#### A Project Report On

**User Vitality Ranking and Prediction in Social Networking Services**

A dissertation work submitted in partial fulfillment of the requirement for the award of degree

# Bachelor of Technology

In

**Computer Science and Engineering**

### Submitted By

C.Mahitha [16J21A0514]

G.Lekha Sri [16J21A0526]

J.S.L.Sameera [16J21A0532]

K.Vaishnavi [16J21A0535]

### Under the Guidance Of

**Mrs.B.SWAPNA B.Tech, M.Tech**

**Assistant Professor**



**Department of Computer Science and Engineering JOGINPALLY B.R. ENGINEERING COLLEGE**

### (Approved by AICITE & Affiliated to Jawaharlal Nehru Technological University Hyderabad &Accredited by NAAC with B++, Yenkapally, Moinabad (Mandal) R.R (Dist)-500075, Telangana, India)

#### 2016-2020

**JOGINPALLY B.R. ENGINEERING COLLEGE**

### (Approved by AICITE & Affiliated to Jawaharlal Nehru Technological University Hyderabad &Accredited by NAAC with B++, Yenkapally, Moinabad (Mandal) R.R (Dist)-500075, Telangana, India)



**CERTIFICATE**

This is to certify that the project entitled “**User Vitality Ranking and Prediction in Social Networking Services**” that is being submitted by **C.Mahitha [16J21A0514], G.Lekha Sri[16J21A0526], J.S.L.Sameera [16J21A0532], K.Vaishnavi [16J21A0535],** in partial fulfillment of the award of Bachelor of Technology in Computer Science and Engineering to Jawaharlal Nehru Technological University Hyderabad. It is record of bonafide work carried out guidance and supervision. The results embodied in this thesis have not been submitted to any other university or Institute for award of any degree. In my opinion, this thesis is of standard required for the degree of Bachelor of Technology.

**Internal Guide** **HOD CSE**

**Mrs.B.Swapna** B.Tech,M.Tech **Dr.M.Giri** B.Tech,M.Tech,PhD.

Assistant Professor

External Examiner

# DECLARATION

We hereby declare that the project work entitled “**User Vitality Ranking and Prediction in Social Networking Services”**, developed under the supervision of **Mrs.B**.**Swapna,** Assistant Professor, JBREC and submitted to Joginpally B.R Engineering College is original and has not been submitted in part or whole for Bachelor degree to any other university.

C.Mahitha [16J21A0514]

G.Lekha Sri [16J21A0526]

J.S.L.Sameera [16J21A0532]

K.Vaishnavi [16J21A0535]

# ACKNOWLEDGMENT

We would like to express our deep sense of gratitude to **Dr.M.Giri**, head of the department, Computer Science and Engineering, Joginpally B R Engineering College.

We would like to extend my profound sense of gratitude to my internal guide **Mrs.B.Swapna**, Assistant Professor, JBREC for installing confidence in us by appreciating this work on “**User Vitality Ranking and Prediction in Social Networking Services”** since beginning.

We thank

Our Principal **Dr. V. Usha Shree** for her encouragement and help to us through the

Course; we also thank the college management for their support, cooperation and encouragement.

We are thankful to our parents who dreamed my Bachelor Degree.

C.Mahitha [16J21A0514]

G.Lekha Sri [16J21A0526]

J.S.L.Sameera [16J21A0532]

K.Vaishnavi [16J21A0535]

**ABSTRACT**

Social networking services have been prevalent at many online communities such as Twitter.com and Weibo.com, where millions of users keep interacting with each other every day. One interesting and important problem in the social networking services is to rank users based on their vitality in a timely fashion. An accurate ranking list of user vitality could benefit many parties in social network services such as the ads providers and site operators. Although it is very promising to obtain a vitality-based ranking list of users, there are many technical challenges due to the large scale and dynamics of social networking data. In this paper, we propose a unique perspective to achieve this goal, which is quantifying user vitality by analyzing the dynamic interactions among users on social networks. Examples of social network include but are not limited to social networks in micro blog sites and academicals collaboration networks. Intuitively, if a user has many interactions with his friends within a time period and most of his friends do not have many interactions with their friends simultaneously, it is very likely that this user has high vitality. Based on this idea, we develop quantitative measurements for user vitality and propose our first algorithm for ranking users based vitality. Also we further consider the mutual Influence between users while computing the vitality measurements and propose the second ranking algorithm, which computes user vitality in an iterative way. Other than user vitality ranking, we also introduce a vitality prediction problem, which is also of great importance for many applications in social networking services. Along this line, we develop a customized prediction model to solve the vitality prediction problem. To evaluate the performance of our algorithms, we collect two dynamic social network data sets. The experimental results with both data sets clearly demonstrate the advantage of our ranking and prediction methods.

**LIST OF CONTENTS**

Page Nos.

1. Introduction

1.1 Purpose

1.2 Scope and Objective

2. Literature Survey

3. Fundamental Concepts on (Domain)

4. System Analysis

4.1 Existing System

4.1.1 Drawbacks

4.2 Problem statement

4.3 Proposed System

4.3.1 Advantages

4.4 Modules Description

4.5 Feasibility Study

4.5.1 Economic Feasibility

4.5.2 Operational Feasibility

4.5.3 Technical Feasibility

5. System Requirements Specification

5.1 Introduction

5.2 Purpose

5.3 Functional Requirements

5.4 Non Functional Requirements

5.5 Input & Output Design

5.6 Hardware Requirements

5.6 Software Requirements

6. System Design

6.1 System Specifications

6.2 System Components

6.3 DFD’s

6.4 UML Diagrams

6.5 Data Dictionaries and ER Diagram

7. Implementation

7.1 Technology Description

8. Coding

9. System Testing

9.1 Testing Methodologies

9.2 Test cases

9.3 Result Analysis

10. Output Screens

11. Reports

12. Conclusion

13. Future Enhancements

14. References

15. Bibliography

1. **INTRODUCTION**

With the development of web technology, social networking service has been prevalent at many online platforms. The social networking service facilitates the building of social networks or social relations among users who, for instance, share interest, activities, background and physical connections. Through such service, users could stay connected with each other and be informed of friends’ behaviors such as posting at a platform, and consequently be influenced by each other. For instance, in today’s Twitter and Weibo (one of the most popular social networking sites in China), a user can get the instant updates about his connected friends’ postings and could further retweet or comment the postings. Within a time period, millions of users may take different actions such as posting and retweeting at these social networking sites. One interesting and important problem is how to rank users based on their vitality with historical data . An accurate vitality ranking of users will provide great insight for many applications in most online social networking sites. For instance, online ads providers may make better strategy

for delivering their ads via considering the ranked vitality of users; site operators may design better practices for online campaigns (e.g., online survey) via leveraging the ranking list. While it is very promising for many parties to provide a vitality ranking of users, there are many technical challenges to tackle this problem. First, to decide the vitality of a user, we could not only examine his own interaction with others, but also need to look into the interactions of other users collectively.

For instance, suppose one user has had many interactions with most of his friends in a time period, we may conclude different vitality of this user when most of his friends also have had many interactions in the same time period versus when most of his friends do not have had many interactions. Second, as the scale of social networks increases, it becomes more challenging to rank the vitality of users because a large number of nodes (users) may influence the vitality of an individual node (user). Third, as the social networks in many online sites evolve over time, the vitality of users may also change over time. Thus efficient methods are needed to dynamically obtain the vitality of users at different times.

* 1. **Purpose**

The purpose of the project is the which is based on dynamic interactions between users on social While the first algorithm works based on the developed two user vitality measurements, the second algorithm further takes into account the mutual influence among users while computing the vitality measurements. Then we presented a user vitality prediction problem and introduced a regressionbased method for the prediction task. Intensive experiments on two real-world data sets that are collected from different domains clearly demonstrate the effectiveness of our ranking and prediction methods. The accurate results of both user vitality ranking and prediction could benefit many parties in different social networking services, e.g., a user vitality ranking list could help ads providers to better display their ads to active users and reach more audiences

* 1. **Scope and Objective**

**Scope:-** The scope of the project is the it is very likely that this user has high vitality. Based on this idea, we develop quantitative measurements for user vitality and propose our first algorithm for ranking users based vitality. Also we further consider the mutual influence between users while computing the vitality measurements and propose the second ranking algorithm, which computes user vitality in an iterative way. Other than user vitality ranking, we also introduce a vitality prediction problem, which is also of greatimportance for many applications in social networking services

**Objective:--**

Social networking services have been prevalent at many online communities such as Twitter.com and Weibo.com, where millions of users keep interacting with each other every day. One interesting and important problem in the social networking services is to rank users based on their vitality in a timely fashion. An accurate ranking list of user vitality could benefit many parties in social network services such as the ads providers and site operators. Although it is very promising to obtain a vitality-based ranking list of users, there are many technical challenges due to the large scale and dynamics of social networking data. In this paper, we propose a unique perspective to achieve this goal, which is quantifying user vitality by analyzing the dynamic interactions among users on social networks. Examples of social network include but are not limited to social networks in micro blog sites and academicals collaboration networks. Intuitively, if a user has many interactions with his friends within a time period and most of his friends do not have many interactions with their friends simultaneously, it is very likely that this user has high vitality. Based on this idea, we develop quantitative measurements for user vitality and propose our first algorithm for ranking users based vitality. Also we further consider the mutual Influence between users while computing the vitality measurements and propose the second ranking algorithm, which computes user vitality in an iterative way. Other than user vitality ranking, we also introduce a vitality prediction problem, which is also of great importance for many applications in social networking services. Along this line, we develop a customized prediction model to solve the vitality prediction problem. To evaluate the performance of our algorithms, we collect two dynamic social network data sets. The experimental results with both data sets clearly demonstrate the advantage of our ranking and prediction methods.

**2. LITERATURE SURVEY : USER VITALITY**

**Everyone’s an influencer: quantifying influence on twitter**

* In this paper we investigate the attributes and relative influence of 1.6M Twitter users by tracking 74 million diffusion events that took place on the Twitter follower graph over a two month interval in 2009.
* Unsurprisingly, we find that the largest cascades tend to be generated by users who have been influential in the past and who have a large number of followers.
* We also find that URLs that were rated more interesting and/or elicited more positive feelings by workers on Mechanical Turk were more likely to spread.
* In spite of these intuitive results, however, we find that predictions of which particular user or URL will generate large cascades are relatively unreliable.
* We conclude, therefore, that wordof-mouth diffusion can only be harnessed reliably by targeting large numbers of potential influencers, thereby capturing average effects.
* Finally, we consider a family of hypothetical marketing strategies, defined by the relative cost of identifying versus compensating potential “influencers.”
* We find that although under some circumstances, the most influential users are also the most cost-effective, under a wide range of plausible assumptions the most cost-effective performance can be realized using “ordinary influencers”— individuals who exert average or even less-than-average in- fluence.
* Word-of-mouth diffusion has long been regarded as an important mechanism by which information can reach large populations, possibly influencing public opinion, adoption of innovations, new product market share, or brand awareness.
* In recent years, interest among researchers and marketers alike has increasingly focused on whether or not diffusion can be maximized by seeding a piece of information or a new product with certain special individuals, often called “influentials” or simply “influencers,” who exhibit some combination of desirable attributes—whether personal attributes like credibility, expertise, or enthusiasm, or network attributes such as connectivity or centrality that allows them to influence a disproportionately large number of others, possibly indirectly via a cascade of influence.
* Although appealing, the claim that word-of-mouth diffusion is driven disproportionately by a small number of key influencers necessarily makes certain assumptions about the underlying influence process that are not based directly on empirical evidence.
* Empirical studies of diffusion are therefore highly desirable, but historically have suffered from two major difficulties.
* First, the network over which word-ofmouth influence spreads is generally unobservable, hence influence is difficult to attribute accurately, especially in instances where diffusion propagates for multiple steps.
* And second, observational data on diffusion are heavily biased towards “successful” diffusion events, which by virtue of being large are easily noticed and recorded; thus inferences regarding the attributes of success may also be biased , especially when such events are rare.
* For both of these reasons, the micro-blogging service Twitter presents a promising natural laboratory for the study of diffusion processes.
* Unlike other user-declared networks (e.g. Facebook), Twitter is expressly devoted to disseminating information, in that users subscribe to broadcasts of other users; thus the network of “who listens to whom” can be reconstructed by crawling the corresponding “follower graph”.
* In addition, because users frequently wish to share web-content, and because tweets are restricted to 140 characters in length, a popular strategy has been to use URL shorteners (e.g. bit.ly, TinyURL, etc.), which effectively tag distinct pieces of content with unique, easily identifiable tokens.
* Together these features allow us to track the diffusion patterns of all instances in which shortened URLs are shared on Twitter, regardless of their success, thereby addressing both the observability and sampling difficulties outlined above.
* The Twitter ecosystem is also well suited to studying the role of influencers.
* In general, influencers are loosely defined as individuals who disproportionately impact the spread of information or some related behavior of interest.
* Unfortunately, however, this definition is fraught with ambiguity regarding the nature of the influence in question, and hence the type of individual who might be considered special.
* Ordinary individuals communicating with their friends, for example, may be considered influencers, but so may subject matter experts, journalists, and other semi-public figures, as may highly visible public figures like media representatives, celebrities, and government officials.
* Clearly these types of individuals are capable of influencing very different numbers of people, but may also exert quite different types of influence on them, and even transmit influence through different media.
* For example, a celebrity endorsing a product on television or in a magazine advertisement presumably exerts a different sort of influence than a trusted friend endorsing the same product in person, who in turn exerts a different sort of influence than a noted expert writing a review.
* In light of this definitional ambiguity, an especially useful feature of Twitter is that it not only encompasses various types of entities, but also forces them all to communicate in roughly the same way: via tweets to their followers.
* Although it remains the case that even users with the same number of followers do not necessarily exert the same kind of influence, it is at least possible to measure and compare the influence of individuals in a standard way, by the activity that is observable on Twitter itself.
* In this way, we avoid the need to label individuals as either influencers or non-influencers, simply including all individuals in our study and comparing their impact directly.
* We note, however, that our use of the term influencer corresponds to a particular and somewhat narrow definition of influence, specifically the user’s ability to post URLs which diffuse through the Twitter follower graph.
* We restrict our study to users who “seed” content, meaning they post URLs that they themselves have not received through the follower graph.
* We quantify the influence of a given post by the number of users who subsequently repost the URL, meaning that they can be traced back to the originating user through the follower graph.
* We then fit a model that predicts influence using an individual’s attributes and past activity and examine the utility of such a model for targeting users.
* Our emphasis on prediction is particularly relevant to our motivating question.
* In marketing, for example, the practical utility of identifying influencers depends entirely on one’s ability to do so in advance.
* Yet in practice, it is very often the case that influencers are identified only in retrospect, usually in the aftermath of some outcome of interest, such as the unexpected success of a previously unknown author or the sudden revival of a languishing brand.
* By emphasizing ex-ante prediction of influencers over ex-post explanation, our analysis highlights some simple but useable insights that we believe are of general relevance to word of mouth marketing and related activities.
* The remainder of the paper is organized as follows.
* We review related work on modeling diffusion and quantifying influence in Section 2.
* In Sections 3 and 4 we provide an overview of the collected data, summarizing the structure of URL cascades on the Twitter follower graph. I
* n Section 5, we present a predictive model of influence, in which cascade sizes of posted URLs are predicted using the individuals’ attributes and average size of past cascades.
* Section 6 explores the relationship between content as characterized by workers on Amazon’s Mechanical Turk and cascade size.
* Finally, in Section 7 we use our predictive model of cascade size to examine the cost-effectiveness of targeting individuals to seed content.
* A number of recent empirical papers have addressed the matter of diffusion on networks in general, and the attributes and roles of influencers specifically.
* In early work, Gruhl et al attempted to infer a transmission network between bloggers, given time-stamped observations of posts and assuming that transmission was governed by an independent cascade model.
* Contemporaneously, Adar and Adamic used a similar approach to reconstruct diffusion trees among bloggers, and shortly afterwards Leskovec et al. used referrals on an e-commerce site to infer how individuals are influenced as a function of how many of their contacts have recommended a product.
* A limitation of these early studies was the lack of “ground truth” data regarding the network over which the diffusion was taking place.
* Addressing this problem, more recent studies have gathered data both on the diffusion process and the corresponding network.
* For example, Sun et al. studied diffusion trees of fan pages on Facebook, Bakshy et al. studied the diffusion of “gestures” between friends in Second Life, and Aral et al. studied adoption of a mobile phone application over the Yahoo! messenger network.
* Most closely related to the current research is a series of recent papers that examine influence and diffusion on Twitter specifically.
* Compared three different measures of influence number of followers, page-rank, and number of retweets finding that the ranking of the most influential users differed depending on the measure.
* Cha et alalso compared three different measures of influence number of followers, number of retweets, and number of mentions and also found that the most followed users did not necessarily score highest on the other measures.
* Finally, Weng et al. compared number of followers and page rank with a modified page-rank measure that accounted for topic, again finding that ranking depended on the influence measure.
* The present work builds on these earlier contributions in three key respects.
* First, whereas previous studies have quantified influence either in terms of network metrics (e.g. page rank) or the number of direct, explicit retweets, we measure influence in terms of the size of the entire diffusion tree associated with each event (Kwak et al also compute what they call “retweet trees” but they do not use them as a measure of influence).
* While related to other measures, the size of the diffusion tree is more directly associated with diffusion and the dissemination of information (Goyal et al , it should be noted, do introduce a similar metric to quantify influence; however, their interest is in identifying community “leaders,” not on prediction.).
* Second, whereas the focus of previous studies has been largely descriptive (e.g. comparing the most influential users), we are interested explicitly in predicting influence; thus we consider all users, not merely the most influential.
* Third, in addition to predicting diffusion as a function of the attributes of individual seeds, we also study the effects of content.
* We believe these differences bring the understanding of diffusion on Twitter closer to practical applications, although as we describe later, experimental studies are still required.
* To study diffusion on Twitter, we combined two separate but related sources of data. First, over the two-month period of September 13 2009 - November 15 2009 we recorded all 1.03B public tweets broadcast on Twitter, excluding October 14-16 during which there were intermittent outages in the Twitter API.
* Of these, we extracted 87M tweets that included bit.ly URLs and which corresponded to distinct diffusion “events,” where each event comprised a single initiator, or “seed,” followed by some number of repostings of the same URL by the seed’s followers, their followers, and so on1 .
* Finally, we identified a subset of 74M diffusion events that were initiated by seed users who were active in both the first and second months of the observation period; thus enabling us to train our regression model on first month performance in order to predict second-month performance .
* In total, we identified 1.6M seed users who seeded an average of 46.33 bit.ly URLs each.

