {
        $objectName = "Platform_Data_". ucfirst($name);
        foreach($this->_data as $data)
        {
            if(is_object($data))
            {
                if(get_class($data) == $objectName)
                {
                    return $data;
                }
            }
        }
    }

    public function __set($name,$val)
    {
        $this->_data[$name] = $val;
    }

```

```
}
```

The response object in this case “Lessons” is set and passed back to the controller and sent to the View as defined in the routes.php file. It then loads the HTML content page as seen in the previous section.

5.3 Data-Tier Implementation

This is layer where our data persistence and retrieval occur. As mentioned in our previous sections, we have used MySQL for our data storage purpose. In MySQL we have created a database called “*hobbyhorse*” using the following SQL command:

```
CREATE DATABASE hobbyhorse;
```

Inside hobbyhorse database we have created following 14 tables. Few of them are as follows:

- Table “*user*” : This stores all the user details that user provide while registering with HobbyHorse. The field loginTypeId tells us if the user logs in with Facebook credentials or HobbyHorse credentials
- Table “*lessonType*”: This lists all the categories of lessons, for example, Arts, Music etc.
- Table “*lesson*” : All the newly created lessons are stored here. It has parameters like sessionId that is used when a lesson session is initiated by the user(expert user). User who creates a lesson is always assumed as expert hobbyists. In other words only expert hobbyists are able to create lessons. Users who join the lessons are novice hobbyists. Therefore user role is only defined when a lesson created or a user joins the lesson.
- Table “*role*”: Lists the roles of user by userId.
- Table “*participants*”: Lists the participants associated with a particular lesson.

Below shown is a SQL create statement for “*user*” table:

```
CREATE TABLE `user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) COLLATE utf8_bin NOT NULL,
  `description` text COLLATE utf8_bin,
  `isDeleted` tinyint(1) NOT NULL,
  `createdBy` varchar(255) COLLATE utf8_bin NOT NULL,
  `lastUpdatedBy` varchar(255) COLLATE utf8_bin NOT NULL,
```



```

`createDate` datetime NOT NULL,
`lastUpdateDate` datetime NOT NULL,
`username` varchar(255) COLLATE utf8_bin NOT NULL,
`password` varchar(255) COLLATE utf8_bin NOT NULL DEFAULT 'admin',
`email` varchar(255) COLLATE utf8_bin DEFAULT NULL,
`skills` text COLLATE utf8_bin,
`hobbies` text COLLATE utf8_bin,
`location` varchar(255) COLLATE utf8_bin DEFAULT NULL,
`loginTypeId` int(11) NOT NULL,
PRIMARY KEY (`id`),
UNIQUE KEY `id` (`id`),
KEY `id_2` (`id`),
KEY `loginTypeId` (`loginTypeId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin
AUTO_INCREMENT=1 ;

```

Similar CREATE sql statements are used for creating other tables.

We have POJO classes to represent these tables in our code. Thus these POJO classes acts as the Data Transfer Objects(DTOs). These classes can be found in the package `com.spring.datasource`. Below shown is an example of such a class:

```

package com.spring.datasource;
import java.util.Date;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name = "user")
public class User {
    private long id;
    private String name;
    private String description;
    private long isDeleted;
    private String createdBy;
    private String lastUpdatedBy;
    private Date createDate;
    private Date lastUpdateDate;
    private String username;
    private String userpassword;
    private String email;
    private String skills;
    private String hobbies;
    private String location;
    private long loginTypeId;

    public long getId() {
        return id;
    }

    @XmlElement
    public void setId(long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

```

