

**A
Mini Project Report
On
Friendbook : A Semantic-based Friend
Recommendation System for Social Networks**

Submitted to
JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY

in partial fulfillment of the requirement
for the award of the degree of
**BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING**



SUBMITTED BY:

Utkarsh Agarwal	(HTNO: 17TR1A0585)
B.Jyothi	(HTNO: 18TR5A0502)
M.Vaishnavi	(HTNO: 17TR1A0549)
K. Akhil	(HTNO: 16TR1A0548)

Under the Guidance of
Mr. Peddi Kishor
Associate Professor
Head of the Department

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SREE CHAITANYA INSTITUTE OF TECHNOLOGICAL SCIENCES
LMD COLONY, KARIMNAGAR-505527
(Approved by AICTE, Affiliated to JNTUH, Hyderabad)

SREE CHAITANYA INSTITUTE OF TECHNOLOGICAL SCIENCES
LMD COLONY, KARIMNAGAR-505527
(Approved by AICTE, Affiliated to JNTUH)

CERTIFICATE



Certified that this Mini Project Report entitled, “**Friendbook: A Semantic – based Friend Recommendation Systems for Social Networks**” is the bonafide work of **Utkarsh Agarwal (H.T.No. 17TR1A0585), B.Jyothi (H.T.No. 18TR5A0502), M.Vaishnavi (H.T.No. 17TR1A0549) and K. Akhil (H.T.No. 16TR1A0548) of IV Year, CSE** in the year 2021 in partial fulfillment of the requirements to award the Degree of Bachelor of Technology in **Computer Science & Engineering Branch** of Sree Chaitanya Institute of Technological Sciences, Karimnagar.

Mr. Peddi Kishor
Associate Professor
Project Guide

Mr. PEDDI KISHOR
Associate Professor
Head of the Department

External Examiner

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our Project Guide, ***Mr. Peddi Kishor, Associate Professor of CSE Department*** whose knowledge and guidance has motivated us to achieve goals we never thought possible. The time we have spent working under his supervision has truly been a pleasure.

We are thankful to **Mr. B. RAMESH, Assistant Professor, & Project Coordinator** of CSE Department for his effort, guidance and all faculty members of CSE Department for their help during my course. Thanks to programmers and non-teaching staff of CSE Department, Sree Chaitanya Institute of Technological Sciences.

We also thank **Mr. PEDDI KISHOR, HOD & Associate Professor of CSE Department** for providing seamless support and knowledge for the entire project work and also for providing right suggestions at every phase of the development of the project. He has consistently been a source of motivation, encouragement, and inspiration.

It is a great pleasure to convey our thanks to our principal **Dr. A. PRASAD RAJU, Principal**, Sree Chaitanya Institute of Technological Sciences and the College Management for permitting us to undertake this project and providing excellent facilities to carry out our project work.

Finally Special thanks to our parents, sisters and brothers for their support and encouragement throughout my life and this course. Thanks to all our friends and well wishers for their constant support.

DECLARATION

We hereby declare that the work which is being presented in this report entitled, **“Friendbook: A Semantic – based Friend Recommendation Systems for Social Networks”** submitted towards the partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science & Engineering**, Sree Chaitanya Institute of Technological Sciences, Karimnagar is an authentic record of our own work carried out under the supervision of **Mr. Peddi Kishor, Associate Professor, Department of CSE**, Sree Chaitanya Institute of Technological Sciences, Karimnagar.

To the best of our knowledge and belief, this project bears no resemblance with any report submitted to Sree Chaitanya Institute of Technological Sciences or any other University for the award of any degree or diploma.

Utkarsh Agarwal

H.T.No: 17TR1A0585

B.Jyothi

H.T.No: 18TR5A0502

M.Vaishnavi

H.T.No: 17TR1A0549

K. Akhil

H.T.No: 16TR1A0548

ABSTRACT

Existing social networking services recommend friends to users based on their social graphs, which may not be the most appropriate to reflect a user's preferences on friend selection in real life. In this project, we present Friendbook, a novel semantic-based friend recommendation system for social networks, which recommends friends to users based on their life styles instead of social graphs. By taking advantage of sensor-rich smartphones, Friendbook discovers life styles of users from user-centric sensor data, measures the similarity of life styles between users, and recommends friends to users if their life styles have high similarity. Inspired by text mining, we model a user's daily life as life documents, from which his/her life styles are extracted by using the Latent Dirichlet Allocation algorithm. We further propose a similarity metric to measure the similarity of life styles between users, and calculate users' impact in terms of life styles with a friend-matching graph. Upon receiving a request, Friendbook returns a list of people with highest recommendation scores to the query user. Finally, Friendbook integrates a feedback mechanism to further improve the recommendation accuracy. We have implemented Friendbook on the Android-based smartphones, and evaluated its performance on both small-scale experiments and large-scale simulations. The results show that the recommendations accurately reflect the preferences of users in choosing friends.

LIST OF FIGURES

S.no	Figure Description	Page No
1	How Java Compiler works	13
2	Java Platform Independence	13
3	Java Technology Architecture	14
4	Java2 SDK	17
5	Flow of execution of Java Program	23
6	Client Server Computing	24
7	Total Address at TCP/IP	26
8	Web Application Directory Structure	34
9	System Architecture	39
10	Data Flow Diagram	41
11	Use Case Diagram	44
12	Class Diagram	45
13	Sequence Diagram	46
14	Activity Diagram	47
15	Index Page	51
16	Abstract Content	51
17	Admin Login	52
18	Admin Home Page	52
19	Admin Home Page	53
20	User List	53
21	New User Registration	54
22	User Login	55
23	User Homepage	55
24	User Home page	56
25	Adding Friends	56
26	My Friend List	57
27	Recommend Sites From Friend	57

TABLE OF CONTENTS

Abstract	V
List of Figures	VI

	Page Nos Start – End
Chapter 1: INTRODUCTION	1-3
Chapter 2: LITERATURE SURVEY	4-7
Chapter 3: SYSTEM ANALYSIS	8-9
Chapter 4: SYSTEM STUDY	10-11
Chapter 5: SOFTWARE ENVIRONMENT	12-37
Chapter 6: SYSTEM REQUIREMENTS	38
Chapter 7: SYSTEM DESIGN.....	39-47
Chapter 8: IMPLEMENTATION.....	48-50
Chapter 9: SCREENSHOTS	51-57
Chapter 10: INPUT DESIGN AND OUTPUT DESIGN.....	58-60
Chapter 11: SYSTEM TESTING.....	61-65
Chapter 12: CONCLUSION.....	66
REFERENCES.....	67

CHAPTER 1

INTRODUCTION

What Is A Social Network?

Wikipedia defines a social network service as a service which “focuses on the building and verifying of online social networks for communities of people who share interests and activities, or who are interested in exploring the interests and activities of others, and which necessitates the use of software.”

A report published by OCLC provides the following definition of social networking sites: “Web sites primarily designed to facilitate interaction between users who share interests, attitudes and activities, such as Facebook, Mixi and MySpace.”

What Can Social Networks Be Used For?

Social networks can provide a range of benefits to members of an organisation:

Support for learning: Social networks can enhance informal learning and support social connections within groups of learners and with those involved in the support of learning.

Support for members of an organisation: Social networks can potentially be used by all members of an organisation, and not just those involved in working with students. Social networks can help the development of communities of practice.

Engaging with others: Passive use of social networks can provide valuable business intelligence and feedback on institutional services (although this may give rise to ethical concerns).

Ease of access to information and applications: The ease of use of many social networking services can provide benefits to users by simplifying access to other tools and applications. The Facebook Platform provides an example of how a social networking service can be used as an environment for other tools.

Common interface: A possible benefit of social networks may be the common interface which spans work / social boundaries. Since such services are often used in a personal capacity the interface and the way the service works may be familiar, thus minimising training and support needed to exploit the services in a professional context. This can, however, also be a barrier to those who wish to have strict boundaries between work and social activities.

Examples of Social Networking Services

Examples of popular social networking services include:

Facebook: Facebook is a social networking Web site that allows people to communicate with their friends and exchange information. In May 2007 Facebook launched the Facebook Platform which provides a framework for developers to create applications that interact with core Facebook features

MySpace: MySpace is a social networking Web site offering an interactive, user-submitted network of friends, personal profiles, blogs and groups, commonly used for sharing photos, music and videos..

Ning: An online platform for creating social Web sites and social networks aimed at users who want to create networks around specific interests or have limited technical skills.

Twitter: Twitter is an example of a micro-blogging service. Twitter can be used in a variety of ways including sharing brief information with users and providing support for one's peers.

Note that this brief list of popular social networking services omits popular social sharing services such as Flickr and YouTube.

Opportunities and Challenges

The popularity and ease of use of social networking services have excited institutions with their potential in a variety of areas. However effective use of social networking services poses a number of challenges for institutions including long-term sustainability of the services, user concerns over use of social tools in a work or study context, a variety of technical issues and legal issues such as copyright, privacy, accessibility, etc.

Institutions would be advised to consider carefully the implications before promoting significant use of such services.

CHAPTER 2

LITERATURE SURVEY

1) “Probabilistic mining of socio geographic routines from mobile phone data”

AUTHORS: K. Farrahi and D. Gatica-Perez

There is relatively little work on the investigation of large-scale human data in terms of multimodality for human activity discovery. In this project, we suggest that human interaction data, or human proximity, obtained by mobile phone Bluetooth sensor data, can be integrated with human location data, obtained by mobile cell tower connections, to mine meaningful details about human activities from large and noisy datasets. We propose a model, called bag of multimodal behavior that integrates the modeling of variations of location over multiple time-scales, and the modeling of interaction types from proximity. Our representation is simple yet robust to characterize real-life human behavior sensed from mobile phones, which are devices capable of capturing large-scale data known to be noisy and incomplete. We use an unsupervised approach, based on probabilistic topic models, to discover latent human activities in terms of the joint interaction and location behaviors of 97 individuals over the course of approximately a 10-month period using data from MIT's Reality Mining project. Some of the human activities discovered with our multimodal data representation include “going out from 7 pm-midnight alone” and “working from 11 am-5 pm with 3-5 other people,” further finding that this activity dominantly occurs on specific days of the week. Our methodology also finds dominant work patterns occurring on other days of the week. We further demonstrate the feasibility of the topic modeling framework for human routine discovery by predicting missing multimodal phone data at specific times of the day.

2. Collaborative and structural recommendation of friends using weblog-based social network analysis

AUTHORS: W. H. Hsu, A. King, M. Paradesi, T. Pydimarri, and T. Weninger

In this project, we address the problem of link recommendation in weblogs and similar social networks. First, we present an approach based on collaborative recommendation using the link structure of a social network and content-based recommendation using mutual declared interests. Next, we describe the application of this approach to a small representative subset of a large real-world social network: the user/community network of the blog service Live Journal. We then discuss the ground features available in Live Journal's public user information pages and describe some graph algorithms for analysis of the social network. These are used to identify candidates, provide ground truth for recommendations, and construct features for learning the concept of a recommended link. Finally, we compare the performance of this machine learning approach to that of the rudimentary recommender system provided by Live Journal.

3. Reality Mining: Sensing Complex Social Systems.

AUTHORS: N. Eagle and A. S. Pentland

We introduce a system for sensing complex social systems with data collected from 100 mobile phones over the course of 9 months. We demonstrate the ability to use standard Bluetooth-enabled mobile telephones to measure information access and use in different contexts, recognize social patterns in daily user activity, infer relationships, identify socially significant locations, and model organizational rhythms.

4. Understanding Transportation Modes Based on GPS Data for Web Applications.

AUTHORS: Y. Zheng, Y. Chen, Q. Li, X. Xie, and W.-Y. Ma.