**Reprint of: The anatomy of a largescale hypertextual web search engine**

* present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems.
* The prototype with a full text and hyperlink database of at least 24 million pages is available .
* To engineer a search engine is a challenging task. Search engines index tens to hundreds of millions of Web pages involving a comparable number of distinct terms.
* They answer tens of millions of queries every day.
* Despite the importance of large-scale search engines on the Web, very little academic research has been done on them.
* Furthermore, due to rapid advance in technology and Web proliferation, creating a Web search engine today is very different from three years ago.
* This paper provides an in-depth description of our large-scale Web search engine - the first such detailed public description we know of to date.
* Apart from the problems of scaling traditional search techniques to data of this magnitude, there are new technical challenges involved with using the additional information present in hypertext to produce better search results.
* This paper addresses this question of how to build a practical large-scale system which can exploit the additional information present in hypertext.
* Also we look at the problem of how to effectively deal with uncontrolled hypertext collections where anyone can publish anything they want. 0 1998 Published by Elsevier Science B.V. All rights reserved.
* The amount of information on the Web is growing rapidly, as well as the number of new users inexperienced in the art of Web research.
* People are likely to surf the Web using its link graph, often starting with high quality human maintained indices such as Yahoo! 3 or with search engines.
* Human maintained lists cover popular topics effectively but are subjective, expensive to build and maintain, slow to improve, and cannot cover all esoteric topics.
* Automated search engines that rely on keyword matching usually return too many low quality matches.
* To make matters worse, some advertisers attempt to gain people’s attention by taking measures meant to mislead automated search engines.
* We have built a large-scale search engine which addresses many of the problems of existing systems.
* It makes especially heavy use of the additional structure present in hypertext to provide much higher quality search results.
* We chose our system name, Google, because it is a common spelling of googol, or 10100 and fits well with our goal of building very large-scale search engines.
* Search engine technology has had to scale dramatically to keep up with the growth of the Web.
* In 1994, one of the first Web search engines, the World Wide Web Worm (WWWW) [6] had an index of 110,000 Web pages and Web accessible documents.
* As of November. 1997, the top search engines claim to index from 2 million (WebCrawler) to 100 million Web documents (from Search Engine Watch4).
* It is foreseeable that by the year 2000, a comprehensive index of the Web will contain over a billion documents.
* At the same time, the number of queries search engines handle has grown incredibly too.
* In March and April 1994, the World Wide Web Worm received an average of about 1500 queries per day.
* In November 1997, Altavista claimed it handled roughly 20 million queries per day.
* With the increasing number of users on the Web, and automated systems which query search engines, it is likely that top search engines will handle hundreds of millions of queries per day by the year 2000.
* The goal of our system is to address many of the problems, both in quality and scalability, introduced by scaling search engine technology to such extraordinary numbers.
* These tasks are becoming increasingly difficult as the Web grows. However, hardware performance and cost have improved dramatically to partially offset the difficulty.
* There are, however, several notable exceptions to this progress such as disk seek time and operating system robustness.
* In designing Google, we have considered both the rate of growth of the Web and technological changes. Google is designed to scale well to extremely large data sets.
* It makes efficient use of storage space to store the index. Its data structures are optimized for fast and efficient access .
* Further, we expect that the cost to index and store text or HTML will eventually decline relative to the amount that will be available .
* This will result in favorable scaling properties for centralized systems like Google Design goals.
* Improved search quality he Web grows.
* However, hardware performance and cost have improved dramatically to partially offset the difficulty.
* There are, however, several notable exceptions to this progress such as disk seek time and operating system robustness.
* In designing Google, we have considered both the rate of growth of the Web and technological changes.
* Google is designed to scale well to extremely large data sets. It makes efficient use of storage space to store the index.
* Its data structures are optimized for fast and efficient access .
* Further, we expect that the cost to index and store text or HTML will eventually decline relative to the amount that will be available.
* This will result in favorable scaling properties for centralized systems like Googlearch engines.
* In 1994, some people believed that a complete search index would make it possible to find anything easily.
* According to Best of the Web 1994 - Navigators5, “The best navigation service should make it easy to find almost anything on the Web (once all the data is entered).”
* However, the Web of 1997 is quite different. Anyone who has used a search engine recently, can readily testify that the completeness of the index is not the only factor in the quality of search results.
* “Junk results” often wash out any results that a user is interested in.
* In fact, as of November 1997. only one of the top four commercial search engines finds itself (returns its own search page in response to its name in the top ten results).
* One of the main causes of this problem is that the number of documents in the indices has been increasing by many orders of magnitude, but the user’s ability to look at documents has not.
* People are still only willing to look at the first few tens of results.
* Because of this, as the collection size grows, we need tools that have very high precision (number of relevant documents returned, say in the top tens of results).
* Indeed, we want our notion of “relevant” to only include the very best documents since there may be tens of thousands of slightly relevant documents.
* This very high precision is important even at the expense of recall (the total number of relevant documents the system is able to return).
* There is quite a bit of recent optimism that the use of more hypertextual information can help improve search and other applications .
* In particular, link structure 171 and link text provide a lot of information for making relevance judgments and quality filtering.
* Academic search engine research Aside from tremendous growth, the Web has also become increasingly commercial over time. In 1993, 1.5% of Web servers were on .com domains.
* This number grew to over 60% in 1997. At the same time, search engines have migrated from the academic domain to the commercial.
* Up until now most search engine development has gone on at companies with little publication of technical details.
* This causes search engine technology to remain largely a black art and to be advertising oriented (see Appendix A in the full version).
* With Google, we have a strong goal to push more development and understanding into the academic realm.
* Another important design goal was to build systems that reasonable numbers of people can actually use.
* Usage was important to us because we think some of the most interesting research will involve leveraging the vast amount of usage data that is available from modern Web systems
* . For example, there are many tens of millions of searches performed every day. However, it is very difficult to get this data. mainly because it is considered commercially valuable.
* Our final design goal was to build an architecture that can support novel research activities on largescale Web data.
* To support novel research uses, Google stores all of the actual documents it crawls in compressed form.
* One of our main goals in designing Google was to set up an environment where other researchers can come in quickly, process large chunks of the Web, and produce interesting results that would have been very difficult to produce otherwise.
* In the short time the system has been up, there have already been several papers using database generated by Google, and many others are underway.
* Another goal we have is to set up a Spacelab-like environment where researchers or even students can propose and do interesting experiments on our largescale Web data.
* The Google search engine has two important features that help it produce high precision results.
* First, it makes use of the link structure of the Web to calculate a quality ranking for each Web page. This ranking is called PageRank and is described in detail in .
* Second, Google utilizes links to improve search results.
* The citation (link) graph of the Web is an important resource that has largely gone unused in existing Web search engines.
* We have created maps containing as many as 518 million of these hyperlinks, a significant sample of the total.
* These maps allow rapid calculation of a Web page’s “PageRank”, an objective measure of its citation importance that corresponds well with people’s subjective idea of importance.
* Because of this correspondence, PageRank is an excellent way to prioritize the results of Web keyword searches. For most popular subjects, a simple text matching search that is restricted to Web page titles performs admirably when PageRank prioritizes the results (demo available at google.stanford.edu).
* For the type of full text searches in the main Google system, PageRank also helps a great deal.

**Measuring User Influence in Twitter: The Million Follower Fallacy**

* Directed links in social media could represent anything from intimate friendships to common interests, or even a passion for breaking news or celebrity gossip.
* Such directed links determine the flow of information and hence indicate a user’s influence on others a concept that is crucial in sociology and viral marketing.
* In this paper, using a large amount of data collected from Twitter, we present an in-depth comparison of three measures of influence: indegree, retweets, and mentions.
* Based on these measures, we investigate the dynamics of user influence across topics and time.
* We make several interesting observations.
* First, popular users who have high indegree are not necessarily influential in terms of spawning retweets or mentions.
* Second, most influential users can hold significant influence over a variety of topics.
* Third, influence is not gained spontaneously or accidentally, but through concerted effort such as limiting tweets to a single topic.
* We believe that these findings provide new insights for viral marketing and suggest that topological measures such as indegree alone reveals very little about the influence of a user.
* Influence has long been studied in the fields of sociology, communication, marketing, and political science (Rogers 1962; Katz and Lazarsfeld 1955).
* The notion of influence plays a vital role in how businesses operate and how a society functions—for instance, see observations on how fashion spreads (Gladwell 2002) and how people vote (Berry and Keller 2003).
* Studying influence patterns can help us better understand why certain trends or innovations are adopted faster than others and how we could help advertisers and marketers design more effective campaigns.
* Studying influence patterns, however, has been difficult.
* This is because such a study does not lend itself to readily available quantification, and essential components like human choices and the ways our societies function cannot be reproduced within the confines of the lab.
* Nevertheless, there have been important theoretical studies on the diffusion of influence, albeit with radically different results.
* Traditional communication theory states that a minority of users, called influentials, excel in persuading others (Rogers 1962).
* This theory predicts that by targeting these influentials in the network, one may achieve a large-scale chain-reaction of influence driven by word-ofmouth, with a very small marketing cost (Katz and Lazarsfeld 1955). A more modern view, in contrast, de-emphasizes the role of influentials.
* Instead, it posits that the key factors determining influence are (i) the interpersonal relationship among ordinary users and (ii) the readiness of a society to adopt an innovation (Watts and Dodds 2007; Domingos and Richardson 2001).
* This modern view of influence leads to marketing strategies such as collaborative filtering.
* These theories, however, are still just theories, because there has been a lack of empirical data that could be used to validate either of them.
* The recent advent of social networking sites and the data within such sites now allow researchers to empirically validate these theories.
* Moving from theory into practice, we find that there are many other unanswered questions about how influence diffuses through a population and whether it varies across topics and time.
* People have different levels of expertise on various subjects.
* When it comes to marketing, however, this fact is generally ignored.
* Marketing services actively search for potential influencers to promote various items.
* These influencers range from “cool” teenagers, local opinion leaders, all the way to popular public figures.
* However, the advertised items are often far outside the domain of expertise of these hired individuals.
* So how effective are these marketing strategies? Can a person’s influence in one area be transferred to other areas? In this paper, we present an empirical analysis of influence patterns in a popular social medium.
* Using a large amount of data gathered from Twitter, we compare three different measures of influence: indegree, retweets, and mentions.
* Focusing on different topics, we examine how the three types of influential users performed in spreading popular news topics.
* We also investigate the dynamics of an individual’s influence by topic and over time.
* Finally, we characterize the precise behaviors that make ordinary individuals gain high influence over a short period of time.
* The Twitter dataset used in this paper consists of 2 billion follow links among 54 million users who produced a total of 1.7 billion tweets.
* We refer readers to our project webpage http://twitter.mpi-sws.org/ for a detailed description of the dataset and our data sharing plan.
* Our study provides several findings that have direct implications in the design of social media and viral marketing:
* Analysis of the three influence measures provides a better understanding of the different roles users play in social media. Indegree represents popularity of a user; retweets represent the content value of one’s tweets; and mentions represent the name value of a user. Hence, the top users based on the three measures have little overlap.
* Our finding on how influence varies across topics could serve as a useful test for answering how effective advertisement in Twitter would be if one is to employ influential users. Our analysis shows that most influential users hold significant influence over a variety of topics.
* Ordinary users can gain influence by focusing on a single topic and posting creative and insightful tweets that are perceived as valuable by others, as opposed to simply conversing with others. These findings provide new insights for viral marketing.
* The first finding in particular indicates that indegree alone reveals little about the influence of a user.
* This has been coined the million follower fallacy by Avnit (Avnit 2009), who pointed to anecdotal evidence that some users follow others simply for etiquette it’s polite to follow someone who’s following you and many do not read all the broadcast tweets.
* We have empirically demonstrated that having a million followers does not always mean much in the Twitter world.
* Instead, we claim that it is more influential to have an active audience who retweets or mentions the user.
* We start by reviewing studies of diffusion of influence and related work on influence propagation on Twitter.
* There are a number of conflicting ideas and theories about how trends and innovations get adopted and spread.
* The traditional view assumes that a minority of members in a society possess qualities that make them exceptionally persuasive in spreading ideas to others.
* These exceptional individuals drive trends on behalf of the majority of ordinary people.
* They are loosely described as being informed, respected, and well-connected; they are called the opinion leaders in the two-step flow theory (Katz and Lazarsfeld 1955), innovators in the diffusion of innovations theory (Rogers 1962), and hubs, connectors, or mavens in other work (Gladwell 2002).
* The theory of influentials is intuitive and compelling.
* By identifying and convincing a small number of influential individuals, a viral campaign can reach a wide audience at a small cost.
* The theory spread well beyond academia and has been adopted in many marketing businesses, e.g., RoperASW and Tremor (Gladwell 2002; Berry and Keller 2003).
* In contrast, a more modern view of information flow emphasizes the importance of prevailing culture more than the role of influentials.
* Some researchers have reasoned that people in the new information age make choices based on the opinions of their peers and friends, rather than by influentials (Domingos and Richardson 2001).
* These researchers argued that direct marketing through influentials would not be as profitable as using “network”-based advertising such as collaborative filtering.
* The traditional influentials theory has also been criticized because its information flow process does not take into account the role of ordinary users.
* In order to compare the role of influentials and ordinary users, researchers have developed a series of simulations, in which information flows freely between users, and a user adopts an innovation when he is influenced by more than a threshold of the sample population (Watts and Dodds 2007).
* Influentials were defined as those in the top 10% of influence distribution.
* The simulation showed that influentials initiated more frequent and larger cascades than average users, but they were neither necessary nor sufficient for all diffusions, as suggested in the traditional theory.
* Moreover, in homogeneous networks, influentials were no more successful in running long cascades than ordinary users.
* This means that a trend’s success depends not on the person who starts it, but on how susceptible the society is overall to the trend (Watts 2007).
* In fact, a trend can be initiated by any one, and if the environment is right, it will spread.
* Therefore, Watts dubbed early adopters or opinion leaders “accidental” influentials.
* The above competing ideas have remained as hypotheses for several reasons. First is the lack of data that could be used to empirically test them.
* Although there exist a handful of empirical studies on word-of-mouth influence (Leskovec, Adamic, and Huberman 2007; Cha, Mislove, and Gummadi 2009), no work has been conducted on the relative order of influence among individuals.
* A second issue is the variety of ways that influence has been defined (Watts 2007; Goyal, Bonchi, and Lakshmanan 2010).
* It has been unclear what exactly influence means. Finally, decades have passed since the influentials theory appeared.
* Even if the theory was reasonably accurate when it was proposed, things have changed and now we have much more variability in the flow of influence.
* In particular, online communities have become a significant way we receive new information and influence in such communities needs to be explored.
* In this paper, we investigate the notion of influence using a large amount of data collected from a popular social medium, Twitter.
* The Merriam-Webster dictionary defines influence as “the power or capacity of causing an effect in indirect or intangible ways.”
* Despite the large number of theories of influence in sociology, there is no tangible way to measure such a force nor is there a concrete definition of what influence means, for instance, in the spread of news.
* In this paper, we analyze the Twitter network as a news spreading medium and study the types and degrees of influence within the network.
* Focusing on an individual’s potential to lead others to engage in a certain act, we highlight three “interpersonal” activities on Twitter.
* First, users interact by following updates of people who post interesting tweets. Second, users can pass along interesting pieces of information to their followers.
* This act is popularly known as retweeting and can typically be identified by the use of RT @username or via @username in tweets.
* Finally, users can respond to (or comment on) other people’s tweets, which we call mentioning.
* Mentioning is identified by searching for @username in the tweet content, after excluding retweets.
* A tweet that starts with @username is not broadcast to all followers, but only to the replied user.
* A tweet containing @username in the middle of its text gets broadcast to all followers.
* These three activities represent the different types of influence of a person:
  + Indegree influence, the number of followers of a user, directly indicates the size of the audience for that user.
  + Retweet influence, which we measure through the number of retweets containing one’s name, indicates the ability of that user to generate content with pass-along value.
  + Mention influence, which we measure through the number of mentions containing one’s name, indicates the ability of that user to engage others in a conversation.
* Several recent efforts have been made to track influence on Twitter.
* The Web Ecology Project tracked 12 popular Twitter users for a 10-day period and grouped a user’s influence into two types: conversation-based and contentbased (Leavitt et al. 2009)
* . This work concluded that news media are better at spreading content, while celebrities are better at simply making conversation.
* Our work extends their notion of influence and uses extensive data to further examine the spatial and temporal dynamics of influence.
* More recently, a PageRank-like measure has been proposed to quantify influence on Twitter (Weng et al. 2010).
* The authors found high link reciprocity (72%) from a nonrandom sample of 6,748 Singapore-based users, and argued that high reciprocity is indicative of homophily.
* They then exploited this fact in computing a user’ influence rank.
* Our study, however, contradicts the observation about high reciprocity; near-complete data of Twitter shows low reciprocity (10%).
* Thus, we predict that social links on Twitter represent an influence relationship, rather than homophily.
* Accordingly, we ask what are the different activities on Twitter that represent influence of a user and to what extent a person’s influence varies across tweet topic and time.
* Describe how we collected the Twitter data and present the characteristics of the top users based on three influence measures: indegree, retweets, and mentions.
* We asked Twitter administrators to allow us to gather data from their site at scale.
* They graciously white-listed the IP address range containing 58 of our servers, which allowed us to gather large amounts of data.
* We used the Twitter API to gather information about a user’s social links and tweets.
* We launched our crawler in August 2009 for all user IDs ranging from 0 to 80 million.
* We did not look beyond 80 million, because no single user in the collected data had a link to a user whose ID was greater than that value. Out of 80 million possible IDs, we found 54,981,152 in-use accounts, which were connected to each other by 1,963,263,821 social links.
* We gathered information about a user’s follow links and all tweets ever posted by each user since the early days of the service. In total, there were 1,755,925,520 tweets.
* Nearly 8% of all user accounts were set private, so that only their friends could view their tweets. We ignore these users in our analysis.
* The social link information is based on the final snapshot of the network topology at the time of crawling and we do not know when the links were formed.
* The network of Twitter users comprises a single disproportionately large connected component (containing 94.8% of users), singletons (5%), and smaller components (0.2%).
* The largest component contains 99% of all links and tweets.
* Our goal is to explore influence of users, hence we focus on the largest component of the network, which is conceptually a single interaction domain for users.
* Because it is hard to determine influence of users who have few tweets, we borrowed the concept of “active users” from the traditional media research (Levy and Windhal 1985) and focused on those users with some minimum level of activity.
* We ignored users who had posted fewer than 10 tweets during their entire lifetime.
* We also ignored users for whom we did not have a valid screen name, because this information is crucial in identifying the number of times a user was mentioned or retweeted by others.
* After filtering, there were 6,189,636 users, whom we focus on in the remainder of this paper.
* To measure the influence of these 6 million users, however, we looked into how the entire set of 52 million users interacted with these active users.

