Social Networking Website-My World using Ruby on Rails

Ishita Sheth
8000 York Road
Department of Computer Science
Towson University
Towson, MD 21252
Isheth1@towson.students.edu

Bimal Thapa
8000 York Road
Department of Computer Science
Towson University
Towson, MD 21252
Bthapa2@towson.students.edu

Abstract
In this report, we discuss about Social networking website which is created to connect so many people who can share ideas ,find friends and able to create blogs by using technology Ruby on Rails. We describe the design, implementation and testing part of the website creation. In this paper we also addresses the various plugins which are used in website for searching friends, uploading images and creating map using Google API.

Keywords
Social networking, Ruby on Rails, Ajax Mashup, Acts_as_ferret, Google API Attachment_fu,

1 Introduction
Ruby on Rails as it is commonly referred to as, is a web framework built upon the Ruby language. It integrates Model-View-Controller style connectivity between the data storage, business logic, and interface. By using ruby on rails we create social networking application which has login facility by which new user can signup and administrative can login in system. Register user can create their own profile, blogs and also can make it public and private. General user can see other register user location within mention criteria to find friends and to see their location and also able to get direction from given address to friend address. Registered user can also upload their own photos and add friend's category wise. They can comment on other blogs and able to reply to their own blog's comment. We have use various plugins, Ajax and Google API for creating mashup.

2 Design
For Social Networking website creation we have created Use case diagram diagram, Class diagram, Sequence Diagram and data flow diagram to show the website design and flow of activity.

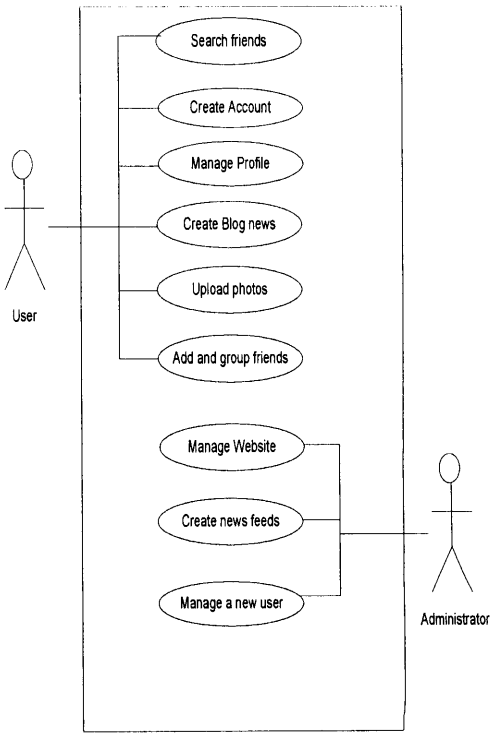


Figure 1. Use Case Diagram

Figure 1 shows that how the actors interact with the system to identify an initial set of use cases Another UML diagram in form of class and relationship that shows the brief description about attribute and models we used in our website design.

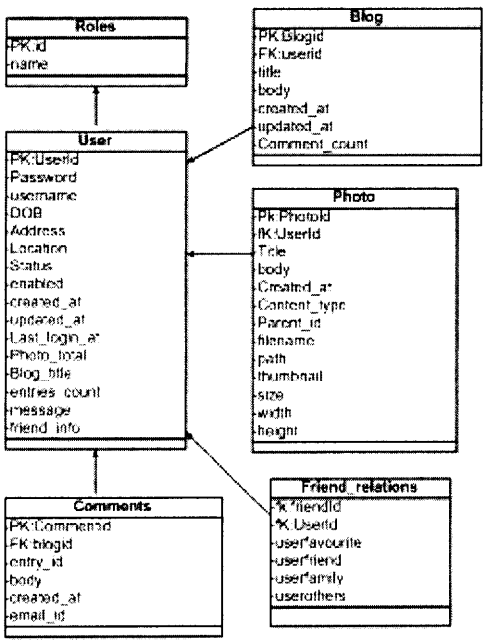


Figure 2. Class Diagram

Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order

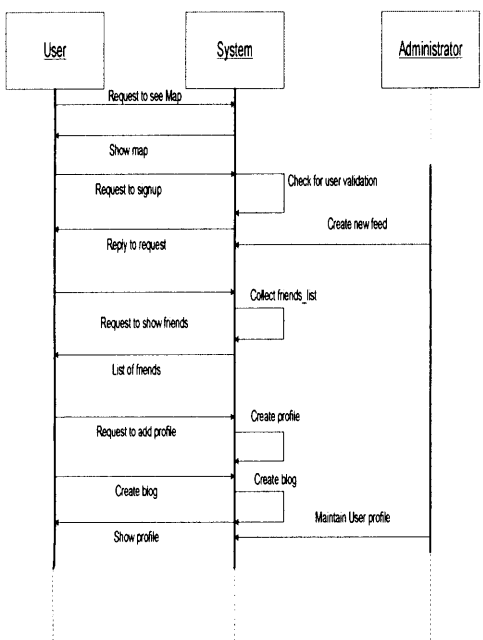


Figure 3. Sequence Diagram

3 Implementation

We have implemented our website using ruby on rails programming language. This is basically a web application framework, which is written in ruby language. It contains an Apache web server and Mysql. Ruby on rails programming language usages a Modal-View-Controller architecture and based on object oriented. We created below modules in our website. All modules are listed below:

- User
- Photo
- Blog
- Comment
- Friend
- Category(friend,family,favourite,other)
- Map

3.1 User module and functionality

By using User module user can create own profile by opening new account by clicking on sign up link as shown in figure 5. Figure 4 shows the main page of the website which can be access by any user.

Figure 4. Main page

Existing user can directly login by giving username and password as per shown in figure 6. Another user we have created for administration purpose. Administrator can make enable or disable other users .Registered user can upload new photos, make their own blog: public or private, reply back to their blog comments, add friends and group them as per category.

They can also view other user's location on the map and find direction using from the given address to user address.

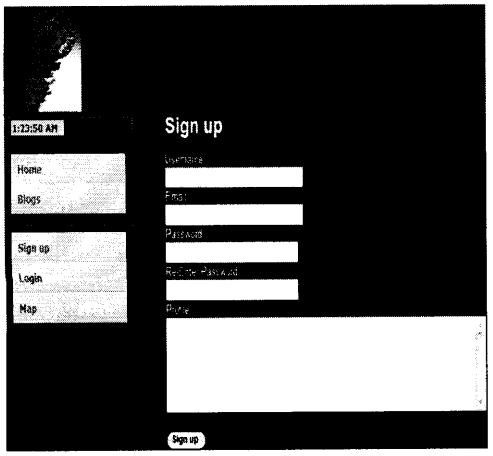


Figure 5. Sign up page

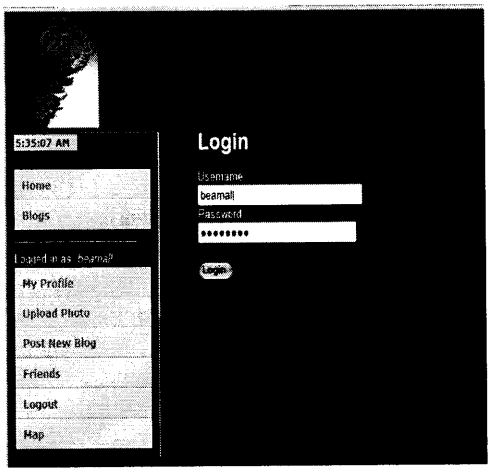


Figure 6. Login page

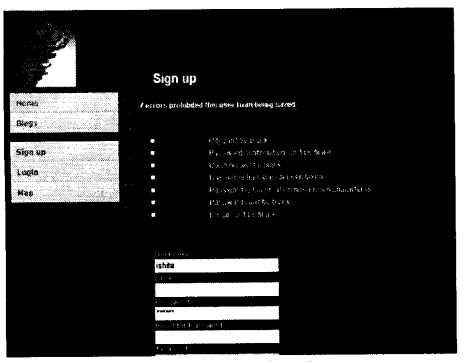


Figure 7. validations in sign up page

3.2 Photo module and functionality

By using above module user can create own photo gallery. We have use attachment_fu plugin to upload and resize photo. User can also zoom out photo by moving curser on the photo. User can also delete photos. We have also use Rmagick in this module to upload photo and add various functionality. User can maintain its Photo gallery as shown in figure

3.3 Blog module and functionality

By using this module user can create own new blog as shown in figure 8 and share ideas with other by keeping it public. User can also make it private so other user or general user cannot see it. User can search for any blog as per his choice. User can edit or delete his blog at any time. User is able to make number of entries in his own blog. User can see all new post on the main page which is public. User can search for blog from the main page as shown in figure. We have use Ajax in search blog to retrieve the search result.

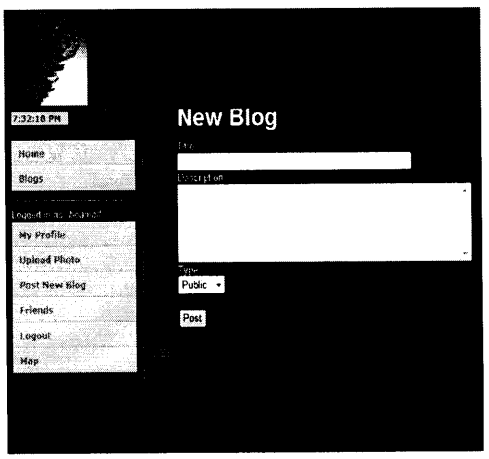


Figure 8. Blog creation

User can see all new post on the main page which is public. User can search for blog from the main page as shown in figure. We have use Ajax in search blog to retrieve the search result.

3.4 Comment module and functionality

User can comment on the listed post and reply back to their listed post. We have also use Ajax in listing comment. We have a add function which automatically make total of comment made on listed post and show total comment on the main page as per blog

entry wise. If any general user want to comment on any post listed on main page than that user has to first login and after only user can make comment on it.

3.5 Friend module and functionality

Friend module is created to store the all information about user and its friends. General user can search for friend by using map. User can see all registered users of website as per scope of the given criteria User has to fill up address and miles to show the registered user who are located in that particular scope. Registered user can search and add friends by using search

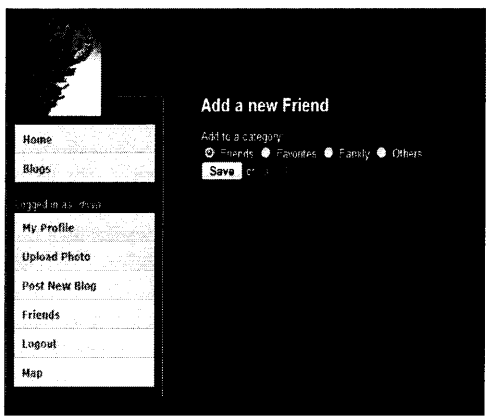


Figure 9. Add friend category wise

engine which we created by using Ferret plugin.User than get a list of friends as a result of search engine. If system does not found any result than it will show the message as no result found for particular search so user can try again. If user get any result from search than user can add that friend in his friend list by choosing category as shown in figure 9.After adding that friend user can see all recent activity done by all friend. User can delete and change friend from one group to another

3.5 Category module and functionality

Category module is created for grouping of friends as per category wise. We have created four categories of friends as shown in figure 13. Those are family, friends, favourite and others. User can make grouping of friends by choosing the category from the list and by choosing category user can see the recent activity done by those friend listed in category. User can change friend from one category to another category.

3.6 Map module and functionality

Map module is created using Google API And Ym4r and Geokit plugin. General user can find the location of registered user in particular area by giving address and miles as input. User can see all friends on the map who are located in that particular given area. We have use ym4r plugin to get coordinates and Geokit to address the location and find direction. Another application we have created by using map is we can find the location of our self on the map and find our friend location on map. Figure 12 shows the result of registered users whose addresses are near to given address and comes under chosen miles.

4 About Technology

We have use various plugins for searching, photo uploading and to create a map using Google API.We have use Ajax in searching blog

4.1 Ajax

Ajax web application sends the input field to the server in the background and updates the affected portion of the current web page in place. This dramatically increases the responsiveness of the user interface and makes it feel much more like a desktop application.

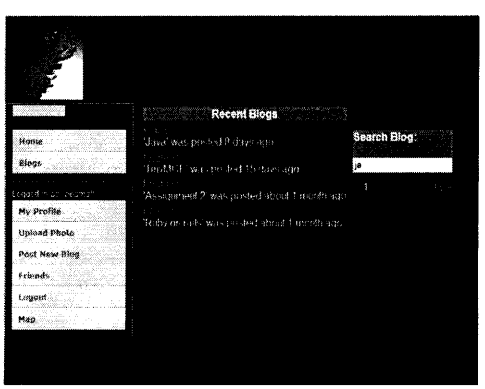


Figure 10. Search blog name using Ajax

Once the browser has rendered and displayed the initial web page, different user actions cause it to display a new web page (like any traditional web app) or trigger an

Ajax operation. Trigger action could be the user clicking on a button or link, the user making changes to the data on a form or in a field, or just a periodic trigger. Data associated with the trigger is sent asynchronously to an action handler on the server.The server-side action handler takes some action based on the data, and returns an HTML fragment as its response. The client-side JavaScript (created automatically by Rails) receives the HTML fragment and uses it to update a specified part of the current page's HTML, often the content of a tag. We have created search function using Ajax which shows the result without changing whole page. We have use Ajax in blog search to find the blog from the list which are created publically. It gives faster result compare to general application

4.2 Attachment_fu Plugin

Attachment_fu is a plug-in and successor to acts_as_attachment. Attachment_fu facilitates file uploads in Ruby on Rails. There are three storage options for files uploaded through attachment_fu:

- File system
- Database file
- Amazon S3

We have use file system for photo storage. For all three of these storage options a table of metadata is required. This table will contain information about the file and its location. This table has no restrictions on naming, unlike the extra table required for database storage, which must have a table name of db_files (and by convention a model of DbFile).If we check thumbnail is nil or not by calling filename function than it returns the public path to the file if a thumbnail prefix is specified it will return the public file path to the corresponding thumbnail. For image processing it's needed to install one of the following packages as well:

- ImageScience
- Rmagick
- minimagick

We have use Rmagick package for photo uploading. Rmagick is an interface between the Ruby programming language and the ImageMagick.In our application we have a photo model to which we want to associate images. An image is submitted by user and is associated to one single photo, has many / belongs to association between a photo and the associated images.

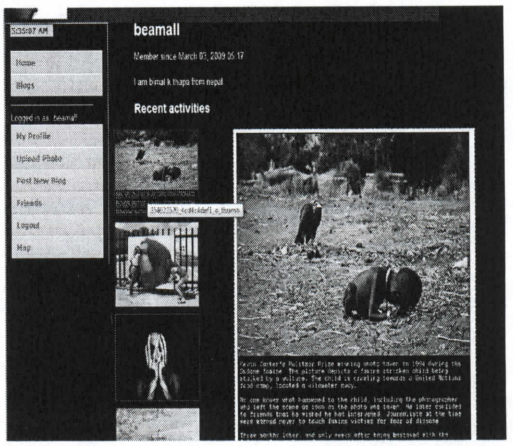


Figure 11. Photo uploading using attachment _fu

By using this we can store image as a thumbnail or in any size so it easy to retrieve it from database and storing in particular file location. We can give as many effects to photo by retrieving it from file and display them as shown in figure 11 and we can zoom out that image by changing it's size

4.3 Mashup

In web development, mashup is a Web application that combines data or functionality from two or more sources into a single integrated application. The term mashup implies easy, fast integration, frequently done by access to open APIs and data sources to produce results that were not the original reason for producing the raw source data. Google API's consist basically of specialized Web services and programs and specialized scripts that enable Internet application developers to better find and process information on the Web. The Google Maps key is the primary configuration setting to get started.We has use two plugins for Google maps.

- Ym4r
- Geokit

As per shown in figure 12 we have use two plugin for creating Gmaps to find location on map and to get direction from given address. YM4R add maps and map features such as icons and information windows while Geokit derive the latitudinal and longitudinal coordinates of the desired locations, calculating the distance between two points, finding all points within a specified radius, and identifying user locations.

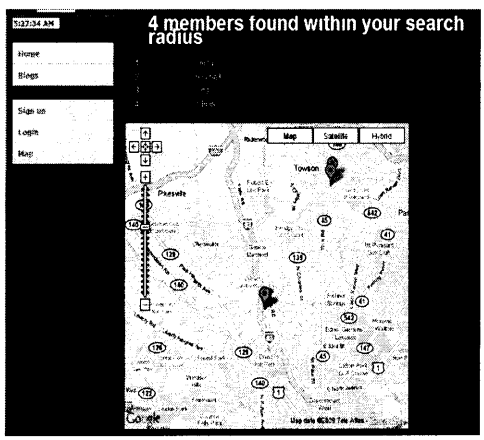


Figure 12. Google map using ym4r and geokit plugin

- gem install ym4r

By typing above we can install ym4r plugin in our application. That is same for geokit to install it. After adding all these plugin we have to make changes in configuration file to make it work using Google API

4.4 Acts_as_ferret

Ferret is a Ruby high-performance text search engine. The Acts As Ferret plugin adds additional search methods to the Active Record Model, User put friend name into the index. This index will auto-magically be updated. ActsAsFerret then calls Ferret's Search each function on our index. We have use Act as ferret plugin for friend search as shown in figure 13 user can search friend by adding friend name in search field. The ferret engines find the friend name and list them by making query and if there is no result than it will print message that no result found.

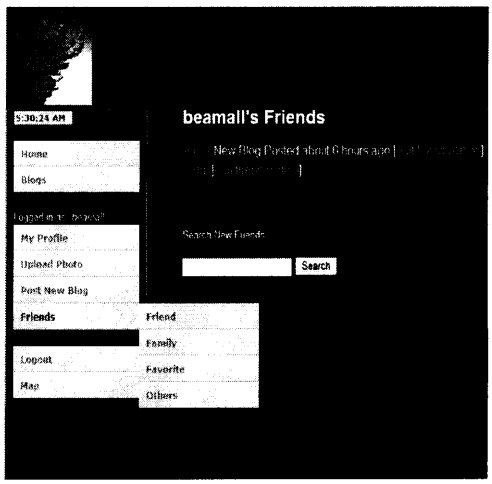


Figure 13 Friend search by using Acts as ferret

5 Testing

Unit testing is a software verification and validation method where the programmer gains confidence that individual units of source code are fit for use.

```
require File.dirname(__FILE__) + '/../test_helper'

class UserTest < Test::Unit::TestCase
  fixtures :users

def test_create_user user =
User.new(:username=>'billu',:email=>'billu@yahoo.com',
:password=>'123asdf',:password_confirmation=>'123asdf', :profile
=> 'I am billu ram!') assert user.save
end

def test_for_invalid_user user =
User.new(:username=>'bimal',:email=>'billu@yahoo.com',
:password=>'123asdf',:password_confirmation=>'123asdf', :profile
=> 'I am billu ram!') assert !user.save
end

end
```

Figure 14. User test case

A unit is the smallest testable part of an application. In object-oriented programming, the smallest unit is a method, which may belong to a base or super class,

abstract class or derived or child class Each test case is independent from the others: substitutes like method stubs can be used to assist testing a module in isolation. Unit tests are created to ensure that code meets its requirements and behaves as intended. Its implementation can vary from being very manual to being formalized as part of build automation.Rails generate tests based on the Test::Unit framework that comes preinstalled with RubyRails generates tests based on the Test::Unit framework that comes preinstalled with Ruby

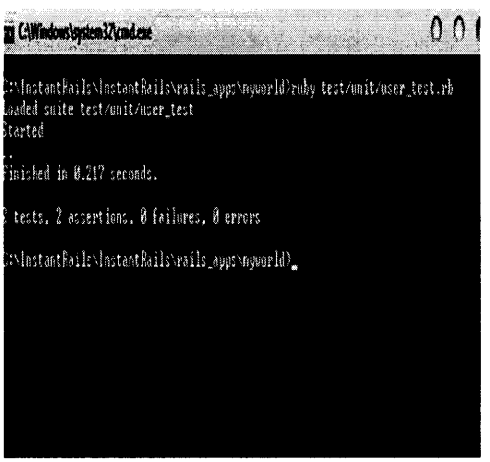


Figure 15. Test Result

We have created two unit tests. One for user and other one for photo .as shown in figure 14. We have included the user fixture into this test. Rails generated are the method test truth. We have put two assertions to check user validity in user test and below is the result of that test case which shows no error in result. We have created so many validations for user field and for user login, Photo uploading, blog creation and in friend module.