User mobility has given rise to a variety of Web applications, in which the global positioning system (GPS) plays many important roles in bridging between these applications and end users. As a kind of human behavior, people's transportation modes, such as walking and driving, can provide pervasive computing systems with more contextual information and enrich a user's mobility with informative knowledge. In this article, we report on an approach based on supervised learning to automatically infer users' transportation modes, including driving, walking, taking a bus and riding a bike, from raw GPS logs. Our approach consists of three parts: a change point-based segmentation method, an inference model and a graph-based post-processing algorithm. First, we propose a change point-based segmentation method to partition each GPS trajectory into separate segments of different transportation modes. Second, from each segment, we identify a set of sophisticated features, which are not affected by differing traffic conditions (e.g., a person's direction when in a car is constrained more by the road than any change in traffic conditions). Later, these features are fed to a generative inference model to classify the segments of different modes. Third, we conduct graph-based post-processing to further improve the inference performance. This post-processing algorithm considers both the commonsense constraints of the real world and typical user behaviors based on locations in a probabilistic manner. The advantages of our method over the related works include three aspects. 1) Our approach can effectively segment trajectories containing multiple transportation modes. 2) Our work mined the location constraints from user-generated GPS logs, while being independent of additional sensor data and map information like road networks and bus stops. 3) The model learned from the dataset of some users can be applied to infer GPS data from others. Using the GPS logs collected by 65

people over a period of 10 months, we evaluated our approach via a set of experiments. As a result, based on the change-point-based segmentation method and Decision Tree-based inference model, we achieved prediction accuracy greater than 71 percent. Further, using the graph-based post-processing algorithm, the performance attained a 4-percent enhancement.

5. Online friend recommendation through personality matching and collaborative filtering

AUTHORS: L. Bian and H. Holtzman

Most social network websites rely on people's proximity on the social graph for friend recommendation. In this project, we present Matchmaker, a collaborative filtering friend recommendation system based on personality matching. The goal of Matchmaker is to leverage the social information and mutual understanding among people in existing social network connections, and produce friend recommendations based on rich contextual data from people's physical world interactions. Matchmaker allows users' network to match them with similar TV characters, and uses relationships in the TV programs as parallel comparison matrix to suggest to the users friends that have been voted to suit their personality the best. The system's ranking schema allows progressive improvement on the personality matching consensus and more diverse branching of users' social network connections. Lastly, our user study shows that the application can also induce more TV content consumption by driving users' curiosity in the ranking process.

CHAPTER 3

SYSTEM ANALYSIS

EXISTING SYSTEM:

Most of the friend suggestions mechanism relies on pre-existing user relationships to pick friend candidates. For example, Facebook relies on a social link analysis among those who already share common friends and recommends symmetrical users as potential friends. The rules to group people together include:

- 1) Habits or life style
- 2) Attitudes
- 3) Tastes
- 4) Moral standards
- 5) Economic level, and
- 6) People they already know.

Apparently, rule #3 and rule #6 are the mainstream factors considered by existing recommendation systems.

DISADVANTAGES OF EXISTING SYSTEM:

- Existing social networking services recommend friends to users based on their social graphs, which may not be the most appropriate to reflect a user's preferences on friend selection in real life

PROPOSED SYSTEM:

- A novel semantic-based friend recommendation system for social networks, which recommends friends to users based on their life styles instead of social graphs.
- By taking advantage of sensor-rich smartphones, Friendbook discovers life styles of users from user-centric sensor data, measures the similarity of life styles between users, and recommends friends to users if their life styles have high similarity.
- We model a user's daily life as life documents, from which his/her life styles are extracted by using the Latent Dirichlet Allocation algorithm.
- Similarity metric to measure the similarity of life styles between users, and calculate users'
- Impact in terms of life styles with a friend-matching graph.
- We integrate a linear feedback mechanism that exploits the user's feedback to improve recommendation accuracy.

ADVANTAGES OF PROPOSED SYSTEM:

- Recommend potential friends to users if they share similar life styles.
- The feedback mechanism allows us to measure the satisfaction of users, by providing a user interface that allows the user to rate the friend list

CHAPTER 4

SYSTEM STUDY

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ◆ **ECONOMICAL FEASIBILITY**
- ◆ **TECHNICAL FEASIBILITY**
- ◆ **SOCIAL FEASIBILITY**

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 5

SOFTWARE ENVIRONMENT

Java Technology

Java technology is both a programming language and a platform.

The Java Programming Language

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded
- Robust
- Dynamic
- Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes*—the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the

computer. Compilation happens just once, interpretation occurs each time the program is executed. The following figure illustrates how this works.

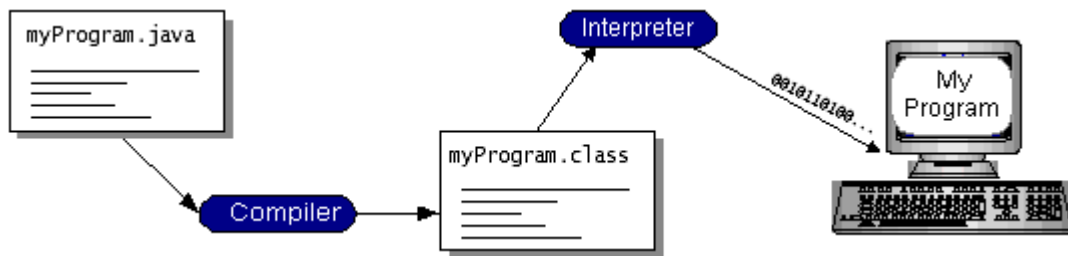


Fig.no. 1

You can think of Java byte codes as the machine code instructions for the *Java Virtual Machine* (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make “write once, run anywhere” possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.

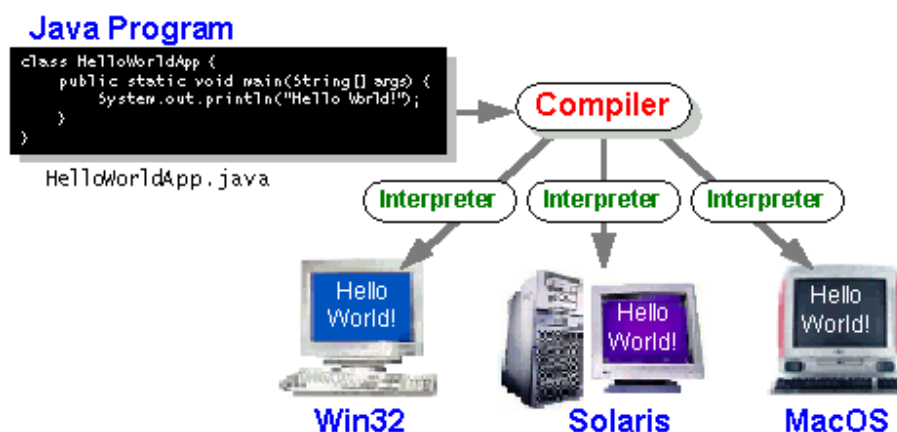


Fig.no. 2

The Java Platform

A *platform* is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

- The *Java Virtual Machine* (Java VM)
- The *Java Application Programming Interface* (Java API)

You've already been introduced to the Java VM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces, these libraries are known as *packages*. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.

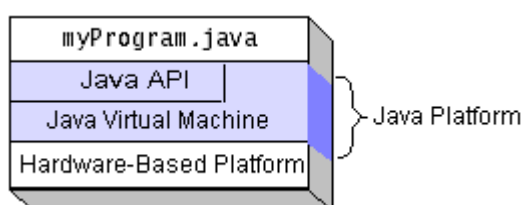


Fig.no. 3

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart

compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

What Can Java Technology Do?

The most common types of programs written in the Java programming language are *applets* and *applications*. If you've surfed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser. However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a *server* serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a *servlet*. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

- **The essentials:** Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
- **Applets:** The set of conventions used by applets.
- **Networking:** URLs, TCP (Transmission Control Protocol), UDP (User Datagram Protocol) sockets, and IP (Internet Protocol) addresses.
- **Internationalization:** Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
- **Security:** Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
- **Software components:** Known as JavaBeans™, can plug into existing component architectures.
- **Object serialization:** Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC™):** Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.

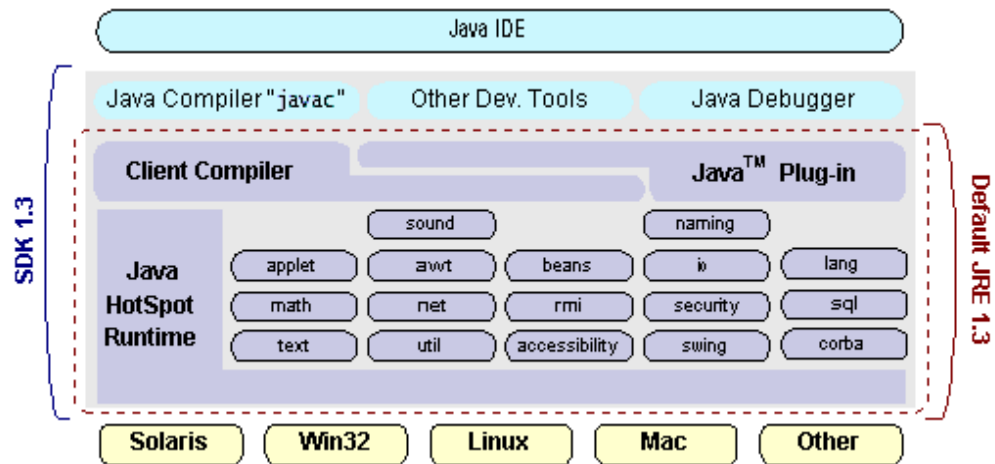


Fig.no. 4

How Will Java Technology Change My Life?

We can't promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

- **Get started quickly:** Although the Java programming language is a powerful object-oriented language, it's easy to learn, especially for programmers already familiar with C or C++.
- **Write less code:** Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
- **Write better code:** The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people's tested code and introduce fewer bugs.
- **Develop programs more quickly:** Your development time may be as much as twice as fast versus writing the same program in C++.

Why? You write fewer lines of code and it is a simpler programming language than C++.

- **Avoid platform dependencies with 100% Pure Java:** You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure Java™ Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.
- **Write once, run anywhere:** Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.
- **Distribute software more easily:** You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded “on the fly,” without recompiling the entire program.

ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a *de facto* standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data

source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources.

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn't change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn't as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers

has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

JDBC

In an effort to set an independent database standard API for Java, Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of “plug-in” database connectivity modules, or *drivers*. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC’s framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

JDBC Goals

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals,

in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

1. SQL Level API

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to “generate” JDBC code and to hide many of JDBC’s complexities from the end user.

2. SQL Conformance

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

3. JDBC must be implemental on top of common database interfaces

The JDBC SQL API must “sit” on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.

4. Provide a Java interface that is consistent with the rest of the Java system

Because of Java’s acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

5. **Keep it simple**

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

6. **Use strong, static typing wherever possible**

Strong typing allows for more error checking to be done at compile time, also, less error appear at runtime.

7. **Keep the common cases simple**

Because more often than not, the usual SQL calls used by the programmer are simple SELECT's, INSERT's, DELETE's and UPDATE's, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible.

Java ha two things: a programming language and a platform. Java is a high-level programming language that is all of the following

Simple	Architecture-neutral
Object-oriented	Portable
Distributed	High-performance
Interpreted	multithreaded
Robust	Dynamic
Secure	

Java is also unusual in that each Java program is both compiled and interpreted. With a compile you translate a Java program into an intermediate language called Java byte codes the platform-independent code instruction is passed and run on the computer.

Compilation happens just once, interpretation occurs each time the program is executed. The figure illustrates how this works.

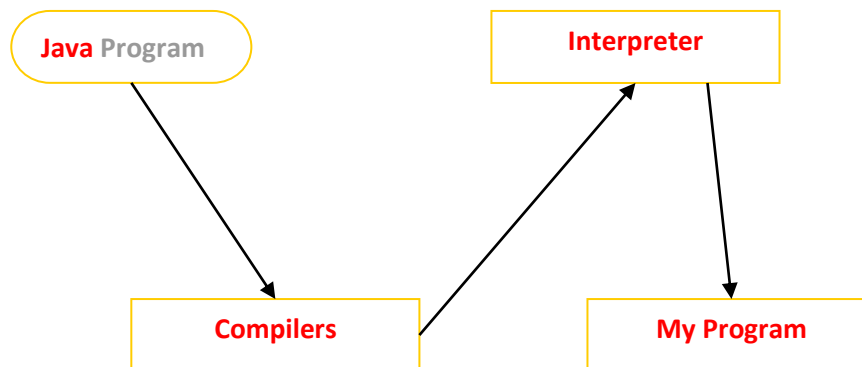


Fig.no. 5

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java byte codes help make “write once, run anywhere” possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintos

Networking

TCP/IP stack

The TCP/IP stack is shorter than the OSI one:

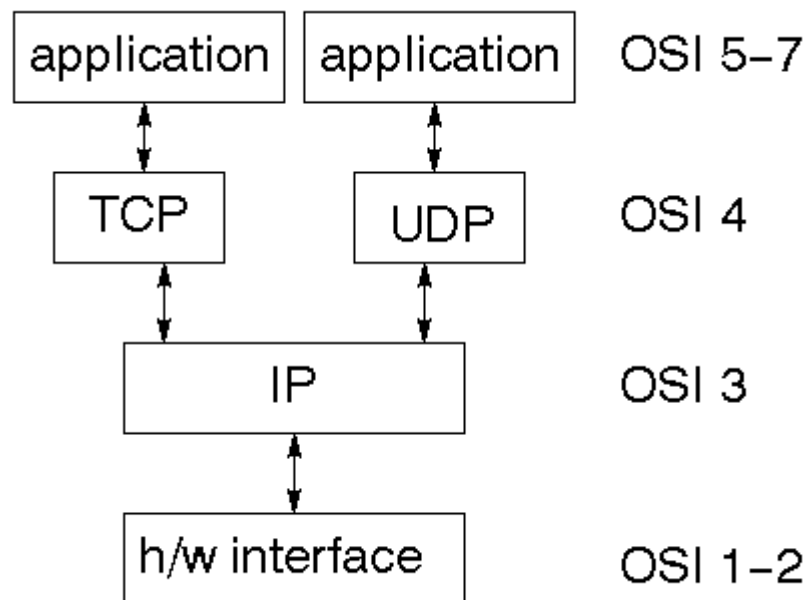


Fig.no. 6

TCP is a connection-oriented protocol, UDP (User Datagram Protocol) is a connectionless protocol.