**Measuring user influence on twitter using modified k-shell decomposition**

* Social influence can be described as power the ability of a person to influence the thoughts or actions of others. Identifying influential users on online social networks such as Twitter has been actively studied recently.
* In this paper, we investigate a modified k-shell decomposition algorithm for computing user influence on Twitter.
* The input to this algorithm is the connection graph between users as defined by the follower relationship.
* User influence is measured by the k-shell level, which is the output of the k-shell decomposition algorithm.
* Our first insight is to modify this k-shell decomposition to assign logarithmic k-shell values to users, producing a measure of users that is surprisingly well distributed in a bell curve.
* Our second insight is to identify and remove peering relationships from the network to further differentiate users. In this paper, we include findings from our study.
* Social influence can be described as power - the ability of a person to influence the thoughts or actions of others. Information and influence propagation in social networks has been actively studied for decades in the fields of psychology, sociology, communication, marketing, and political science.
* For online social networks, (Capece et al. 2009) summarized the social structures into three categories: Pyramid, Circular, and Hybrid.
* An example of the pyramid structure is Twitter. Influencers such as CNN have millions of followers, while the influencer doesn't follow back.
* Facebook is an example of a circular social structure, where Facebook users befriend only a select number of people or brands.
* The hybrid social structure combines the circular and pyramid-shaped community frameworks.
* Automatically detecting influencers on online social networks has recently received great attention from both research and industry.
* In industry, Klout.com tracks influence of users on online social networks including Twitter and Facebook.
* It measures users' influence using the Klout score, which is calculated based on 35 variables such as Follower Follow ratio, unique retweeters, unique messages retweeted, and username mention count.
* The scores range from 1-100 with higher scores representing a wider and stronger sphere of influence.
* The size of this sphere is calculated by measuring true reach (engaged followers and friends vs. spam bots, dead accounts, etc.).
* The strength of influence is calculated by tracking interactions across a user’s social graph to determine the likelihood of someone listening to or acting upon any specific message.
* TurnRank.com is another tool to measure user influence on Twitter.
* The TurnRank score is a reflection of both how much attention your followers can directly give you and how much attention they bring you from their network followers.
* In research, there has been a broad spectrum of algorithms proposed to measure influence on online social networks such as the number of retweets, the number of followers, the number of mentions, PageRank (Page et al. 1999), Hirsch-index or H-index (Hirsch 2005), and the Passive-Influence (PI) algorithm (Romero et al. 2010).
* PageRank is a link analysis algorithm which assigns a numerical weight called the PageRank value (Page et al. 1999).
* For online social networks, the higher a user’s value, the more influential he is. (Weng et al. 2010) extended PageRank to consider both the topical similarity between users and the link structure between the user accounts.
* The Hirsch index (or H-index) is used in the scientific community in order to measure the productivity and impact of a scientist. In a social network such as Twitter, a user will have H-index i if i of his messages have been retweeted or mentioned at least i times each.
* The higher the H-index is, the more influential the user is expected to be.
* The PI algorithm was recently proposed in (Romero et al. 2010) to address the observation that the majority of users on Twitter act as passive information consumers and do not forward the content to the network.
* The PI algorithm interactively estimates the influence and passivity of users based on their information forwarding activity.
* As we see above, most influence measures address the dynamic characteristics of the social network such as retweets and user mentions.
* While these types of measures may be good at identifying influence related to frequent use, they typically are not sensitive to influence related to infrequent use (unless a given measure includes data for extended periods).
* In this paper we focus on measuring a user’s full potential influence inherent in the user connectivity network, which is relatively more static.
* In particular, we propose a variant k-shell decomposition algorithm to estimate the influence of users on Twitter.
* The k-shell algorithm, described in (Kitsak et al. 2010), has been shown to be effective at identifying influential spreaders in complex networks.
* As is the case with infectious diseases within a society, the most efficient spreaders of information are found to be at the core of the network as identified by k-shell decomposition.
* To use k-shell decomposition effectively for the Twitter network, we make two significant changes.
* The first change alters the basic algorithm for determining k-shells to produce logarithmic values.
* The second change is to the manner in which the network structure is interpreted (once with peers included, once with peers excluded).
* User influence is measured by the logarithmic k-shell value, which is the output of the
* Our experiments show that these k-shell values for users are surprisingly well distributed in a bell curve.
* The remainder of this paper elaborates on the algorithm and our analysis of it using a large amount of Twitter data.
* In the Data section we describe the Twitter data set we use. The Algorithms section introduces the original k-shell decomposition algorithm and our modifications for the
* In the Experiment section, we share our analysis and findings. We conclude the paper in the last Section.
* In this paper, we use two twitter datasets. One was collected by KAIST in 2009.
* Details and analysis of this dataset was reported in (Kwak et al. 2010).
* It includes 41.7 million user profiles (User Data), and 1.47 billion social relations (Network Data). T
* The other dataset is Usage Data, which was collected by Lehigh University and includes a sampling of more than 80 million actual tweets from October 2009, representing more than 7 million users, or about 17% of the total Twitter user community.
* Network data is provided as pairs of values denoting the arcs/edges of the follower relationship for users in the Twitter network.
* For each pair, the first value is the User ID of a given Twitter user and the second value is the User ID of a known follower. For instance, “998700 342100” and “998700 531240” indicate that users with IDs 342100 and 531240 are followers of the user with ID 998700.
* User data contains information on individual users such as Account ID, Account Name, Owner Name, Description, and Create Date.
* Usage data comes in hourly dumps containing information on individual tweets such as Tweet ID, Tweet Timestamp, Tweet Text, Sender ID, and other status information relevant to the tweet and its sender.
* The k-shell decomposition algorithm is a well-established method for detecting the core and the hierarchical structure of a given network (Seidman 1983) (Carmi et al. 2007).
* Recently, (Kitsak et al. 2010) proposed using k-shell decomposition as a technique for identifying the most efficient, or influential, spreaders in a complex network.
* This applies to the spread of infectious disease in a society as well as the spread of information in a social network.
* Viewed as nodes in a graph, the higher the k-shell level assigned, the closer the node is to the core of the graph.
* The assumption is that, if these nodes are users in a social network, the users in the higher k-shell levels are more influential in the network than users in lower k-shell levels.
* The k-shell decomposition algorithm groups all nodes in a network that have k (or less) connections or that are only connected to other nodes with k (or less) connections.
* Once a node has been identified, it is marked (and removed from the network for purposes of the algorithm) and the search continues until all nodes in shell k have been found. The process then moves to the next larger k-shell value (and continues until all nodes have been marked).
* In this basic algorithm, k-shell values are assigned in a linear fashion. That is, each k-shell value is equivalent to the analyzed connection count.
* An example of this algorithmic result for a simple network, from (Kitsak et al. 2010).
* Note that, in this example, no nodes have fallen into k-shell level 0.
* This is because the connections in this example are undirected.
* Any node with at least one connection will minimally fall into k-shell level 1. In the Twitter data, connections are directional.
* It is possible to have a user who follows any number of other Twitter users but has no followers of his own; such a user would be placed in kshell level 0.
* Any user that only has followers from k-shell level 0 would also be placed in k-shell 0.
* We initially used the basic algorithm to analyze the Twitter network (using the Network Data described in the Data section).
* We observed that the results were highly skewed, with most nodes (or users) falling into the first few (low) k-shell levels, with user counts peaking at k-shell level 4, and the remaining users tailing off over thousands of additional non-empty (higher) k-shell levels.
* This distribution of nodes made statistical observations hard. Motivated by this difficulty, we modified the original algorithm by applying a logarithmic mapping, where each k-shell level represents roughly the log value of the analyzed connection count. So, whereas the original k-shell decomposition algorithm placed nodes with k (or less) connections into k-shell level k, our modified algorithm places nodes with 2^k - 1 (or less) connections into k-shell level k, effectively consolidating the higher k-shell levels.
* This modified algorithm produces fewer and more meaningful k-shell values.
* As the Experiment section will show, the placement of users in the logarithmic k-shell levels produces a more useful distribution.
* We also found the modified algorithm to be faster than the original algorithm.
* For a given network, when the original algorithm iterated n times to find a solution, the modified algorithm only needed to iterate log2n times.
* In this experiment, the modified k-shell decomposition algorithm is used to measure user influence in the Twitter network.
* These measures are validated against Twitter usage data. But, to discuss these measures, it is also necessary to clarify some of the terminology we will use .
* The Twitter Network Data only defines connections in terms of users and followers.
* We define here other relations that are implicit in these connections.
* We say that a user A with a follower B will also be called a leader of user B. For a given user A, its set of leaders and its set of followers can overlap.
* We call the users in this overlap the peers of A (i.e., all users B for which A follows B and B follows A).
* A follower relationship with peers allowed is called a peered follower relationship.
* A follower relationship with the peers removed is called a non-peered follower relationship. The same is true for the leader relationship.
* Once the peers are identified, a distinction can be made between reach and authority.
* Reach measures the potential audience for a user’s message, either directly through tweets or indirectly through retweets (via the follower relationship).
* Authority is a similar, but non-peered, measure in that it excludes peers when interpreting the follower relationship.
* This is because peering implies a bidirectional (leader/follower) relationship between users whereas authority is inherently uni-directional.
* This experiment produces and compares user influence measures based on both reach and authority (and a combination of the two).

**Authoritative sources in a hyperlinked environment**

* The network structure of a hyperlinked environment can be a rich source of information about the content of the environment, provided we have effective means for understanding it.
* We develop a set of algorithmic tools for extracting information from the link structures of such environments, and report on experiments that demonstrate their effectiveness in a variety of contexts on the World Wide Web.
* The central issue we address within our framework is the distillation of broad search topics, through the discovery of “authoritative” information sources on such topics.
* We propose and test an algorithmic formulation of the notion of authority, based on the relationship between a set of relevant authoritative pages and the set of “hub pages” that join them together in the link structure.
* Our formulation has connections to the eigenvectors of certain matrices associated with the link graph; these connections in turn motivate additional heuristics for link-based analysis.
* The network structure of a hyperlinked environment can be a rich source of information about the content of the environment, provided we have effective means for understanding it. In this work, we develop a set of algorithmic tools for extracting information from the link structures of such environments, and report on experiments that demonstrate their effectiveness in a variety of contexts on the World Wide Web (www).
* In particular, we focus on the use of links for analyzing the collection of pages relevant to a broad search topic, and for discovering the most “authoritative” pages on such topics.
* While our techniques are not specific to the www, we find the problems of search and structural analysis particularly compelling in the context of this domain.
* The www is a hypertext corpus of enormous complexity, and it continues to expand at a phenomenal rate.
* Moreover, it can be viewed as an intricate form of populist hypermedia, in which millions of on-line participants, with diverse and often conflicting goals, are continuously creating hyperlinked content.
* Thus, while individuals can impose order at an extremely local level, its global organization is utterly unplanned high-level structure can emerge only through a posteriori analysis.
* Our work originates in the problem of searching on the www, which we could define roughly as the process of discovering pages that are relevant to a given query.
* The quality of a search method necessarily requires human evaluation, due to the subjectivity inherent in notions such as relevance.
* We begin from the observation that improving the quality of search methods on the www is, at the present time, a rich and interesting problem that is in many ways orthogonal to concerns of algorithmic efficiency and storage.
* In particular, consider that current search engines typically index a sizable portion of the www and respond on the order of seconds.
* Although there would be considerable utility in a search tool with a longer response time, provided that the results were of significantly greater value to a user, it has typically been very hard to say what such a search tool should be computing with this extra time. Clearly we are lacking objective functions that are both concretely defined and correspond to human notions of quality. Queries and Authoritative Sources
* . We view searching as beginning from a usersupplied query. It seems best not to take too unified a view of the notion of a query; there is more than one type of query, and the handling of each may require different techniques. Consider, for example, the following types of queries.
* Concentrating on just the first two types of queries for now, we see that they present very different sorts of obstacles.
* The difficulty in handling specific queries is centered, roughly, around what could be called the Scarcity Problem: there are very few pages that contain the 1 required information, and it is often difficult to determine the identity of these pages.
* For broad-topic queries, on the other hand, one expects to find many thousand relevant pages on the www; such a set of pages might be generated by variants of term-matching, or by more sophisticated means.
* Thus, there is not an issue of scarcity here. Instead, the fundamental difficulty lies in what could be called the Abundance Problem: The number of pages that could reasonably be returned as relevant is far too large for a human user to digest.
* To provide effective search methods under these conditions, one needs a way to filter, from among a huge collection of relevant pages, a small set of the most “authoritative” or “definitive” ones.
* This notion of authority, relative to a broad-topic query, serves as a central focus in our work.
* One of the fundamental obstacles we face in addressing this issue is that of accurately modeling authority in the context of a particular query topic.

**An expert user ranking algorithm in online communities**

* As computer-mediated communication services in the Web 2.0 arena, online communities have become very important places for Web users to share knowledge and experiences.
* One important research issue in online communities is how to find expert users in the community.
* In this paper, we investigate the expertise that users displayed in online communities, especially in discussion groups and propose an effective expert ranking algorithm, which integrates both discussion thread contents and social network extracted from massive social interactions.
* We present a vector space model to compute the content relevance part and a PageRank style algorithm for the expert network part.
* Considering the expert spamming issue caused by mutually referencing in a small group, we modify the original PageRank algorithm and propose a novel ranking algorithm.
* The two parts are lastly integrated using a cascade strategy.
* The experimental results show that our so-called ExpertRank algorithm is an effective expert ranking algorithm, which can guarantee that the highly ranked experts are both highly relevant to the specific queries and highly authoritative in corresponding areas.

**3. FUNDAMENTAL CONCEPTS ON (DOMAIN)**

Implementation is the stage where the theoretical design is turned in to working system. The most crucial stage is achieving a new successful system and in giving confidence on the new system for the users that it will work efficiently and effectively.

The system can be implemented only after through testing is done and if it found to work according to the specification. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the change over and an evaluation of change over methods a part from planning. Two major tasks of preparing the implementation are education and training of the users and testing of the system.

The more complex the system being implemented, the more involved will be the systems analysis and design effort required just for implementation. The implementation phase comprises of several activities. The required hardware and software acquisition is carried out. The System may require some hardware and software acquisition is carried out. The system may require some software to be developed. For this, programs are written and tested. The user then changes over to his new fully tested system and the old system is discontinued.

Implementation is the process of having systems personnel check out and put new equipment in to use, train users, install the new application, and construct any files of data needed to it.

Depending on the size of the organization that will be involved in using the application and the risk associated with its use, system developers may choose to test the operation in only one area of the firm, say in one department or with only one or two persons. Sometimes they will run the old and new systems together to compare the results. In still other situations, developers will stop using the old system one-day and begin using the new one the next. As we will see, each implementation strategy has its merits, depending on the business situation in which it is considered. Regardless of the implementation strategy used, developers strive to ensure that the system’s initial use in trouble-free.