6 Conclusion

By using ruby on rails it's possible to make reduction in the code lengths and volume that enhances the speed of the project completion. It also helps in reducing the debugging pressure. Ruby creates an environment that makes to work more clearly, faster. By using MVC web framework which is provided in ruby, we can lighten the configuration task compare to java and .net framework. The simple conventions and self-managing reflections also reduce the configuration burden. It supports so many plugin so by installing them we can create rich web application.

References

[1] Agile web development with Rails by David Hansson

[2] Cascaded Style Sheet information Available at "www.dynamicdrive.com"

[3] Plugin for Google map Available at "www.scribd.com/doc"

[4] Google API key Available at "www.google.com/apis/maps"

Technical Report System

Ting Zhang
Computer and Information Sciences Department
Towson University
7800 York Road, Towson, Maryland, 21204
+1 443 682 4152
tzhang2@towson.edu

Jun Tao
Computer and Information Sciences Department
Towson University
7800 York Road, Towson, Maryland, 21204
+1 443 615 5907
jtao1@towson.edu

Guanghui Ma
Computer and Information Sciences Department
Towson University
7800 York Road, Towson, Maryland, 21204
+1 443 869 0297
gma1@towson.edu

ABSTRACT
More and more educational material can be shared in the web as it is convenient and easy. In this project, we have designed a technical paper system which is aimed at to let faculties and students share valuable papers. The work of this project involves several processes such as the class design, database design, implementation and unit test. This paper will demonstrate those detailed process and show the screenshot of this system

Keywords
Report, system, design, user

1. INTRODUCTION
As development of modern technique, there are several new tools for communication between faculties and students except for textbook, lecture notes and so on. The Technical Report System is targeted at providing technical papers available online for faculties and students so that they can share valuable papers online instead of hard copy. This system mainly provide following functionalities:

- Allow faculties can upload the latest papers to students with author, abstract, title, and links to the full paper (PDF format).
- Students can download the papers which are uploaded by faculties.
- "Search for paper" is provided so that faculties and students can search the specific papers by criteria such as date, author, and category and so on.
- Both faculties and students can view download history as well as most popular paper and best contributor to system.
- Administration is needed to add or delete users and grant user authority.

This Tech Report System will be a convenient platform for faculties and students to share the precious technical papers so that students can learn much more useful knowledge and information.

2. DESIGN
Base on the description of the system mentioned above, there are three different roles in action:

- Administrator who is charge of the users including faculties and students to add or delete them and grant authority to them.
- Faculties who provide technical papers to students.
- Student who can download the papers from faculties.

So User Case is shown in Figure 1.

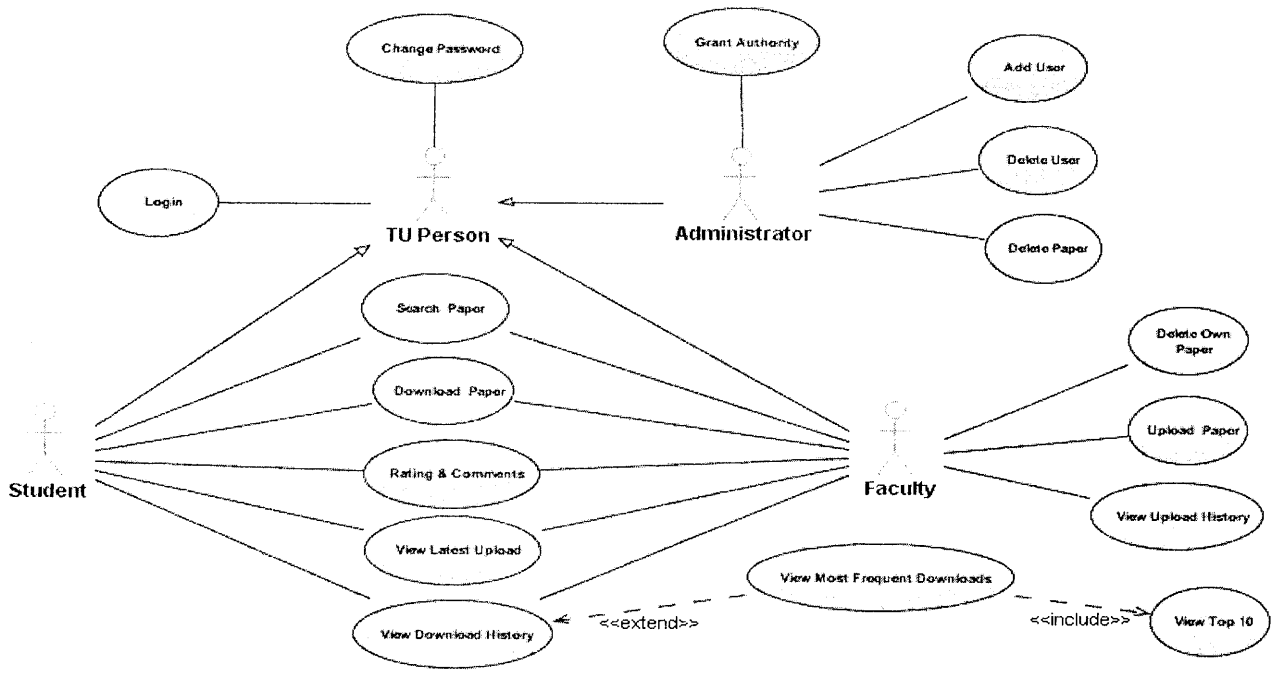


Figure 1 User Case

Our database designs are below:

First, Figure 2 will show our Entity-Relationship Diagram.

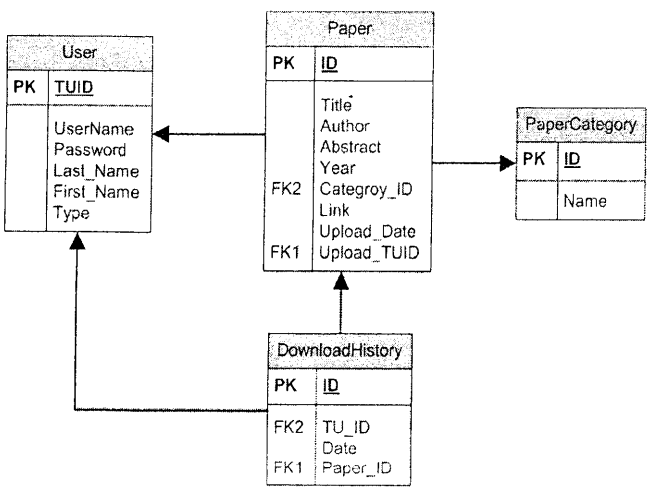


Figure 2 Entity-Relationship Diagram

Second, Figure 3 will show the schema of a database system

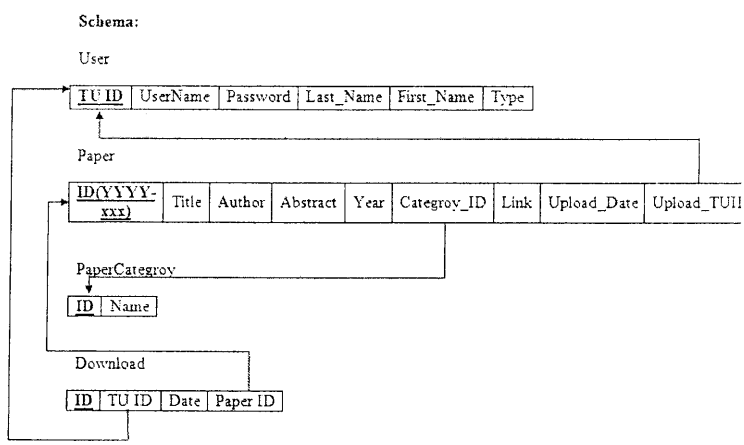


Figure 3 The schema of a database system

Third, as the different data of input and output, we create four tables including TechReport_DownloadHistory, TechReport_Paper, TechReport_Category, and TechReport_User. The structure of each table is shown in following.

Column Name	Data Type
download_id	int
download_tuid	varchar(10)
download_paper_id	varchar(8)
download_dt	smalldatetime

Figure 4. TechReport_DownloadHistory

In this table, download_id is the primary key, while download_tuid indicates the student or faculty who download specific paper which is described by download_paper_id; download_dt presents the specific date of downloading

Column Name	Data Type
paper_id	varchar(8)
paper_title	varchar(100)
paper_author	varchar(50)
paper_abstract	varchar(MAX)
paper_year	varchar(50)
paper_category	int
paper_link	varchar(100)
paper_upload_tuid	varchar(50)
paper_upload_dt	smalldatetime

Figure 5. TechReport_Paper

TechReport_Paper is concerned with paper. Paper_id is unique variable for each paper. Title, author, abstract, year, category, link are indicated by paper_title, paper_author, paper_abstract, paper_year, paper_category, paper_link. Paper_upload_tuid is used to imply the person who uploads one paper. And paper_upload_dt indicates the specific date for upload.

Column Name	Data Type
category_id	int
category_name	varchar(50)

Figure 6. TechReport_Category

TechReport_Category is related with category of paper. It includes primary key which is category_id and category_name which is used to represent the different area of paper such as software engineering, network security and so on.

Column Name	Data Type
user_tuid	varchar(10)
user_username	varchar(20)
user_password	varchar(100)
user_first_name	varchar(50)
user_last_name	varchar(50)
user_type	varchar(50)

Figure 7. TechReport_User

This is the user table. User_tuid is the primary key which is unique for each student and faculty. User_username is the name which is created by user to login to the system. User_password is also created by user for safety. User_first_name and user_last_name are students and faculties' real name and be added by administrator. User_type includes three different values which are student, faculty or administrator.

The basic class design is shown in Figure 8.

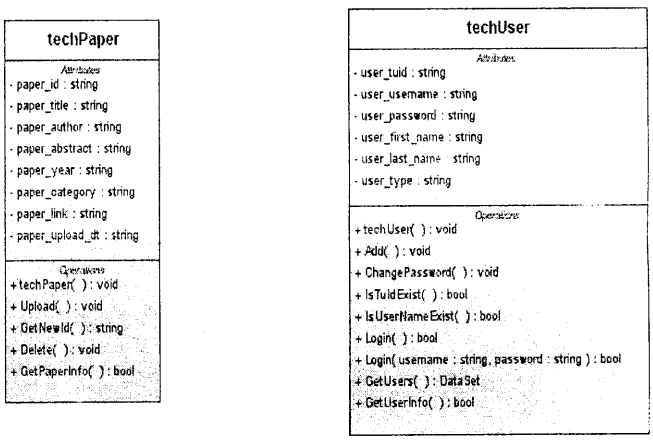


Figure 8. Class Design

Before implementation, work flow of each function should be considered. The following is "search" work flow written in activity diagram.

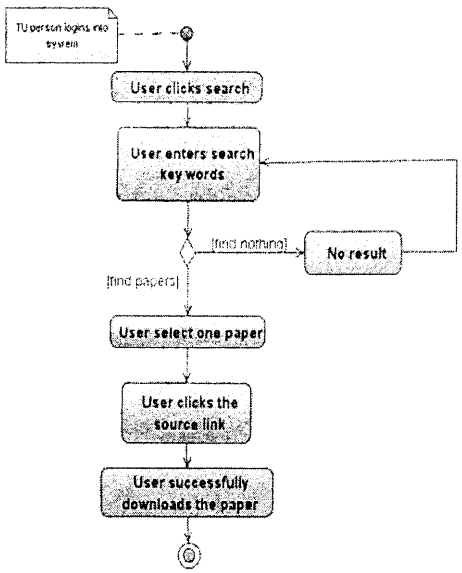


Figure 9. Activity Diagram——Search

Users can search specific paper by key words and then download it.

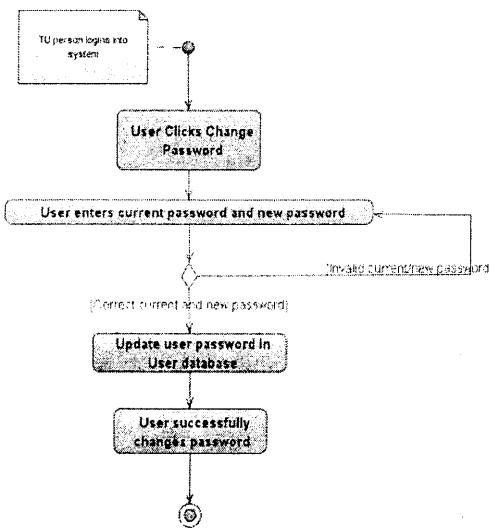


Figure 10. Activity Diagram——Change Password

Users enter current and new password and if approved they can use new password to login. Following is the "login" activity diagram.

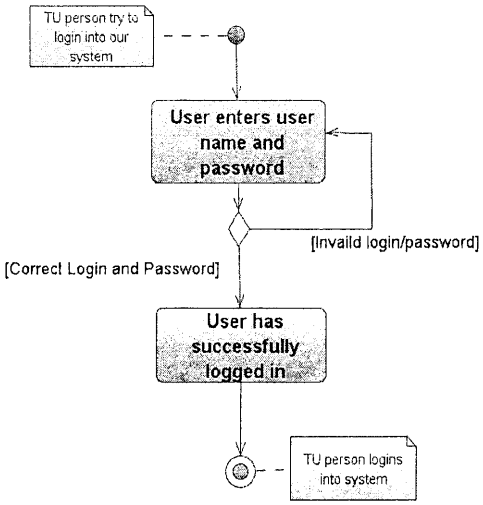


Figure 11. Activity Diagram——Login

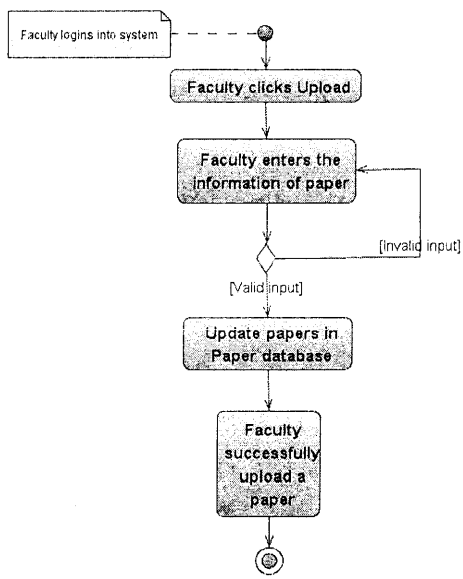


Figure 12. Activity Diagram——Upload

Faculty can upload paper by entering title, author, abstract and so on. The uploaded paper will be stored in the database.

3. IMPLEMENTATION

The platform and tools we use in this project include:

- Programming Language: C#
- Platform: ASP.NET 2.0
- Database: Microsoft SQL Server 2005 (Express Edition)

3.1 Functionalities

Here are the detailed functionalities of the system:

For faculty:

- Upload Paper
- Download paper
- Search paper
- View latest upload
- View download history
- View upload history
- View most frequent downloads
- View best contributor
- Login/Logoff
- Change password

For student:

- Download paper
- Search paper
- View latest upload
- View most frequent downloads
- View best contributor
- Login/Logoff

For administrator:

- Login/Logoff
- Grant authority
- Add user
- Delete user

Here are the main screenshots of the system.

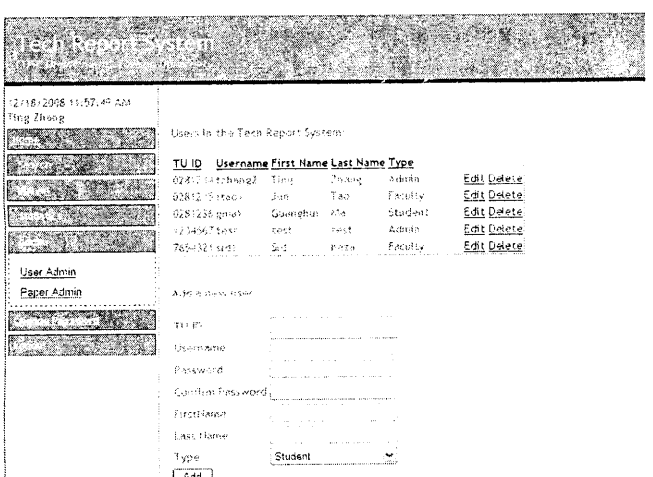


Figure 13. Administrator——User Admin

Administrator can add one user by TUID, Username, Password, first name, last name and type. And he/she can also edit one user by changing his/her authority and completely delete him/her.

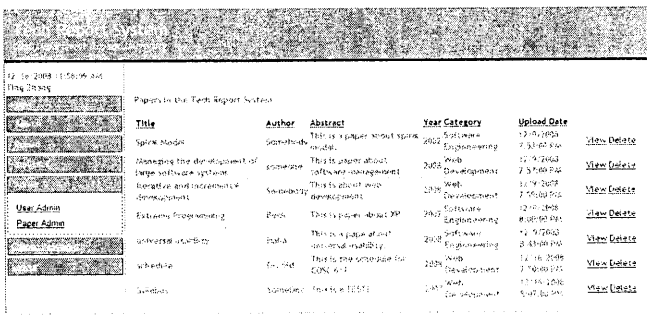


Figure 14. Administrator——Paper Admin

"Paper Admin" involves viewing and deleting papers. When administrator click "view" button, he/she can download the paper. Another feature is that clicking the attributes such as title, author, it will be sorted.

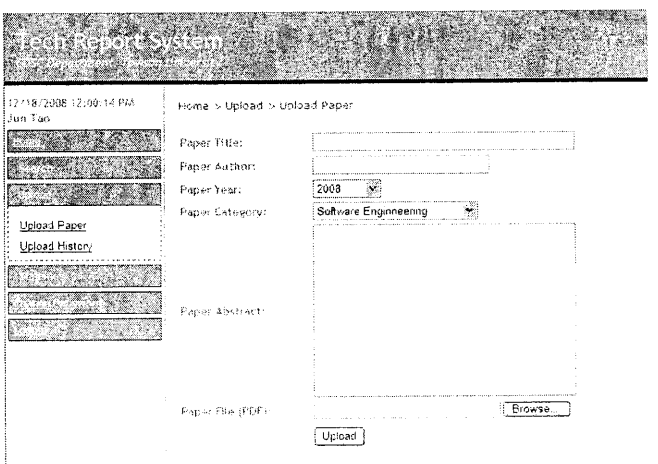


Figure 15. Administrator——Upload Paper

Faculty clicks on "Upload Paper" button to upload some paper by entering the title, author, publish year, abstract, category which includes software engineering, network security and so on. Also he/she need provide the local .PDF fine and then click "Upload". If action successfully, there will be a message to notify.