```

    }
    @XmlElement
    public void setName(String name) {
        this.name = name;
    }
    public String getDescription() {
        return description;
    }
    @XmlElement
    public void setDescription(String description) {
        this.description = description;
    }
    public long getIsDeleted() {
        return isDeleted;
    }
    @XmlElement
    public void setIsDeleted(long isDeleted) {
        this.isDeleted = isDeleted;
    }
    public String getCreatedBy() {
        return createdBy;
    }
    @XmlElement
    public void setCreatedBy(String createdBy) {
        this.createdBy = createdBy;
    }
    public String getLastUpdatedBy() {
        return lastUpdatedBy;
    }
    @XmlElement
    public void setLastUpdatedBy(String lastUpdatedBy) {
        this.lastUpdatedBy = lastUpdatedBy;
    }
    public Date getCreateDate() {
        return createDate;
    }
    @XmlElement
    public void setCreateDate(Date createDate) {
        this.createDate = createDate;
    }
    public Date getLastUpdateDate() {
        return lastUpdateDate;
    }
    @XmlElement
    public void setLastUpdateDate(Date lastUpdateDate) {
        this.lastUpdateDate = lastUpdateDate;
    }
    public String getUsername() {
        return username;
    }
}

```

```
@XmlElement
public void setUsername(String username) {
    this.username = username;
}

public String getEmail() {
    return email;
}

@XmlElement
public void setEmail(String email) {
    this.email = email;
}

public String getSkills() {
    return skills;
}

@XmlElement
public void setSkills(String skills) {
    this.skills = skills;
}

public String getHobbies() {
    return hobbies;
}

@XmlElement
public void setHobbies(String hobbies) {
    this.hobbies = hobbies;
}

public String getLocation() {
    return location;
}

@XmlElement
public void setLocation(String location) {
    this.location = location;
}

public long getLoginTypeId() {
    return loginTypeId;
}

@XmlElement
public void setLoginTypeId(long loginTypeId) {
    this.loginTypeId = loginTypeId;
}

public String getUserpassword() {
    return userpassword;
}

public void setUserpassword(String userpassword) {
    this.userpassword = userpassword;
}
```

```
}
```

Then we have the DAOImplementations classes to implement our DAO interface. These are the Model classes that do the data exchange between the business layer and the data layer. These classes are written inside the package com.spring.dao. A typical DAO implemented class is as shown below:

```
package com.spring.dao;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

import com.spring.datasource.User;
import com.spring.manager.UserRowMapper;
import com.spring.util.Query;

public class UserDao {
    private static UserRowMapper rowMapper = new UserRowMapper();
    static DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
    static Date date = new Date();
    private static final String CURRENT_TIMESTAMP = dateFormat.format(date);
    private static final String SELECT_ALL = "SELECT * FROM user WHERE
isDeleted=0";
    private static final String SELECT_BY_USERNAME = "SELECT * FROM user WHERE
isDeleted=0 and username=";
    private static final String SELECT_BY_USERNAME_AND_PASSWORD = "SELECT *
FROM user WHERE isDeleted=0,username=";
    private static final String SELECT_BY_ID = "SELECT * FROM user WHERE
isDeleted=0 and id=";
    private static final String DELETE_BY_USERNAME = "UPDATE user SET isDeleted=1
where username=";

    private static final String INSERT_USER = "INSERT INTO user(name, description,
isDeleted, createdBy, lastUpdatedBy, createDate, lastUpdateDate, username, password,
email, skills, hobbies, location, loginTypeId) VALUES";

    public Query query = new Query();
    ArrayList<User> users = new ArrayList<User>();

    public ArrayList<User> getAllUsers(Connection conn) {
        ResultSet rs = query.executeQuery(SELECT_ALL, conn);
        try {
            users = rowMapper.convertUserBean(rs);
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return users;
    }
}
```

```

    }

    public ArrayList<User> getUserByUsername(Connection conn, String username) {
        ResultSet rs = query.executeQuery(SELECT_BY_USERNAME + "" +
username
        + "", conn);

        try {
            users = rowMapper.convertUserBean(rs);
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return users;
    }

    public ArrayList<User> getUserById(Connection conn, long id) {
        System.out.println("In DAO class " + id);
        ResultSet rs = query.executeQuery(SELECT_BY_ID + id, conn);
        return users;
    }

    public ArrayList<User> getUserByUsernameAndPassword(Connection conn, String
username, String password) {
        ResultSet rs =
query.executeQuery(SELECT_BY_USERNAME_AND_PASSWORD + "" + username + "" + " and
password=" + "" + password + "", conn);
        try {
            users = rowMapper.convertUserBean(rs);
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return users;
    }