Once installed, applications are often used for many years. However, both the organization and the users will change, and the environment will be different over the weeks and months. Therefore, the application will undoubtedly have to be maintained. Modifications and changes will be made to the software, files, or procedures to meet the emerging requirements.

**4. System Analysis**

The **Systems Development Life Cycle (SDLC)**, or *Software Development Life Cycle* in [systems engineering](http://en.wikipedia.org/wiki/Systems_engineering), [information systems](http://en.wikipedia.org/wiki/Information_systems) and [software engineering](http://en.wikipedia.org/wiki/Software_engineering), is the process of creating or altering systems, and the models and [methodologies](http://en.wikipedia.org/wiki/Methodologies) that people use to develop these systems.

In software engineering the SDLC concept underpins many kinds of [software development methodologies](http://en.wikipedia.org/wiki/Software_development_methodologies). These methodologies form the framework for planning and controlling the creation of an information system the [software development process](http://en.wikipedia.org/wiki/Software_development_process).

**SOFTWARE MODEL OR ARCHITECTURE ANALYSIS:**

Structured project management techniques (such as an SDLC) enhance management’s control over projects by dividing complex tasks into manageable sections. A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. But none of the SDLC models discuss the key issues like Change management, Incident management and Release management processes within the SDLC process, but, it is addressed in the overall project management. In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer. In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer. The ―one size fits all‖ approach to applying SDLC methodologies is no longer appropriate. We have made an attempt to address the above mentioned defects by using a new hypothetical model for SDLC described elsewhere. The drawback of addressing these management processes under the overall project management is missing of key technical issues pertaining to software development process that is, these issues are talked in the project management at the surface level but not at the ground level.

**WHAT IS SDLC?**

A software cycle deals with various parts and phases from planning to testing and deploying software. All these activities are carried out in different ways, as per the needs. Each way is known as a Software Development Lifecycle Model (SDLC). A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. A descriptive model describes the history of how a particular software system was developed. Descriptive models may be used as the basis for understanding and improving software development processes or for building empirically grounded prescriptive models.

**SDLC models** \* **The Linear model (Waterfall)** - Separate and distinct phases of specification and development. - All activities in linear fashion. - Next phase starts only when first one is complete. \* **Evolutionary development** - Specification and development are interleaved (Spiral, incremental, prototype based, Rapid Application development). - Incremental Model (Waterfall in iteration), **-** RAD(Rapid Application Development) **-** Focus is on developing quality product in less time, - **Spiral Model** - We start from smaller module and keeps on building it like a spiral. It is also called Component based development. \* **Formal systems development** - A mathematical system model is formally transformed to an implementation. \* **Agile Methods.** - Inducing flexibility into development. \* **Reuse-based development** - The system is assembled from existing components.

**The General Model**

Software life cycle models describe phases of the software cycle and the order in which those phases are executed. There are tons of models, and many companies adopt their own, but all have very similar patterns. Each phase produces deliverables required by the next phase in the life cycle. Requirements are translated into design. Code is produced during implementation that is driven by the design. Testing verifies the deliverable of the implementation phase against requirements.

**SDLC Methodology:**

**Spiral Model**

The spiral model is similar to the incremental model, with more emphases placed on risk analysis.  The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation.  A\ software project repeatedly passes through these phases in iterations (called Spirals in this model).  The baseline spiral, starting in the planning phase, requirements is gathered and risk is assessed.  Each subsequent spirals builds on the baseline spiral. Requirements are gathered during the planning phase.  In the risk analysis phase, a process is undertaken to identify risk and alternate solutions.  A prototype is produced at the end of the  
risk analysis phase. Software is produced in the engineering phase, along with testing at  
the end of the phase.  The evaluation phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral. In the spiral model, the angular component represents progress, and the radius of the spiral represents cost. Spiral Life Cycle Model.

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article, “A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

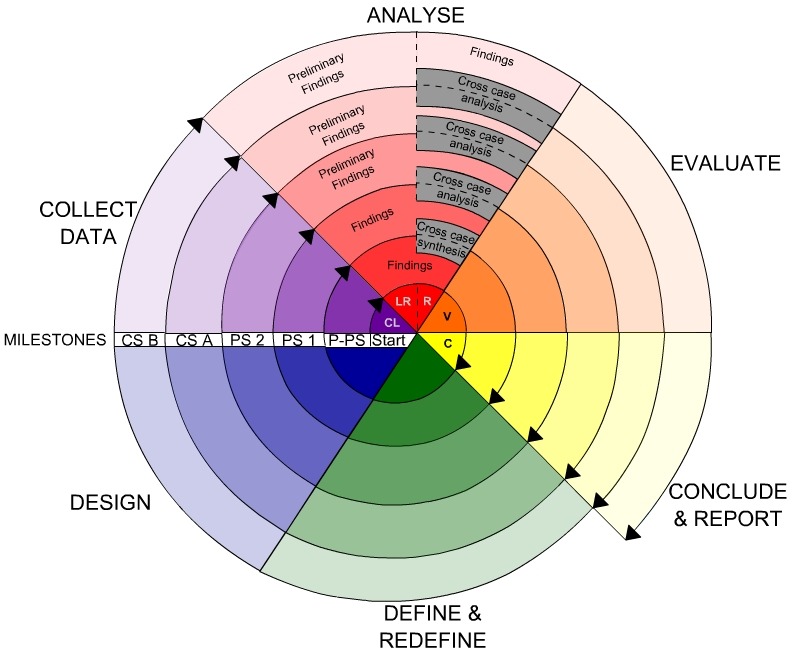
As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

The steps for Spiral Model can be generalized as follows:

* The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
* A preliminary design is created for the new system.
* A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
* A second prototype is evolved by a fourfold procedure:

1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
2. Defining the requirements of the second prototype.
3. Planning an designing the second prototype.
4. Constructing and testing the second prototype.

* At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involved development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer’s judgment, result in a less-than-satisfactory final product.
* The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
* The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
* The final system is constructed, based on the refined prototype.
* The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.



**Fig -Spiral Model**

**Advantages**

* High amount of risk analysis
* Good for large and mission-critical projects.
* Software is produced early in the software life cycle.

**4.1 Existing System**

In today’s Twitter and Weibo (one of the most popular social networking sites in China), a user can get the instant updates about his connected friends’ postings and could further retweet or comment the postings. Within a time period, millions of users may take different actions such as posting and retweeting at these social networking sites. One interesting and important problem is how to rank users based on their vitality with historical data. An accurate vitality ranking of users will provide great insight for many applications in most online social networking sites. For instance, a Twitter user ranking algorithm was proposed to identify authoritative users who often submit useful information. The proposed algorithm mainly works based on the user-tweet graph, rather than the user-user social graph. An extension of Page Rank algorithm named Twitter Rank was developed to rank Twitter users based on their influence. They first build topic-specific relationship network among users, then apply the Twitter Rank algorithm for ranking. In a modified K-shell decomposition algorithm is developed to measure the user influence in Twitter. Furthermore, some explicit measurements such as retweets and mentions are developed to measure and rank user influence in Twitter. However, most of these measurements quantify the influence in an isolated way, rather than in a collective way.

**4.1.1 Drawbacks**

Within a time period, millions of users may take different actions such as posting and retweeting at these social networking sites. One interesting and important problem is how to rank users based on their vitality with historical data

**4.2 Problem statement**

For enterprise systems running on public clouds in which the servers are outside the control domain of the enterprise, access control that was traditionally executed by reference monitors deployed on the system servers can no longer be trusted. Hence, a self-contained security scheme is regarded as an effective way for protecting outsourced data. However, building such a scheme that can implement the access control policy of the enterprise has become an important challenge

**4.3 Proposed System**

we propose two types of node vitality ranking algorithms that analyze the vitality of all nodes in a collective way. First, for a node A that has many interactions with his friends in a time period, if most of his friends do not have many interactions with their friends, it is very likely that the node A has high vitality. Based on this intuition, we define two measurements to quantify the vitality level of each node and propose the first algorithm. Second, by exploiting the mutual dependency of vitality among all users within a social network, we propose the second algorithm that infers the vitality level of users in an iterative way. Through the iteration, all nodes measurements propagate through the network and affect each other. Thus the second algorithm is able to collectively analyze the vitality score of all nodes by considering the whole network. Furthermore, upon our in-depth understanding about user vitality, we propose an improved model to predict the vitality of users. The successful prediction results will further benefit many applications on social networking sites. Finally, we conduct intensive experiments on both user vitality ranking and prediction with two large-scale real world data sets. The experimental results demonstrate the effectiveness and efficiency of our methods.

**4.3.1 Advantages**

**Advantages:**

Thus the second algorithm is able to collectively analyze the vitality score of all nodes by considering the whole network. Furthermore, upon our in-depth understanding about user vitality, we propose an improved model to predict the vitality of users. The successful prediction results will further benefit many applications on social networking sites. Finally, we conduct intensive experiments on both user vitality ranking and prediction with two large-scale real world data sets. The experimental results demonstrate the effectiveness and efficiency of our methods.

**4.4 Modules Description**

**Modules**

**Initial Ranking Algorithm**

**Iterative Ranking Algorithm**

**Predicting user Vitality**

**Performance on vitality Ranking**

**Initial Ranking Algorithm:**

In this Algorithm examples of interactions among users over different time periods in a social network. The number on the top of link denotes the number of interaction. The accumulated number of interactions per user can be defined as follows.

**Iterative Ranking Algorithm:**

Introduce the iterative ranking algorithm, which takes an iterative process to measure the vitality of users within a social network. we actually allocate the interactions between two users equally as shown in Equation 1, which essentially assumes each of both users makes the same contribution to the interactions. However, this assumption may not be perfect in practice. For instance, one of them may be very active to interact, while the other one may be relatively passive.

**Predicting user Vitality**

In this section, we introduce and address the problem of predicting the user vitality based on the model and inference of user vitality in a social network.

**Performance on vitality Ranking**

In this section, we present the results of ranking algorithms on both micro blog and academic network data. We will get different ranking lists of users with different ranking algorithms in multiple time periods and discuss the effectiveness of ranking results based on the role of users in real network systems.

**Feasibility Study**

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

* Technical Feasibility
* Operational Feasibility
* Economical Feasibility

**4.5.1Economic Feasibility**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

**4.5.2 Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following: -

* Is there sufficient support for the management from the users?
* Will the system be used and work properly if it is being developed and implemented?
* Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

**4.5.3 Technical Feasibility**

The technical issue usually raised during the feasibility stage of the investigation includes the following:

* Does the necessary technology exist to do what is suggested?
* Do the proposed equipments have the technical capacity to hold the data required to use the new system?
* Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
* Can the system be upgraded if developed?
* Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hard requirements for the development of this project are not many and are already available in-house at NIC or are available as free as open source. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

**5. System Requirements Specification**

**5.1 Introduction**

A **Software Requirements Specification** (**SRS**) – a [requirements specification](http://en.wikipedia.org/wiki/Requirements_specification) for a [software system](http://en.wikipedia.org/wiki/Software_system) – is a complete description of the behavior of a system to be developed. It includes a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. [Non-functional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements) are requirements which impose constraints on the design or implementation (such as [performance engineering](http://en.wikipedia.org/wiki/Performance_engineering) requirements, [quality](http://en.wikipedia.org/wiki/Quality_%28business%29) standards, or design constraints).

**System requirements specification:** A structured collection of information that embodies the requirements of a system. A [business analyst](http://en.wikipedia.org/wiki/Business_analyst), sometimes titled [system analyst](http://en.wikipedia.org/wiki/System_analyst), is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the [systems development life cycle](http://en.wikipedia.org/wiki/Systems_development_life_cycle) domain, typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* [**Business requirements**](http://en.wikipedia.org/wiki/Business_requirements) describe in business terms *what* must be delivered or accomplished to provide value.
* **Product requirements** describe properties of a system or product (which could be one of

Several ways to accomplish a set of business requirements.)

* **Process requirements** describe activities performed by the developing organization. For instance, process requirements could specify specific methodologies that must be followed, and constraints that the organization must obey.

Product and process requirements are closely linked. Process requirements often specify the activities that will be performed to satisfy a product requirement. For example, a maximum development cost requirement (a process requirement) may be imposed to help achieve a maximum sales price requirement (a product requirement); a requirement that the product be maintainable (a Product requirement) often is addressed by imposing requirements to follow particular development styles

**5.2 Purpose**

An systems engineering, a **requirement** can be a description of *what* a system must do, referred to as a [Functional Requirement](http://en.wikipedia.org/wiki/Functional_requirements). This type of requirement specifies something that the delivered system must be able to do. Another type of requirement specifies something about the system itself, and how well it performs its functions. Such requirements are often called [Non-functional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements), or 'performance requirements' or 'quality of service requirements.' Examples of such requirements include usability, availability, reliability, supportability, testability and maintainability.

A collection of requirements define the characteristics or features of the desired system. A 'good' list of requirements as far as possible avoids saying *how* the system should implement the requirements, leaving such decisions to the system designer. Specifying how the system should be implemented is called "implementation bias" or "solution engineering". However, *implementation constraints* on the solution may validly be expressed by the future owner, for example for required interfaces to external systems; for interoperability with other systems; and for commonality (e.g. of user interfaces) with other owned products.

In software engineering, the same meanings of requirements apply, except that the focus of interest is the software itself.

**5.3 Functional Requirements**

**1)admin login then check the accumulated number of interaction and interact with the members**

**2)admin can be relative increase of interaction of the user in particular time period**

**3)the admin also check the avg interaction for user**

**4)admin check the user vitality score**

**5)user can be post the new articles**

**6)user also send the messages check the messages**

**7)user also send the friend request and see all friend**

**8)user also search friends in particular user**

**5.4 Non Functional Requirements**

The major non-functional Requirements of the system are as follows

**Usability**

The system is designed with completely automated process hence there is no or less user intervention.

**Reliability**

The system is more reliable because of the qualities that are inherited from the chosen platform java. The code built by using java is more reliable.

**Performance**

This system is developing in the high level languages and using the advanced front-end and back-end technologies it will give response to the end user on client system with in very less time.

**Supportability**

The system is designed to be the cross platform supportable. The system is supported on a wide range of hardware and any software platform, which is having JVM, built into the system.

**Implementation**

The system is implemented in web environment using struts framework. The apache tomcat is used as the web server and windows xp professional is used as the platform.

Interface the user interface is based on Struts provides HTML Tag

**5.5 Input & 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.

**5.6 Hardware Requirements**

Processor : Pentium IV

Hard Disk : 40GB or more

RAM : 512MB or more

**5.6 Software Requirements**

Operating System : Windows XP/2003 or Linux (Any OS)

User Interface : HTML, CSS

Client-side Scripting : JavaScript

Programming Language : Java

Web Applications : JDBC, Servlets, JSP

IDE/Workbench : My Eclipse 8.6

Database : Oracle 10g

Server Deployment : Tomcat 6.0

**6. System Design**

The purpose of the design phase is to plan a solution of the problem specified by the requirement document. This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed, design takes us toward how to satisfy the needs. The design of a system is perhaps the most critical factor affection the quality of the software; it has a major impact on the later phase, particularly testing, maintenance. The output of this phase is the design document. This document is similar to a blueprint for the solution and is used later during implementation, testing and maintenance. The design activity is often divided into two separate phases System Design and Detailed Design.

System Design also called top-level design aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. At the end of the system design all the major data structures, file formats, output formats, and the major modules in the system and their specifications are decided.

During, Detailed Design, the internal logic of each of the modules specified in system design is decided. During this phase, the details of the data of a module is usually specified in a high-level design description language, which is independent of the target language in which the software will eventually be implemented.

In system design the focus is on identifying the modules, where as during detailed design the focus is on designing the logic for each of the modules. In other works, in system design the attention is on what components are needed, while in detailed design how the components can be implemented in software is the issue.

Design is concerned with identifying software components specifying relationships among components. Specifying software structure and providing blue print for the document phase. Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts. In such a manner , the interaction between parts is minimal clearly specified.

During the system design activities , Developers bridge the gap between the requirements specification , produced during requirements elicitation and analysis , and the system that is delivered to the user.

Design is the place where the quality is fostered in development . Software design is a process through which requirements are translated into a representation of software.

**6.2 System Specifications**

**Introduction to UML**

The unified Modeling Language (UML) is a standard language for writing software blueprints. The UML may be used to visualize, specify , construct and document the artifacts of software-intensive system.

The goal of UML is to provide a standard notation that can be used by all object - oriented methods and to select and integrate the best elements .UML is itself does not prescribe or advice on how to use that notation in a software development process or as part of an object - design methodology. The UML is more than just bunch of graphical symbols. Rather , behind each symbol in the UML notation is well-defined semantics.

The system development focuses on three different models of the system.

* Functional model
* Object model
* Dynamic model

**Functional model**  in UML is represented with use case diagrams , describing the functionality of the system from user point of view.

**Object model** in UML is represented with class diagrams , describing the structure of the system in terms of objects , attributes , associations and operations.

**Dynamic model**  in UML is represented with sequence diagrams , start chart diagrams and activity diagrams describing the internal behaviour of the system.

**6.3 Scenarios**

A Use Case is an abstraction that all describes all possible scenarios involving the described functionality . A scenario is an instance of a use case describing a concrete set of actions.