IP datagram's

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The IP layer supplies a checksum that includes its own header. The header includes the source and destination addresses. The IP layer handles routing through an Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end.

UDP

UDP is also connectionless and unreliable. What it adds to IP is a checksum for the contents of the datagram and port numbers. These are used to give a client/server model - see later.

TCP

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

Internet addresses

In order to use a service, you must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32 bit integer which gives the IP address. This encodes a network ID and more addressing. The network ID falls into various classes according to the size of the network address.

Network address

Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16 bit network addressing. Class C uses 24 bit network addressing and class D uses all 32.

Subnet address

Internally, the UNIX network is divided into sub networks. Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.

Host address

8 bits are finally used for host addresses within our subnet. This places a limit of 256 machines that can be on the subnet.

Total address

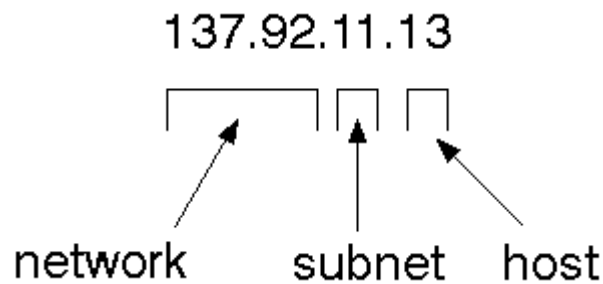


Fig.no. 7

The 32 bit address is usually written as 4 integers separated by dots.

Port addresses

A service exists on a host, and is identified by its port. This is a 16 bit number. To send a message to a server, you send it to the port for that service of the host that it is running on. This is not location transparency! Certain of these ports are "well known".

Sockets

A socket is a data structure maintained by the system to handle network connections. A socket is created using the call `socket`. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with `Read File` and `Write File` functions.

```
#include <sys/types.h>
```

```
#include <sys/socket.h>
```

```
int socket(int family, int type, int protocol),
```

Here "family" will be `AF_INET` for IP communications, `protocol` will be zero, and `type` will depend on whether TCP or UDP is used. Two processes wishing to communicate over a network create a socket each. These are similar to two ends of a pipe - but the actual pipe does not yet exist.

JFree Chart

JFreeChart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. JFreeChart's extensive feature set includes:

- A consistent and well-documented API, supporting a wide range of chart types,

- A flexible design that is easy to extend, and targets both server-side and client-side applications,

- Support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG),

JFreeChart is "open source" or, more specifically, free software. It is distributed under the terms of the GNU Lesser General Public Licence (LGPL), which permits use in proprietary applications.

1. Map Visualizations

Charts showing values that relate to geographical areas. Some examples include: (a) population density in each state of the United States, (b) income per capita for each country in Europe, (c) life expectancy in each country of the world. The tasks in this project include:

- Sourcing freely redistributable vector outlines for the countries of the world, states/provinces in particular countries (USA in particular, but also other areas),

- Creating an appropriate dataset interface (plus default implementation), a rendered, and integrating this with the existing XYPlot class in JFreeChart,

- Testing, documenting, testing some more, documenting some more.

2. Time Series Chart Interactivity

Implement a new (to JFreeChart) feature for interactive time series charts --- to display a separate control that shows a small version of ALL the time series data, with a sliding "view" rectangle that allows you to select the subset of the time series data to display in the main chart.

3. Dashboards

There is currently a lot of interest in dashboard displays. Create a flexible dashboard mechanism that supports a subset of JFreeChart chart types (dials, pies, thermometers, bars, and lines/time series) that can be delivered easily via both Java Web Start and an applet.

4. Property Editors

The property editor mechanism in JFreeChart only handles a small subset of the properties that can be set for charts. Extend (or reimplement) this mechanism to provide greater end-user control over the appearance of the charts.

What is a Java Web Application?

A Java web application generates interactive web pages containing various types of markup language (HTML, XML, and so on) and dynamic content. It is typically comprised of web components such as JavaServer Pages (JSP), servlets and JavaBeans to modify and temporarily store data, interact with databases and web services, and render content in response to client requests.

Because many of the tasks involved in web application development can be repetitive or require a surplus of boilerplate code, web frameworks can be applied to alleviate the overhead associated with common activities. For example, many

frameworks, such as JavaServer Faces, provide libraries for templating pages and session management, and often promote code reuse.

What is Java EE?

Java EE (Enterprise Edition) is a widely used platform containing a set of coordinated technologies that significantly reduce the cost and complexity of developing, deploying, and managing multi-tier, server-centric applications. Java EE builds upon the Java SE platform and provides a set of APIs (application programming interfaces) for developing and running portable, robust, scalable, reliable and secure server-side applications.

Some of the fundamental components of Java EE include:

- Enterprise JavaBeans (EJB): a managed, server-side component architecture used to encapsulate the business logic of an application. EJB technology enables rapid and simplified development of distributed, transactional, secure and portable applications based on Java technology.
- Java Persistence API (JPA): a framework that allows developers to manage data using object-relational mapping (ORM) in applications built on the Java Platform.

JavaScript and Ajax Development

JavaScript is an object-oriented scripting language primarily used in client-side interfaces for web applications. Ajax (Asynchronous JavaScript and XML) is a Web 2.0 technique that allows changes to occur in a web page without the need to perform a page refresh. JavaScript toolkits can be leveraged to implement Ajax-enabled components and functionality in web pages.

Web Server and Client

Web Server is a software that can process the client request and send the response back to the client. For example, Apache is one of the most widely used web server. Web Server runs on some physical machine and listens to client request on specific port.

A web client is a software that helps in communicating with the server. Some of the most widely used web clients are Firefox, Google Chrome, Safari etc. When we request something from server (through URL), web client takes care of creating a request and sending it to server and then parsing the server response and present it to the user.

HTML and HTTP

Web Server and Web Client are two separate softwares, so there should be some common language for communication. HTML is the common language between server and client and stands for **HyperText Markup Language**.

Web server and client needs a common communication protocol, HTTP (**HyperText Transfer Protocol**) is the communication protocol between server and client. HTTP runs on top of TCP/IP communication protocol.

Some of the important parts of HTTP Request are:

- **HTTP Method** – action to be performed, usually GET, POST, PUT etc.
- **URL** – Page to access
- **Form Parameters** – similar to arguments in a java method, for example user,password details from login page.

Sample HTTP Request:

1GET /FirstServletProject/jsps/hello.jsp HTTP/1.1

2Host: localhost:8080

3Cache-Control: no-cache

Some of the important parts of HTTP Response are:

- **Status Code** – an integer to indicate whether the request was success or not. Some of the well known status codes are 200 for success, 404 for Not Found and 403 for Access Forbidden.
- **Content Type** – text, html, image, pdf etc. Also known as MIME type
- **Content** – actual data that is rendered by client and shown to user.

MIME Type or Content Type: If you see above sample HTTP response header, it contains tag “Content-Type”. It’s also called MIME type and server sends it to client to let them know the kind of data it’s sending. It helps client in rendering the data for user. Some of the mostly used mime types are text/html, text/xml, application/xml etc.

Understanding URL

URL is acronym of Universal Resource Locator and it’s used to locate the server and resource. Every resource on the web has it’s own unique address. Let’s see parts of URL with an example.

`http://localhost:8080/FirstServletProject/jsps/hello.jsp`

http:// – This is the first part of URL and provides the communication protocol to be used in server-client communication.

localhost – The unique address of the server, most of the times it’s the hostname of the server that maps to unique IP address. Sometimes multiple hostnames point to same IP addresses and web server virtual host takes care of sending request to the particular server instance.

8080 – This is the port on which server is listening, it’s optional and if we don’t provide it in URL then request goes to the default port of the protocol. Port

numbers 0 to 1023 are reserved ports for well known services, for example 80 for HTTP, 443 for HTTPS, 21 for FTP etc.

FirstServletProject/jsps/hello.jsp – Resource requested from server. It can be static html, pdf, JSP, servlets, PHP etc.

Why we need Servlet and JSPs?

Web servers are good for static contents HTML pages but they don't know how to generate dynamic content or how to save data into databases, so we need another tool that we can use to generate dynamic content. There are several programming languages for dynamic content like PHP, Python, Ruby on Rails, Java Servlets and JSPs.

Java Servlet and JSPs are server side technologies to extend the capability of web servers by providing support for dynamic response and data persistence.

Web Container

Tomcat is a web container, when a request is made from Client to web server, it passes the request to web container and it's web container job to find the correct resource to handle the request (servlet or JSP) and then use the response from the resource to generate the response and provide it to web server. Then web server sends the response back to the client.

When web container gets the request and if it's for servlet then container creates two Objects `HttpServletRequest` and `HttpServletResponse`. Then it finds the correct servlet based on the URL and creates a thread for the request. Then it invokes the servlet `service()` method and based on the HTTP method `service()` method invokes `doGet()` or `doPost()` methods. Servlet methods generate the dynamic page and write it to response. Once servlet thread is complete, container converts the response to HTTP response and send it back to client.

Some of the important work done by web container are:

- **Communication Support** – Container provides easy way of communication between web server and the servlets and JSPs. Because of container, we don't need to build a server socket to listen for any request from web server, parse the request and generate response. All these important and complex tasks are done by container and all we need to focus is on our business logic for our applications.
- **Lifecycle and Resource Management** – Container takes care of managing the life cycle of servlet. Container takes care of loading the servlets into memory, initializing servlets, invoking servlet methods and destroying them. Container also provides utility like JNDI for resource pooling and management.
- **Multithreading Support** – Container creates new thread for every request to the servlet and when it's processed the thread dies. So servlets are not initialized for each request and saves time and memory.
- **JSP Support** – JSPs doesn't look like normal java classes and web container provides support for JSP. Every JSP in the application is compiled by container and converted to Servlet and then container manages them like other servlets.
- **Miscellaneous Task** – Web container manages the resource pool, does memory optimizations, run garbage collector, provides security configurations, support for multiple applications, hot deployment and several other tasks behind the scene that makes our life easier.

Web Application Directory Structure

Java Web Applications are packaged as Web Archive (WAR) and it has a defined structure. You can export above dynamic web project as WAR file and unzip it to check the hierarchy. It will be something like below image.

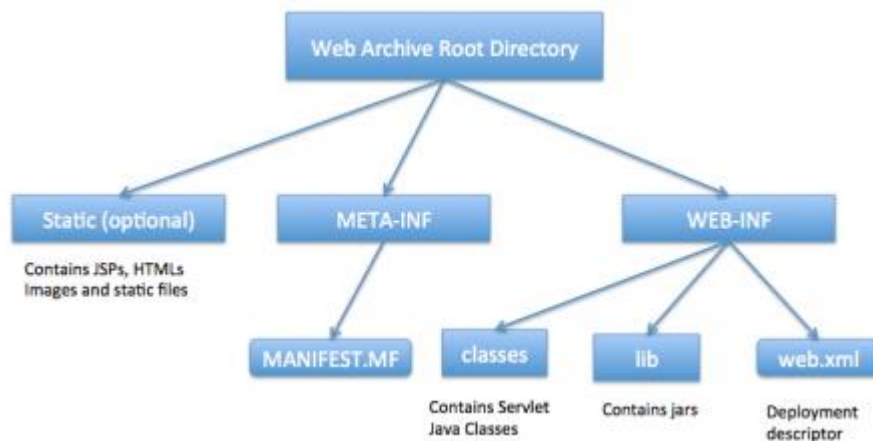


Fig.no. 8

Deployment Descriptor

web.xml file is the deployment descriptor of the web application and contains mapping for servlets (prior to 3.0), welcome pages, security configurations, session timeout settings etc.