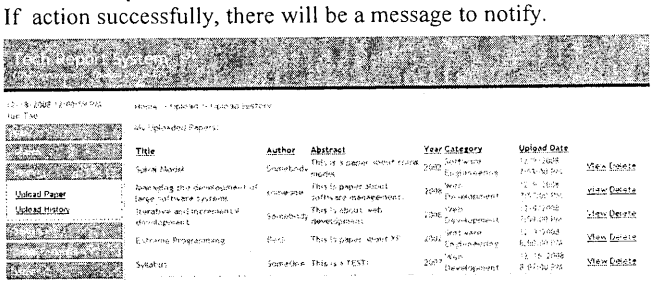


Figure 16. Administrator——Upload History

Faculty can view his/her upload history by clicking on "Upload History". And then he/she can view or delete his/her uploaded paper if he/she wants.

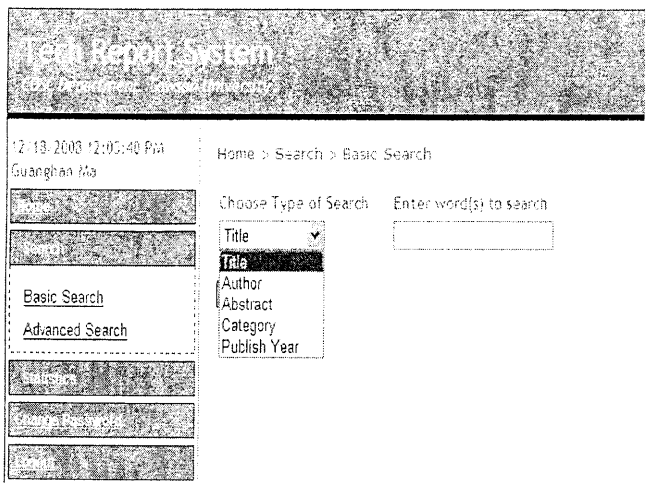


Figure 17. Basic Search

"Search" function provides two ways which include basic search and advanced search (advanced search will be explained later). Using basic search users can search specific paper by entering key words of different criteria like title, author, abstract and so on.

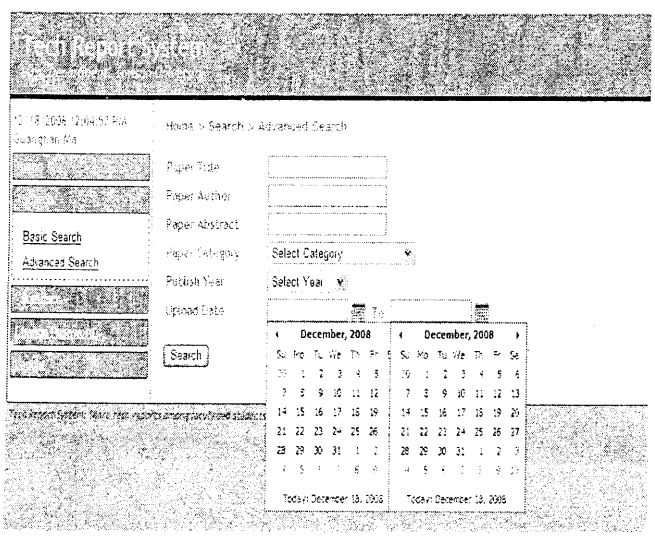


Figure 18. Advanced Search

Advanced Search let user to search by multiple criteria such as title, author, abstract, category and publish year. There is another feature need to be mentioned. User can use calendar to identify the range of upload date to search.

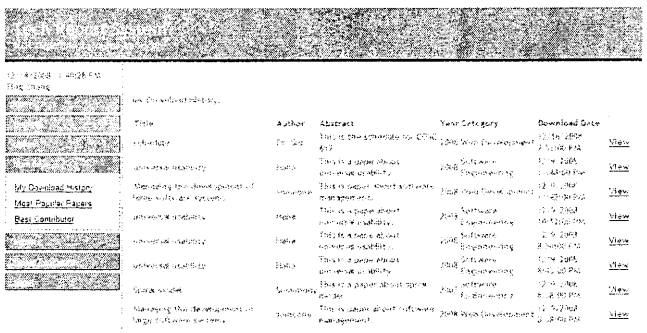


Figure 19. Download History

"Download History" shows the all downloaded paper by user and clicking the button "view" can view the information of paper.

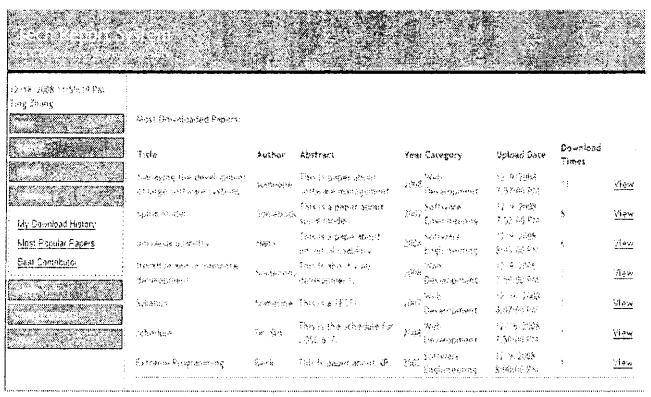


Figure 20. Most Downloaded Paper

"Most Downloaded Paper" lies out that which paper is the most frequently downloaded by user.

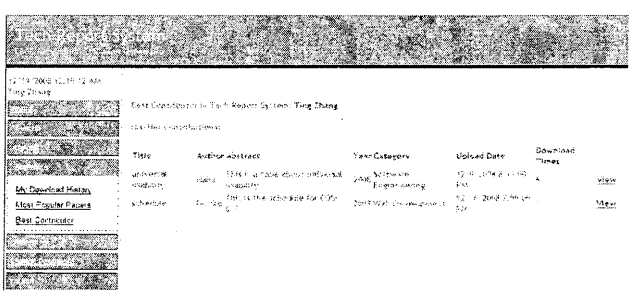


Figure 21. Best Contributor

"Best Contributor" indicates the person who uploads paper most frequently.

4. Technical Details

4.1 Master Page

In Tech Report System, we use Master page mechanism which is provided in Visual Studio 2005. It is used to keep the whole web site look consistent. It is like a template of the web site.

A single master page defines the look and feel and standard behavior for all of the pages in the application. Individual content pages contain the content for each page to display. When users request the content pages, they merge with the master page to produce output that combines the layout of the master page with the content from the content page.

In this application, the top part (above the black line) and left part (the menu) are included in the master page. So, they are for all of the web pages of this application.

4.2 AJAX

AJAX (Asynchronous JavaScript and XML) is used to improve this application's interactivity and performance. For this application, Microsoft Ajax Control Tool Kit is used.

To use this control tool kit, ASP.NET AJAX Extension 1.0 has to be installed. AJAX controls are used in this application for calendars and menu.

Calendars are used for date textboxes. It is quite easy for user to enter a date by clicking the small calendar image and choosing a date from the pop up calendar. When calendar pops up, the current date is highlighted. And user can click arrows and numbers in the calendar to choose a different month or year. Besides the efficiency of selecting a date, calendar also has the benefit of keep the date format correct.

Menu in this application is also created using AJAX control. There are many web pages for this application. So, we use Accordion to make the menu flexible. The Accordion is a web control that allows you to provide multiple panes and display them one at a time. It is like having several collapsible panels where only one can be expanded at a time.

The Accordion is implemented as a web control that contains AccordionPane web controls. Each AccordionPane control has a template for its header and its content. So, in the menu of this system, each section is an AccordionPane, the section name is the header, and the links to pages are the content. For example, the "Search" section is an AccordionPane. The name of this pane is "Search", and the content of this pane is two links ("Basic Search" and "Advanced Search"). Only one AccordionPane can be expended at a time. So the menu would not be too long to display on the page.

4.3 Sessions

Sessions are used in this application for two purposes. One is to keep the information of current user. The other is to keep the section number of current page.

For the first purpose, three sessions are used, including Session ["Name_of_User"], Session ["Type_of_User"], and Session ["TUID"].

Session ["Name_of_User"] is used for displaying user name in the left upper side of each page.

Session ["TUID"] is used whenever we want to update or retrieve some data from database about the current user, like inserting a download or upload record.

Session ["Type_of_User"] contains type of current user (Admin, Faculty, or Student), which is used to decide which pages can be accessed by current user.

Actually, the information stored in Session ["Name_of_User"] and Session ["Type_of_User"] can both be got by using Session ["TUID"]. However, that will require database connections each time we need that information. Therefore, to improve the system performance, we use three sessions to keep user's information.

Session ["section"] is used for the second purpose. The menu is implemented with Ajax Accordion. Only one accordion pane can be extended at one time. To force the pane which current page belongs to keep extended, the section number of the current page needs to be kept in a session. If we don't keep this number, the extended menu section would always be the default one whenever a page is reloaded. With the section number of current page, the corresponding pane can be set to be extended when page is loading.

4.4 Unit test

In our Tech Report System, we can use many test tools which is provided in Visual Studio 2008 Professional Edition. Visual Studio Team System provides a suite of test tools that are integrated closely with Visual Studio; they can work not only in their own testing framework, but also within a larger framework of software life cycle tools.

Visual Studio testing tools provides several test types that we can use for our testing purposes. However, Visual Studio 2008 Professional Edition only provides Unit Test, so we can't do Web Test (Integration testing), Load Test (Smoke, Stress, Performance and Capacity Planning), and Generic Test.

In computer programming, unit testing is a method of testing that verifies the individual units of source code are working properly. Below in Figure 22, there is our Unit Test Screenshot for one of our system methods: Login.

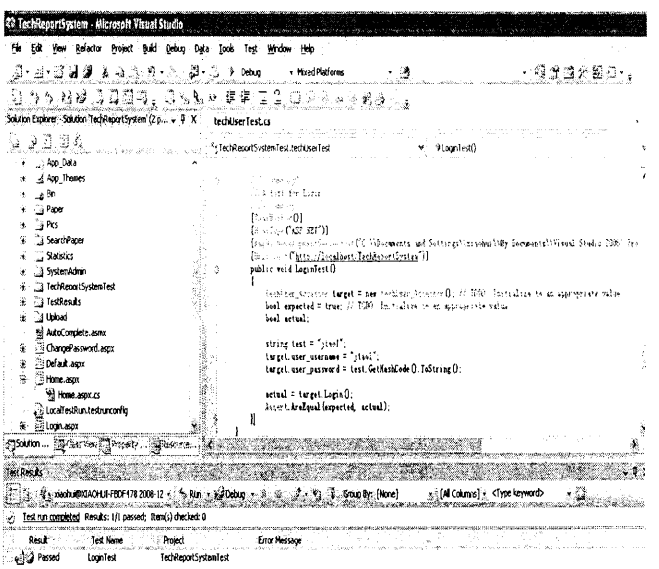


Figure 22. Unit Test

Also, we faced many challenges for our test. For many methods, we can't go pass for its unit test, and we got error message. That is because many methods will call other methods, and test can't recognize the unknown other methods.

5. TASK ALLOCATION AND TIMELINE

	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	**Phase 1: Initialization**	**10 days**	**Tue 9/9/08**	**Mon 9/22/08**		
2	Overall Goal	3 days	Tue 9/9/08	Thu 9/11/08		Guanghui Ma
3	Requirement Analysis	2 days	Fri 9/12/08	Mon 9/15/08	2	Ting Zhang
4	Major Classes Design	2 days	Tue 9/16/08	Wed 9/17/08	3	Jun Tao
5	Use Case Design	2 days	Tue 9/16/08	Wed 9/17/08		Jun Tao
6	Proposal	3 days	Thu 9/18/08	Mon 9/22/08	5	Guanghui Ma,Jun Tao,Ting Zhang
7	**Phase 2: Design**	**20 days**	**Tue 9/23/08**	**Mon 10/20/08**		
8	Database Design	7 days	Tue 9/23/08	Wed 10/1/08		Guanghui Ma
9	Functional Design	7 days	Mon 9/29/08	Tue 10/7/08		Jun Tao
10	GUI Design	5 days	Wed 10/8/08	Tue 10/14/08	9	Ting Zhang
11	Design Report	4 days	Wed 10/15/08	Mon 10/20/08	10	Guanghui Ma,Jun Tao,Ting Zhang
12	**Phase 3: Implementation**	**30 days**	**Tue 10/21/08**	**Mon 12/1/08**		
13	Coding and Unit Testing	30 days	Tue 10/21/08	Mon 12/1/08		Guanghui Ma,Jun Tao,Ting Zhang
14	Progress Report	3 days	Fri 11/14/08	Tue 11/18/08		Guanghui Ma
15	**Phase 4: Testing**	**5 days**	**Tue 12/2/08**	**Mon 12/8/08**		
16	Testing Plan	2 days	Tue 12/2/08	Wed 12/3/08		Jun Tao
17	Integrate Testing	3 days	Thu 12/4/08	Mon 12/8/08	16	Guanghui Ma,Ting Zhang
18	**Phase 5: Project Release**	**9 days**	**Tue 12/9/08**	**Fri 12/19/08**		
19	Environment Setting Up	2 days	Tue 12/9/08	Wed 12/10/08		Ting Zhang
20	Presentation Preparation	5 days	Tue 12/9/08	Mon 12/15/08		Guanghui Ma,Jun Tao,Ting Zhang
21	Project Final Report	9 days	Tue 12/9/08	Fri 12/19/08		Guanghui Ma,Jun Tao,Ting Zhang

Figure 23. Timeline

6. CONCLUSIONS

Technique paper system provides a convenient platform to students and faculties to share papers. To implement this system, we do the class design, database design, coding, test, and so on. Also we use much technical knowledge, such as Master Page, AJAX, and Sessions. According to this, we create several tables and classes, and a real database-driven web software application. In final, this system can be in use.

7. REFERENCES

[1] http://www.asp.net/

[2] http://msdn.microsoft.com/en-us/default.aspx

Tiger Version Control (TVC)

Pamela Maldeis
Computer Science Graduate
Towson University
8000 York Road
Towson, Maryland 21252
pmalde1@students.towson.edu

Anuradha Mahalingam
Computer Science Graduate
Towson University
8000 York Road
Towson, Maryland 21252
amahal1@students.towson.edu

Leon Bernard
Computer Science Graduate
Towson University
8000 York Road
Towson, Maryland 21252
lberna1@students.towson.edu

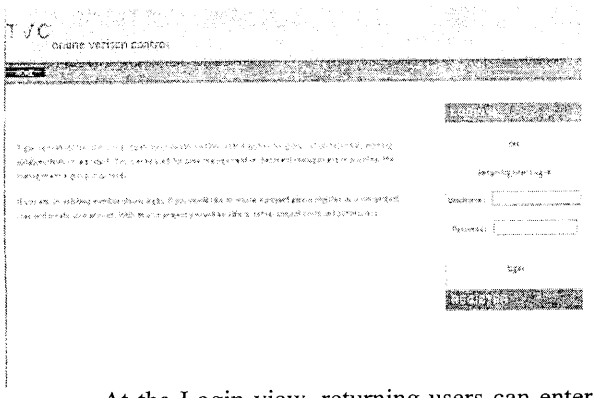
ABSTRACT
In this paper, we discuss our project design and implementation.

Keywords

INTRODUCTION
Tiger Version Control (TVC) application is an online version control system. This application was develop using Ruby on Rails and MYSQL.

Implementation
We will go through each of the features of the TVC application.

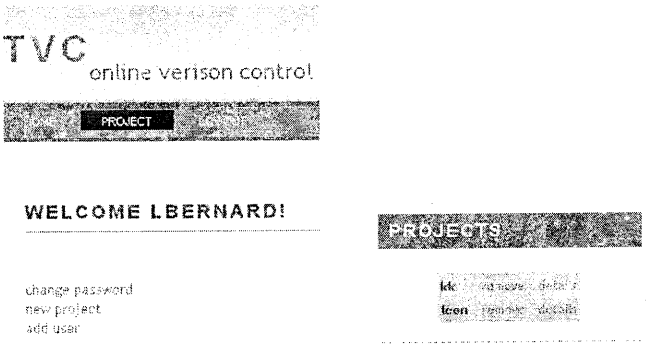
LOGIN



At the Login view, returning users can enter their username and password and then click the login button. The login method in the login_controller is executed once the login button is clicked. If the password and username submitted matches the data in the user model then the user will see the project view. If the data doesn't match what's in the database, the user will see the login page and has the ability to re-enter his/her information.

New users can register by clicking the register button. This will trigger the register_user method in the login_controller. The users will see the new user screen which allows them to add a user to the user model. Once the user is added successfully to the user model, the user will see the login view which allows them to login.

PROJECT



At the project view, users will be able change their password, add a new project, add a new user, remove a project and view the details of a project.

When the user clicked the change password link, the change_password method in the project_controller is triggered. The change_password method will render the change_password partial of the user view. This view will allow them to modify their password in the user model.

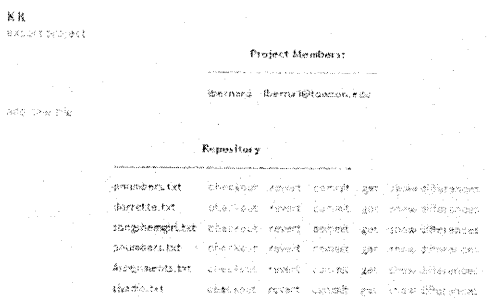
The add a new project link will fired the add_project method in the project_controller. This method

will then render the new project partial view. This will allow user to add a project to the project model.

The add user link will fire the newUser method in the project_controller. This method will then render the new user partial view. User can use the new user view to add users to the user model.

When a user clicks the remove project link to the right of the screen the remove method in the project_controller will be fired. This method will remove the project from the project model and then refresh the project main view to display the changes.

PROJECT DETAILS



Users can manage files from the Project Details screen. They can add a new file, checkout, commit and get existing files. Users can also revert changes made to a file and compare two versions of a file by clicking the show differences link. Users are able to export the entire project and save to disk.

Clicking the add_new_file link will display the upFileDiv. This is initially hidden and it also the users to browse their file system for the file to add to TVC.

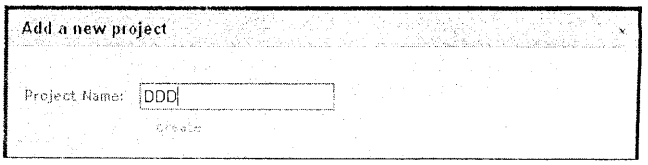
When a user clicks the checkout or get link, the checkout method in the project_controller is fired. This method will send the current version of the file to user for download. The difference between the checkout and the get link is that the checkout will cause the file to be lock. This means that other users cannot checkout the file. This information is stored in the proj_file model in the status field.

The commit link will display the upFileDiv. From the upFileDiv users can save the changes made to checkout file by creating a new version of the file.

Clicking the revert link will trigger the displayVersions method in the project_controller. This will then render the versionList partial view. From this view the user will be able to choose a version to restore.

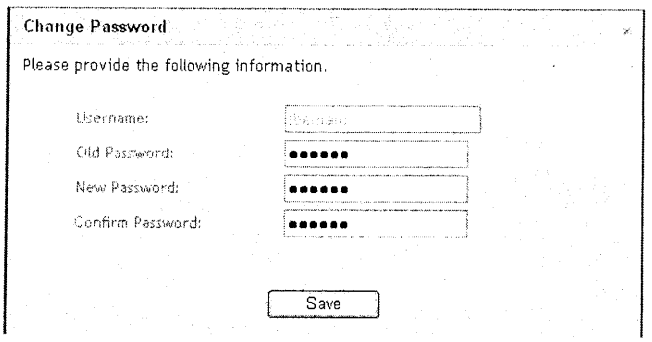
When a user clicked the show difference link the diff_select method in the project_controller is fired. This method will render the versionList partial view. From this view the user will be able to choose the versions to compare.