* The **name** of the scenario enables us to refer it ambiguously. The name of scenario is underlined to indicate it is an instance.
* The **Participating actor instance** field indicates which actor instance are involved in this scenario. Actor instance also have underlined names.
* The **Flow of Events** of scenario describe the sequence of events step by step.

**6.3.1 Use Case Model**

Use case diagrams represent the functionality of the system from a user point of view. A Use case describes a function provided by the system that yields a visible result for an actor. an actor describe any entity that interacts with the system. The identification of actors and use cases results in the definition of the boundary of the system, which is , in differentiating the tasks accomplished by the system and the tasks accomplished by its environment. The actors outside the boundary of the system, where as the use cases are inside the boundary of the system

A Use case contains all the events that can occur between an actor and a set of scenarios that explains the interactions as sequence of happenings.

**Actors**

Actors represent external entities that interact with the system. An actor can be human or external system.

Actor are not part of the system. They represent anyone or anything that interact with the system.

An Actor may

* Only input information to the system.
* Only receive information from the system.
* Input and receive information from to and from the system.

During this activity , developers indentify the actors involved in this system are:

**User:**

User is an actor who uses the system and who performs the operations like data classifications and execution performance that are required for him.

**Use Cases:**

Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use case focus on the behaviour of the system from an external point of view. The identification of actors and use cases results in the definition of the boundary of the system , which is , in differentiating the tasks accomplished by the system and the tasks accomplished by its environment. The actors are outside the boundary of the system , where as the use cases are inside the boundary of the system.

**6.3 DFD’s**

**Data Flow Diagrams:**

A graphical tool used to describe and analyze the moment of data through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input to output, through processes, may be described logically and independently of the physical components associated with the system. The DFD is also know as a data flow graph or a bubble chart.

DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system’s structure charts. The Basic Notation used to create a DFD’s are as follows:

**1. Dataflow:** Data move in a specific direction from an origin to a destination.

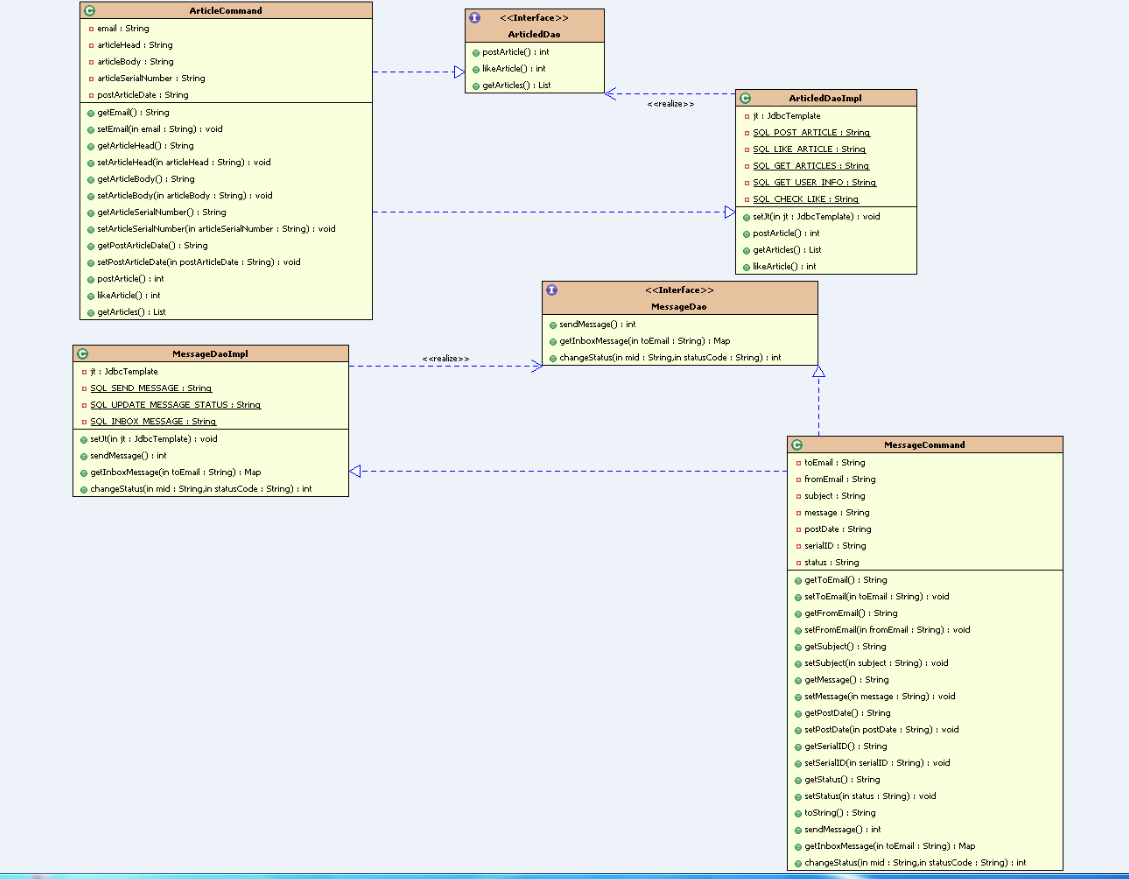
**2. Process:** People, procedures, or devices that use or produce (Transform) Data. The physical component is not identified.

**3. Source:** External sources or destination of data, which may be People, programs, organizations or other entities.

**4. Data Store:** Here data are stored or referenced by a process in the System.

**6.4 UML Diagrams**

Class diagram



UML DIAGRAM:

Admin usecase diagram



User use case diagram;



Admin sequence diagram



User sequence diagram



User collaboration



Admin collaboration



Admin activity



User activity



State chart diagram for admin



User state chart diagram



Dployement diagram;





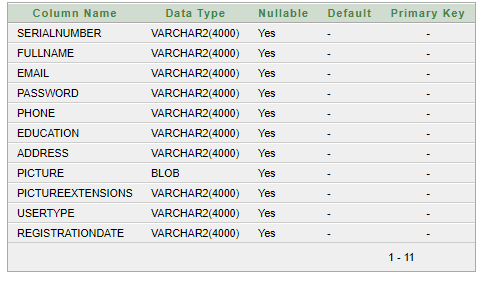
component



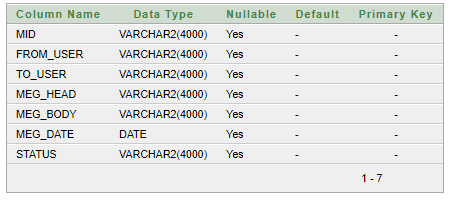


**Data dictionary:**

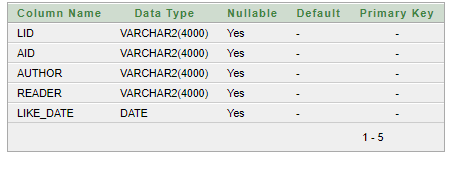
**USER\_ACCOUNT**



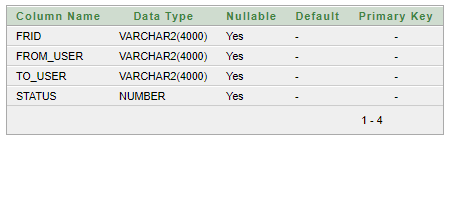
**MESSAGE**



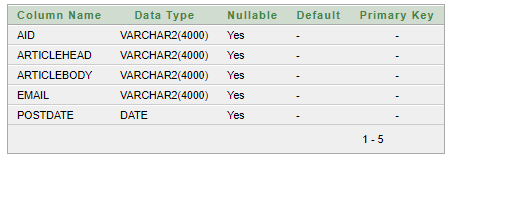
**LIKES**



**FRIEND\_REQUEST**



**ARTICLE**



**7. Implementation**

Implementation is the stage where the theoretical design is turned in to working system. The most crucial stage is achieving a new successful system and in giving confidence on the new system for the users that it will work efficiently and effectively.

The system can be implemented only after through testing is done and if it found to work according to the specification. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the change over and an evaluation of change over methods a part from planning. Two major tasks of preparing the implementation are education and training of the users and testing of the system.

The more complex the system being implemented, the more involved will be the systems analysis and design effort required just for implementation. The implementation phase comprises of several activities. The required hardware and software acquisition is carried out. The System may require some hardware and software acquisition is carried out. The system may require some software to be developed. For this, programs are written and tested. The user then changes over to his new fully tested system and the old system is discontinued.

Implementation is the process of having systems personnel check out and put new equipment in to use, train users, install the new application, and construct any files of data needed to it.

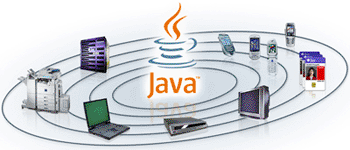
Depending on the size of the organization that will be involved in using the application and the risk associated with its use, system developers may choose to test the operation in only one area of the firm, say in one department or with only one or two persons. Sometimes they will run the old and new systems together to compare the results. In still other situations, developers will stop using the old system one-day and begin using the new one the next. As we will see, each implementation strategy has its merits, depending on the business situation in which it is considered. Regardless of the implementation strategy used, developers strive to ensure that the system’s initial use in trouble-free.

Once installed, applications are often used for many years. However, both the organization and the users will change, and the environment will be different over the weeks and months. Therefore, the application will undoubtedly have to be maintained. Modifications and changes will be made to the software, files, or procedures to meet the emerging requirements.

**7.1 Technology Description**

**About the Java Technology**

The Java platform consists of the Java application programming interfaces (APIs) and the Java virtual machine (JVM).



The following Java technology lets developers, designers, and business partners develop and deliver a consistent user experience, with one environment for applications on mobile and embedded devices. Java meshes the power of a rich stack with the ability to deliver customized experiences across such devices.

Java APIs are libraries of compiled code that you can use in your programs. They let you add ready-made and customizable functionality to save you programming time.  
Java programs are run (or interpreted) by another program called the Java Virtual Machine. Rather than running directly on the native operating system, the program is interpreted by the Java VM for the native operating system. This means that any computer system with the Java VM installed can run Java programs regardless of the computer system on which the applications were originally developed.

In the Java programming language, all source code is first written in plain text files ending with the .java extension. Those source files are then compiled into .class files by the javac compiler. A .class file does not contain code that is native to your processor; it instead contains bytecodes — the machine language of the Java Virtual Machine (Java VM). The java launcher tool then runs your application with an instance of the Java Virtual Machine.

Because the Java VM is available on many different operating systems, the same .class files are capable of running on Microsoft Windows, the Solaris TM Operating System (Solaris OS), Linux, or Mac OS.

Java technology is both a programming language and a platform.

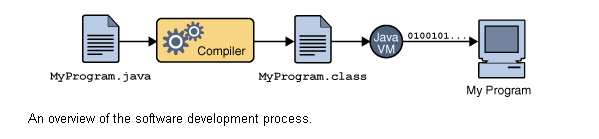
**The Java Programming Language**

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

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

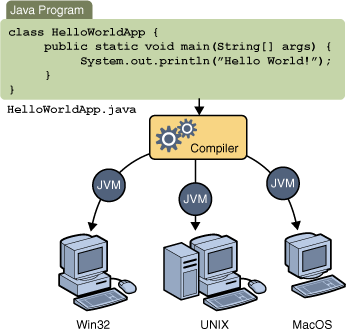
Each of the preceding buzzwords is explained in [*The Java Language Environment*](http://java.sun.com/docs/white/langenv/) , a white paper written by James Gosling and Henry McGilton.

In the Java programming language, all source code is first written in plain text files ending with the .java extension. Those source files are then compiled into .class files by the javac compiler. A .class file does not contain code that is native to your processor; it instead contains *bytecodes* — the machine language of the Java Virtual Machine[1](http://download.oracle.com/javase/tutorial/getStarted/intro/definition.html#FOOT) (Java VM). The java launcher tool then runs your application with an instance of the Java Virtual Machine.



An overview of the software development process.

Because the Java VM is available on many different operating systems, the same .class files are capable of running on Microsoft Windows, the Solaris™ Operating System (Solaris OS), Linux, or Mac OS. Some virtual machines, such as the [Java HotSpot virtual machine](http://java.sun.com/products/hotspot/), perform additional steps at runtime to give your application a performance boost. This include various tasks such as finding performance bottlenecks and recompiling (to native code) frequently used sections of code



Through the Java VM, the same application is capable of running on multiple platforms.

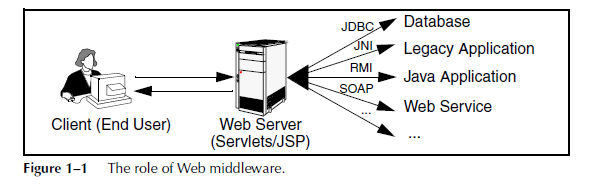
**Servlet and JSP technology**

Servlet and JSP technology has become the technology of choice for developing online stores, interactive

**A Servlet’s Job**

Servlets are Java programs that run on Web or application servers, acting as a middle layer between requests coming from Web browsers or other HTTP clients and databases or applications on the HTTP server. Their job is to perform the following tasks,

as illustrated in Figure 1–1.



1. **Read the explicit data sent by the client.**

The end user normally enters this data in an HTML form on a Web page. However, the data could also come from an applet or a custom HTTP client program. Chapter 4 discusses how servlets read this data.

2. **Read the implicit HTTP request data sent by the browser.**

Figure 1–1 shows a single arrow going from the client to the Web server (the layer where servlets and JSP execute), but there are really *two* varieties of data: the explicit data that the end user enters in a form and the behind-the-scenes HTTP information. Both varieties are critical. The HTTP information includes cookies, information about media types and compression schemes the browser understands,

3. **Generate the results.**

This process may require talking to a database, executing an RMI or EJB call, invoking a Web service, or computing the response directly. Your real data may be in a relational database. Fine. But your database probably doesn’t speak HTTP or return results in HTML, so the Web browser can’t talk directly to the database. Even if it could, for security reasons, you probably would not want it to. The same argument applies to most other applications. You need the Web middle layer to extract the incoming data from the HTTP stream, talk to the application, and embed the results inside a document.

4. **Send the explicit data (i.e., the document) to the client.**

This document can be sent in a variety of formats, including text (HTML or XML), binary (GIF images), or even a compressed format like gzip that is layered on top of some other underlying format. But, HTML is by far the most common format, so an important servlet/JSP task is to wrap the results inside of HTML.

5. **Send the implicit HTTP response data.**

Figure 1–1 shows a single arrow going from the Web middle layer (the servlet or JSP page) to the client. But, there are really *two* varieties of data sent: the document itself and the behind-the-scenes HTTP information. Again, both varieties are critical to effective development. Sending HTTP response data involves telling the browser or other client what type of document is being returned (e.g., HTML), setting cookies and caching parameters

**The Advantages of Servlets Over “Traditional” CGI**

Java servlets are more efficient, easier to use, more powerful, more portable, safer, and cheaper than traditional CGI and many alternative CGI-like technologies. With traditional CGI, a new process is started for each HTTP request. If the CGI program itself is relatively short, the overhead of starting the process can dominate the execution time. With servlets, the Java virtual machine stays running and handles each request with a lightweight Java thread, not a heavyweight operating system process. Similarly, in traditional CGI, if there are *N* requests to the same CGI program, the code for the CGI program is loaded into memory *N* times. With servlets, however, there would be *N* threads, but only a single copy of the servlet class would be

loaded. This approach reduces server memory requirements and saves time by instantiating fewer objects. Finally, when a CGI program finishes handling a request, the program terminates. This approach makes it difficult to cache computations, keep database connections open, and perform other optimizations that rely on persistent data. Servlets, however, remain in memory even after they complete a response, so it is straightforward to store arbitrarily complex data between client requests.

**Convenient**

Servlets have an extensive infrastructure for automatically parsing and decoding HTML form data, reading and setting HTTP headers, handling cookies, tracking sessions, and many other such high-level utilities. In CGI, you have to do much of this yourself. Besides, if you already know the Java programming language, why learn Perl too? You’re already convinced that Java technology makes for more reliable and reusable code than does Visual Basic, VBScript, or C++. Why go back to those languages for server-side programming?

**Powerful**

Servlets support several capabilities that are difficult or impossible to accomplish with regular CGI. Servlets can talk directly to the Web server, whereas regular CGI programs cannot, at least not without using a server-specific API. Communicating with the Web server makes it easier to translate relative URLs into concrete path names, for instance. Multiple servlets can also share data, making it easy to implement database connection pooling and similar resource-sharing optimizations. Servlets can also maintain information from request to request, simplifying techniques like session tracking and caching of previous computations.

**Portable**

Servlets are written in the Java programming language and follow a standard API. Servlets are supported directly or by a plug-in on virtually *every* major Web server. Consequently, servlets written for, say, Macromedia Run can run virtually unchanged on Apache Tomcat, Microsoft Internet Information Server (with a separate plug-in), IBM Web Sphere, planet Enterprise Server, Oracle9i AS, or Star Nine Webster. They are part of the Java 2 Platform, Enterprise Edition (J2EE; see <http://java.sun.com/j2ee/>), so industry support for servlets is becoming even more pervasive.