    public void deleteUser(Connection conn, String username) {
        query.executeUpdate(DELETE_BY_USERNAME + "" + username + "",
conn);
    }

    public ArrayList<User> saveUser(Connection conn, User user) {
        String insertQuery = INSERT_USER + "(" + user.getName() + "," +
        + user.getDescription() + "," + user.getIsDeleted() + "," +
        + user.getCreatedBy() + "," + user.getLastUpdatedBy() +
        + CURRENT_TIMESTAMP + "," + CURRENT_TIMESTAMP
        + user.getUsername() + "," + user.getUserpassword() +
        + user.getEmail() + "," + user.getSkills() + "," +
        + user.getHobbies() + "," + user.getLocation() + "," +
user.getLoginTypeId() + ")";

        query.executeUpdate(insertQuery, conn);
        return users;
    }

```

```

    }
}

```

The method “*saveUser(..)*” inserts user data in the “*user*” table. Similarly, “*deleteUser(..)*” deletes a user from the user table. In reality when this method is called, user details are not actually deleted from the table; instead a flag called “*isDeleted*” is set to 1 in the “*user*” table. Method “*getUserByUsernameAndPasword(..)*” is called during user login authentication.

Additional Project Implementation Features

Apart from the above mentioned technical implementations, we would like highlight a few of our other features.

5.3.1 Video Calling:

The implementation of “Multiple Video Calling” is the technical highlight of our project, after Data Mining. When the expert hobbyist starts the lesson, multiple novice hobbyists can start the same lesson and be active in the same sessions. This video calling has enabled voice and webcam interfacing.

When the video starts, the user is first asked for permissions to access his / her webcam and microphone and only after the user agrees, the session starts. Thus, the question of interfering into one’s privacy or security is taken care of. The user can end the lesson whenever he/she wants and need not wait till the end of the lesson.

Currently, we have decided to have a limit of 3 novice hobbyists for the reasons of simplicity. The user can start a lesson, only if he / she have enrolled (joined) previously. The user cannot enroll into an ongoing lesson.

We have used an open-source library called “OpenTok” [23] for the simple implementation of multiple video calling. This library has multiple video calling implementations in various languages like Java, JavaScript, PHP and Ruby. Since our client-side implementation is completely done using PHP, it was quite seamless for us to implement and add this functionality using the existing internal and external libraries.

The technical aspects of this feature are that multiple users can join the same session provided they have the same sessionId. This sessionId is randomly generated by the opentok libraries when we call the “`create_session()`” function. To keep it simple, we have generated a sessionId in this way, each time a lesson is created. This sessionId is then stored in the database and then used for the lesson whenever it needs to be started. The code for the same is found in the view “`lesson/create`”:

```
/** Include the required libraries**/
require_once 'Platform/opentok/API_Config.php';
require_once 'Platform/opentok/OpenTokSDK.php';

$apiObj = new OpenTokSDK(API_Config::API_KEY, API_Config::API_SECRET);

/** Create the sessionId **/
$session = $apiObj->create_session($_SERVER["REMOTE_ADDR"]);
$sessionId = $session->getSessionId();
```

Now, when the lesson needs to start, the code to start the Video calling is done in JavaScript and can be found in the view “`lesson/main`”:

There is a button that has the event to start the video lesson:

```
<input type="button" value="Start Lesson" onclick="startLesson('<?php echo $l-
>sessionId ?>');showEndLessonButton();"/>
```

The onClick event of the above button fires the following JavaScript function. This function accepts the sessionId as the parameter to maintain the uniqueness of each lesson session. This sessionId is fetched from the database, that is stored on a new lesson creation.