ADD NEW PROJECT



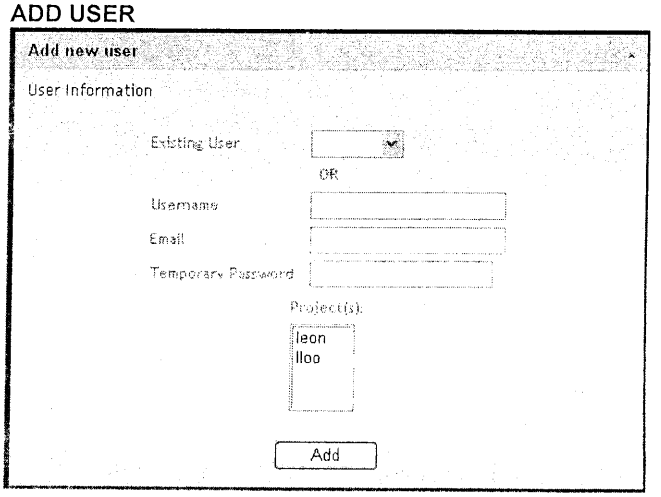
From this screen the user can enter the name of the project they want to create. When the user clicked the create link, the add_project method in the project_controller is fired. This method will add the project to the project model and take the user back to the project main view.

CHANGE PASSWORD



This screen allows the users to change their password. When the save button is clicked the save_password method in the project_controller is executed. This method will update the user model with the new password entered. This password is hashed using a salt and the SHA1 encrypt function.

ADD USER

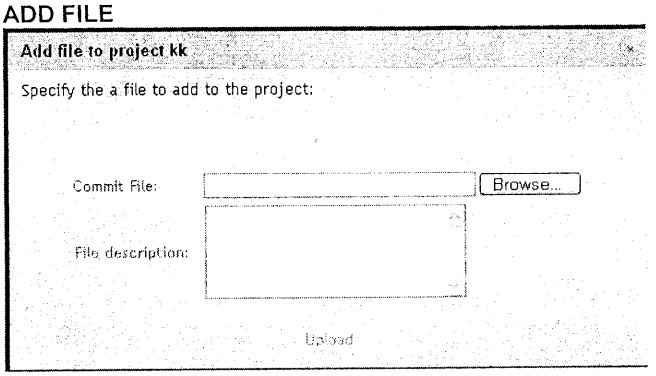


The add user screen can only be access by the site and project administrators. From this screen new users are created and assigned to a project. Once the add button is clicked the add_user method in the project_controller is executed. This method will add the user to the user model and also add setup the user's project permissions in the permissions model.

EXPORT PROJECT

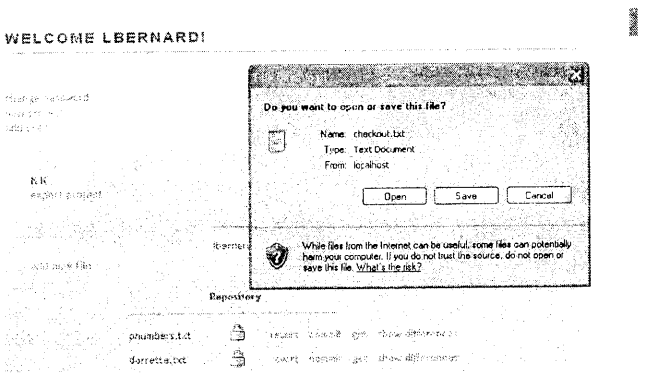
This screen is displayed when the user clicked the export project link from the project details screen. User can save the entire project as a zip or tar file. User can also cancel the export project task.

ADD FILE



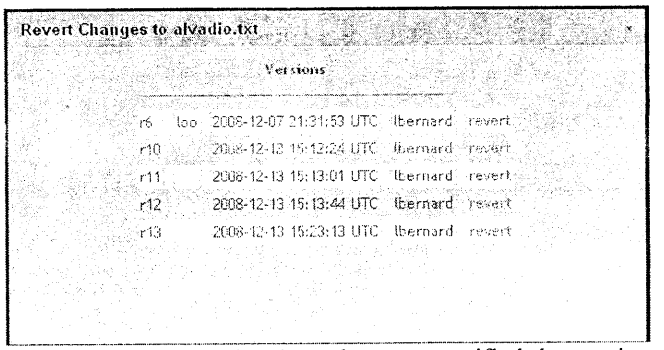
From this screen the user can add a new file to the project. I check is made to see if the file already exists in the proj_file model. When they clicked the upload link the commit method in the project_controller will then add the file to the proj_file model and also save the content of the file as the first version of the file in the file_versions model.

CHECKOUT FILE



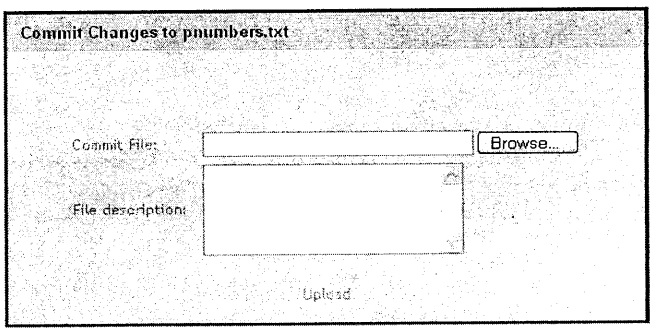
This screen is displayed when the user clicked the checkout link from the project details screen. User can save or open the current version of the file. User can also cancel the checkout task.

REVERT FILE



This screen is where the user specified the version of the file they want to revert. When the user clicked the revert link the revert method in the project_controller is fired. This revert method will update the file current version in the proj_file model.

COMMIT FILE

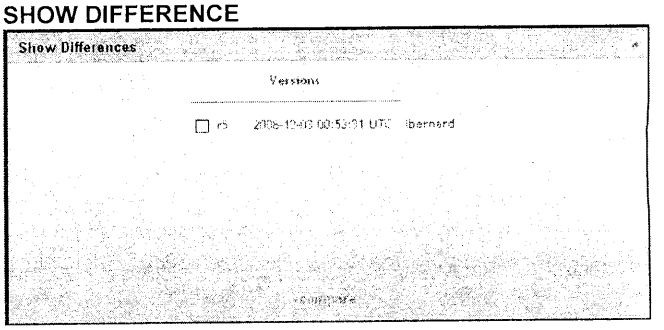


From this screen the user can select the file with the changes from their file system. When they clicked the upload link the commit method in the project_controller will then save the changes as a new version of the file in the file_versions model and also update the current version of the file in the proj_file model to match the commit file.

GET FILE

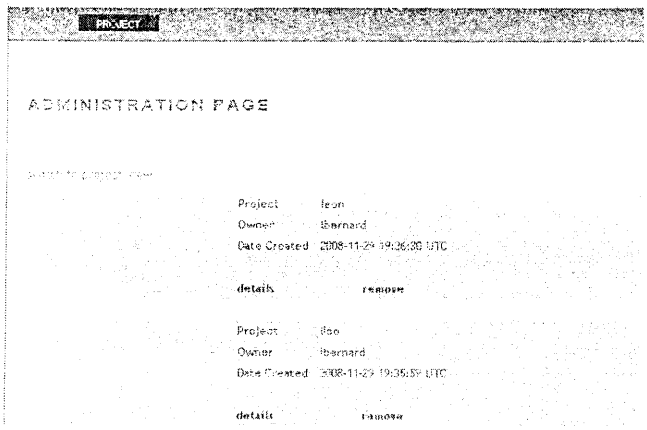
See Check out...

SHOW DIFFERENCE



The show difference screen allows the users to choose the version of the file they want to compare. The showDifference method is fired once the compare link is clicked. This method will store the versions of the file in temporary files and make a system call which highlight the differences between the two versions of the file

ADMINISTRATION PAGE



This screen is only visible to the site administrator. From the screen the site administrator can perform remove a project and also display the project. The site administrator can also switch to the project view by clicking the link in the top left corner of the screen. This will take the site administrator to the main project view which is seen by the project admin and regular users. The site administrator can perform all the operations performed by both project administrator and users.

JAVASCRIPTS, AJAXS AND CSS

Javascript in combination with ajax is heavily used throughout the TVC web application to give the users a more responsive desktop like interface. A few specific examples are displaying a project's detailed information, adding or deleting users, and revert and show difference methods. Cascading style sheets (CSS) were used to give a cohesive pretty design to the TVC application.

CONCLUSION

TVC provides a robust environment to manage file and their versions. It's developed using several technologies and provides an integrated solution to project development. TVC illustrates how to successful developed a web application using agile development methods. The model-view-controller framework is extensive used in this project and improves the application's scalability. TVC performance is relatively fast due to the ORM called ActiveRecord provide by Ruby on Rails.

TrackMyDiet – A diet and fitness tracker application

Maria Gizinski
Towson University
1717 Cannongate Rd.
Forest Hill, MD 21050
410-967-4867
maria_dia@hotmail.com

Kai Zhu
Towson University
306 Garden Rd,
Apt B, Towson, MD-21286
410-688-1206
zhukai85@gmail.com

Sowmini Gundamraju
Towson University
7911 Knollwood Rd,
Apt C, Towson, MD-21286
571-213-9316
sowmini2087@gmailcom

ABSTRACT
The course outline discusses the significance of developing fast, rich and high quality Web 2.0 applications using modern software development techniques. To accelerate the application development process and to increase its adaptability and flexibility to changing requirements and delivery schedule, Agile Programming techniques were adopted. To understand the current web application UI and database design standards and to be on par with the current web technologies, we developed a database driven web 2.0 application called 'TrackMyDiet'. This application allows the users to track their daily diet and workout routines as well as creating and tracking body progress goals. The application allows users to collaborate with different related communities by associating user profile to his/her Facebook account and updating their status on Facebook. The application was developed in an incremental iterative fashion thereby following the agile standards.

Categories and Subject Descriptors
H.3.5 **[Online Services]**: Web Development – *Web-based services, Ruby on Rails.*

General Terms
Design, Human Factors, Languages.

Keywords
Diet tracker, Ruby on Rails, Agile programming. screen scraping.

1. INTRODUCTION
TrackMyDiet is a Web 2.0 application that allows the users to keep track of various food items consumed by them daily. It provides the users with a comprehensive list of food items extracted from various public sources on web. The food recipes data extraction is performed using web screen scraping scripts.

It is a diet-cum-fitness tracking online application that facilitates the user to keep track of his/her workouts. The application is designed in a user-friendly manner allowing the users to choose a specific workout and creates an exclusive workout plan for each user thereby tracking his exercise progress.

The application maintains a personal profile for each user wherein it stores all the contact information and specifications of the user. It also allows the users to create goals for themselves providing them a platform to update and view their goal periodically. It calculates the daily calorie intake of the user and presents a detailed report on the analysis.

The application consists of an extensive workout module that helps the users to track daily workouts by choosing an appropriate exercise from the exercise database and any associated information needed to help the user track their progress.

The outstanding feature of the application is its mashup with a popular social networking site - Facebook. It provides the users with a 'communities' corner wherein the user collaborates with other health related communities and posts his comments. The communities feature allows the user to login to Facebook and post his/her comments on the wall thereby providing an interface to connect to facebook.

2. PURPOSE
A person's diet and exercise is a personal part of their life and is an activity that people perform throughout their day. Most users are not always near a desktop or even a laptop computer to go to a website to record information. When tracking things like what a user eats or the kind of exercise they are performing it is more effective to record this information right away and not rely on human memory. Most users carry cell phones which if they have access to the web can login in to TrackMyDiet site and record information right away such as after eating a meal or after performing a series of strength exercise. The more accurate the information the more helpful it is to the user. Considering this fact, we decided on developing a diet tracking application that provides a user with the knowledge of the diet consumed daily and the nutritional value. It informs them about the fitness available and their benefits and motivates them to create fitness goals and work towards it. For the purpose of functionality and usability, a mobile version of this application is in progress. Due to the increased use of mobile technology in the current world, a mini-version of the application was developed. It displays an exercise catalog with the feature to add an exercise to the catalog. Due to time constraints no tracking functionality has been developed, however the potential is there for it.

3. CONSTRAINTS
A major constraint for developing this application was the lack of availability of food recipes data in the required format. To suffice this, a web spider script was written in Ruby to extract the required information from multiple online sources. Since it was difficult extracting the required fields for a specific food item, many data inconsistency issues had to be handled which led to change in design and implementation decisions.

The design decisions varied over the course of development process because of the mobile browser views. It required us to develop a new application that was compatible for viewing in mobiles.

4. REQUIREMENTS

4.1 Hardware

The TrackMyDiet does not require any special hardware components. The application will run comfortably on any platform with default configuration

4.2 Software

Ruby on Rails web framework was used to develop TrackMyDiet application along with MySQL database. Rails is an open-source web framework written in Ruby programming language and is a extensive, self-contained, transactional SQL database. Database can be changed to SQL Server, SQLite, or preferred database based on future need.

The application is configured to run on any platform with Ruby language version of 1.8.6, Rails framework version of 2.3.5, and MySQL 2.10.0.2.

In addition to the web framework, an open-source CSS template was used[1] and JQuery, a JavaScript library, version 1.4.2 was also used for better user interactions and client-side validations. JQuery plugins under GPL and MIT licenses were used to include features like jquery modal box, fading in and fading out were used.

'Facebooker' gem was installed to associate the application with Facebook. 'Hpricot' gem was installed to screen scrape food-recipe data from webpages. Other ruby gems faker and populator were used to generate users test data.

Since the project was developed in an Agile fashion, versions of code was maintained using two versioning softwares – Tortoise SVN and Git. There were few problems with tortoise SVN, so the project was later versioned using Git.

```
config.gem "authlogic", version => ">= 2.1.3"
config.gem "searchlogic", version => ">= 2.4.11"
config.gem "ramaze", version => ">= 1.0.2"
config.gem "will_paginate", version => ">= 2.3.12"
config.gem "paperclip", version => ">= 2.3.1.1"
config.gem "jammit", version => ">= 0.4.4"
config.gem "fantastic", version => ">= 0.9.7"
```

Figure 1 Gem list

5. DESIGN

5.1 Context

The application provides the users to view information about the functionality of the website and encourages them to join and create an account.

5.1.1 User

Users register and create an account to log into the web application. The application provides a 'remember me' option to store the user login information and generates it when logged in next time. This is done with the help of cookies.

5.1.2 User Specifications

Specs basically create a profile for the user to view and edit his/her account information. It also allows the users to navigate and use other features of the application like tracking nutrition, body and workouts.

The 'Track body' feature allows the user to create, update and view user's body specifications. It also allows the users to create a new body related goal by changing his/her height and weight specs. This information is updated and the user is notified if the goal is accomplished.

The 'Track nutrition' control redirects the user to a whole list of food items and provides many actions like adding a meal and so on.

The 'Track workouts' control redirects the user to log his workout information by selecting a whole list of exercises.

5.1.3 Nutrition

The nutrition module allows the user to select the food item that he needs to add and creates a user specific meal plan. The meal plan displays the user the meal type for a specific item and the date and time when he had the food.

5.1.4 Exercises

Exercises have two categories: Strength and Cardio. Different information is associated with each exercise type. A strength exercise puts more than normal strain on muscles to increase muscle strength.

Information related to this exercise would be Reps, Sets and Pounds Lifted. Reps is the number of repetitions the user performed the exercise. A set is a grouping of repetitions. Each set can have a different amount of reps. The pounds lifted is the weight that was used in the exercise. A cardio exercise is an exercise that increases a person's heart rate and uses large muscle movement over some period of time. The information related to cardio is Minutes, Distance and Calories Burned. Calories Burned is a number to represent the energy a user might have burned to perform this exercise. This burn of energy is what helps people lose weight. By keeping a record of this information the user can see where progress is being made or where improvements can be incorporated, for example adding an extra day of exercise

5.1.5 Blogs & Facebook Mashup

The blogs module was implemented in an advanced layout of the application. This allows the users to create blogs and posts. Each post allows the users to comment on them and the comments are specific to the related post. The Facebook mashup facilitates the user to post a comment onto his/her Facebook wall.

5.2 Data Structures

Structure of data was determined based on the requirement of capturing appropriate fields.

5.2.1 User

The User table contains general user login fields with appropriate field types.

id	int(11)
username	varchar(255)
firstname	varchar(255)
lastname	varchar(255)
password	varchar(255)
created_at	datetime
updated_at	datetime
email	varchar(255)

Figure 2 User table schema

5.2.2 Nutrition

Nutrition contains a list of edible food with appropriate calorie content and nutritional value.

id	int(11)
food_id	int(11)
genres	varchar(255)
itemName	varchar(255)
servingSize	varchar(255)
carbs	int(11)
fat	int(11)
calories	int(11)
created_at	datetime
updated_at	datetime

Figure 3 Nutrition table schema

5.2.3 Exercises

Exercises contain a list of exercises names and exercise types allowing the user to choose the one he desires.

Field	Type
id	int(11)
exerciseName	varchar(255)
exerciseType	varchar(255)
description	varchar(255)
imageUrl	varchar(255)
created_at	datetime
updated_at	datetime

Figure 4 Exercise table schema

5.2.4 Meal Plan

Meal Plan table stores all the food items consumed by the user daily and also stores the meal type, that is, breakfast, lunch etc. associated with it.

Field	Type
id	int(11)
user_id	int(11)
nutrition_id	int(11)
mealtype	varchar(255)
created_at	datetime
updated_at	datetime

Figure 5 MealPlan table schema

5.2.5 Workouts

Workouts contain the type of workouts done by the user at a particular time of day.

id	int(11)
useridFK	int(11)
exerciseidFK	int(11)
date	date
timeofday	varchar(255)
sets	varchar(255)
reps	varchar(255)
lbs	varchar(255)
minutes	varchar(255)
distance	varchar(255)
caloriesBurned	varchar(255)
created_at	datetime
updated_at	datetime

Figure 6 Workout table schema

5.2.6 Specs

Specs tables contain the profile information along with other body related information of the user.

id	int(11)
user_id	int(11)
first_name	varchar(255)
last_name	varchar(255)
gender	varchar(255)
birthdate	date
height	int(11)
weight	int(11)
city	varchar(255)
state	varchar(255)
zip_code	varchar(255)
weight_goal	int(11)
height_goal	int(11)

Figure 7 Specs table schema

5.3 Architectural and Component-level Design

5.3.1 Architecture

The TrackMyDiet application was developed using Model View Controller architecture. For the purpose of utilizing MVC architecture, project was coded using Ruby on Rails.

Model View Controller(MVC) architecture was chosen as it best suited TrackMyDiet website requirements rather than traditional tier-architecture websites as data accessed through a single model of user could be used to selectively display content depending on input. MVC allows for finer control of data and user interaction with that data. In a MVC architecture, all browser requests are parsed and routed to controller which then decides whether to get data from suitable model or whether to render view.

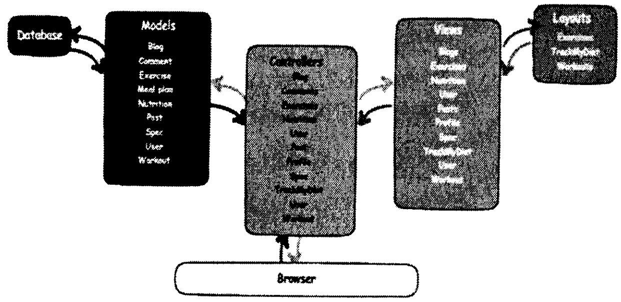


Figure 8 MVC Architecture of Track my diet

Structure of data was determined based on the requirement of capturing appropriate fields.

5.3.2 Components

Data associations are defined in models of TrackMyDiet application. Figure depicts data associations in models.