Thats all for the java web application startup tutorial, we will explore Servlets and JSPs more in future posts.

MySQL:

MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation.

The MySQL Web site (<http://www.mysql.com/>) provides the latest information about MySQL software.

- **MySQL is a database management system.**

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large

amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

- **MySQL databases are relational.**

A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one-to-one, one-to-many, unique, required or optional, and “pointers” between different tables. The database enforces these rules, so that with a well-designed database, your application never sees inconsistent, duplicate, orphan, out-of-date, or missing data.

The SQL part of “MySQL” stands for “Structured Query Language”. SQL is the most common standardized language used to access databases. Depending on your programming environment, you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language, or use a language-specific API that hides the SQL syntax.

SQL is defined by the ANSI/ISO SQL Standard. The SQL standard has been evolving since 1986 and several versions exist. In this manual, “SQL-92” refers to the standard released in 1992, “SQL:1999” refers to the standard released in 1999, and “SQL:2003” refers to the current version of the standard. We use the phrase “the SQL standard” to mean the current version of the SQL Standard at any time.

- **MySQL software is Open Source.**

Open Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything. If you wish, you may study the source code and change it to suit your needs. The MySQL software uses the GPL (GNU General Public License), <http://www.fsf.org/licenses/>, to define what you may and may not do with the software in different situations. If you feel uncomfortable with the GPL or need to embed MySQL code into a commercial application, you can buy a commercially licensed version from us. See the MySQL Licensing Overview for more information (<http://www.mysql.com/company/legal/licensing/>).

- **The MySQL Database Server is very fast, reliable, scalable, and easy to use.**

If that is what you are looking for, you should give it a try. MySQL Server can run comfortably on a desktop or laptop, alongside your other applications, web servers, and so on, requiring little or no attention. If you dedicate an entire machine to MySQL, you can adjust the settings to take advantage of all the memory, CPU power, and I/O capacity available. MySQL can also scale up to clusters of machines, networked together. You can find a performance comparison of MySQL Server with other database managers on our benchmark page.

MySQL Server was originally developed to handle large databases much faster than existing solutions and has been successfully used in highly demanding production environments for several years. Although under constant development, MySQL Server today offers a rich and useful set of functions. Its connectivity, speed, and security make MySQL Server highly suited for accessing databases on the Internet.

- **MySQL Server works in client/server or embedded systems.**

The MySQL Database Software is a client/server system that consists of a multi-threaded SQL server that supports different backends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs).

We also provide MySQL Server as an embedded multi-threaded library that you can link into your application to get a smaller, faster, easier-to-manage standalone product.

- **A large amount of contributed MySQL software is available.**

MySQL Server has a practical set of features developed in close cooperation with our users. It is very likely that your favorite application or language supports the MySQL Database Server.

The official way to pronounce “MySQL” is “My Ess Que Ell” (not “my sequel”), but we do not mind if you pronounce it as “my sequel” or in some other localized way.

CHAPTER 6

SYSTEM REQUIREMENTS:

HARDWARE REQUIREMENTS:

- System : Intel® Core™ i3-7020U 2.30 GHz
- Hard Disk : 1 TB
- Monitor : 15 VGA Colour.
- Mouse : Logitech.
- RAM : 4 GB
- System Type : 64-bit OS

SOFTWARE REQUIREMENTS:

- Operating system : Windows 7/8/10
- Coding Language : JAVA/J2EE
- IDE : NetBeans 12.2
- Database : MYSQL 8.0.23

CHAPTER 7

SYSTEM DESIGN

SYSTEM ARCHITECTURE:

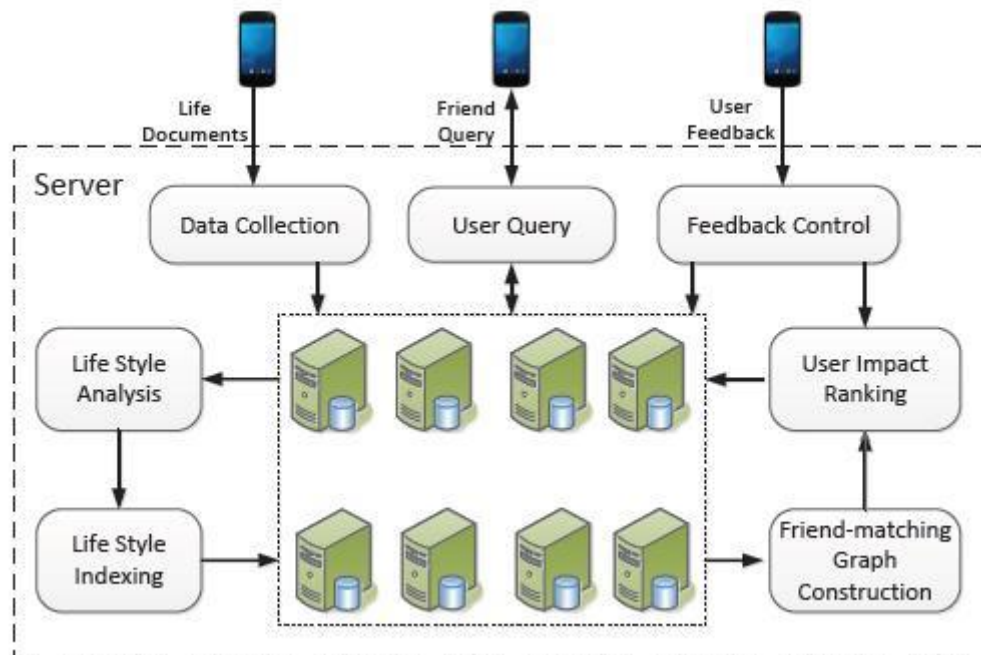


Fig.no. 9

DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

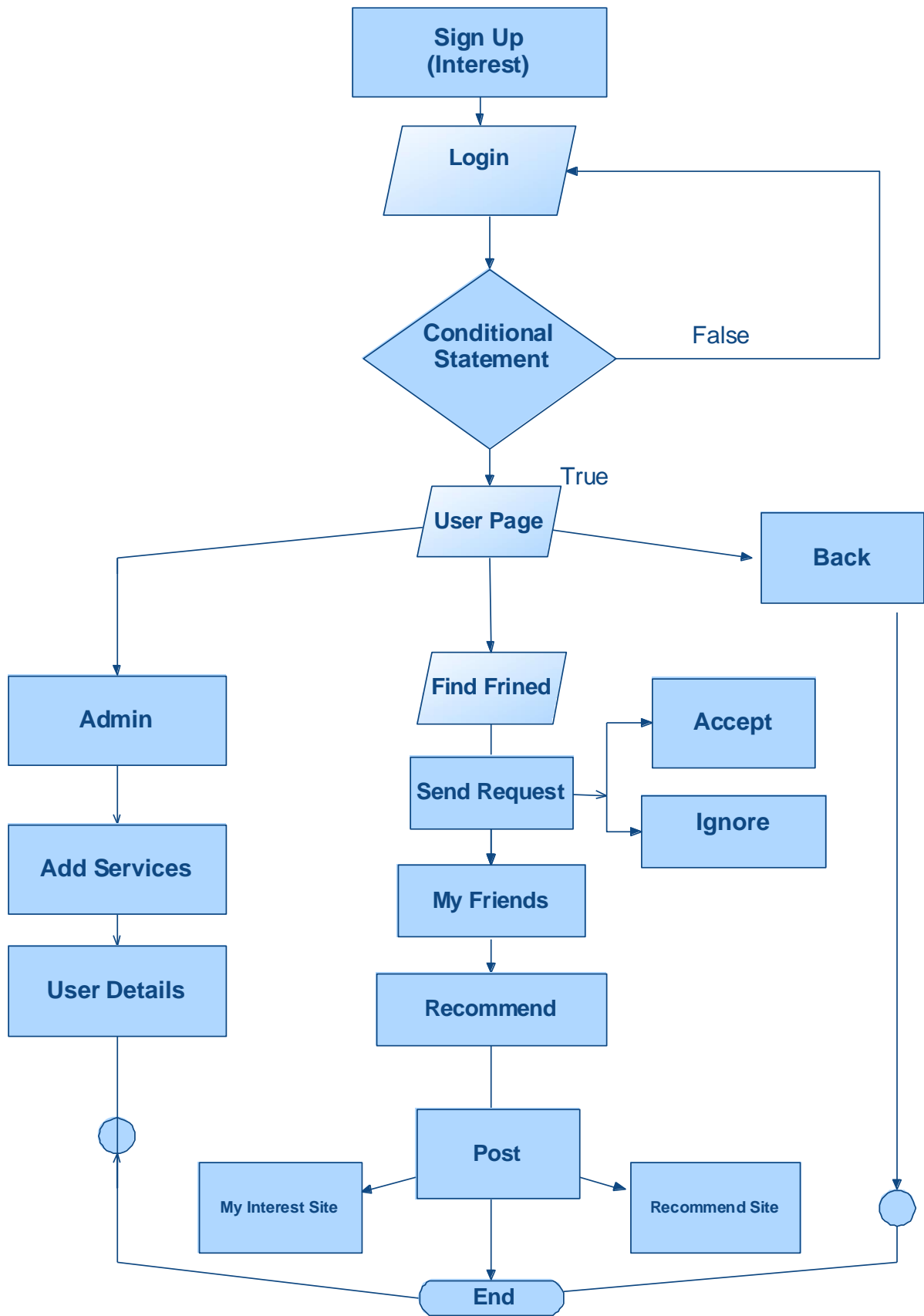


Fig.no. 10

UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to, or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.