```
<script type="text/javascript">
function showEndLessonButton() {
    document.getElementById('endLesson').innerHTML = "<a href='<?php echo
base_url()?>index.php/lesson/joinedLesson'><input type='button' name='closeLesson' value='End
My Lesson' /></a>";
}
function startLesson(sessionId) {
    var apiKey = '14098051';
```

```

    //var sessionId =
    '2_MX41ODc1MjMxfjEyNy4wLjAuMX4yMDEyLTA0LTEyIDA1OjAxOjA3LjYyMTM4MCSwMDowMH4
    wLjc3OTU1NTAzMjQxOX4';
    var token = 'devtoken';

    TB.setLogLevel(TB.DEBUG);

    var session = TB.initSession(sessionId);
    session.addEventListener('sessionConnected', sessionConnectedHandler);
    session.addEventListener('streamCreated', streamCreatedHandler);
    session.connect(apiKey, token);

    var publisher;

    function sessionConnectedHandler(event) {
        publisher = session.publish('myPublisherDiv');

        // Subscribe to streams that were in the session when we connected
        subscribeToStreams(event.streams);
    }

    function streamCreatedHandler(event) {
        // Subscribe to any new streams that are created
        subscribeToStreams(event.streams);
    }

    function subscribeToStreams(streams) {
        for (var i = 0; i < streams.length; i++) {
            // Make sure we don't subscribe to ourself
            if (streams[i].connection.connectionId == session.connection.connectionId) {
                return;
            }

            // Create the div to put the subscriber element in to
            var div = document.createElement('div');
            div.setAttribute('id', 'stream' + streams[i].streamId);
            document.body.appendChild(div);

            // Subscribe to the stream
            session.subscribe(streams[i], div.id);
        }
    }
}
</script>

```

5.3.2 Badges:

As mentioned earlier, each user can join multiple lessons and can either be an expert hobbyist or a novice hobbyist in that lesson. The novice hobbyists will be given a certain level, or what we call here “Badges” depending upon the number of lessons he completed. For example, a novice hobbyists who has completed a minimum of ten lessons of any lesstypewould receive a Badge of a “Foal” (meaning a newborn

horse). When he/she completes 20 lessons, he/she earns the badge of “Weanling”(just little older). The next badge level he/she would receive is “Yearling” when he/she completes 35 or more lessons. After he/she completes 50 lessons he/she earns the badge of “Stallion”(if male) or “Mare”(if female). ”. Thus the assignment of these badges will be on certain pre-defined rules in our system. Users can use these badges to earn points that he can reimburse for exciting gifts for example, discount coupons for an online purchase or at some coffee shop, or a free movie ticket, or a discounted ticket to some concert. The points can be exchanged for a gift that can come from a variety of choices users wish to pick. While badge-level assignment and points earnings have been actualized, reimbursement of the points for a gift is yet to be implemented. This later part remains as a part of our future enhancement. This will add extra implementation and interfacing of external third parties that provide such coupons to us. .

This feature is one of our marketing highlights that would keep the users hooked up to our system. Users would definitely want to come back in desire of acquiring some free gifts or discount coupons that he can use for his personal purpose. For example, one who is passionate about learning Guitar would be definitely excited to receive a discounted coupon for buying a guitar of his choice.

Pseudo Code for the Badge-level assignment:

The joinLesson(..) method is invoked when user wishes to join a lesson. The method first gets the lesson details from the database and then checks if the lesson has expired or not. If the lesson exists and has not expired, it adds the user to the lesson session and also updates the user lesson bucket. User’s lesson bucket keeps the count of the lessons the user has attended.

```
joinLesson(userId, lessonId)
```

```
    lessonObj = getLessonDetailsFromDatabase(lessonId)
```

```
    if(lessonObj.lesson exists && lessonObj.lesson not expired)
```

```
        addUserDetailsToLessonSession(userId)
```

```
        addLessonToUserLessonBucket(lessonId,userId)
```

```
    else
```

```
return errorMessage
```

The below method will periodically check how many lessons a user has attended. Based on the number, it will assign badge-level and points to the user.