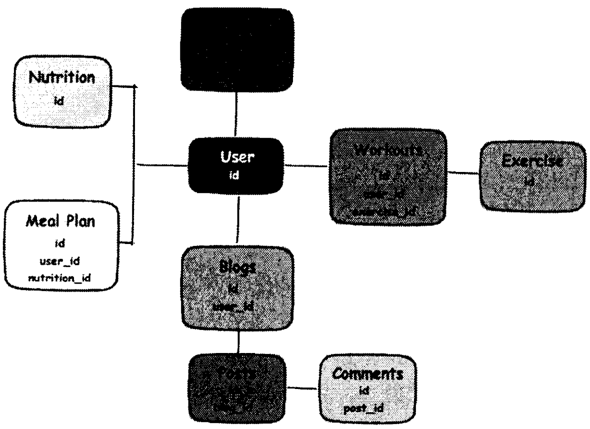


Figure 9 Model associations in track my diet

5.3.3 Screen scraping

The application required extensive food and nutritional data and to resemble it closely to a real-world application, we ran ruby scripts that downloaded the web pages containing the related data and extracted the required data fields required for our application. Data fields like food item names, serving size, calories, carbs and fat related to a particular food recipe were extracted. In order to accomplish this task, a ruby gem called "Hpricot" was used to extract data.

```
# This Ruby code downloads the webpage that you're trying to scrape

require 'open-uri'
require 'rubygems'
require 'hpricot'

doc = open("http://whatscookingamerica.net/NutritionalChart.htm") { |f| Hpricot(f) }

title = (doc/"head/title")[0].inner_html

(doc/"a").each do |a|
  a.after(' ' + a.get_attribute('href') + ' ') if a.has_attribute?('href')
end

# Writes data to 'testHtml' file
open('testHtml.out', 'w'){ |f|
  f.puts doc
}
```

Figure 10 Ruby script to download webpage

```
# [illegible]
# [illegible]

require [illegible]
require [illegible]

# [illegible]
# [illegible]
doc = [illegible](open("[illegible].html"))

# [illegible]

print "[illegible]ITEMS[illegible]"

# [illegible]
items = (doc/"html/body/div/center/table/tr/td[@width=543]/font")
valueText = items.inner_html.gsub("<br />", "").gsub("<strong>","").gsub("</strong>","").gsub(" ","")
open('items.out', 'w'){ |f|
    f.puts  valueText
}
print "ITEMS copied"

print "[illegible]SERVING SIZE[illegible]"
servingSize = (doc/"/html/body/div/center/table/tr/td[@width=543]/font")
valueText = servingSize.inner_html.gsub("<br />", "").gsub("<strong>","").gsub("</strong>","").gsub(" "
```

Figure 11 Ruby script to scrape data form webpage

6. USER INTERFACE

Two different layouts were developed for TrackMydiet application in order to make the web application compatible to view both in mobile and on system browser.

The first layout is a simple and fully-functional application layout containing minimal javascript and mainly targeting mobiles. The second layout is a full-fledged web 2.0 application with all the fancy elements of ajax, javascript and mashups.

Figure 12 Trackmydiet homepage

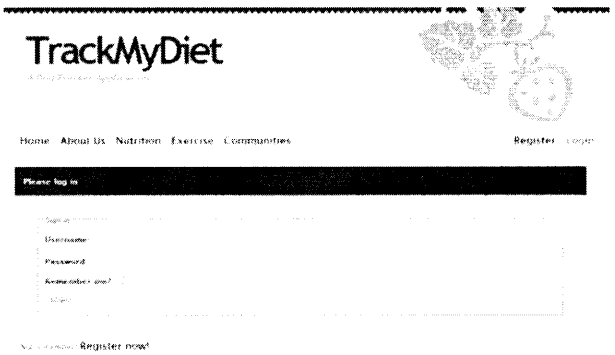


Figure 13 User Login page

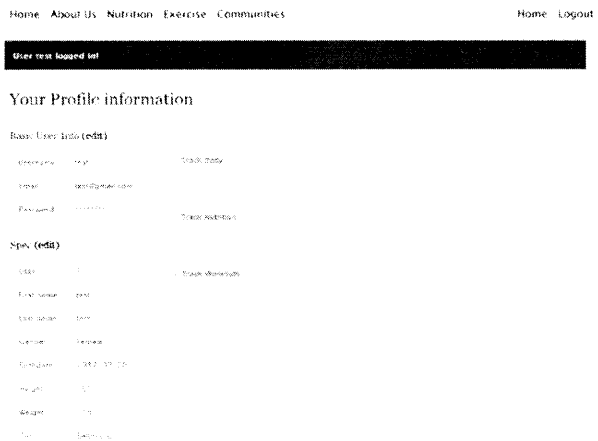


Figure 14 User specs

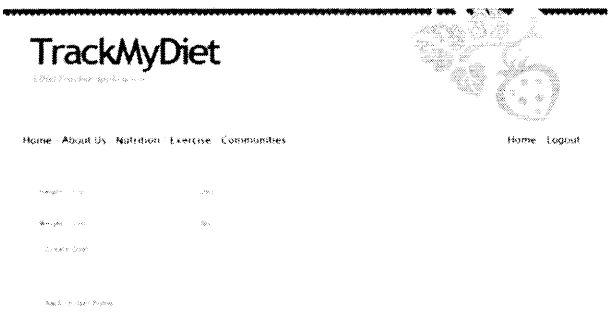


Figure 15 Create/Update goals

Figure 16 User body specifications

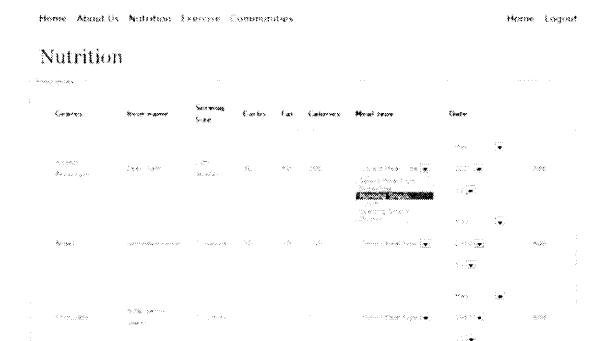


Figure 17 Add food item to meal plan

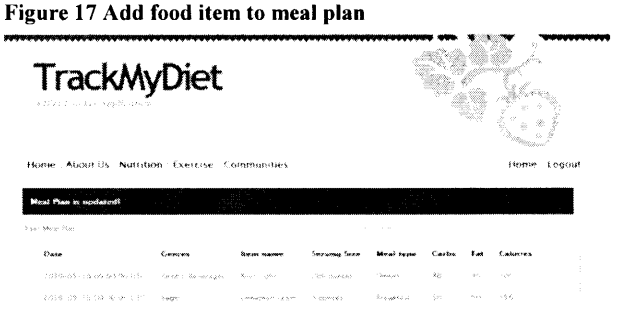


Figure 18 View added item

Figure 19 Exercises list

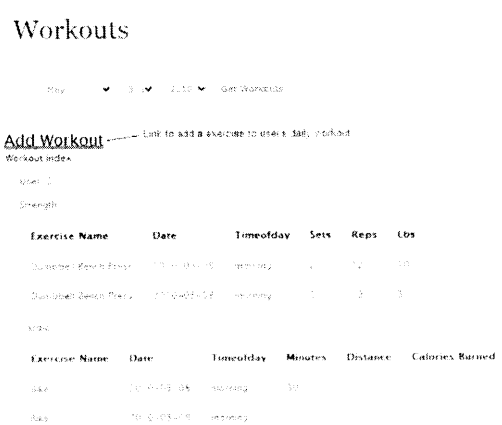


Figure 20 Add workouts to user schedule

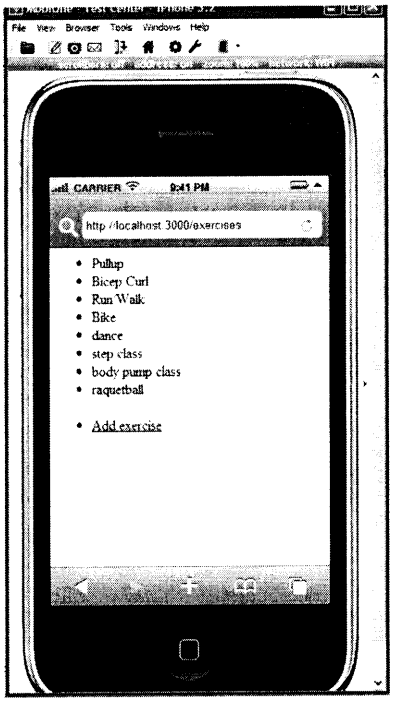


Figure 21 Mobile view of Exercises module

\

7. SECURITY

Security in TrackMyDiet application was provided through various methods.

7.1 Database

Rails inbuilt HTML escaping method was used when displaying any database retrieved data so as not to run any harmful code when viewing data. Rails escaping method also provides certain amount of security against Cross-Site-Scripting(XSS) attacks.

```
<table>
    <tr>
      <td class="label">Username:</td>
      <td><%=h @user.username %></td>
    </tr>
    <tr>
      <td class="label">Email:</td>
      <td><%=h @user.email %></td>
    </tr>
    <tr>
      <td class="label">Password:</td>
      <td>********</td>
    </tr>
</table>
```

Figure 22 Html escaping to prevent XSS

Database table field types were set based on the data they are supposed to contain. Before insertion, type of data being inserted is also validated to ensure that incompatible or incorrect data will not be pushed onto the database.

User and committee member passwords are hashed using MD5 to maintain password integrity. SSN is encrypted using AES 256 key encryption with a secret key salt.

```
private
def secure_unique_identifier
    Digest::SHA1::hexdigest("#{username}:#{password}")
end
```

Figure 23 Encrypting user password

7.2 URL

MVC architectures map url's to controllers, if needed, models, and then to views. To disable the user from navigating to restricted sections without appropriate authentication, methods in controllers are protected by ensuring that they are not called without valid credentials, available through sessions.

To provide further security, even if a browser url is mapped to a valid controller method or view, access is blocked as the origin of the request is designated as untrusted. Session variables that are hard to tamper with are used to generate any request from a controller.

7.3 Routing

Rails routing system interprets a request URL and generates the controller action for that particular URL. The routing system performs this by using routing rules which are defined in config/routes.rb file

Figure 24 Application's routes

.

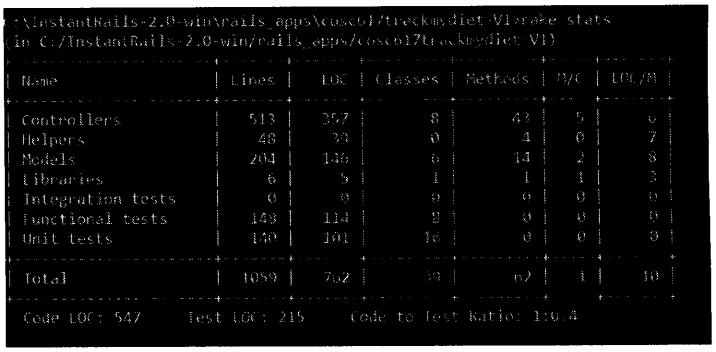
8. TESTING AND VALIDATIONS

8.1 Testing

Functional tests and Unit tests were written using rails for user module inbuilt test base. Tests ensured that all layouts and views were being successfully rendered with HTTP success code of 200, along with tests of correct title tag data and correct notifications regarding user/member actions.

Valid and invalid user logins was tested using data supplied through Rails fixtures. Application creation, edits, and updates were also tested.

The code versus test code ratio for the application is shows below using '`rake stats`' command:



Name	Lines	LOC	Classes	Methods	M/C	LOC/M
Controllers	513	357	8	43	5	6
Helpers	48	33	0	4	0	7
Models	204	146	6	14	2	8
Libraries	6	5	1	1	1	3
Integration tests	0	0	0	0	0	0
Functional tests	148	114	2	0	0	0
Unit tests	140	101	10	0	0	0
Total	1059	762	30	62	1	10

Code LOC: 547 Test LOC: 215 Code to Test Ratio: 1:0.4

Figure 25 Code versus test code - rake stats

8.2 Validations

Validations were enforced in models, before data was saved to database.

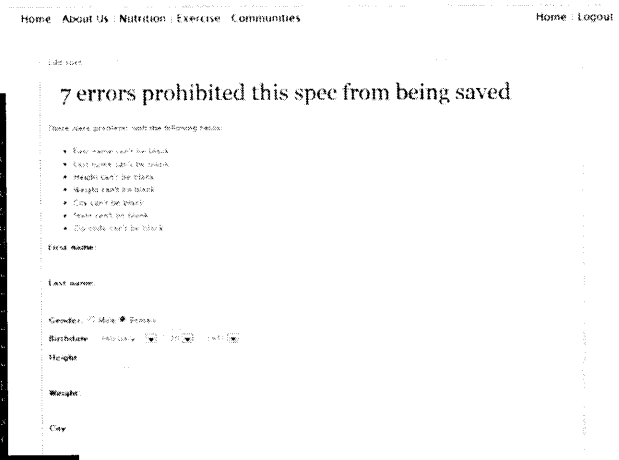


Figure 26 Rails model validations

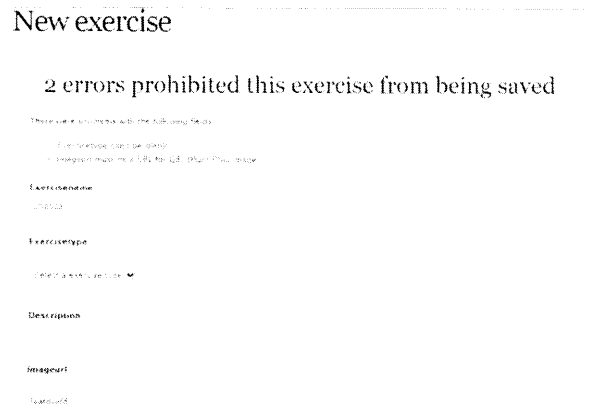


Figure 27 Rails model validations

9. FUTURE DEVELOPMENTS

The application has a great scope of development in future as lot of progress can be done on improving the layouts of the mobile views. The database can be updated with more data by creating and implementing more generalized scraping scripts.

Due to time constraint, some of the features were not implemented. The next version of this application can have features like providing the user with permissions to add data into the database. The layout of the application could be improved to accommodate the whole month diet schedule of the user.

The blogs module currently functions at the user level, so an administrator module can be added so that the user accounts, posts and comments could be managed thereby improving the functionality.

The code versus test code ratio could be improved by adding more functional, unit and integration tests.

10. CONCLUSION

It was a great experience developing a diet tracker application on Ruby on Rails environment using Agile programming methods. The MVC architecture of Rails speeded the build process and agile practices greatly contributed to the development process by ensuring a working prototype at any time during the development cycle.

11. REFERENCES

[1] Micheal Hartl, Aurelius Prochazka 2008. RailsSpace: building a social networking website with Ruby on Rails.

[2] Clark, Mike and Rails Community May 2008. Advanced Rails Recipes, The Pragmatic Bookshelf, Raleigh NC and Dallas TX

[3] Online Food and nutritional source. http://whatscookingamerica.net/NutritionalChart.htm

[4] Hpricot gem. http://wiki.github.com/hpricot/hpricot/

[5] Facebooker gem. http://facebooker.rubyforge.org/

[6] MobiOne - iphone emulator for windows http://www.genuitec.com/mobile/.

[7] Abrahamsson, Pekka, Salo, Outi, Ronkainen, Jussi, & Warsta, Juhani. Agile software development methods. http://www.pss-europe.com/P478.pdf

[8] GitHub version control system. http://github.com/

[9] Tortoise SVN. http://tortoisesvn.tigris.org/

CardMarkets.com - A Sportscard Trading Website

Ronald M. J. Erdman
Towson University
7800 York Road
Towson, Maryland 21204 USA
+1 443 935 7844
ron@rmje.com

Dandan Maggie Wu
Towson University
7800 York Road
Towson, Maryland 21204 USA
+1 646 262 6994
maggie_wdd@msn.com

ABSTRACT
This paper provides the background, design, and implementation information associated with the creation of CardMarkets.com, a website created using Ruby on Rails.

Keywords
Website, Ruby on Rails, Sports, Sportscards, Collecting, Trading,

INTRODUCTION

Collecting and trading cards - particularly sports-related cards - are one of the oldest and most popular hobbies in the United States and beyond. Originating in the late 1800's, the hobby has become big business for the companies engaged in producing cards. In an age when collectors collectively dump hundreds of millions of dollars into trading cards and can connect with one another instantaneously over the Internet, one would expect that technology would already be embraced to facilitate the mundane and logistical elements of the hobby. However, such is surprisingly not the case. Hobbyists are faced with a wide array of information, tools and resources needed to acquire, sell, trade, value and organize their cards, as well as a myriad of communications channels available for connecting collectors with one another. Each of these is distributed among multiple websites, making the hobby quite labor intensive. For example, a collector might buy cards through a hobby store's online storefront, bid on cards in auctions on eBay, and search for other collectors with whom to trade using Facebook, hobby-related websites, or an endless number of collector-focused online message boards. Keeping track of one's collection requires access to the manufacturers' websites, where one can find checklists detailing all of the cards made in each set for each year of issue, as well as some means for storing these checklists and the collector's inventory of cards. Other online services provide updated pricing information for nearly every card in circulation, allowing collectors to gauge relative value of cards they may be unfamiliar with, or better understand the demand for specific cards among collectors. The net result is that the hobby involves quite a bit of manual labor that could easily be consolidated, collocated, and automated into a single solution. This is the idea behind CardMarkets.com - a "one-stop" hobbyist destination for everything involved in collecting and trading cards of any type.

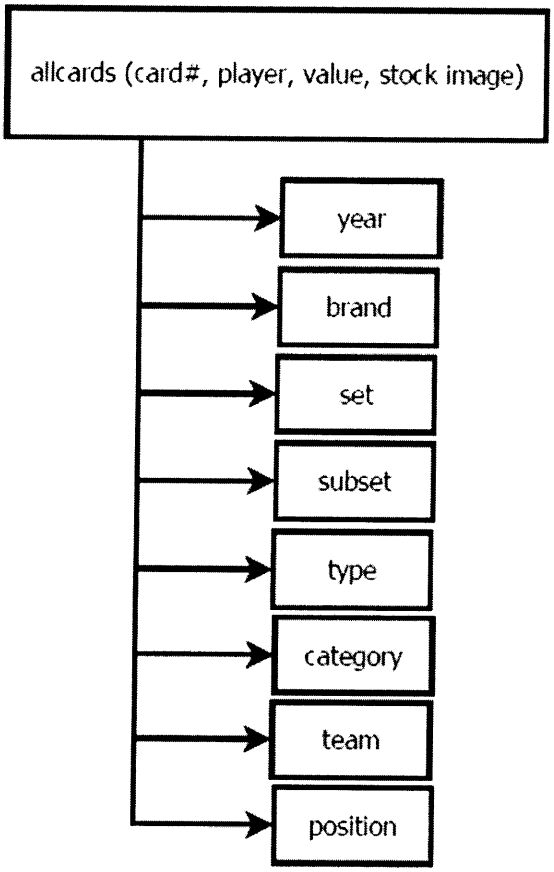
CONCEPTUAL DESIGN

The chosen design solution is centered around three primary data models which drive the website. These include a catalogue of all the different cards that have ever been produced, an inventory of all of the physical cards collectors using the site are keeping track of, and the users themselves. Each of these key models are discussed in detail below:

The Catalogue of All Cards Made (allcards)

Key information regarding the cards available for collecting form the basis of the website. The data for this model includes both relevant attributes of each card (such as the name, team and position of the player(s) depicted on the card, the card number, the year of issue, the brand of the manufacturer, and other similar identifying information. It also includes a proxy for the value of each card - a rough idea of the range of prices for which transactions on that card could be expected to take place. All of this information is important for a variety of reasons. Some collectors collect only specific players, while others collect specific teams. Conversely, other collectors are set-collectors - they seek to collect one or more of every card made for a given year, brand, set, and or subset. Others focus on different types of cards, such as autographed cards, cards featuring game used materials (a swatch of material from a game-worn jersey, for example), or cards made with special attributes such as refractors (made with a reflective surface which bends the light). The important concept is that there are many collectors, and each is searching by a different method for the cards he/she collects. So all of this information must be collected and implemented if it is to be useful to all collectors. This is depicted in Figure 1 on the following page.