6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

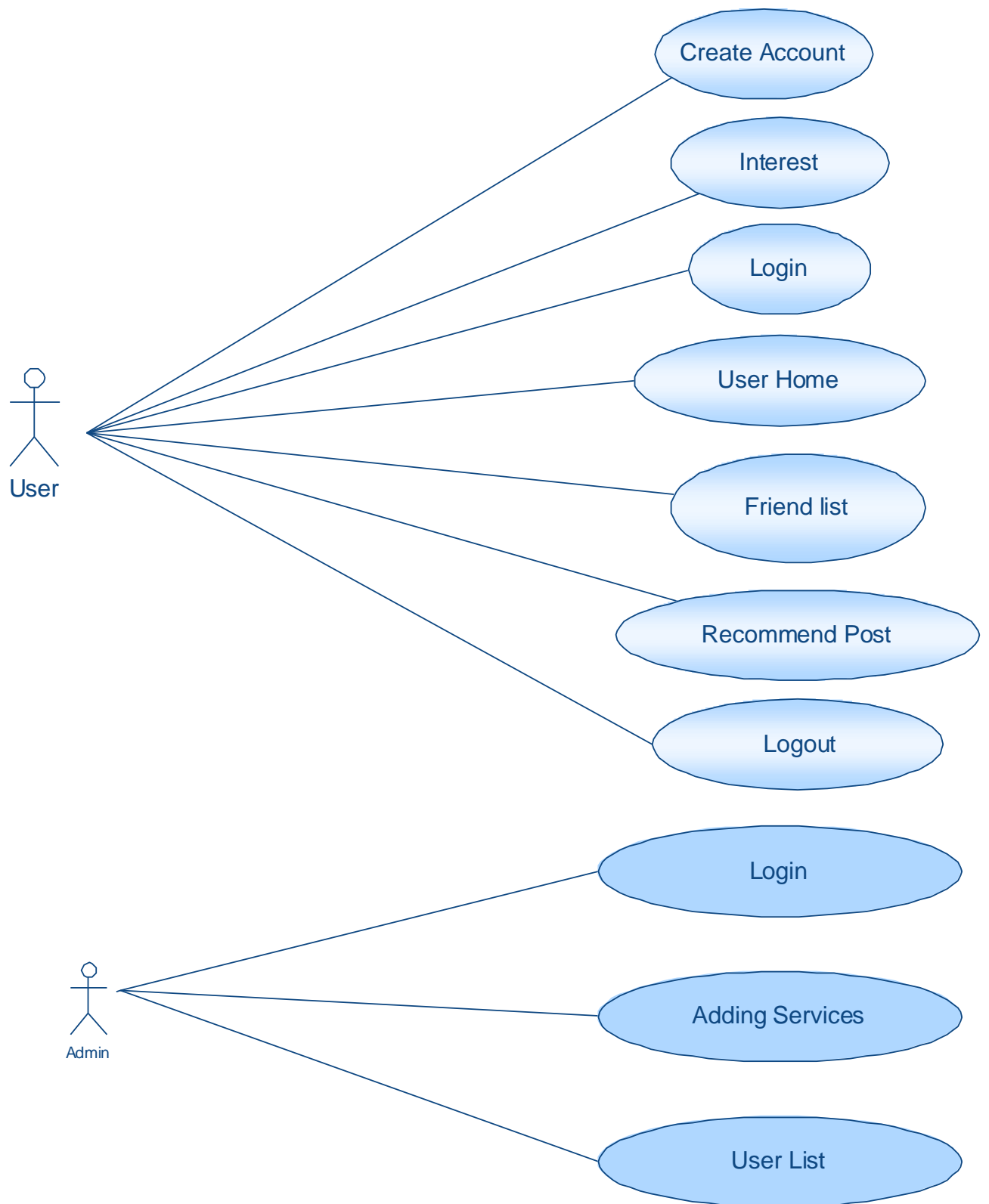


Fig.no. 11

CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

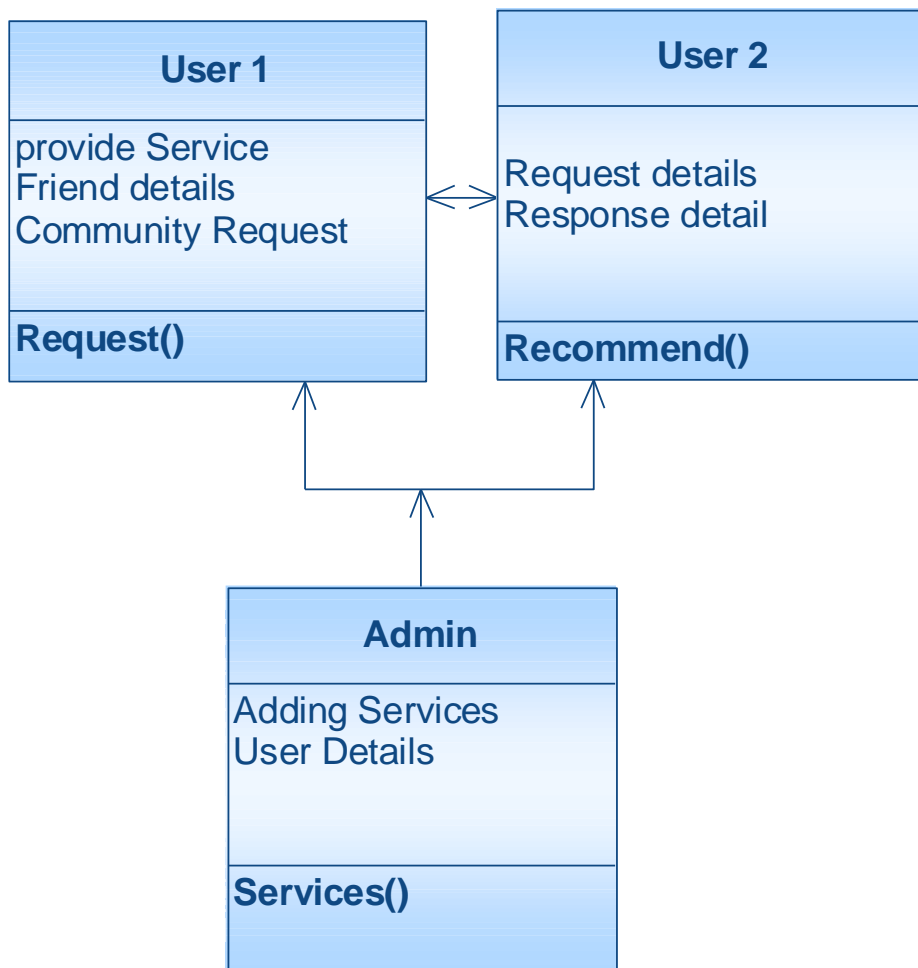


Fig.no. 12

SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

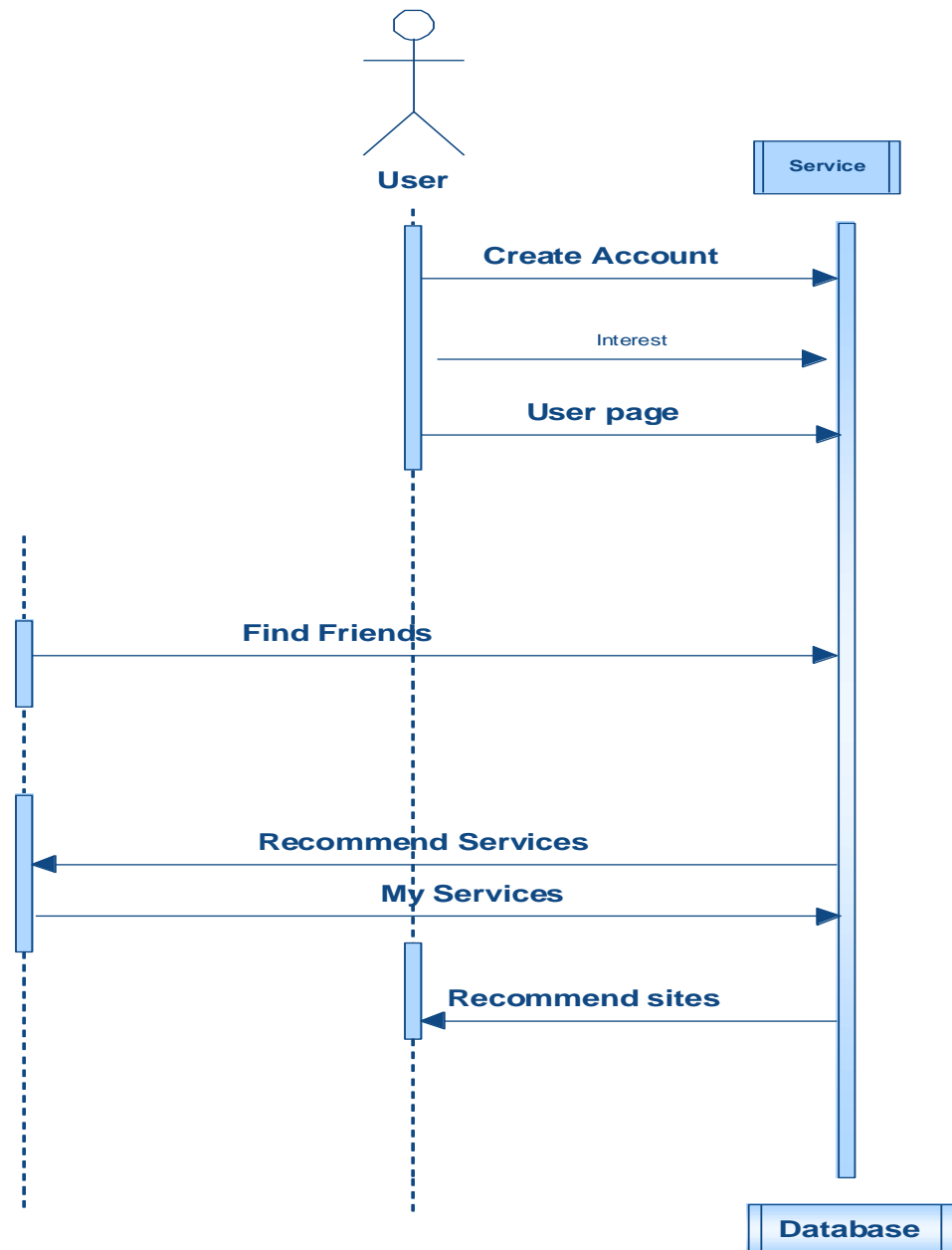


Fig.no. 13

ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

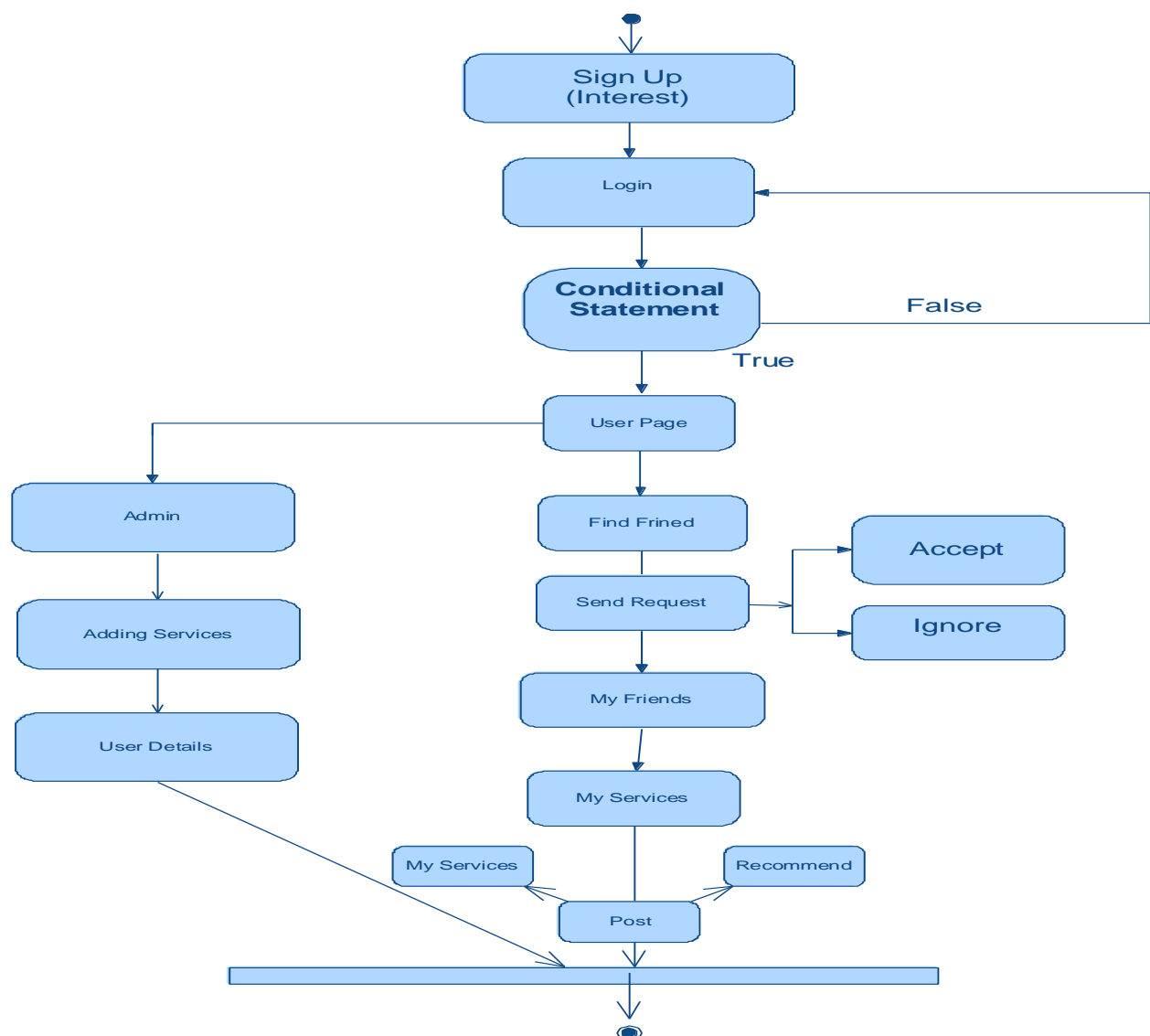


Fig.no. 14

CHAPTER 8

IMPLEMENTATION

MODULES:

- ⊗ Life Style Modeling
- ⊗ Activity Recognition
- ⊗ Friend-matching Graph Construction
- ⊗ User Impact Ranking

MODULES DESCRIPTION:

Life Style Modeling

Life styles and activities are reflections of daily lives at two different levels where daily lives can be treated as a mixture of life styles and life styles as a mixture of activities. This is analogous to the treatment of documents as ensemble of topics and topics as ensemble of words. By taking advantage of recent developments in the field of text mining, we model the daily lives of users as life documents, the life styles as topics, and the activities as words. Given “documents”, the probabilistic topic model could discover the probabilities of underlying “topics”. Therefore, we adopt the probabilistic topic model to discover the probabilities of hidden “life styles” from the “life documents”. Our objective is to discover the life style vector for each user given the life documents of all users.