**Inexpensive**

A number of free or very inexpensive Web servers are good for development use or deployment of low- or medium-volume Web sites. Thus, with servlets and JSP you can start with a free or inexpensive server and migrate to more expensive servers with high-performance capabilities or advanced administration utilities only after your project meets initial success. This is in contrast to many of the other CGI alternatives, which require a significant initial investment for the purchase of a proprietary package. Price and portability are somewhat connected. For example, Marty tries to keep track of the countries of readers that send him questions by email. India was near the top of the list, probably #2 behind the U.S. Marty also taught one of his JSP and servlet

training courses (see http://courses.coreservlets.com/) in Manila, and there was great interest in servlet and JSP technology there. Now, why are India and the Philippines both so interested? We surmise that the answer is twofold. First, both countries have large pools of well-educated software developers.

Second both countries have (or had, at that time) highly unfavorable currency exchange rates against the U.S. dollar. So, buying a special-purpose Web server from a U.S. company consumed a large part of early project funds. But, with servlets and JSP, they could start with a free server: Apache Tomcat (either standalone, embedded in the regular Apache Web server, or embedded in Microsoft IIS). Once the project starts to become successful, they could move to a

server like Caucho Resin that had higher performance and easier administration but that is not free. But none of their servlets or JSP pages have to be rewritten. If their project becomes even larger, they might want to move to a distributed (clustered) environment. No problem: they could move to Macromedia Run Professional, which supports distributed applications (Web farms). Again, none of their servlets or JSP pages have to be rewritten. If the project becomes quite large and complex, they might want to use Enterprise JavaBeans (EJB) to encapsulate their business logic. So, they might switch to BEA Web Logic or Oracle9i AS. Again, none of their servlets

or JSP pages have to be rewritten. Finally, if their project becomes even bigger, they might move it off of their Linux box and onto an IBM mainframe running IBM Web- Sphere. But once again, none of their servlets or JSP pages have to be rewritten

**Secure**

One of the main sources of vulnerabilities in traditional CGI stems from the fact that the programs are often executed by general-purpose operating system shells. So, the CGI programmer must be careful to filter out characters such as backquotes and semicolons that are treated specially by the shell. Implementing this precaution is harder than one might think, and weaknesses stemming from this problem are constantly being uncovered in widely used CGI libraries. A second source of problems is the fact that some CGI programs are processed by languages that do not automatically check array or string bounds. For example, in C and C++ it is perfectly legal to allocate a 100-element array and then write into the 999th “element,” which is really some random part of program memory. So, programmers who forget to perform this check open up their system to deliberate or accidental buffer overflow attacks. Servlets suffer from neither of these problems. Even if a servlet executes a system call (e.g., with Runtime. Exec or JNI) to invoke a program on the local operating system, it does not use a shell to do so. And, of course, array bounds checking and other memory protection features are a central part of the Java programming language.

**Mainstream**

There are a lot of good technologies out there. But if vendors don’t support them and developers don’t know how to use them, what good are they? Servlet and JSP technology is supported by servers from Apache, Oracle, IBM, Sybase, BEA, Macromedia, Caucho, Sun/planet, New Atlanta, ATG, Fujitsu, Ultras, Silver stream, the World Wide Web Consortium (W3C), and many others. Several low-cost plugins add support to Microsoft IIS and Zeus as well. They run on Windows, Unix/Linux, Maces, VMS, and IBM mainframe operating systems. They are the single most popular application of the Java programming language. They are arguably the most popular choice for developing medium to large Web applications. They are used by the airline

industry (most United Airlines and Delta Airlines Web sites), e-commerce (ofoto.com), online banking (First USA Bank, Blanco Popular de Puerto Rico), Web search engines/portals (excite.com), large financial sites (American Century Investments), and hundreds of other sites that you visit every day. Of course, popularity alone is no proof of good technology. Numerous

counter-examples abound. But our point is that you are not experimenting with a

new and unproven technology when you work with server-side Java.

**The Role of JSP**

A somewhat oversimplified view of servlets is that they are Java programs with HTML embedded inside of them. A somewhat oversimplified view of JSP documents is that they are HTML pages with Java code embedded inside of them. For example, compare the sample servlet shown earlier (Listing 1.1) with the JSP page shown below (Listing 1.2). They look totally different; the first looks mostly like a regular Java class, whereas the second looks mostly like a normal HTML page. The interesting thing is that, despite the huge apparent difference, behind the scenes they are the same. In fact, a JSP document is just another way of writing a servlet. JSP pages get translated into servlets, the servlets get compiled, and it is the servlets that run at request time. So, the question is, If JSP technology and servlet technology are essentially equivalent in power, does it matter which you use? The answer is, Yes, yes, yes! The issue is not power, but convenience, ease of use, and maintainability. For example, anything you can do in the Java programming language you could do in assembly language. Does this mean that it does not matter which you use? Hardly. JSP is discussed in great detail starting in Chapter 10. But, it is worthwhile mentioning now how servlets and JSP fit together. JSP is focused on simplifying the creation and maintenance of the HTML. Servlets are best at invoking the business logic and performing complicated operations. A quick rule of thumb is that servlets are best for tasks oriented toward *processing*, whereas JSP is best for tasks oriented toward *presentation*. For some requests, servlets are the right choice. For other requests, JSP is a better option. For still others, neither servlets alone nor JSP alone is best, and a combination of the two (see Chapter 15, “Integrating Servlets and JSP: The Model View Controller (MVC) Architecture”) is best. But the point is that you need *both* servlets and JSP in your overall project: almost no project will consist

entirely of servlets or entirely of JSP. You want both.

**8. Coding**

Login command .java

**package** com.anand.command;

**import** java.util.Date;

**public** **class** LoginCommand {

**private** String fullName;

**private** String email;

**private** String password;

**private** String phone;

**private** String education;

**private** String address;

**private** String picture;

**private** String pictureExtensions;

**private** String serialNumber;

**private** Date registrationDate;

**private** String userType;

**public** String getFullName() {

**return** fullName;

}

**public** **void** setFullName(String fullName) {

**this**.fullName = fullName;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** String getPhone() {

**return** phone;

}

**public** **void** setPhone(String phone) {

**this**.phone = phone;

}

**public** String getEducation() {

**return** education;

}

**public** **void** setEducation(String education) {

**this**.education = education;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** String getPicture() {

**return** picture;

}

**public** **void** setPicture(String picture) {

**this**.picture = picture;

}

**public** String getPictureExtensions() {

**return** pictureExtensions;

}

**public** **void** setPictureExtensions(String pictureExtensions) {

**this**.pictureExtensions = pictureExtensions;

}

**public** String getSerialNumber() {

**return** serialNumber;

}

**public** **void** setSerialNumber(String serialNumber) {

**this**.serialNumber = serialNumber;

}

**public** Date getRegistrationDate() {

**return** registrationDate;

}

**public** **void** setRegistrationDate(Date registrationDate) {

**this**.registrationDate = registrationDate;

}

**public** String getUserType() {

**return** userType;

}

**public** **void** setUserType(String userType) {

**this**.userType = userType;

}

}

Registration command .java

**package** com.anand.command;

**import** java.util.Date;

**public** **class** RegistrationCommand {

**private** String fullName;

**private** String email;

**private** String password;

**private** String phone;

**private** String education;

**private** String address;

**private** String picture;

**private** String pictureExtensions;

**private** String serialNumber;

**private** Date registrationDate;

**private** String userType;

**public** String getFullName() {

**return** fullName;

}

**public** **void** setFullName(String fullName) {

**this**.fullName = fullName;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** String getPhone() {

**return** phone;

}

**public** **void** setPhone(String phone) {

**this**.phone = phone;

}

**public** String getEducation() {

**return** education;

}

**public** **void** setEducation(String education) {

**this**.education = education;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** String getPicture() {

**return** picture;

}

**public** **void** setPicture(String picture) {

**this**.picture = picture;

}

**public** String getPictureExtensions() {

**return** pictureExtensions;

}

**public** **void** setPictureExtensions(String pictureExtensions) {

**this**.pictureExtensions = pictureExtensions;

}

**public** String getSerialNumber() {

**return** serialNumber;

}

**public** **void** setSerialNumber(String serialNumber) {

**this**.serialNumber = serialNumber;

}

**public** Date getRegistrationDate() {

**return** registrationDate;

}

**public** **void** setRegistrationDate(Date registrationDate) {

**this**.registrationDate = registrationDate;

}

**public** String getUserType() {

**return** userType;

}

**public** **void** setUserType(String userType) {

**this**.userType = userType;

}

}

Message command .java

**package** com.anand.command;

**public** **class** MessageCommand {

**private** String toEmail;

**private** String fromEmail;

**private** String subject;

**private** String message;

**private** String postDate;

**private** String serialID;

**private** String status;

**public** String getToEmail() {

**return** toEmail;

}

**public** **void** setToEmail(String toEmail) {

**this**.toEmail = toEmail;

}

**public** String getFromEmail() {

**return** fromEmail;

}

**public** **void** setFromEmail(String fromEmail) {

**this**.fromEmail = fromEmail;

}

**public** String getSubject() {

**return** subject;

}

**public** **void** setSubject(String subject) {

**this**.subject = subject;

}

**public** String getMessage() {

**return** message;

}

**public** **void** setMessage(String message) {

**this**.message = message;

}

**public** String getPostDate() {

**return** postDate;

}

**public** **void** setPostDate(String postDate) {

**this**.postDate = postDate;

}

**public** String getSerialID() {

**return** serialID;

}

**public** **void** setSerialID(String serialID) {

**this**.serialID = serialID;

}

**public** String getStatus() {

**return** status;

}

**public** **void** setStatus(String status) {

**this**.status = status;

}

@Override

**public** String toString() {

StringBuilder builder = **new** StringBuilder();

builder.append("MessageBo [fromEmail=");

builder.append(fromEmail);

builder.append(", message=");

builder.append(message);

builder.append(", postDate=");

builder.append(postDate);

builder.append(", serialID=");

builder.append(serialID);

builder.append(", status=");

builder.append(status);

builder.append(", subject=");

builder.append(subject);

builder.append(", toEmail=");

builder.append(toEmail);

builder.append("]");

**return** builder.toString();

}

}

Login controller .java

package com.anand.controller;

import javax.servlet.ServletContext;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import org.springframework.validation.BindException;

import org.springframework.web.servlet.ModelAndView;

import org.springframework.web.servlet.mvc.SimpleFormController;

import com.anand.dto.AccountDto;

import com.anand.service.AccountService;

import com.anand.service.ArticleService;

@SuppressWarnings("deprecation")

public class LoginController extends SimpleFormController {

private AccountService accountService;

private ArticleService articleService;

public void setArticleService(ArticleService articleService) {

this.articleService = articleService;

}

public void setAccountService(AccountService accountService) {

this.accountService = accountService;

}

@Override

protected ModelAndView onSubmit(HttpServletRequest request,

HttpServletResponse response, Object command, BindException errors)

throws Exception {

HttpSession session = request.getSession();

ServletContext context = getServletContext();

String path = context.getRealPath("/images");

AccountDto accountDto;

accountDto = new AccountDto();

accountDto.setEmail(request.getParameter("email"));

accountDto.setPassword(request.getParameter("password"));

request.getSession();

String userType =null;

accountDto=accountService.authentication(path,accountDto);

userType=accountDto.getUserType();

if (userType==null||"".equals(userType)) {

return new ModelAndView("login", "resMsg","Invalid credentials!!");

} else {

session = request.getSession();

session.setAttribute("USER", userType);

session.setAttribute("USEREMAIL", request.getParameter("email"));

session.setAttribute("USERPROFILE", accountDto);

// get all my friend

session.setAttribute("MYFRIENDLIST",accountService.getFriendList(request.getParameter("email")));

String page=null;

if (userType.equals("ADMIN")) {

page="admin-home";

} else if (userType.equals("USER")) {

session.setAttribute("ARTICILES", articleService.getArticles());

page="user-home";

}

System.out.println(page+userType);

return new ModelAndView(page);

}

}

}

View all friends.java

package com.anand.controller;

import javax.servlet.ServletContext;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import org.springframework.web.servlet.ModelAndView;

import org.springframework.web.servlet.mvc.AbstractController;

import com.anand.dto.AccountDto;

import com.anand.service.SearchFriendsService;

public class ViewAllFriendRequestController extends AbstractController {

SearchFriendsService searchFriendsService;

public void setSearchFriendsService(

SearchFriendsService searchFriendsService) {

this.searchFriendsService = searchFriendsService;

}

@Override

protected ModelAndView handleRequestInternal(HttpServletRequest request,

HttpServletResponse response) throws Exception {

HttpSession session = request.getSession();

AccountDto accountDto = (AccountDto) session

.getAttribute("USERPROFILE");

ServletContext context = getServletContext();

String path = context.getRealPath("/images");

return new ModelAndView("view-friend-request","userList",

searchFriendsService.getFriendRequest(path, accountDto.getEmail()));

}

}

Utility.java

package com.anand.util;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.UUID;

public class Utility {

private SimpleDateFormat simpleDateFormat;

public void setSimpleDateFormat(SimpleDateFormat simpleDateFormat) {

this.simpleDateFormat = simpleDateFormat;

}

public String generateSerialNumber() {

return UUID.randomUUID().toString();

}

public String getCurrentDate() throws ParseException {

return simpleDateFormat.format(new Date());

}

}

Mobile vo.java

**package** com.anand.vo;

**public** **class** MobileVo {

**private** String brand;

**private** String color;

**private** String ram;

**private** String operatingSystem;

**private** String internalStorage;

**private** String networkType;

**private** String screenSize;

**private** String batteryCapacity;

**private** String simType;

**private** String primaryCamera;

**private** String secondaryCamera;

**private** String clockSpeed;

**private** String emi;

**private** String productDescription;

**private** String productAvailability;

**private** String offers;

**public** String getBrand() {

**return** brand;

}

**public** **void** setBrand(String brand) {

**this**.brand = brand;

}

**public** String getColor() {

**return** color;

}

**public** **void** setColor(String color) {

**this**.color = color;

}

**public** String getRam() {

**return** ram;

}

**public** **void** setRam(String ram) {

**this**.ram = ram;

}

**public** String getOperatingSystem() {

**return** operatingSystem;

}

**public** **void** setOperatingSystem(String operatingSystem) {

**this**.operatingSystem = operatingSystem;

}

**public** String getInternalStorage() {

**return** internalStorage;

}

**public** **void** setInternalStorage(String internalStorage) {

**this**.internalStorage = internalStorage;

}

**public** String getNetworkType() {

**return** networkType;

}

**public** **void** setNetworkType(String networkType) {

**this**.networkType = networkType;

}

**public** String getScreenSize() {

**return** screenSize;

}

**public** **void** setScreenSize(String screenSize) {

**this**.screenSize = screenSize;

}

**public** String getBatteryCapacity() {

**return** batteryCapacity;

}

**public** **void** setBatteryCapacity(String batteryCapacity) {

**this**.batteryCapacity = batteryCapacity;

}

**public** String getSimType() {

**return** simType;

}

**public** **void** setSimType(String simType) {

**this**.simType = simType;

}

**public** String getPrimaryCamera() {

**return** primaryCamera;

}

**public** **void** setPrimaryCamera(String primaryCamera) {

**this**.primaryCamera = primaryCamera;

}

**public** String getSecondaryCamera() {

**return** secondaryCamera;

}

**public** **void** setSecondaryCamera(String secondaryCamera) {

**this**.secondaryCamera = secondaryCamera;

}

**public** String getClockSpeed() {

**return** clockSpeed;

}

**public** **void** setClockSpeed(String clockSpeed) {

**this**.clockSpeed = clockSpeed;

}

**public** String getEmi() {

**return** emi;

}

**public** **void** setEmi(String emi) {

**this**.emi = emi;

}

**public** String getProductDescription() {

**return** productDescription;

}

**public** **void** setProductDescription(String productDescription) {

**this**.productDescription = productDescription;

}

**public** String getProductAvailability() {

**return** productAvailability;

}

**public** **void** setProductAvailability(String productAvailability) {

**this**.productAvailability = productAvailability;

}

**public** String getOffers() {

**return** offers;

}

**public** **void** setOffers(String offers) {

**this**.offers = offers;

}

}

**9. System Testing**

**Testing Methodologies**

Testing is the process of finding differences between the expected behavior specified by system models and the observed behavior implemented system. From modeling point of view , testing is the attempt of falsification of the system with respect to the system models. The goal of testing is to design tests that exercise defects in the system and to reveal problems.

The process of executing a program with intent of finding errors is called testing. During testing , the program to be tested is executed with a set of test cases , and the output of the program for the test cases is evaluated to determine if the program is performing as expected . Testing forms the first step in determining the errors in the program. The success of testing in revealing errors in program depends critically on test cases.

**Strategic Approach to Software Testing:**

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirements analysis where the information domain , functions , behavior , performance , constraints and validation criteria for software are established. moving inward along the spiral , we come to design and finally to coding . To develop computer software we spiral in along streamlines that decreases the level of abstraction on each item.A Strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing will progress by moving outward along the spiral to integration testing , where the focus on the design and the concentration of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed . Finally we arrive at system testing , where the software and other system elements are tested as a whole .

UNUNI

UNIT TESTING

MODULE

SUB-SYSTEM

**Component**

SYSTEM TESTING

**Integration Testing**

ACCEPTANCE

**User Testing**

**Different Levels of Testing**

Client Needs Acceptance Testing

Requirements System Testing

Design Integration Testing

Code Unit Testing

Testing is the process of finding difference between the expected behavior specified by system models and the observed behavior of the implemented system.