```
checkUserLessonBucket(userId)

    int lessonCount = getLessonCount(userId)

    if(lessonCount>=10 && lessonCount<20)

        assignBadgeLevelAndPoints("Foal",20,userId)

    else if(lessonCount>=20 && lessonCount<35)

        assignBadgeLevelAndPoints("Weanling",30,userId)

    else if(lessonCount>=35 && lessonCount<50)

        assignBadgeLevelAndPoints("Yearling",40,userId)

    else if(lessonCount>=50)

        assignBadgeLevelAndPoints("Stallion",50,userId)
```

This method is for future implementation. This would require collaboration with other APIs that would provide the gift coupons. The recommendation of gifts would involve some mining logic also.

```
checkUserPoints(userId)

    int Points = getTotalPoints(userId)

    if(Points>=50 && Points<80)

        recommend gift options to user

    else if(Points>=80 && Points< 110)

        recommend gift options to user

    else if(Points>=120)

        recommend gift options to user
```

5.3.3 Ranking:

Similarly, depending on the ratings and comments, the expert hobbyist will be ranked when the user searches for a lesson. For example if the expert hobbyist is given a rating of 6/10, 8/10, 7/10 in the same lesson by different novice hobbyists, the average ranking of the expert hobbyist for that lesson type will be a 7. This deduction is based on the very simple mathematical equation of calculating an average of a sum of numbers.

All the ratings for that user in a particular lesson type will be taken into consideration while determining the ranking. This is because an expert hobbyist may hold lesson sessions of different lesson categories.

This will be displayed on the “All Lessons” page and the user will then be given an option to sort the lessons list by the rankings of the expert hobbyist. Thus, the user can choose the lesson with the best ranked expert hobbyist.

Pseudo Code for Ranking

This method is called when each novice user provides ratings to the expert hobbyist on a particular lesson. Novice hobbyists can rate the expert only when the lesson is conducted.

```
addRatings()
    if lessonCompleted ==TRUE
        giveRatings(ToUser,FromUser,LessonId)
    else
        error_message
```

This method would decide the rank of the expert hobbyist. Since we have currently restricted the total number of joiners (novice hobbyists) for a lesson to five, we have used an array. We get all the ratings that the expert hobbyist has received on a particular lesson and find the average rating to decide his rank.

```
decideRank(userId, lessonId)
```

```
double[] allRatings = new double[5]

allRatings=getAllRatings(userId,lessonId)

double totalRatings=0

for (i=0 to i=allRating.length)

    totalRatings += allRating[i]

double avgRatings = totalRatings /allRatings.length

assignRatingToUser(userId, lessonId, avgRatings)
```

5.3.3 Other Data Mining logic

Our application also provides the users with some recommendation on few lessons the users might be interested to take part in. This implementation we were able to achieve using the Apriori Data mining algorithm. If user John has taken lessons A, B, C and D, user Jenny has taken lessons A,B, D and user Alice has taken lesson A, then based on the fact that lessons A,B, D are frequently attended lessons, user Alice would be recommended lessons B and D.

Chapter 6. Performance and Benchmarks

We have undertaken a SCRUM developmental strategy for implementing our project, HobbyHorse. Hence, whenever we planned to implement a feature, we tested its requirements and specification before we actually implemented the feature. Then after the implementation we tested the feature against the test cases we tested before implementation. Following that we ran an integration test to ensure that the feature developed works with the other features that had already been developed. Integration test also includes the test cases that we prepared before implementing a feature. In this process, we always modified the broken test cases to keep the test plan updated at each increment. This helped us not only to minimize the bugs in the system but also ensured the quality of the application that we have developed so far.

Table 6.1: Testing methodologies and their purpose:

Serial Number	Methodology	Purpose
1.	Smoke Test	Test the major modules of the build.
2.	Bug Verification Test	Bugs fixed recently in the build would be checked in this test.
3.	Regression Test	Old bugs that were fixed would be verified. Also testing would done to check if fixing bugs have resulted in new bugs.
4.	Integration Test	This will test if the modules that were developed individually, when integrated, work together.
5.	Functional Test	Test the functionality of the whole system.
6.	User Acceptance Test(UAT)	Test the whole system by users to see if the application is user-friends,

Some of the tools we are using during testing are as shown below in table 6.2.

Table 6.2 : Testing and Bug Reporting tools

Serial Number	Tool	Usage
1.	JUnit	For unit testing.
2.	Bugzilla	For tracking defects and testing status.

Testing would follow the Agile technique and Bugzilla tool is used for collecting test logs and tracking defects.

Next table 6.3 shows the entrance and exit criteria for Unit test, Integration test and System test.

Table 6.3 Entrance and exit criteria for Unit test

Serial Number	Test	Entrance Criteria	Exit Criteria
1.	Unit Test	A module should be ready to test.	a.Requirement document should be baselined. b.Coding for the phase should be complete.
2.	Integration Test	a.Requirement and Design spec and doc should be baselined. b.Test Plan should exist and should be complete. c.Delivery of a working- software already completed.	a.All the test cases have been successfully executed. b.Bugs are fixed and tracked. c.Test logs are collected.

An example of a JUnit test is described below:

```
import java.util.ArrayList;

import java.util.Date;
import junit.framework.TestCase;
import org.joda.time.DateTime;
import org.junit.Test;
import org.junit.Before;
import org.junit.After;
import com.spring.datasource.User;
import com.spring.manager.*;

public class UserTest extends TestCase {
    private static final Date CURRENT_TIMESTAMP = new DateTime().toDate();

    @Before
    public void setUp() {
    }

    @After
    public void tearDown() {
    }

    @Test
    /*
    * Test the total number of users
    * This function gets all the users and checks the number
    */
    public void testGetAllUsers() throws Exception {
        UserManager o = new UserManager();
        ArrayList<User> res = o.getAllUsers();
        assertEquals(8, res.size());
    }

    @Test
    /*
    * Test the save, getuserbyusername, deleteuser functionality
    * This function first saves a user, then fetches it by username
    * and then deletes it. Thus all 3 checks are accomplished in the
    * same function. Thus integration testing is also achieved here
    */
    public void testGetSaveAndUserByUserNameAndDeleteUser() throws Exception {
        UserManager o = new UserManager();
        User user = makeTestUserBean();
        o.saveUser(user);
        ArrayList<User> res = o.getUserByUsername(user.getUsername());
        assertNotNull(res.get(0));
        assertEquals("David", res.get(0).getName());
        o.deleteUser(o.getUserByUsername(user.getUsername()).get(0)
                    .getUsername());
    }
}
```

```

    /*
    * Create a test user object
    * This is a common function to create a test user object
    * That will be used in all the unit tests
    */
    private User makeTestUserBean() {
        User user = new User();
        user.setName("David");
        user.setDescription("Test User");
        user.setCreateDate(CURRENT_TIMESTAMP);
        user.setLastUpdateDate(CURRENT_TIMESTAMP);
        user.setCreatedBy("admin");
        user.setLastUpdatedBy("admin");
        user.setEmail("david_k@gmail.com");
        user.setIsDeleted(0);
        user.setHobbies("Guitar, dancing");
        user.setSkills("dance");
        user.setLocation("Mountain View");
        user.setUsername("r_k");
        return user;
    }
}

```

As explained in the previous sections, maven provides a seamless testing integration framework; we have defined such test classes for each of the domain classes. A simple command such as “`mvn test`” runs all the tests defined and records the results. The default test package in a maven project is `src/test/java` and the test results are saved in the `surefire-reports` folder under the `target`. The command line output is something like this:

```

-----
TESTS
-----

Running UserTest

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.508 sec

Results:

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0

[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----

```


This is the result obtained when the single maven test is run, which can be run using the command “*mvn -Dtest=UserTest test*”

The test results are saved in an XML file as well as a txt file. It also incorporates the errors if any. The contents of the surefire-reports folder for each test class are:

```
rwadegaonkar:surefire-reports$ ls
```

```
TEST-UserTest.xml      UserTest.txt
```

Chapter 7. Deployment, Operations, Maintenance

7.1 Deployment

We have used Maven for building our project. We have used the following Maven commands for our project:

To compile: `mvn compile`

To clean target directory and install the output generated in their respective directories:
`mvn clean install`

To package as war: `mvn package`

After packaging is successful, we deploy the war file on the Tomcat 7 server. When the Tomcat is restarted, the war unpacks itself into an application. The application is accessed by the following url: <http://localhost:8080:hobbyhorse>. Currently we are running in our local machine. However, we plan to host a live application on a free web hosting site like godaddy.com. The website name would be www.hobbyhorse.edu.

7.2 Operations

Few of the operations our web application is supporting right now are:

Signup : For registering new users.

Login: For existing hobbyhorse users or users with Facebook /LinkedIn accounts.

Available lessons: This provides list of already existing lessons.

Create lesson: This is for creating new lessons. This is where the user role expert hobbyist is defined.

Join lesson: This is for joining a lesson event. Again here the user role as novice hobbyists is defined.

Start lesson: To start the lesson that had been created. Only expert hobbyist can initiate and end. This is a multiple video calling session.

Leave a Comment: After the lesson is over, novice hobbyists can leave back comments and give ratings to the expert hobbyists lesson

7.3 Maintenance

We would use bugzilla to capture bugs from our web application. Users of the website would also be able to report bugs. Depending on its frequency of occurrence, a bug would be analyzed and given a fix at regular intervals. The website would be taken down at regular interval for site maintenance. This would include maintenance of servers, network hardware, operation system, other configurations and optimizations. Besides, there would be also maintenance of the content of the website like the pages, images and videos.

Chapter 8. Summary, Conclusions, and Recommendations

8.1 Summary

As mentioned in the earlier sections too, we aim to build something with a highly social aspect. Hence, there will be a huge social impact as an output of our project.

Users will be able to utilize their time to the maximum, especially the time that they would otherwise invest in just surfing the social networking sites.

Our project, encourages the user to enhance his / her own skill sets or hobbies by still maintaining the social interaction like in Facebook, LinkedIn , Google+ and so on.

8.2 Conclusions

Apart from the social aspects and implementation details of the project, we can definitely conclude that in the process of developing the project, we explored, researched a whole bunch of tools, technologies and algorithms. Tools like PHP, Maven and Spring3 MVC were completely new to us, and we are now comfortable working with it.

Apart from the tools, the social data mining algorithms that we learnt in the process gave a thorough insight of the “behind the scenes” of the current social data oriented websites.

8.3 Recommendations for Further Research

Currently, we have used MySQL as our backend database. However, we believe beyond prototyping if there is a better suitable database that would fit well with our application, it is Cassandra.

Since our project focuses most on social data, scalability and performance is of major importance. Since Cassandra is well known for its distributive nature, it works perfect for scalability of such kind of data.

So as a further research factor, we would definitely like to migrate the whole application using Cassandra as the backend database and we could thus explore NoSQL type of database.

Another factor, where we could expand the scope of the project would be trying to incorporate users various social data accounts like Facebook, LinkedIn and get all the

social data as well as suggest lessons for the users dependent on the interests listed on these social networks for the specified users.