Activity Recognition

We need to first classify or recognize the activities of users. Life styles are usually reflected as a mixture of motion activities with different occurrence probability. Generally speaking, there are two mainstream approaches: supervised learning and unsupervised learning. For both approaches, mature techniques have been developed and tested. In practice, the number of activities involved in the analysis is unpredictable and it is difficult to collect a large set of ground truth data for each activity, which makes supervised learning algorithms unsuitable for our system. Therefore, we use unsupervised learning approaches to recognize activities.

Friend-matching Graph Construction

To characterize relations among users, in this section, we propose the friend-matching graph to represent the similarity between their life styles and how they influence other people in the graph. In particular, we use the link weight between two users to represent the similarity of their life styles. Based on the friend-matching graph, we can obtain a user's affinity reflecting how likely this user will be chosen as another user's friend in the network. We define a new similarity metric to measure the similarity between two life style vectors. Based on the similarity metric, we model the relations between users in real life as a friend-matching graph. The friend-matching graph has been constructed to reflect life style relations among users.

User Impact Ranking

The impact ranking means a user's capability to establish friendships in the network. In other words, the higher the ranking, the easier the user can be made friends with, because he/she shares broader life styles with others. Once the ranking of a user is obtained, it provides guidelines to those who receive the recommendation list on how to choose friends. The ranking itself, however, should be independent from the query user. In other words, the ranking depends only on the graph structure of the friend-matching graph, which contains two aspects: 1) how the edges are connected, 2) how much weight there is on every edge. Moreover, the ranking should be used together with the similarity scores between the query user and the potential friend candidates, so that the recommended friends are those who not only share sufficient similarity with the query user, and are also popular ones through whom the query user can increase their own impact rankings.

CHAPTER 9

SCREEN SHOTS

Index page:

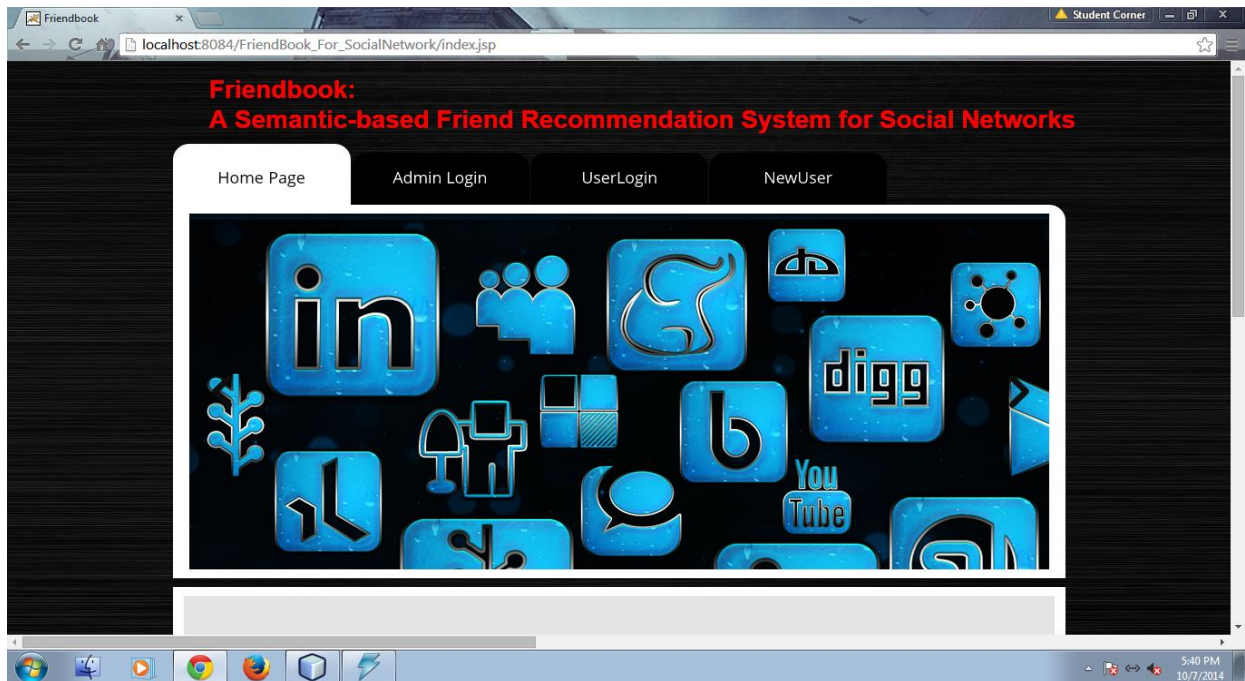


Fig.no. 15

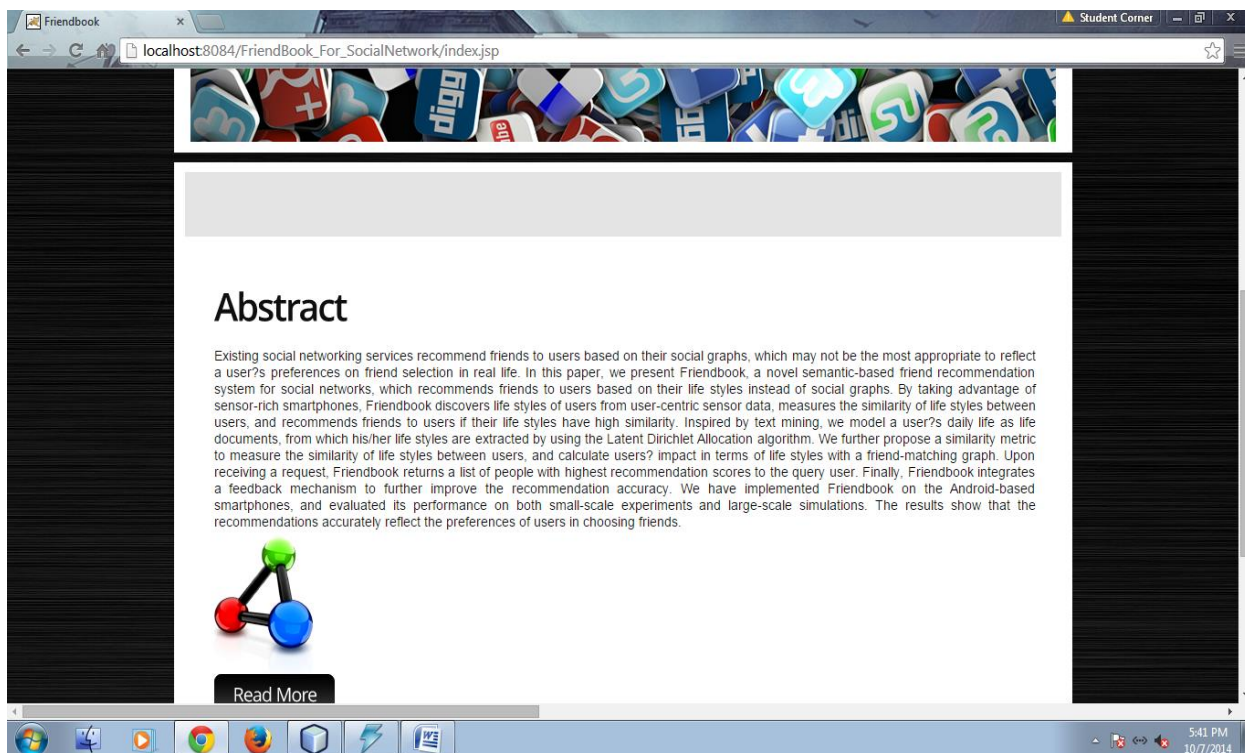


Fig.no. 16

Admin Login:

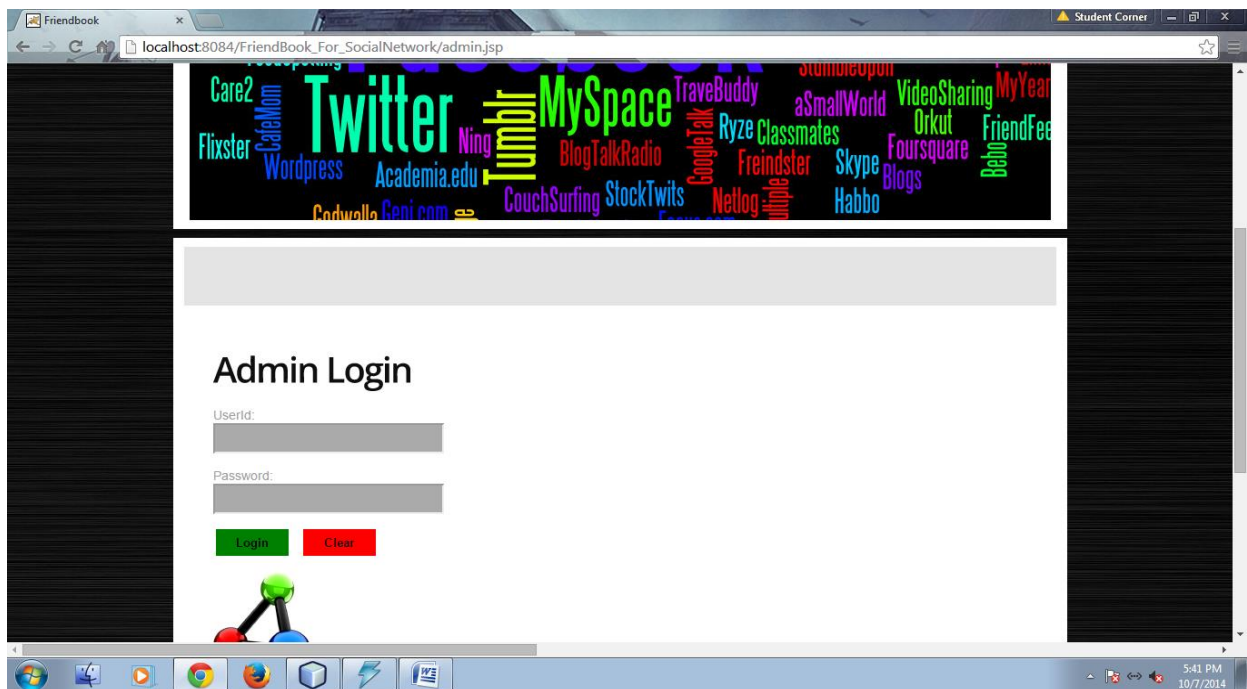


Fig.no. 17

Admin Home page:

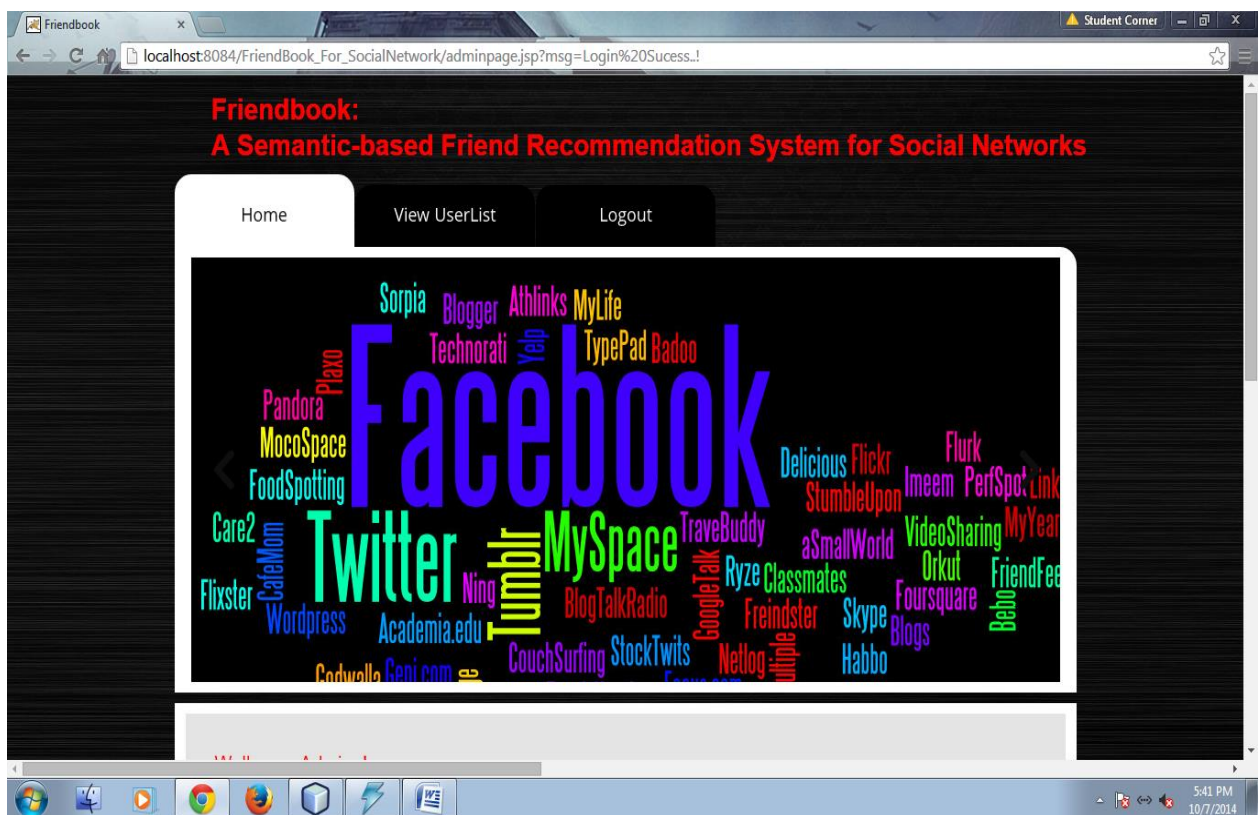


Fig.no. 18

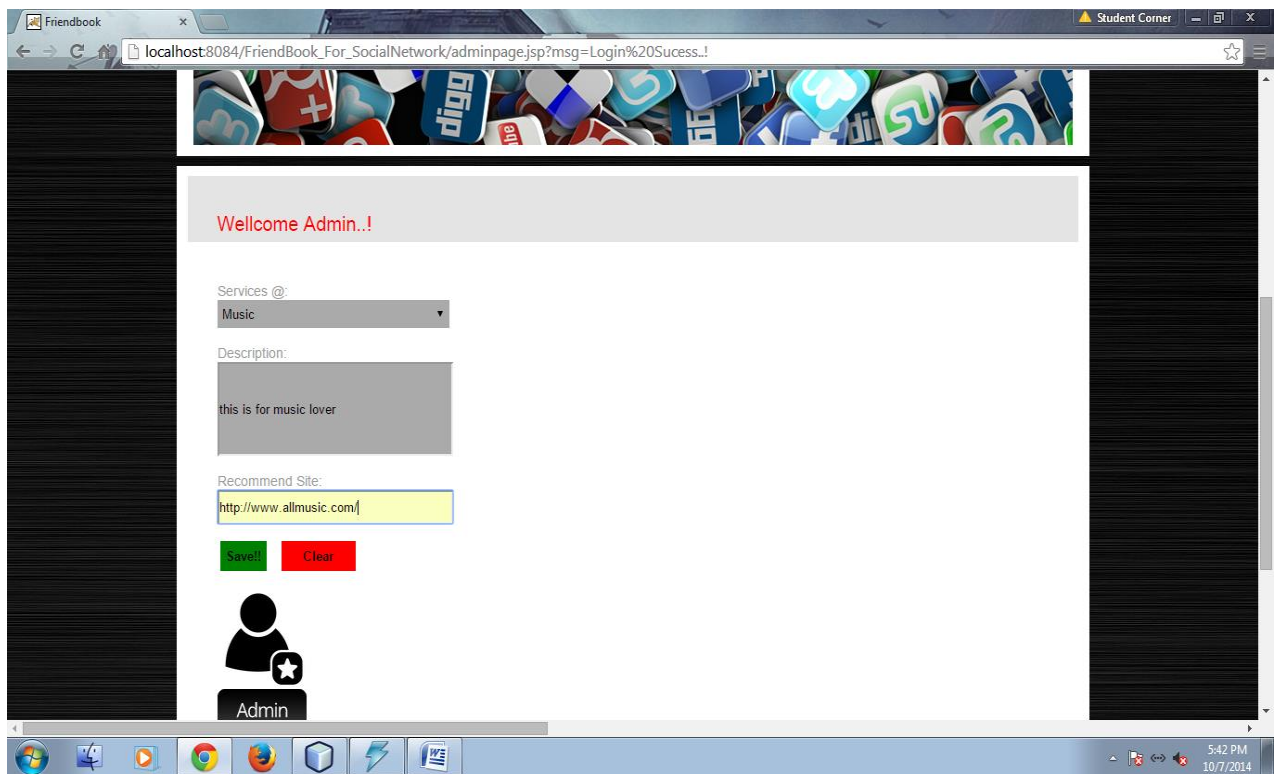


Fig.no. 19

User List:

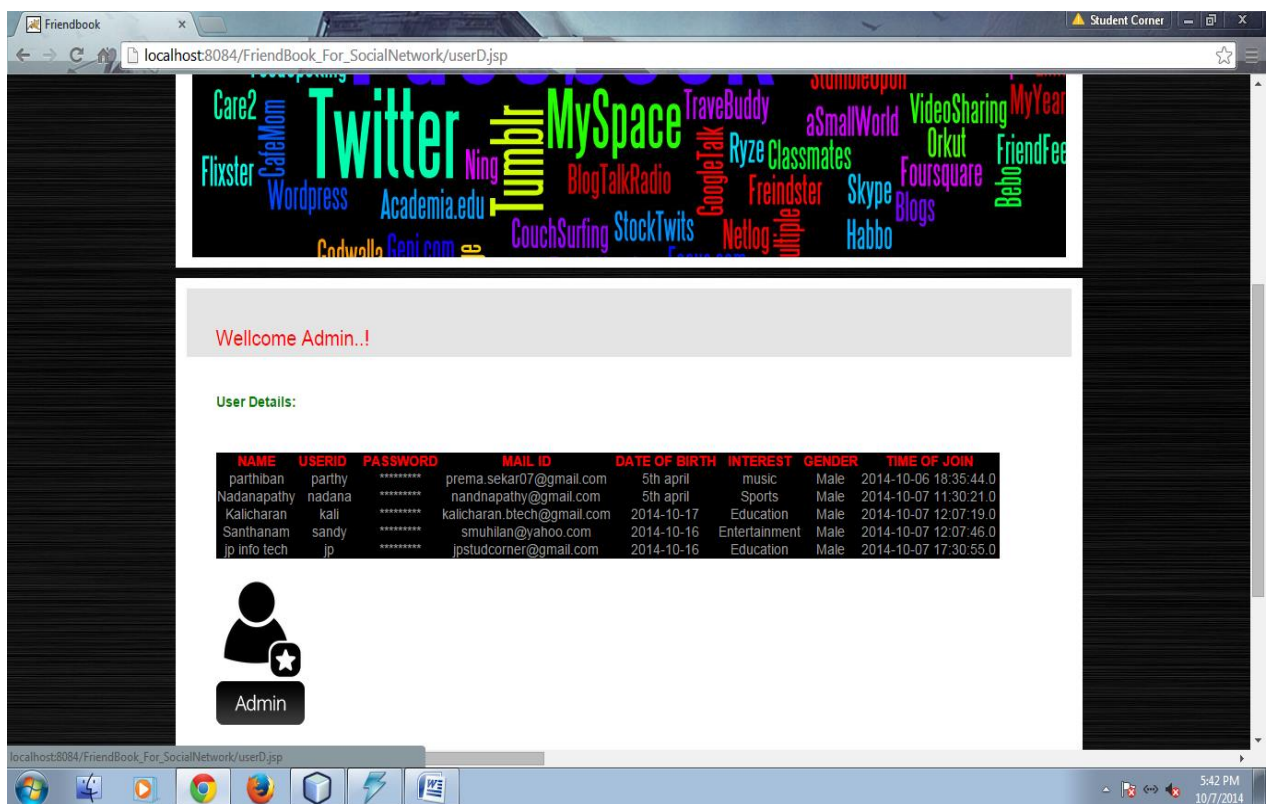
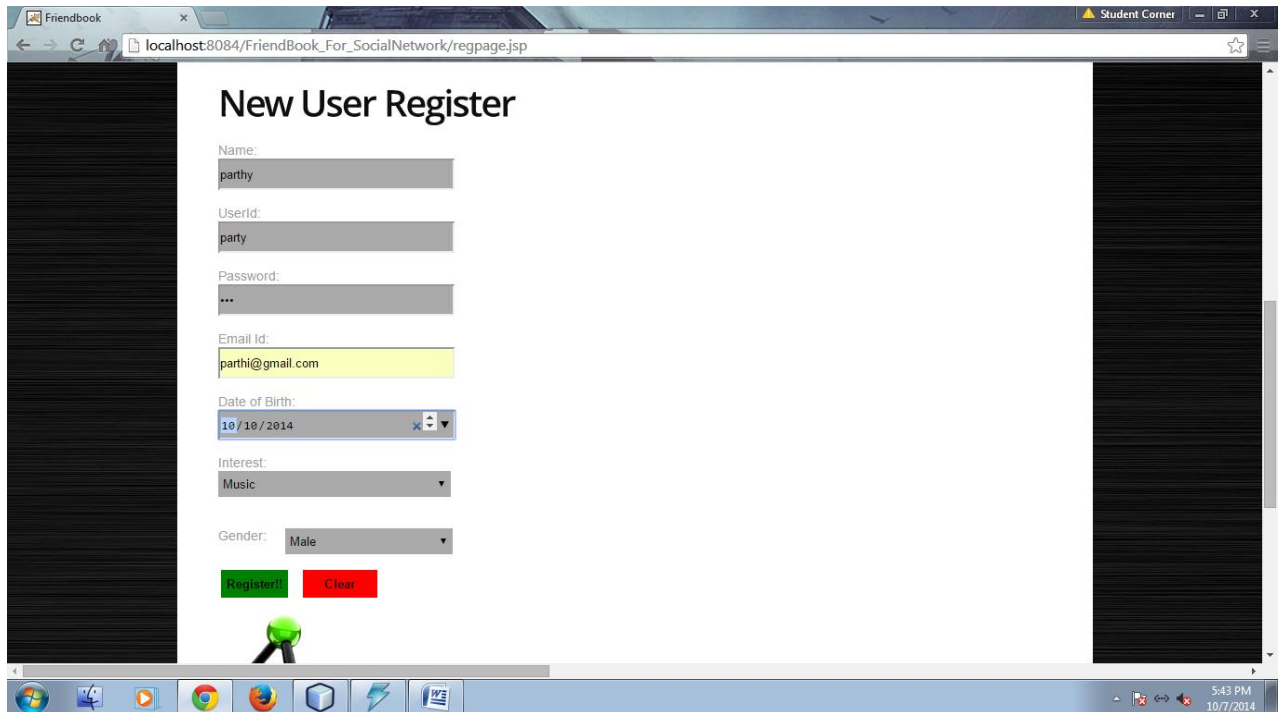


Fig.no. 20

New User Registration:



The screenshot displays a web browser window with the title 'Friendbook'. The address bar shows the URL 'localhost:8084/FriendBook_For_SocialNetwork/regpage.jsp'. The main content area is titled 'New User Register' and contains the following form fields:

- Name:
- UserId:
- Password:
- Email Id:
- Date of Birth:
- Interest:
- Gender:

At the bottom of the form, there are two buttons: a green 'Register' button and a red 'Clear' button. The browser's taskbar at the bottom shows various application icons and the system clock indicating 5:43 PM on 10/7/2014.