**8.2 Testing Activities**

Different levels of testing are used in the testing process , each level of testing aims to test different aspects of the system. the basic levels are:

Unit testing

Integration testing

System testing

Acceptance testing

**Unit Testing**

Unit testing focuses on the building blocks of the software system, that is, objects and sub system . There are three motivations behind focusing on components. First, unit testing reduces the complexity of the overall tests activities, allowing us to focus on smaller units of the system. Second , unit testing makes it easier to pinpoint and correct faults given that few components are involved in this test . Third , Unit testing allows parallelism in the testing activities , that is each component can be tested independently of one another . Hence the goal is to test the internal logic of the module.

**Integration Testing**

In the integration testing, many test modules are combined into sub systems , which are then tested . The goal here is to see if the modules can be integrated properly, the emphasis being on testing module interaction.

After structural testing and functional testing we get error free modules. These modules are to be integrated to get the required results of the system. After checking a module, another module is tested and is integrated with the previous module. After the integration, the test cases are generated and the results are tested.

**System Testing**

In system testing the entire software is tested . The reference document for this process is the requirement document and the goal is to see whether the software meets its requirements. The system was tested for various test cases with various inputs.

**Acceptance Testing**

Acceptance testing is sometimes performed with realistic data of the client to demonstrate that the software is working satisfactory. Testing here focus on the external behavior of the system , the internal logic of the program is not emphasized . In acceptance testing the system is tested for various inputs.

**8.3 Types of Testing**

1. Black box or functional testing
2. White box testing or structural testing

**Black box testing**

This method is used when knowledge of the specified function that a product has been designed to perform is known . The concept of black box is used to represent a system whose inside workings are not available to inspection . In a black box the test item is a "Black" , since its logic is unknown , all that is known is what goes in and what comes out , or the input and output.

Black box testing attempts to find errors in the following categories:

Incorrect or missing functions

Interface errors

Errors in data structure

Performance errors

Initialization and termination errors

As shown in the following figure of Black box testing , we are not thinking of the internal workings , just we think about

What is the output to our system?

What is the output for given input to our system?

**?**

Input Output

The Black box is an imaginary box that hides its internal workings

**White box testing**

White box testing is concerned with testing the implementation of the program. the intent of structural is not to exercise all the inputs or outputs but to exercise the different programming and data structure used in the program. Thus structural testing aims to achieve test cases that will force the desire coverage of different structures . Two types of path testing are statement testing coverage and branch testing coverage.

**INTERNAL WORKING**

Input Output

The White Box testing strategy , the internal workings

**8.4 Test Plan**

Testing process starts with a test plan. This plan identifies all the testing related activities that must be performed and specifies the schedules , allocates the resources , and specified guidelines for testing . During the testing of the unit the specified test cases are executed and the actual result compared with expected output. The final output of the testing phase is the test report and the error report.

**Test Data:**

Here all test cases that are used for the system testing are specified. The goal is to test the different functional requirements specified in Software Requirements Specifications (SRS) document.

**Unit Testing:**

Each individual module has been tested against the requirement with some test data.

**Test Report:**

The module is working properly provided the user has to enter information. All data entry forms have tested with specified test cases and all data entry forms are working properly.

**Error Report:**

If the user does not enter data in specified order then the user will be prompted with error messages. Error handling was done to handle the expected and unexpected errors.

**8.7 Test cases**

A Test case is a set of input data and expected results that exercises a component with the purpose of causing failure and detecting faults . test case is an explicit set of instructions designed to detect a particular class of defect in a software system , by bringing about a failure . A Test case can give rise to many tests.

**TEST CASES:**

Test cases can be divided in to two types. First one is Positive test cases and second one is negative test cases. In positive test cases are conducted by the developer intention is to get the output. In negative test cases are conducted by the developer intention is to don’t get the output.

**+VE TEST CASES**

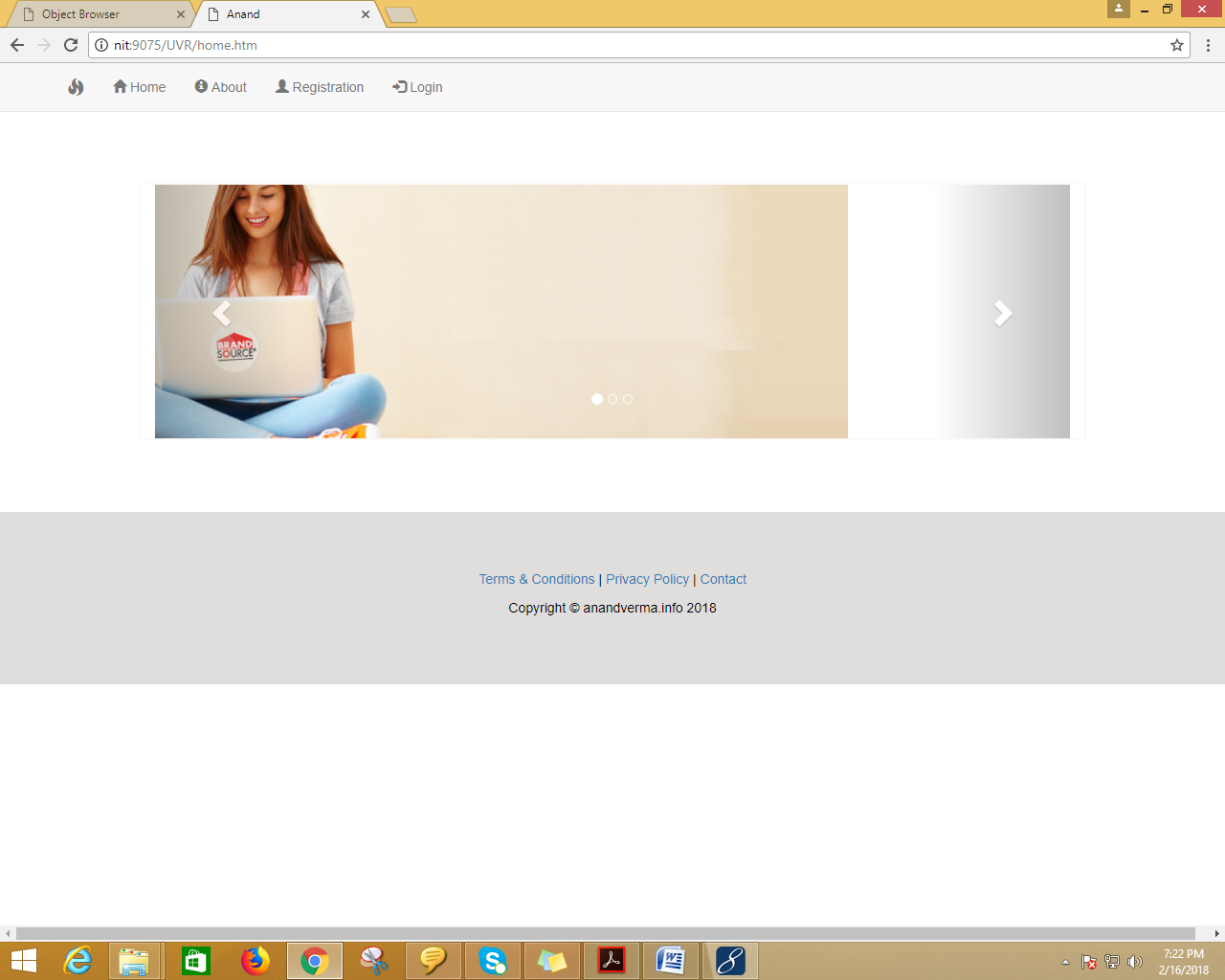
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S .No** | **Test case Description** | **Actual value** | **Expected value** | **Result** |
| **1** | Create new user registration process | Enter the personal info and address info. | Update personal info and address info in to oracle database successfully | True |
| **2** | Enter the username and password | Verification of login details. | Login Successfully | True |
| **3** | Upload file into single cloud , multiple cloud | Enter all fields | Web data uploaded successfully | True |
| **4** | Enter keyword query submission | Enter valid query | Display relevant records based keyword query | True |

**+VE TEST CASES**

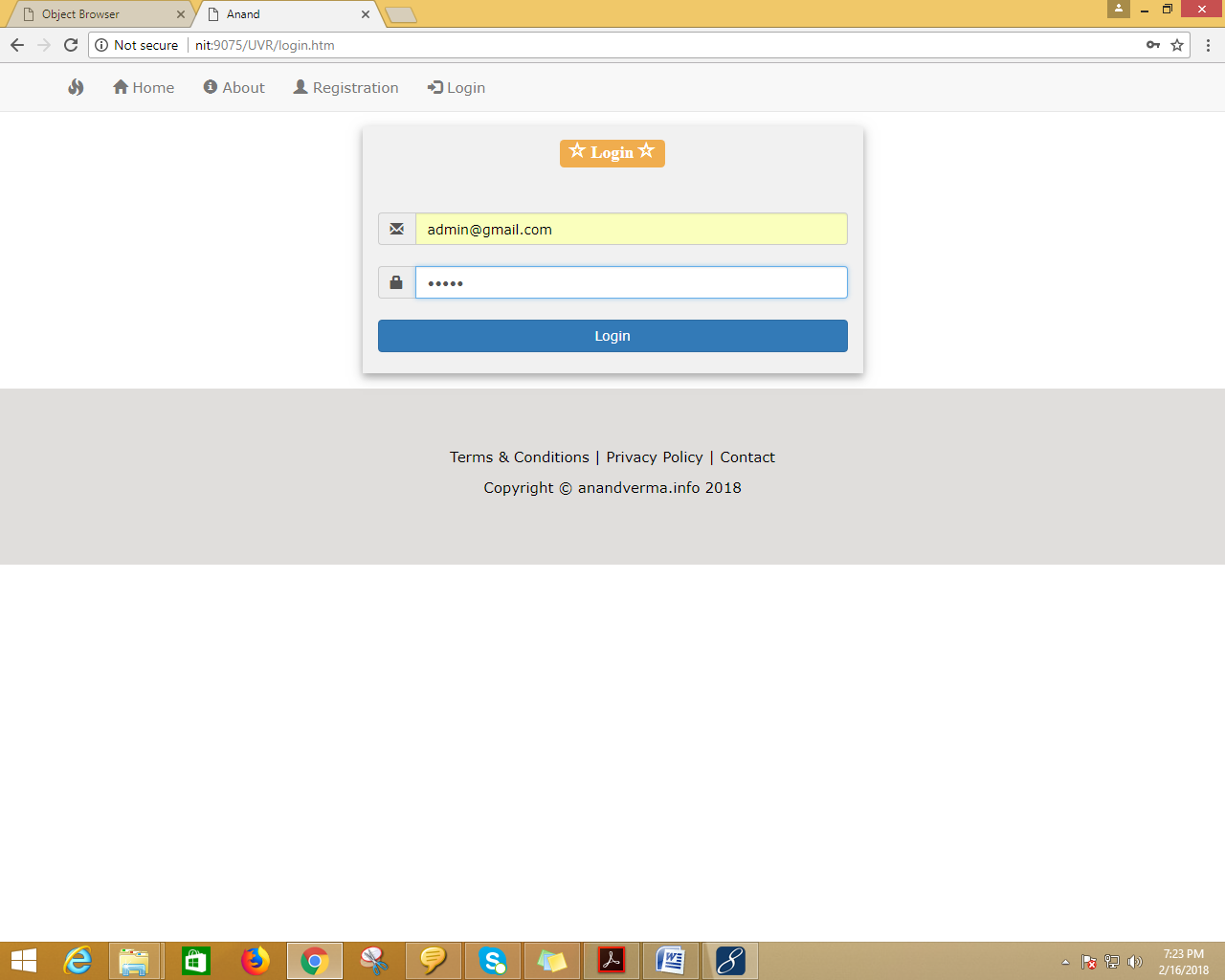
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S .No** | **Test case Description** | **Actual value** | **Expected value** | **Result** |
| **1** | Create the new user registration process | Enter the personal info and address info. | Personal info and address info its not update into database successfully. | False |
| **2** | Enter the username and password | Verification of login details. | Login failed | False |
| **3** | Upload information | Enter all fields | Web data is not create successfully. | False |
| **4** | Enter keyword query submission | Enter valid query | Relevant records are not present in database | False |