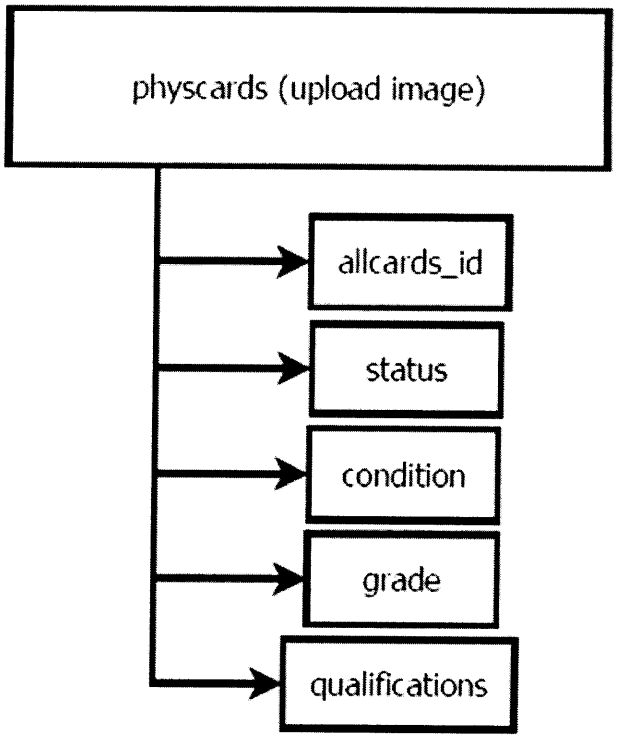
Figure 1



The Inventory of Collectors' Physical Cards (physcards)

To meet the needs of collectors, who have to keep track of the cards they have in an efficient and expedient way if they are to minimize the labor associated with the hobby, a website must provide an interface for a collector to quickly track and manage the cards he/she has. The physcards model captures this information, with each physical card representing a real-world instance of a single card manufactured. In other words, physcards stores the instances of the allcards model. Each physcard has characteristics specific to the real card in the collector's possession (or one that he seeks to acquire), such as its physical condition, grading, grading qualifications, and a means of indicating a collector's interest in the card (whether he/she wishes to keep this card, would be willing to trade it for another card, or whether he/she doesn't actually have the card but is interested in trading other cards in his/her collection to acquire it). Each physcard *also contains* a foreign-key attribute pointing to the card in the allcards catalogue of which it is the physical instance. All of this information can be seen in Figure 2 as follows.

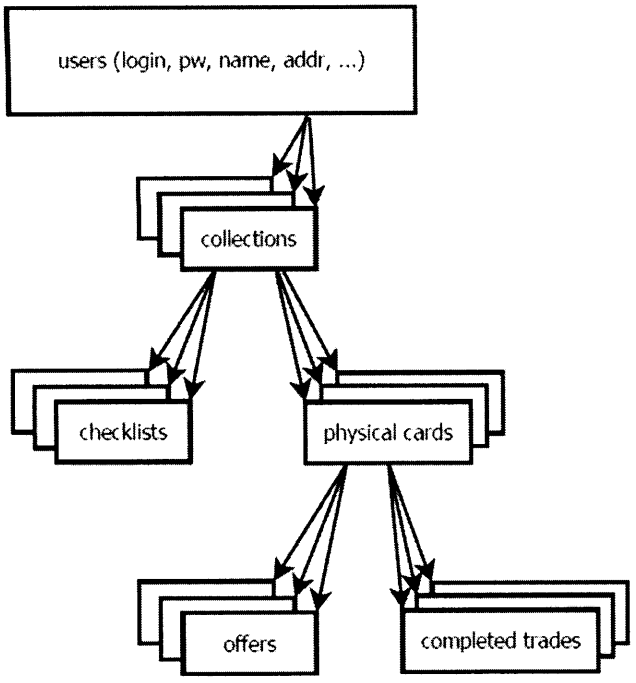
Figure 2



The Collector (users)

Websites are intended for users - the people who visit them and use them. Information on users, from basic personal details such as name, address and contact information, to login credentials, must be stored for retrieval on successive visits, especially if we wish to associate a collector's collection information with the user. The user then becomes the central entity in the data model, to whom all other information is related. For example, a user can have one or more online collections (a means for organizing his/her larger physical collection in a meaningful and useable way). Similarly, a user can have one or more checklists associated with each collection for tracking which cards are available for him to collect, as well as a means for tracking the actual physical cards he/she owns (physcards). Each physical card, in turn, can be involved in one or more offers, which when agreed to by the corresponding user, become completed trades. In this way, the entire process, including all tools, information sources, and means for transacting exchanges, is captured in a series of related models. See Figure 3 on the following page for a clearer understanding of the users model and its relationship with other key models.

Figure 3



DATABASE SCHEMA

The above primary models, as well as a variety of supporting models, are introduced in Appendix A - Database Schema. This graphical representation provides not only the data model entities, but also all necessary attributes associated with each. It also outlines all relationships established between entities.

USE CASES

Any good website, and especially one meant to serve all the needs of its users in a consolidated way, must identify, understand and address all of the use cases involved. The following use cases were identified in the development process for this website:

- New User Registers
- Existing User Logs In
- User Creates a Collection
- User Adds/Removes Checklists to/from Collection
- User Manages Cards in Collection
- User Looks Up Card Values
- User Searches for Cards to Trade (away/for)
- User Proposes Trades to Other Users
- User Accepts/Declines Trades Proposed by Others
- Administrator Manages All Models/Functions

OTHER FEATURES

Authentication

The website provides a secure means for creating an account, allowing a user to log in, as well as a means for logging out when the visit is completed.

Security

Sensitive information, such as user and administrator passwords, is encrypted (a unique plain text password is provided by the user upon registration, and then combined with a unique salt value, and an SHA1 digest is run on the resulting combination, returning a 40-character string of hex digits, which is then stored in the database) for the protection of both. Similarly, administrator areas should be restricted to only those users designated as such. General users should be limited to the areas designed for them.

User Maps

A mapping interface is (intended to be) provided to assist users in identifying collectors in their area. This could be used to facilitate local communications, in-person exchanges of online-generated trades (to save postage costs), or simply to facilitate relationships with other collectors in your area.

Relevant Current Information (Statistics & Reports)

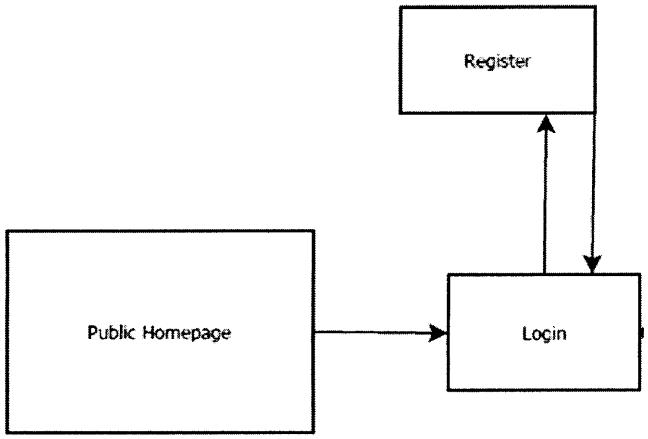
Websites that are based on transactions (this one on trade transactions) generate a wealth of information, which if organized properly, offers significant value to its users. An example of this has been provided on the homepage, providing a current list of recently executed trades. This makes collectors aware of what cards are actively trading between users, as well as which users are actively engaging in trades. Any number of similar reports and statistics could be included for the benefit of users.

IMPLEMENTATION

This website has been implemented based on the following conceptual design components. They are segmented here for discussion purposes, but included in their entirety in Appendix B.

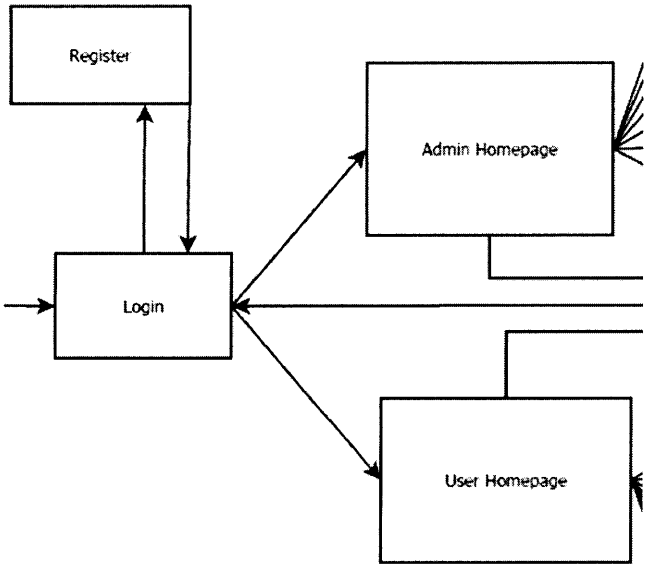
Homepage & User Login/Registration

Users are directed initially to the landing page of the website, which is /public in Rails. This contains the public/index.html.erb page, which provides both the public face of the website and an interface for users to create accounts and login.



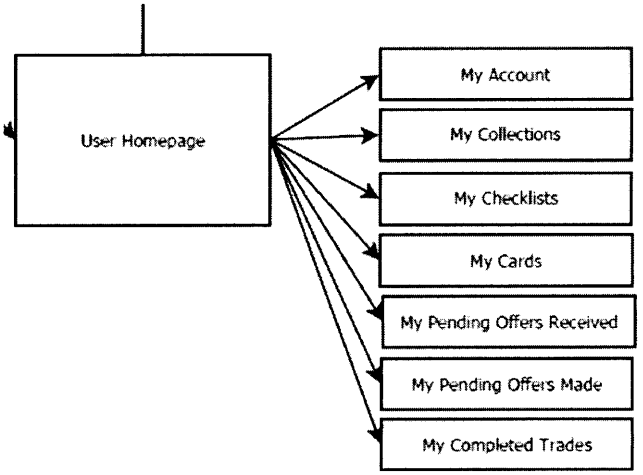
User/Administrator Areas Separated by Usertype

An attribute of the users model, which can only be altered by an existing administrator, defines whether a user is an administrator or a general user (the default is the latter). This determines whether, upon login, a user is directed to the general user homepage or the administrator's homepage.



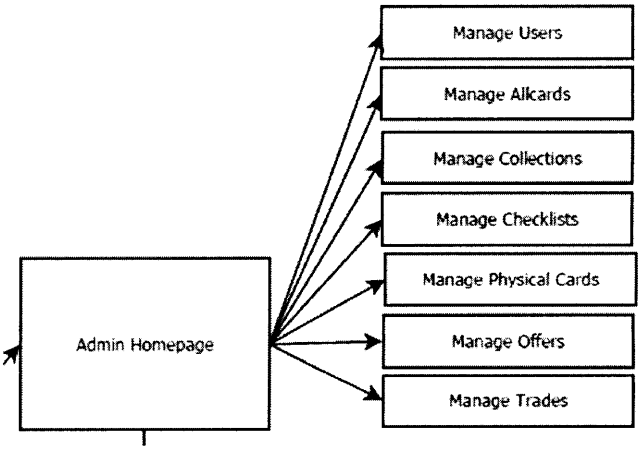
User Functionality

Users are provided with links on the user homepage which enable all of the functionality detailed in the use cases above.



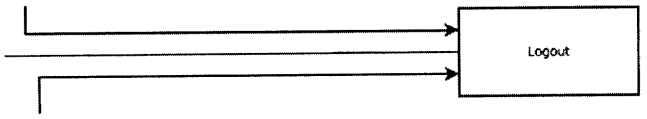
Administrator Functionality

Similarly, administrators are provided with links on the administrator homepage which enable all of the functions seen by the user, but with added scope and more functionality and detail. This is necessary for management of the database, problems that arise in transactions (be they technical or otherwise), and for assisting users who might not have a clear understanding regarding how to accomplish certain tasks.



Logout

A means is provided, through a link at the top of the page, for users of any type to finish their session and securely log out of their account, thereby completing their visit.



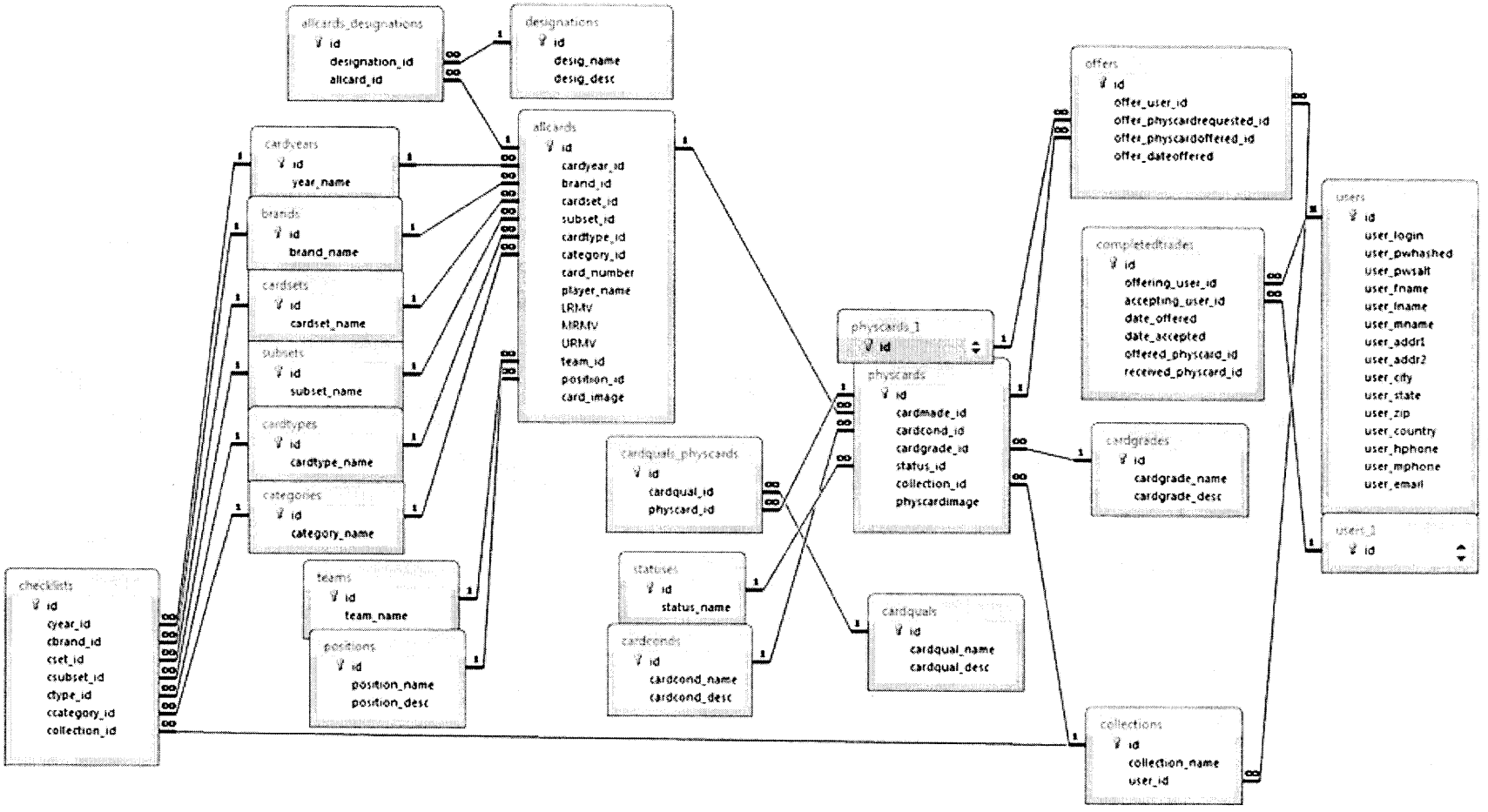
ACKNOWLEDGMENTS

A special thanks to Sowmini, who was kind enough to guide me in different means of extracting image data from websites in order to make that available for this purpose.

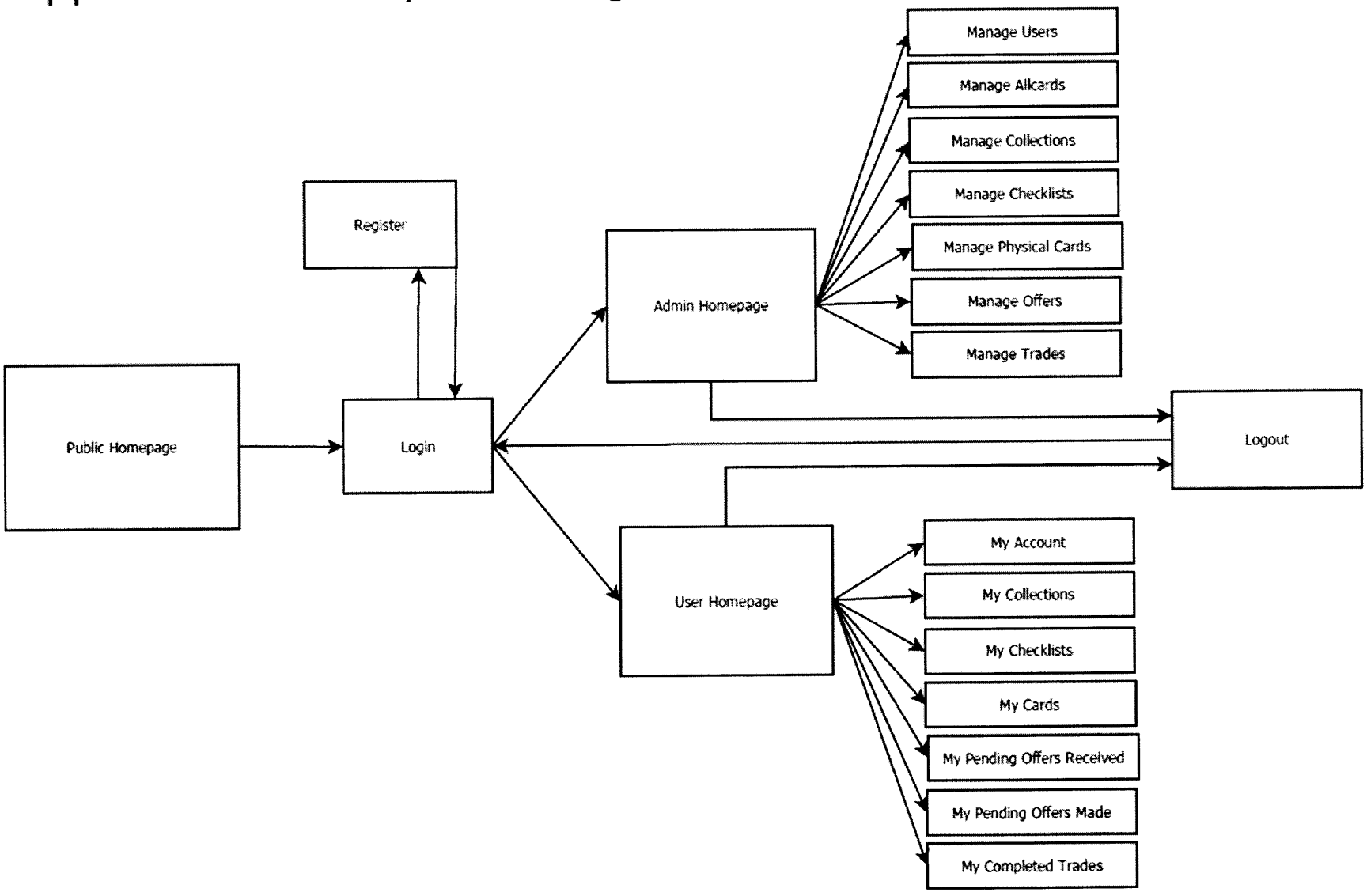
REFERENCES

1. Thomas, Dave & Ruby, Sam. *Agile Web Development with Rails, Third Edition* (2009), entire book - especially the parts on encryption, sessions, and multiple-table forms.
2. Tutorial on Dynamic Select Boxes. Available at http://pullmonkey.com/2008/3/30/dynamic-select-boxes-ruby-on-rails/ - attempted, but not accomplished.