Fig.no. 21

User Login:

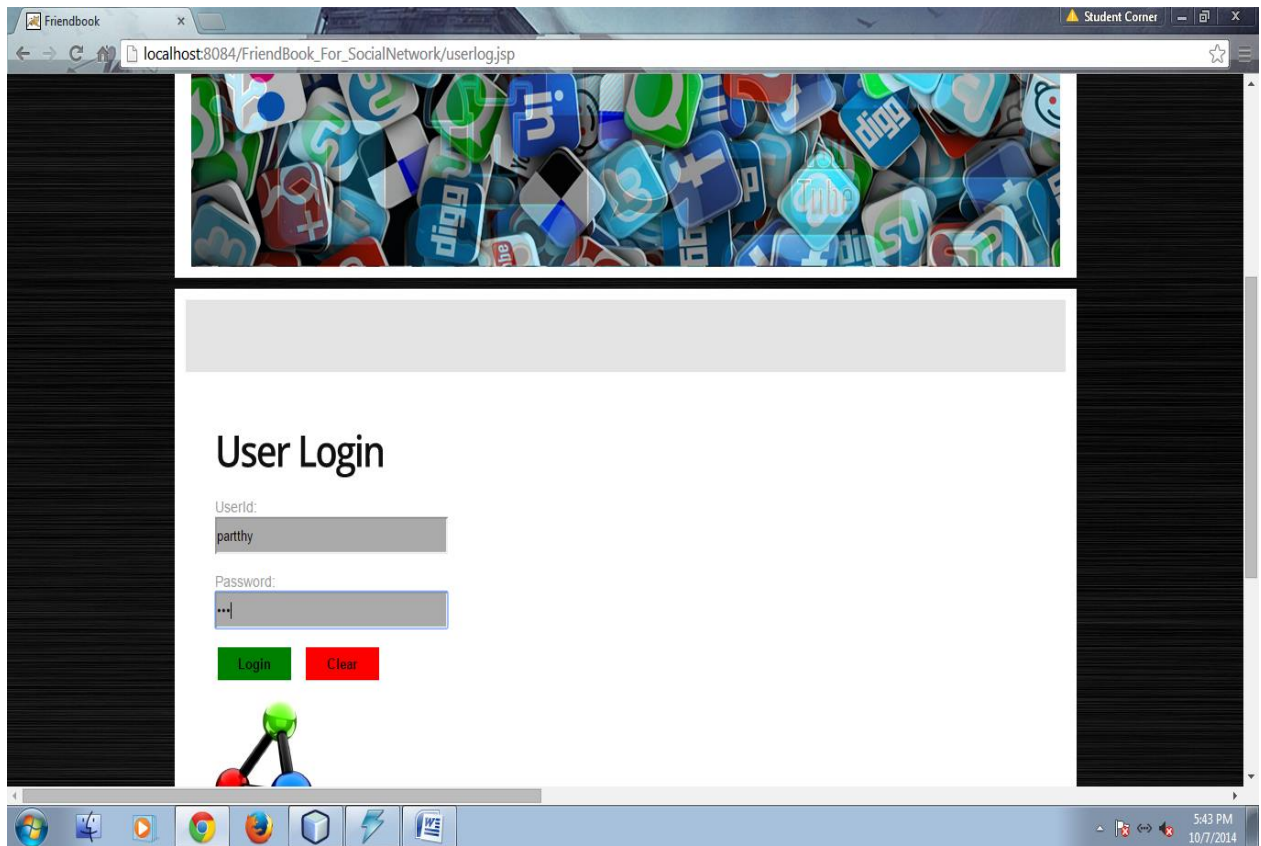


Fig.no. 22

User Home Page:



Fig.no. 23

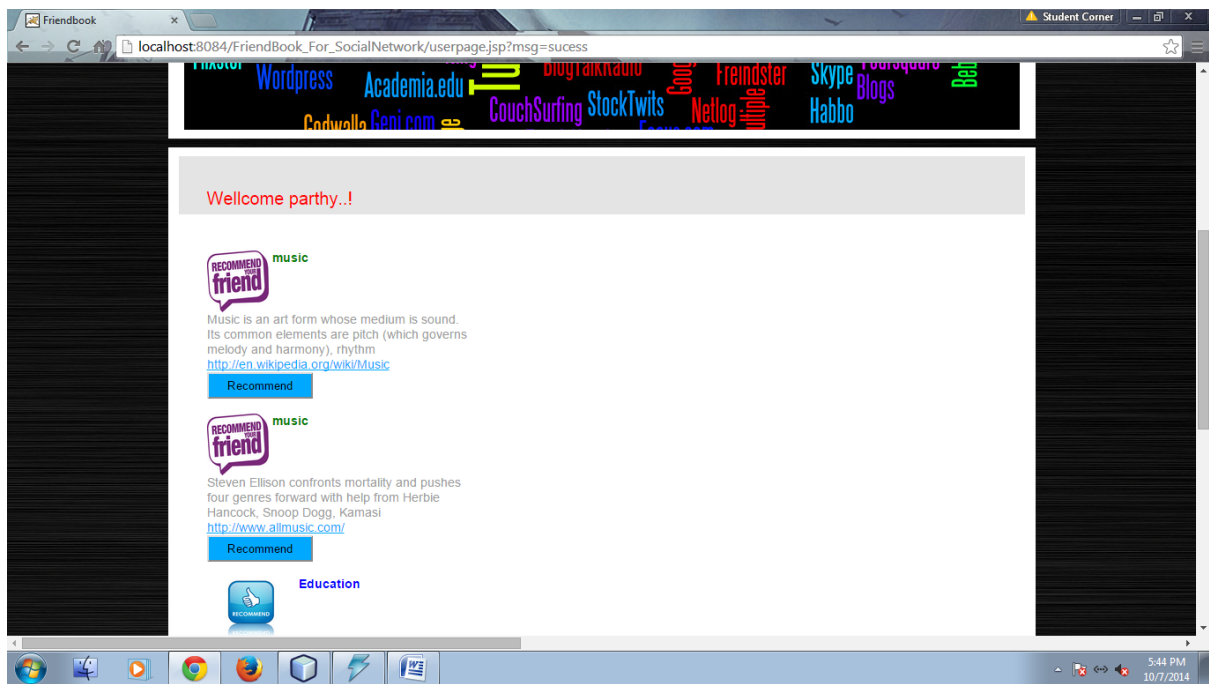


Fig.no. 24

Adding Friends:



Fig.no. 25

My Friends List:



Fig.no. 26

Recommend sites from Friends:

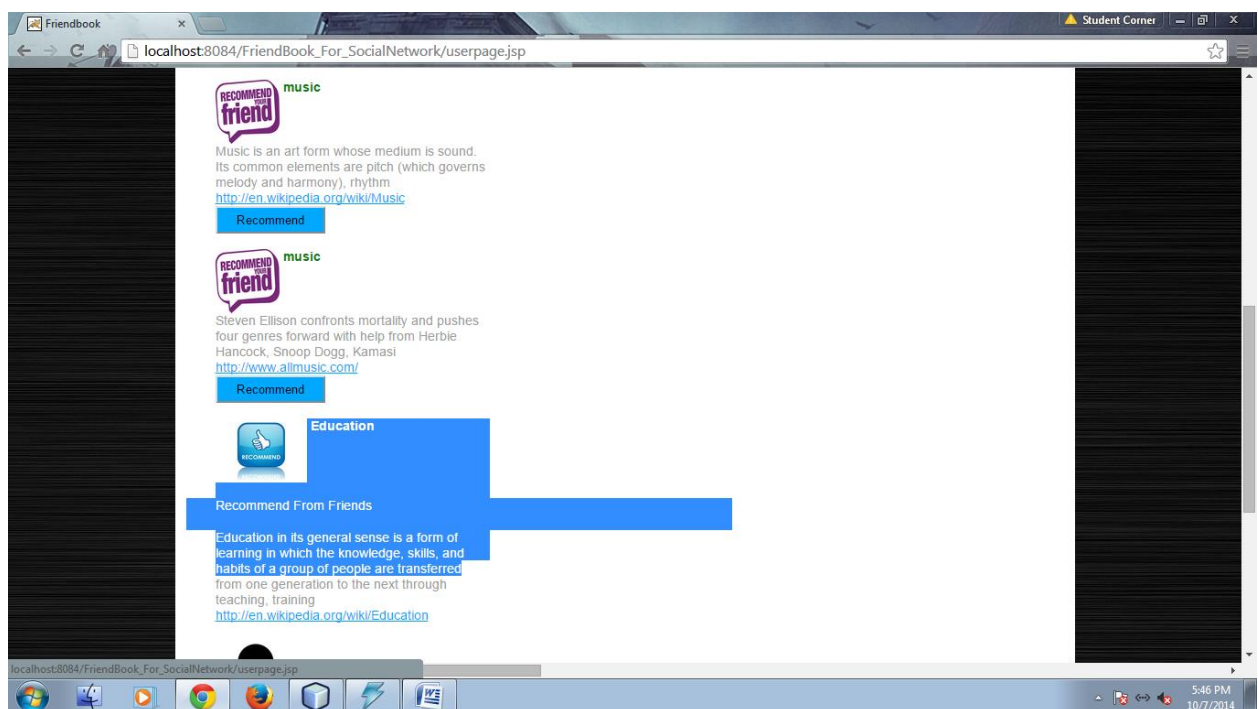


Fig.no. 27

CHAPTER 10

INPUT DESIGN AND OUTPUT DESIGN

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner, the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
2. Select methods for presenting information.
3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- ❖ Convey information about past activities, current status or projections of the
- ❖ Future.
- ❖ Signal important events, opportunities, problems, or warnings.
- ❖ Trigger an action.
- ❖ Confirm an action.

CHAPTER 11

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows, data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

6.1 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

6.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

6.3 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 12

CONCLUSION

In this project, we presented the design and implementation of Friendbook, a semantic-based friend recommendation system for social networks. Different from the friend recommendation mechanisms relying on social graphs in existing social networking services, Friendbook extracted life styles from user-centric data collected from sensors on the smartphone and recommended potential friends to users if they share similar life styles. We implemented Friendbook on the Android-based smartphones. The results showed that the recommendations accurately reflect the preferences of users in choosing friends. Beyond the current prototype, the future work can be four-fold. First, we would like to evaluate our system on large-scale field experiments. Second, we intend to implement the life style extraction using LDA and the iterative matrix-vector multiplication method in user impact ranking incrementally, so that Friendbook would be scalable to large-scale systems. Third, the similarity threshold used for the friend-matching graph is fixed in our current prototype of Friendbook. It would be interesting to explore the adaption of the threshold for each edge and see whether it can better represent the similarity relationship on the friend-matching graph. At last, we plan to incorporate more sensors on the mobile phones into the system and also utilize the information from wearable equipments (e.g., Fitbit, Google glass) to discover more interesting and meaningful life styles. For example, we can incorporate the sensor data source from Fitbit, which extracts the user's daily fitness infograph, and the user's place of interests from GPS traces to generate an infograph of the user as a "document". From the infograph, one can easily visualize a user's life style which will make more sense on the recommendation. Actually, we expect to incorporate Friendbook into existing social services (e.g., Facebook, Twitter, LinkedIn) so that Friendbook can utilize more information for life discovery, which should improve the recommendation experience in the future.

REFERENCES

- [1] G. R. Arce. Nonlinear Signal Processing: A Statistical Approach. John Wiley & Sons, 2005.
- [2] B. Bahmani, A. Chowdhury, and A. Goel. Fast incremental and personalized pagerank. Proc. of VLDB Endowment, volume 4, pages 173-184, 2010.
- [3] J. Biagioni, T. Gerlich, T. Merrifield, and J. Eriksson. EasyTracker: Automatic Transit Tracking, Mapping, and Arrival Time Prediction Using Smartphones. Proc. of SenSys, pages 68-81, 2011.
- [4] L. Bian and H. Holtzman. Online friend recommendation through personality matching and collaborative filtering. Proc. of UBICOMM, pages 230-235, 2011.
- [5] C. M. Bishop. Pattern recognition and machine learning. Springer New York, 2006.
- [6] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. Journal of Machine Learning Research, 3:993-1022, 2003.
- [7] P. Desikan, N. Pathak, J. Srivastava, and V. Kumar. Incremental page rank computation on evolving graphs. Proc. of WWW, pages 1094-1095, 2005.
- [8] N. Eagle and A. S. Pentland. Reality Mining: Sensing Complex Social Systems. Personal Ubiquitous Computing, 10(4):255-268, March 2006.
- [9] K. Farrahi and D. Gatica-Perez. Probabilistic mining of sociogeographic routines from mobile phone data. Selected Topics in Signal Processing, IEEE Journal of, 4(4):746-755, 2010.
- [10] K. Farrahi and D. Gatica-Perez. Discovering Routines from Largescale Human Locations using Probabilistic Topic Models. ACM Transactions on Intelligent Systems and Technology (TIST), 2(1), 2011.
- [11] B. A. Frigyik, A. Kapila, and M. R. Gupta. Introduction to the dirichlet distribution and related processes. Department of Electrical Engineering, University of Washington, UWEETR-2010-0006, 2010.