We could also try to add more social data mining algorithms based on the magnitude of the data, and add more filters if there is huge data returned for each of the current scenarios.

References

- [1] Social Learning. (2011). *Social Learning, Technical Learning, Innovative Learning*. Retrieved Sep 25, 2011 from http://www.innovativelearning.com/teaching/social_learning.html
- [2] Cost Benefit Analysis. (2003, November 21). *Cost Benefit Analysis, Wikipedia*. Retrieved Sep 25, 2011 from http://en.wikipedia.org/wiki/Cost%E2%80%93benefit_analysis
- [3] Memon, N., & Jie Xu, J., & Hicks, D. & Chen, H. (2010). *Data Mining for Social Network Data*, Springer.
- [4] Han, J., & Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques*, Elsevier.
- [5] Kantardzic, M. (2011). *Data Mining: Concepts, Models, Methods and Algorithms*, John Wiley & Sons.
- [6] Viewpath Project Management with Gadget Visualization. (2006). *Gantt Gadget, Google Docs*. Retrieved Oct 23, 2011 from <http://docs.google.com/spreadsheet>
- [7] Study: Ages of social network users (2010, February 16), Pingdom. Retrieved Nov 2, 2011 from <http://royal.pingdom.com/2010/02/16/study-ages-of-social-network-users/>.
- [8] Internet World Stats, Mar 31, 2011. Retrieved Nov 2, 2011 from <http://www.Internetworldstats.com/stats.htm>.
- [9] AttentionMeter, Sep, 2011. Retrieved Nov 2, 2011 from <http://attentionmeter.com/?d1=facebook.com&d2=linkedin.com&d3=myspace.com&d4=twitter.com&d5>.
- [10] Monica Michelle (Feb 21, 2011). *Difference between MySpace, Facebook, Twitter, & LinkedIn | Michele Minutes 185*. Retrieved Nov 2, 2011 from <http://www.examiner.com/music-marketing-in-national/difference-between-myspace-facebook-twitter-linkedin-michelleminutes-185>
- [11] The History and Evolution of Social Media. Retrieved Nov 2, 2011 from <http://www.webdesignerdepot.com/2009/10/the-history-and-evolution-of-social-media/>.

- [12] Jon-Mikel Bailey (Oct 14, 2011) *5Essentials for LinkedIn Success*. Retrieved Nov 2, 2011 from <http://www.woodst.com/blog/wood-street-journal/inbound-marketing/5-essentials-for-linkedin-success/>.
- [13] Han J., & Ng R. *Efficient and Effective Clustering Methods for Spatial Data Mining*. Retrieved Oct 29, 2011.
- [14] Schaeffer S., (2007). *Graph Clustering*. Laboratory for Theoretical Computer Science, Helsinki University of Technology.
- [15] Papagelis, M. & Plexousakis D. (2005). *Qualitative Analysis of user-based and item-based prediction algorithms for recommendation agents*. Department of Computer Science, University of Crete. Melville P., & Mooney, R., & Nagarajan, R. (2002). *Content-Boosted Collaborative Filtering for Improved Recommendations*. Department of Computer Sciences, University of Texas.
- [16] MagForWoman.com *.5 Reasons You Must Pursue A Hobby*. Retrieved Nov 2, 2011 from <http://www.magforwomen.com/5-reasons-you-must-pursue-a-hobby/>.
- [17] Think Inc Team (May 2, 2011). *Why is it important to have a hobby?*. Retrieved Nov 2, 2011 from <http://thinklink.in/why-is-it-important-to-have-a-hobby/>.
- [18] *Video Interactions in Online Video Social Networks* by FABRÍCIO BENEVENUTO, TIAGO RODRIGUES, VIRGILIO ALMEIDA, and JUSSARA ALMEIDA from Federal University of Minas Gerais, Brazil and KEITH ROSS. Retrieved Nov 2, 2011 from <http://cis.poly.edu/~ross/papers/VideoInteractions.pdf>.
- [19] Pew Internet & American Life Project (Feb 3, 2010). *Social Media and Young Adults*. Retrieved Nov 2, 2011, from <http://www.pewInternet.org/Reports/2010/Social-Media-and-Young-Adults.aspx>.
- [20] Linda Reoder. *Why Social Networking?*. Retrieved Nov 2, 2011, from <http://personalweb.about.com/od/easyblogsandwebpages/a/whysocialnetwor.htm>
- [21] J.J. Sandvig, Bamshad Mobasher, and Robin Burke, *Robustness of Collaborative Recommendation Based On Association Rule*, Retrieved April 12, 2012 from <http://maya.cs.depaul.edu/~mobasher/papers/smb-recsys07.pdf>
- [22] Rakesh Agrawal, Ramakrishnan Srikant, *Fast Algorithms for Mining Association Rules*, Retrieved April 12, 2012 from <http://www.rsrikant.com/papers/vldb94.pdf>
- [23] Free Video Chat API by tokbox. Retrieved Feb 24, 2012 from <http://www.tokbox.com/opentok/api>.

Appendices

Appendix A. Description of CDROM Contents