Appendix A - Database Schema



Appendix B - Conceptual Design



"MeSan"- A social networking site

Hui Liu, Menaka Mishra
Computer and Information Science Department,
Towson University
7800 York Building, Towson, MD -21252, USA
hliu3@students.towson.edu
mmishr1@students.towson.edu

Ripal Shah, Somchai Srikitipraphas
Computer and Information Science Department,
Towson University
7800 York Building, Towson, MD -21252, USA
rshah3@students.towson.edu
somnaluk@gmail.com

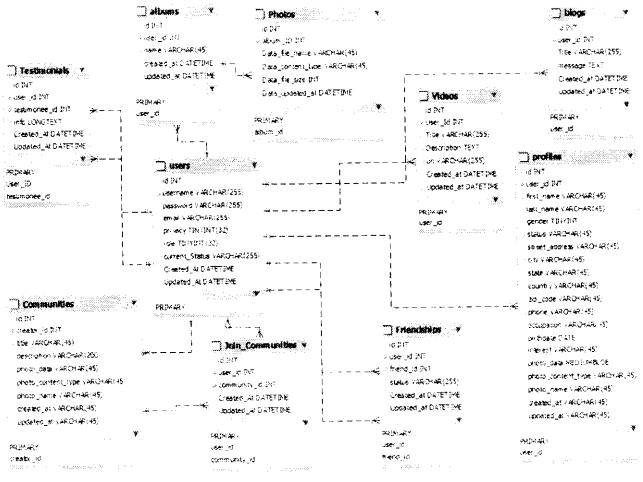
ABSTRACT
In this article, we intend to discuss a social networking site build in ruby and rails which enables quick networking with people online. This social site enables us to connect and communicate with our friends, family and others in a distributed networks world-wide while providing features like usability, reliability and security. Also, in this paper we shall discuss in detail hardware/software configurations, technology and tools along with databases which were used in designing, implementing and testing this website along with the approach we adopted in constructing it. We also shall be discussing all the plug-in's used in building this application. Finally, this paper will talk in detail about all the MeSan functionalities and its successful test cases for each module built.

Keywords
Ruby and Rails, MeSan, MVC architecture, Face book CSS, YouTube embedding, Ajax, ImageMagik, RMagik, Acts_as_ferret, MashUps, Geokit.

INTRODUCTION
What is social networking site? Most of must be familiar with this term but are not quite sure what it means. The social networking is basically, grouping of individuals into specific groups, communities and organizations in person at workplace, universities, high school, it is most popular online. This is because at these places, the interest is filled with millions of individuals who are looking to meet other people, to gather and share first hand experiences about any number of topics like weather news, sports, developing friendships and professional alliances. When it comes to online social networking , websites are commonly used and functions as a online community for internet users. We have build **"MeSan"** as a social networking site. The term "MeSan" means contacting and establishing communication with people and sharing common interests with them.
The MeSan uses Ruby and Rails framework which is based on ruby language. It uses MVC(model, view and controller) architecture style for connectivity. The *'model'* represents an application data and contains logic for accessing and manipulating the data. The *'view'* is responsible for rendering the state of the model and *'controller'* is responsible for intercepting and transforming user input into action to be performed.

The MeSan users must be able to register to the MeSan website by signing up at www.mesan.com . MeSan uses Facebook cascaded style(CSS) sheet as its environment. Once the user had authenticated MeSan account he/she must be able to login with valid username and password. User should be able to update his profile information and profile photo and be able to view his friends profile. The user should be able to add friends from the list of suggestions or can find friend in full-text search box by entering friends name. The user might write blogs, may comment on others blogs and may respond to his own blog comments also add new blogs, delete a blog and show list of blogs, upload photos and albums, videos, join communities and write testimonial to his beloved ones. The most recent activities of the user should be displayed as feeds on the user page. We have used various plug-ins like, RMagik, ImageMagik, Paperclip, Acts_as_ferret, MashUps(Google API), Ajax and HTML. We also used MySQL and MyPhp Admin databases for storing our tables.

DESIGN
For our social networking site –MeSan, we have build class diagram using tool MySQL Workbench to give us overall view of how the intended web application should have looked like. The figure 1 represents ER diagram for MeSan below: (Figure1)



HARDWARE

We used Windows Vista/XP with wireless connectivity.

SOFTWARE

We used Ruby and Rails as a web framework called Instant Rails for building MeSan which is a ruby language based. It has a Apache server with needs databases like MySQL/PhpAdmin connectivity. It works on all browsers like internet explorer, Mozilla Firefox and Google Chrome.

We required integration development environment(IDE's) like Komodo Active 1.5 or Edit Plus 5.0 for writing our codes. It also uses Hypertext Markup Language(HTML) and JavaScript for MeSan view display.

IMPLEMENATION

The MeSan has been built on Instant Rails framework with Apache Server running all the time. We also used open source Active Komodo1.5/Edit Plus5.0 as IDE's for writing the code with MySQL database connectivity. For this project we created a new database called "mesan_development" for storing all the new tables created and data files as text fields.

In the config folder of MeSan we have database.yml file which we edited with the same mesan_development database name. As we determine all the functionalities our social networking site might be needing in the class diagram, here we list all the MeSan key functionalities which make it a very user-friendly, reliable and secure website with platform independence and robustness.

The MeSan functionalities includes:

1. Home
2. User
3. Profile
4. Friends
5. Blog
6. Photos
7. Albums
8. Videos
9. Community
10. Testimonial

Let's see how each functionality is implemented:

1. HOME- The Home page is a default page that consists of two parts. The first one is a signup part where people can register to receive the username and password. The MeSan application does not make this more difficult by requiring more information from people who want to create a new user account. The MeSan requires the username, email address, and password. People can create their own username and password. There are no criteria for this creation. However, if the username already has been used by someone, the application will deny the creation and display the appropriate message to the user in order to come with the different username. The other required in the signup part is the security check known as the CATCHA. Using the CATCHA ensure that the creation of the user account will not be generated by a computer which is considered as the attack. The MeSan shows the CATCHA in form of the number. Along with the username, the email and the password provided, the number typed in must match the CATCHA shown above the field. To randomly generate the CATCHA, the MeSan uses the software called RMagik to convert the text into the number as the image. ImageMagik, which is a rails plug-in for the CATCHA, then is used to display that CATCHA on the web page. Figure 2 illustrates the MeSan home page.

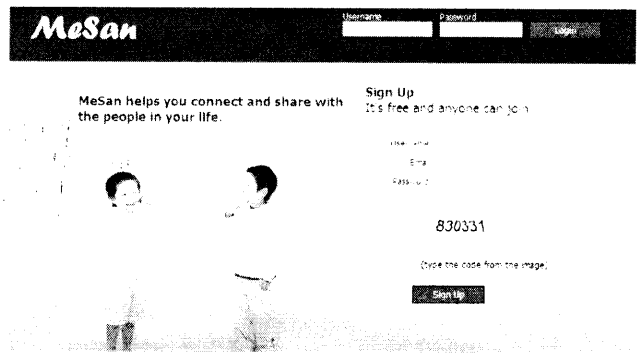


Figure 2. The MeSan Home page.

In addition to the signup part, the login functionality is also part of the home page located on the top bar (see Figure 2). This login functionality is used for authenticating the user before accessing the secure page. The user needs to provide the valid username and password. Otherwise, the error message will be displayed as depicted in Figure 3.

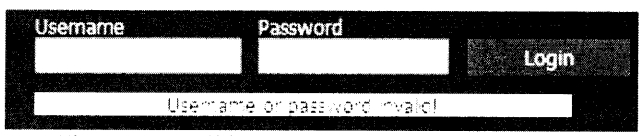


Figure 3. Invalid username or password message.

2. USER- The User functionality can be accessed from the Settings link on the top of the page. The idea is to give the capability for the user to change the email address, the password, and set up the privacy settings. Figure 4 show the Settings page with the current settings.

Figure 4. The Settings page

The user can update this information by clicking on the Change link. Each one needs to get updated individually. In other word, the user can not update the email address and the password at the same time. Once the Change link is clicked, the editable field will display with the Save button. Notice that if the update on the password is triggered, the user must type in the current password, the new password, and the confirmation password. This is to ensure that the right user is changing his/her password. Figure 5 shows the update on the password.

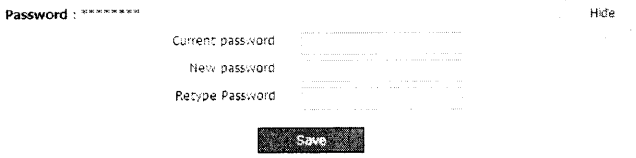


Figure 5. Update on the password.

3. PROFILE- To tell your friend about yourself, the Profile information should be completed. By clicking the Profile menu on the sidebar, the user will redirect to the Profile page. In case of the first time access, all information is blank and the default photo is displayed. To update the profile, click the Change link on the top right, then all fields will become editable with the Save button as shown in Figure 6. Figure 6 also show information filled out and read to save.

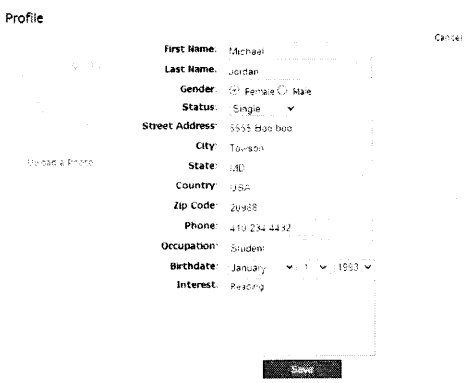


Figure 6. The editable profile page

To update your photo, simply click the Update a Photo link under the default photo. Then the photo upload will display. The MeSan only accepts two types of the image format: jpeg and gif. If the user uploads other format, the MeSan will alert with the error message. On your Site page where your friends can access and view, the profile information and the photo will be displayed to the left of the page.

3. FRIENDS- The Friends functionality gives you the capability to make friend by sending a request to who you want to be a friend. There are three sections on the Friends page. The first section shows your current friends. The section shows all requests from people who request you as a friend. The last section shows friend suggestions from the system. Figure 7 shows the Friends page.

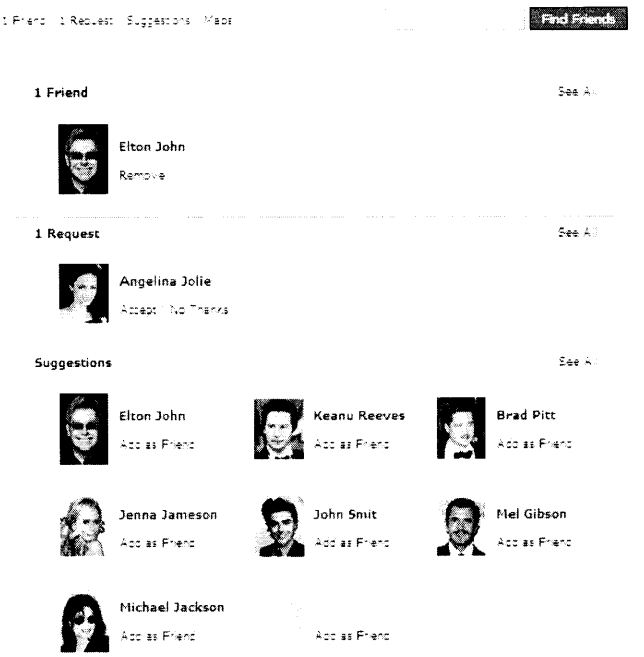


Figure 7. The Friends page

To make a friend, simply click the Add as Friend link then the confirmation message will be popped up. If it is a person who you want to send a request, click OK. Otherwise, select the Cancel button to go back to the Friends page. To accept a request, click the Accept link. Then that person will become your friend. To remove or deny the request, click on the No Thank link. That person will be moved to the friend suggestions section. Another functionality provided in the Friends page is the capability to search friends. You can enter a keyword to search friends. Moreover, the map MashUps is the other functionality. This feature makes use of the Google Map APIs to position your friends on a map as shown in Figure 8. The user can also click a marker to pop up a balloon window having the friend's photo

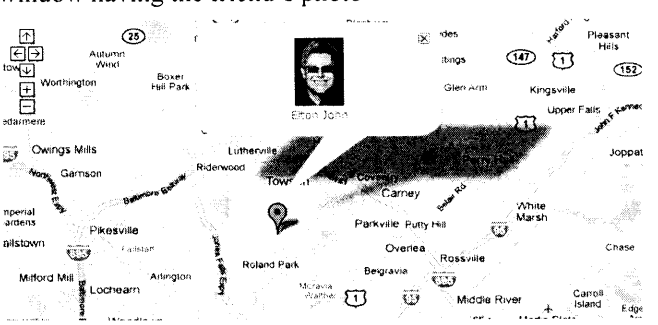
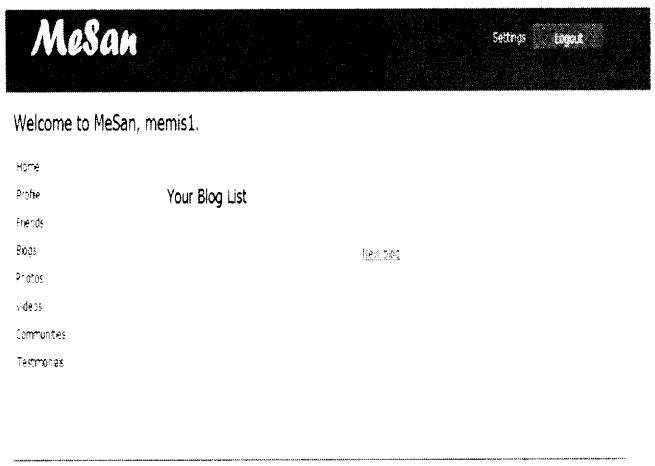
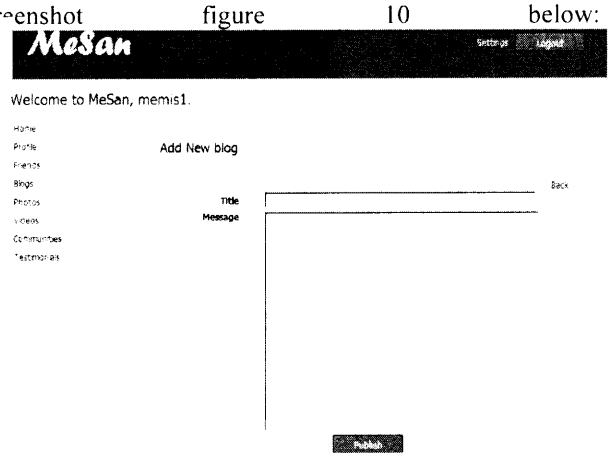


Figure 8. Google Map MashUps.

5. BLOG- The blog is basically a place where you can write a text message. This activity of updating a blog is called 'blogging' and someone who writes blog is a 'blogger'. Blogs are typically updated on daily basis by people with little or no technical expertise to update and create a blog. Postings on a blog are always in a chronological order with the most recent additions featured most prominently on the top of the page with older posts following it. If we are registered user of MeSan we can log-in using correct username and password which redirects us to user page. If a user wants to see his blogs he can do it by clicking on Blog label from the sidebar menu. This will populate *'Your Blog List'* if user already created a blog before. If it does not shows any list it means that you previously created no blog list. This can be shown in screen shot figure 9 below:



So, user can create his own blog by clicking on the *Add New* button. When Add New button is clicked a create method is being called in the blog model, the blog view page will be displayed which shows the Add New Blog form with Title and Message fields. This is shown in the screenshot figure 10 below:



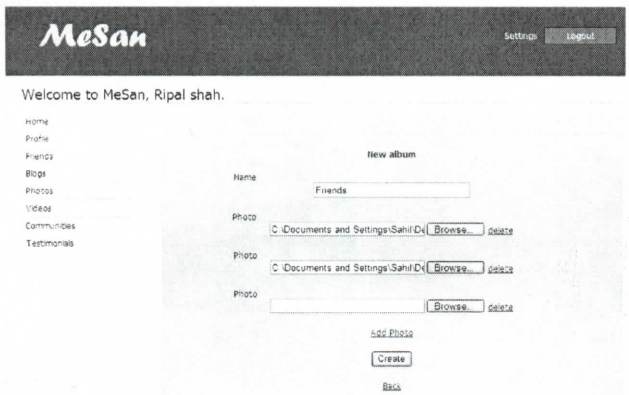
The title (text)field must accept 4- 80 characters as input. If text field is less than 3 and more than 80 characters, the update button will generate an error message that invalid length of title. Also, the message field could be as large as 255 characters long but cannot be left empty. That is both title and message fields must not be empty otherwise again error message will be generating in submitting the publish button and no blog will be created. So, once we provide correct input to both the Title and Message fields in add new blog form and clicking the button 'Publish' it updates the current blog of the user. On the user page, the blogs success message will be generated as "Your blog has been created" And you can see your newly created blog in the 'Your Blog List'. You can see the date and time the blog was created at and updated at. Now, already existing blogs can be edited, deleted and their list can be shown in the Show list. When we click on the edit button next to the blog you want to modify, it will redirect to blog index page with a form with content in its fields which can be altered. Again submitting the Publish button will update the blog for the user. Similarly, when we click on the show lists it displays all the blog lists and now if we wish to delete any blog, we can do it by clicking delete button next to the selected blog and lists shows the remaining blog lists. For rendering this we have used Ajax. Also, user can view other user blogs on their user page and can add a comment to their blog posts. A user can also comment on their own blogs and can update it. All recent activities on blog updates are visible in FEEDS of the user. The screenshot figure 11below represents this:

4. ALBUMS & PHOTOS – The MeSan gives user a functionality to create new album and upload photos into album so it can be shared with friends. User can also view friend's uploaded albums.

ADDING NEW ALBUM:

Album name is a required field and therefore missing album name will fires an error message and prevents user to save the album. MeSan system checks all validation such as album name existence and photos path should be *.jpg, *.jpeg. Now user successfully creates new album

Figure 12



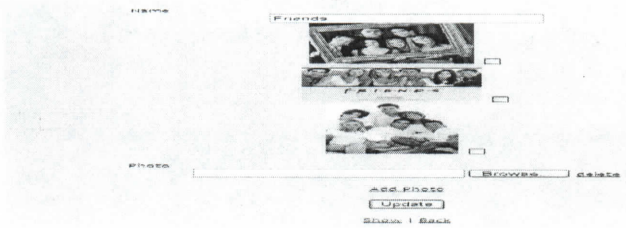
VIEWING AND EDITING ALBUM/PHOTOS

Once album gets created, User can view own album through clicking photos link from left side panel of home page. Considering user's album gets created and contains few photos. User can able to view particular album and all photos of that album. User may want to edit or delete few photos or entire album, MeSan gives this functionality as well. Below is a screen shots shows user seeing own album and photos inside it.

Figure 13

Now, user wants to delete or edit the same album. By clicking on the edit link of the particular album, below screen shot will prompt which will give you functionality to delete particular photos and add new photos.

Figure 13:



CHECKING FRIEND'S ALBUMS/PHOTOS

User can also able to see friend's album and photos uploaded into it by clicking on friend's name. This action shows friend's recent activities. To see friend's album click on photos link on top of menu and you will be redirected to uploaded albums page of friends'. By clicking on any particular album gives you list of all photos.

Figure14:

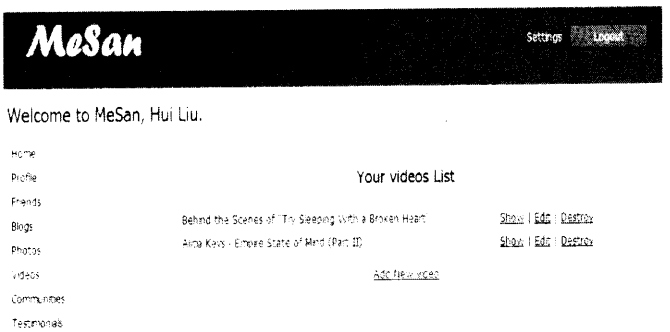
Most importantly, user can not update or destroy friends' album or photos.

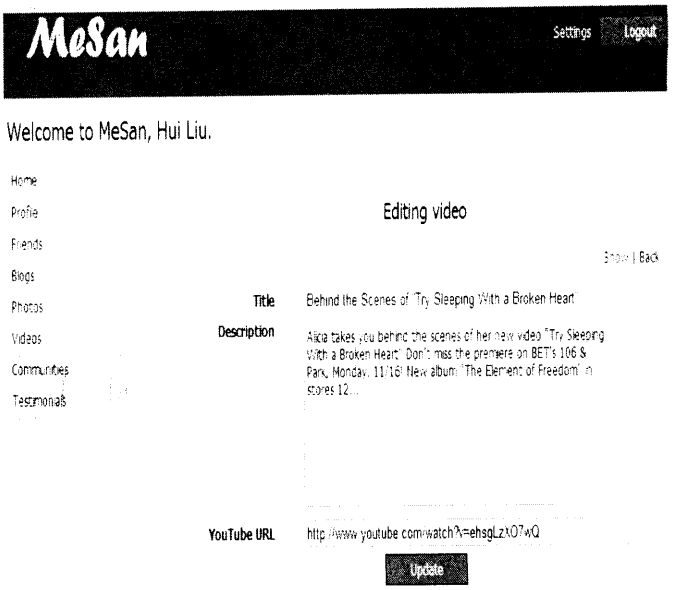
5. VIDEOS- The MeSan has the functionality to allow its users to view their favorite Youtube video in MeSan environment. We discuss in detail all the activities of section Videos.

Viewing video

When we click videos from the sidebar menu, we will see a list of all videos created by myself in descending order of the created time. Each of these video *will contain a title.* Each video will have a way to show it, edit it and delete it. We will also be able to add new videos. When we click show, we will be taken to the show page. The show page will show us the title, description of the video, and the video which we can watch directly in the page. We can go

to edit it or back to the video list from show page. This can be shown in the screen shot figure 15 below:

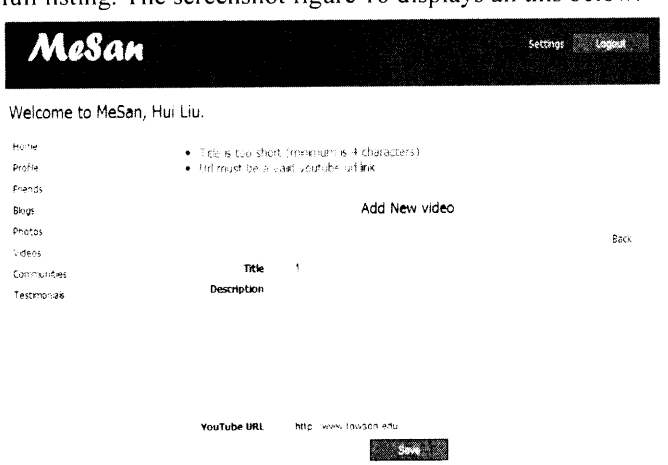




Editing, adding, and deleting from video

When we click edit, we will be taken to the edit page. The edit page will show me all the details of the video and we should be able to change any one of them (or all). After we change, we will like to go to the show page to check the updated result. We should also be able to add a new video and then see the new video in the complete listing. Make sure that the YouTube Url link we input is in correct format.

We should also be able to delete the video just created from the list and then that video will be not visible in the full listing. The screenshot figure 16 displays all this below:



Check friends' videos

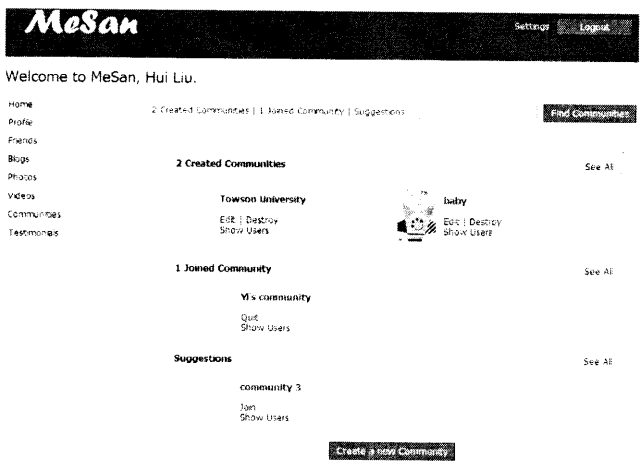
When we enter a friend's page, we should be able to see his video list. Each video will have a way to show it, but no way to edit or delete it.

This can be shown as figure 17below:

6. COMMUNITY- The MeSan allows its users to join group of people and sharing common interest among themselves by forming community.

Community indexing

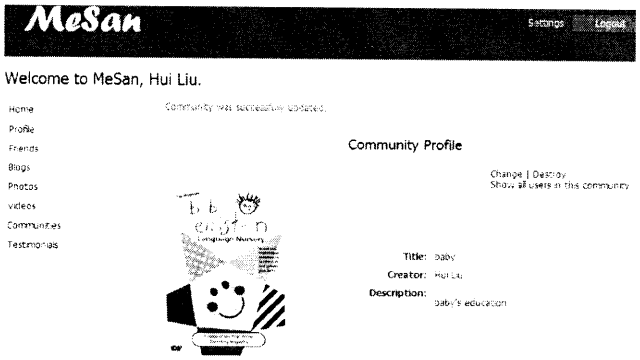
When we click communities, we will see a summary of all communities in 3 categories: created communities, joined communities, and suggestions. There is a total number for the first two categories. Under each category, there is a list of first 3 communities and a "see all" link to show all communities in that category. Each created community will have a way to show it, edit it, destroy it and show users who joined it. Each joined community will have a way to show it, quit it and show users who joined it. Each suggestion will have a way to show it, join it and show users who joined it. We will be able to do full-text search on title and description to find communities. We will also be able to create a new community. Figure 18:

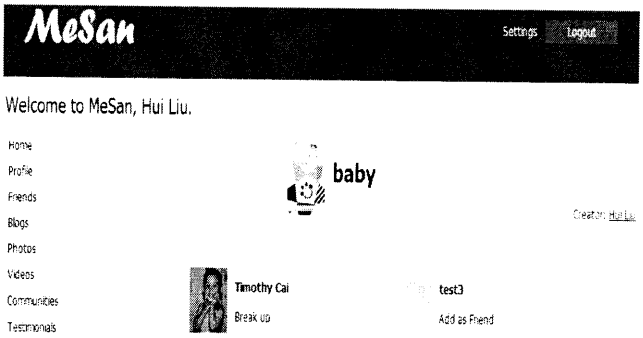


Viewing Community

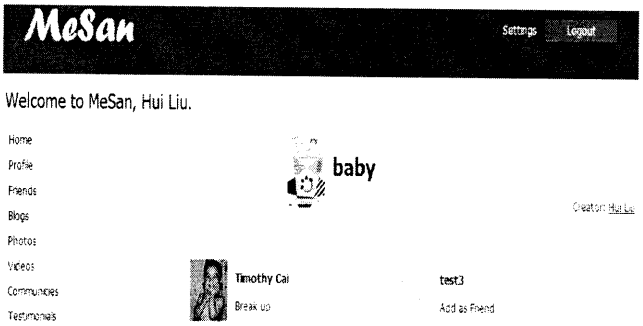
When we click the picture or title of a community, we will see the title, creator, description, picture of the community. I will also have a way to show all users in this community. If I am the creator of this community, I will have a way to change or destroy it. If I haven't joined this community, I will have a way to join it. If I have joined this community, I will have a way to quit it.

The screenshot figure 19 shows this below:

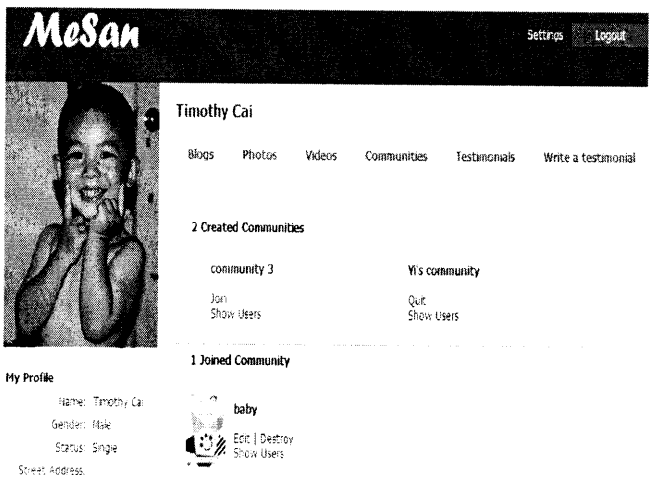




When I click show users or show all users in this community, I will see a list of users who joined in this community. Each user will contain a picture and a name. If he is my friend, I will have a way to see his page and a way to break up with him. If he is not my friend, I will have a way to add him as friend. Figure 20



When I enter a friend's page, I should be able to see his community indexing. And I can join or quit community in his list. Figure 21:

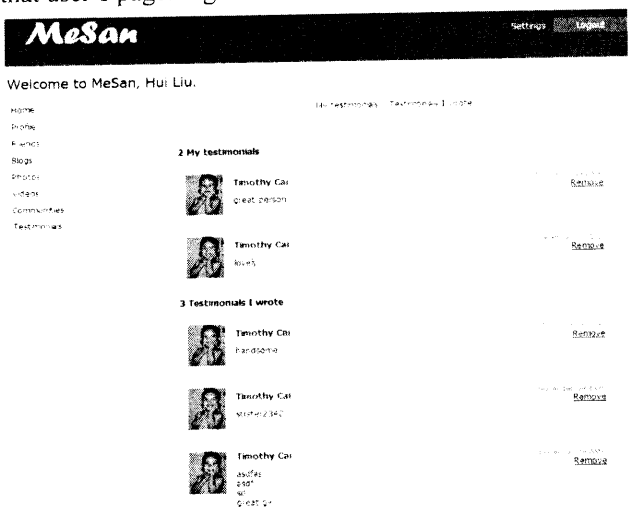


7. TESTIMONIALS

The MeSan allows user to write testimonial to his friend and the user can also receive the testimonial from his friends.

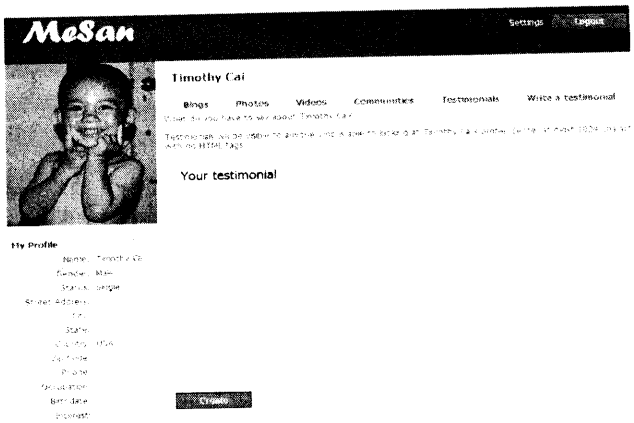
Viewing testimonial

When we click testimonials, we will see two lists: my testimonials and testimonials I wrote. The index page will show only first 3 of each and there is a way to show full testimonial list. Each my testimonial will contain writer's name and photo, full testimonial and create time. Each testimonial we wrote will contain receiver's name and photo, full testimonial and create time. Each testimonial has a way to remove it. Each user name and photo will link to that user's page. Figure 22:



Writing testimonials

When we enter a friend's page, we should be able to see his testimonials. When we click "write a testimonial", we will be able to write a testimonial to that friend and then see the new testimonial in his testimonial list. If we go back to my homepage, the new testimonial will show under feed and in testimonials under "testimonials I wrote". Figure 23:



8. RECENT ACTIVITIES

User should be able to see all his recent activities and feed in user home page.

Recent activities including:

- Who I just made friend with
- Who I just invited to be friend
- Which community I just joined
- Which community I just created

Feed include:

- My most recent blog published
- My most recent video added
- My most recent photo uploaded
- My most recent testimonial I received
- My most recent testimonial I wrote

TECHNOLOGY AND TOOLS

AJAX- The Ajax is a web application that sends the input field to the server in the background and updates the affected portion of the current web page in place. This dramatically increases the responsiveness of the user interface and makes it feel much more like a desktop application. Once the browser has rendered and displayed the initial web page, different user actions cause it to display a new web page (like any traditional web app) or trigger an Ajax operation. Trigger action could be the user clicking on a button or link, the user making changes to the data on a form or in a field, or just a periodic trigger. Data associated with the trigger is sent asynchronously to an action handler on the server.

The server-side action handler takes some action based on the data, and returns an HTML fragment as its response. The client-side JavaScript (created automatically by Rails) receives the HTML fragment and uses it to update a specified part of the current page's HTML, often the content of a tag. We have created blog function using Ajax which shows the result without changing whole page. We have use Ajax in blog in populating the blog list. It gives faster result compare to general application.

MashUps- The MashUps is a webpage or applications that combines data or function from two or more external sources to create a new service, it actuates fast integration, using open API's and data sources to produce rest that were not original reason for producing the raw source data.

Plug-ins-

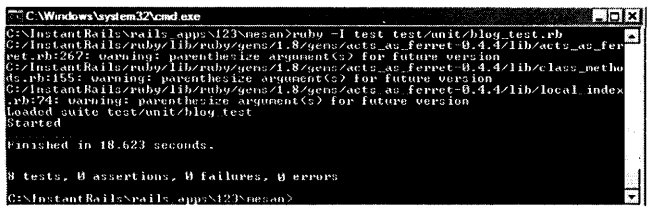
- Geokit- The Geokit is a ruby gem and a rails plug-in which provides geocoding, distance based finders, distance calculation and IP geocoding
- RMagik and ImageMagik-
- Paperclip- The paperclip is a plug-in for ruby which stores metadata such as filename, file type and file size in the database, but the actual file is stored in the local drive, by default it gets stores under public/system/data.
- Acts_as_ferret- The acts_as_ferret is a plug-in for ruby which makes it simple to implement full text search for rails. It builds ferret *ferret which is a ruby port* of *Apache Lucene.* It is a technically suitable for any application that requires full text search.

TESTCASES

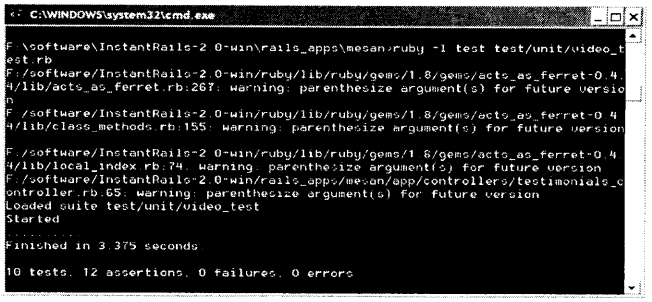
We only have time to do unit test and functional test on Video and Blog part.

For unit test, we did test on empty attributes, valid length and valid format.

Unit testing for Blog:

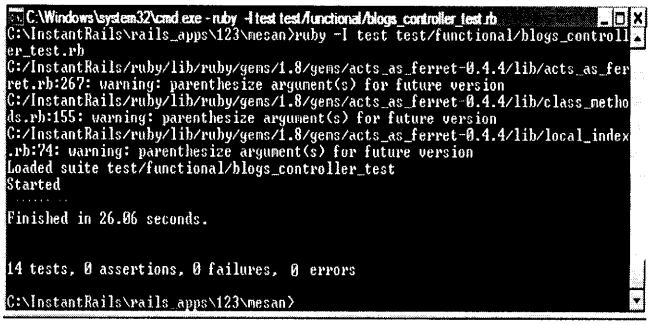


Unit testing for Video:

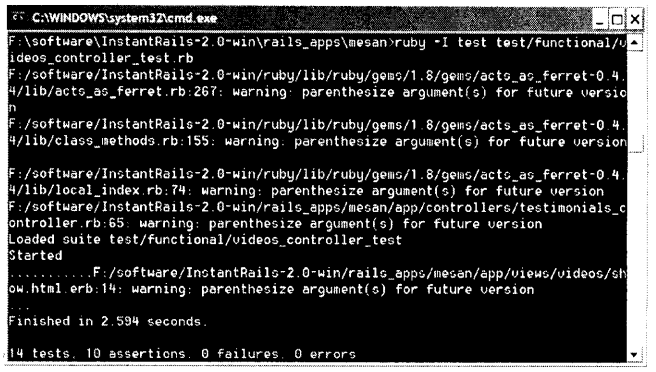


For functional test, we did test on index, new, create, show, edit, update and destroy.

Functional testing of Blogs:



Functional testing of Videos:



MeSan DEMERITS

MeSan is a social networking site and there are dangers associated with social networking including data theft and viruses, which are ever rising. Most often the online predators or individuals who claim to be someone but actually they are imposters. Although dangers are associated with online social networking so are associated in real like too. We often are advised to be cautious while confronting strangers at public places like parks, clubs, school, etc, so we must be careful online also and must use your common sense and wisdom to not to be victim

of unhealthy messages or uncomfortable comments from someone who you are chatting with.

CONCLUSION AND LESSON LEARNED

We need to do the following :

- **Fixtures files must be correct before we do testing**
  When we do unit test on Video model, we got lots of error from other model. We checked into it and found that it's because our fixture files are not correct. These fixture files are generated automatically when we generate the models. But after that, we did lots of changes on the models, and the fixture files did not make those changes. So we have to regenerate fixture files.
- **We can create our own rake command**
  We create an "extract_fixtures.rake" file to regenerate fixture files automatically. By put it into the directory /lib/tasks, we can run the command by:

  rake db:extract_fixtures

- **Plug-ins are buggy**
  We use acts-as-ferret to implement full text search. We followed the lecture notes exactly. However, at first, we always get an empty result. Then we figured out it's because it did not generate the index automatically (which it should be). So we use rebuild index() to recreate the index manually. Then it works fine. However, when we do the testing, we always get the wrong message that it cannot load the acts-as-ferret file in environment.rb. So we have to remove "require acts-as-ferret" from environment.rb to the controller.
  We use paperclip plug-in to implement photo upload. However, we went into trouble when we do the testing, so we have to give up testing on album and photo models as paperclip plug-in needs another gems and software to install for it.

- **Pair Programming**
  We were successful in this project as one agile approach like pair programming actually helped us in defact fixing and it actually actuated our work. Also, it sped out project growth and finally building the social networking site as desired.

ACKNOWLEDGMENTS

We thank Dr. Siddhart Kaza for his invaluable guidance and support throughout the project which helped us make this project article a success.

REFERENCES

1. www. google.com
2. www.wikipedia.com
3. www.youtube.com
4. www.sorceforge.com
5. Agile Ruby and Rails, third edition, by Dave Thomas