**10. Output Screens**

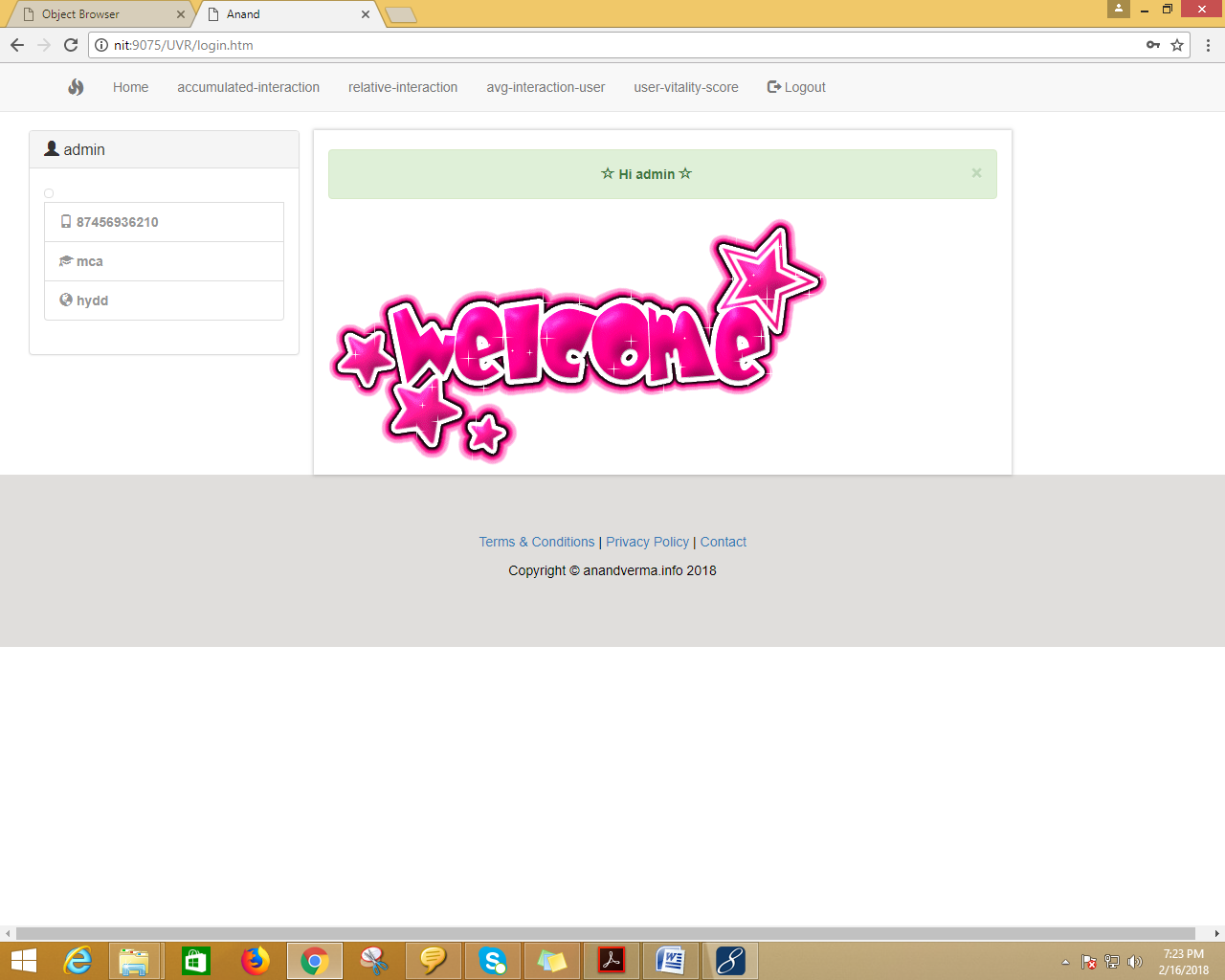
Home page



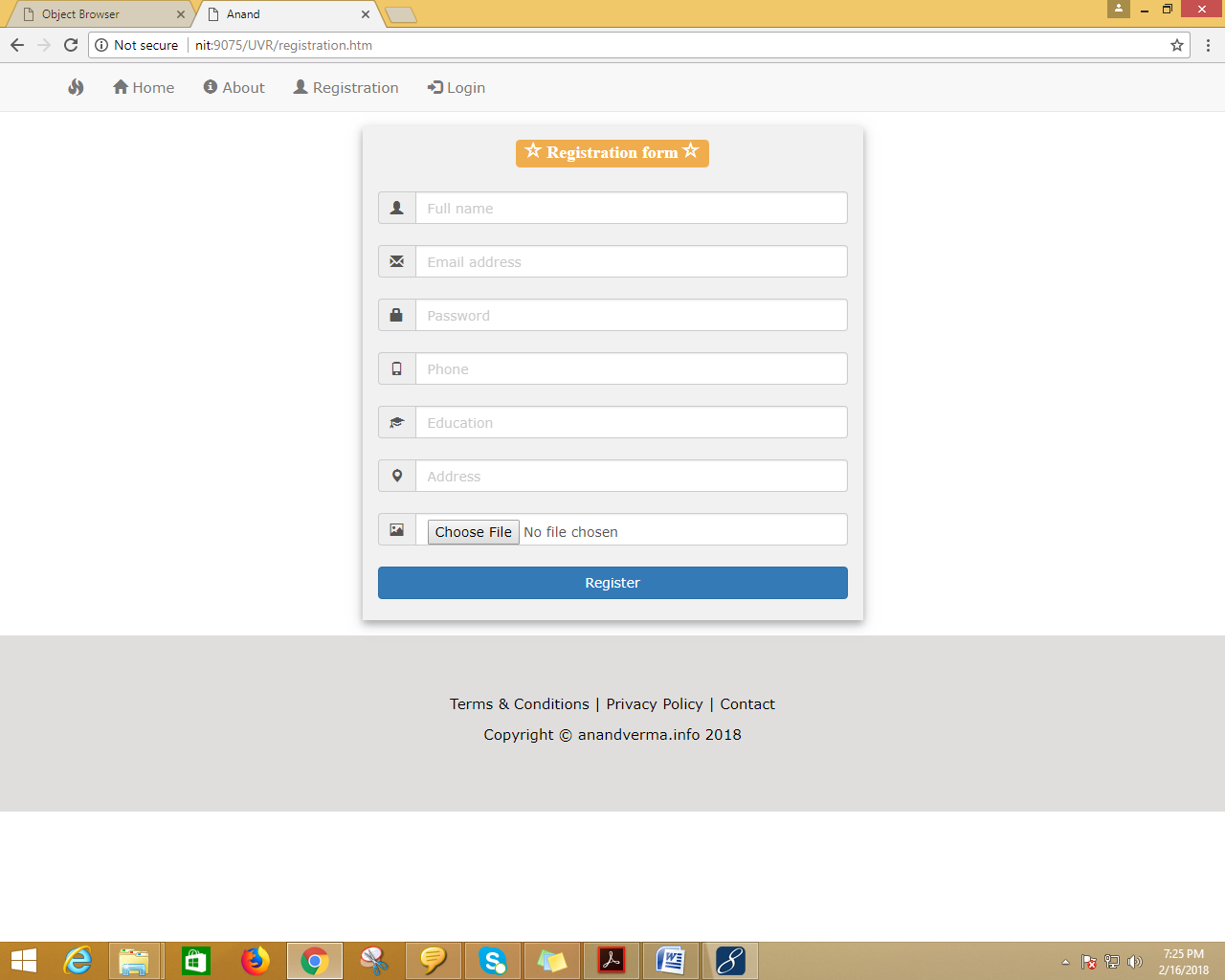
Admin login page



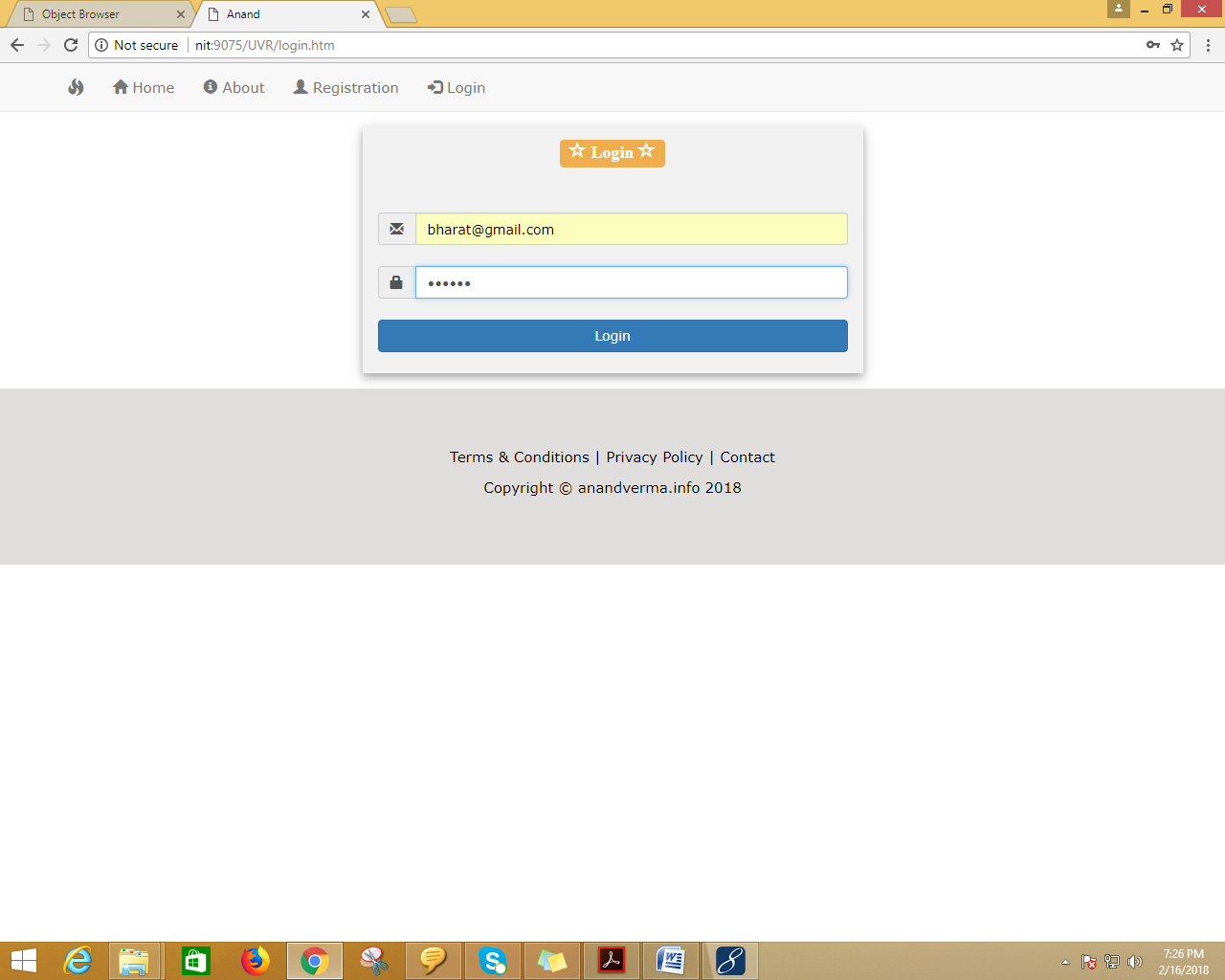
Admin home page



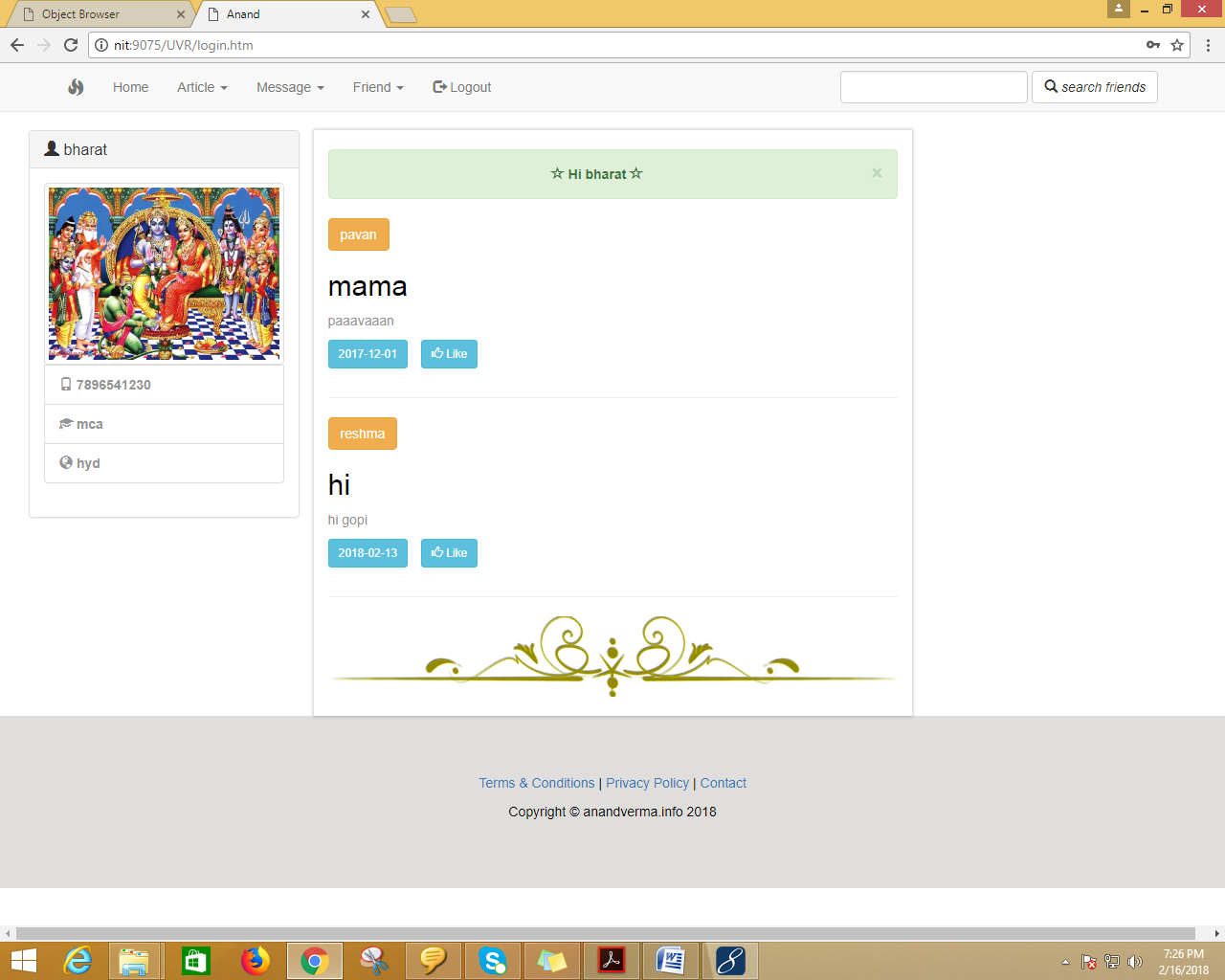
User registration form



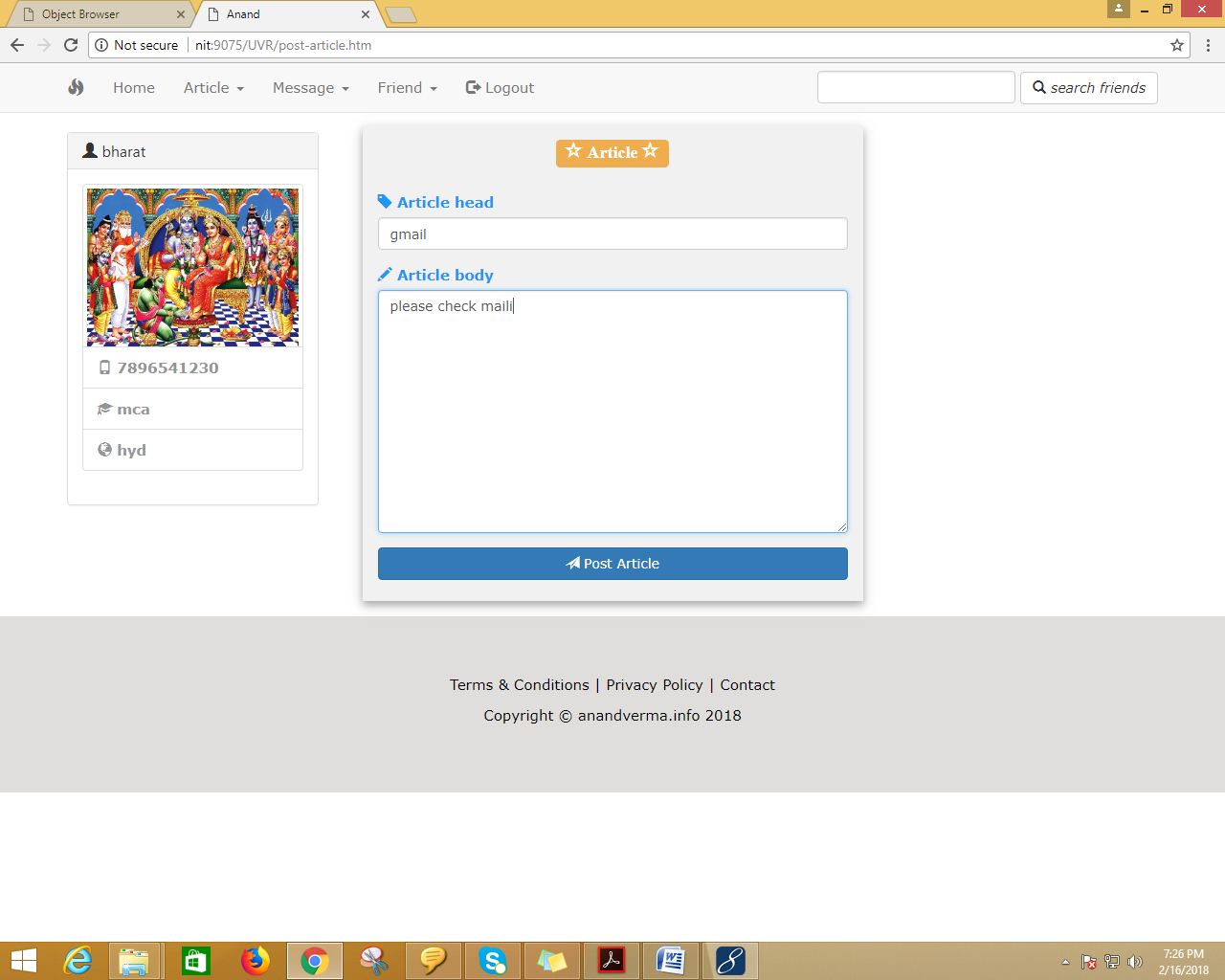
User login form:



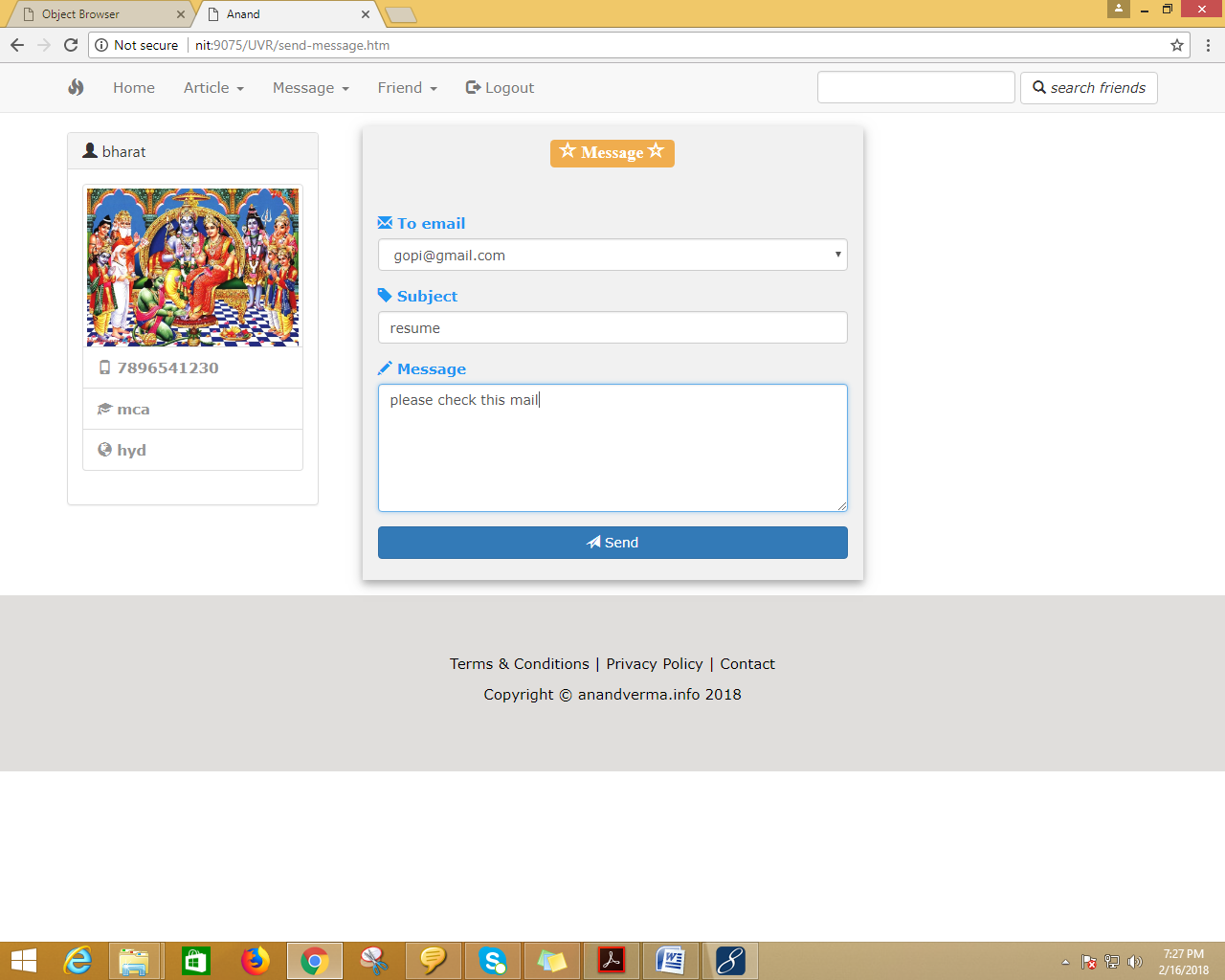
User home



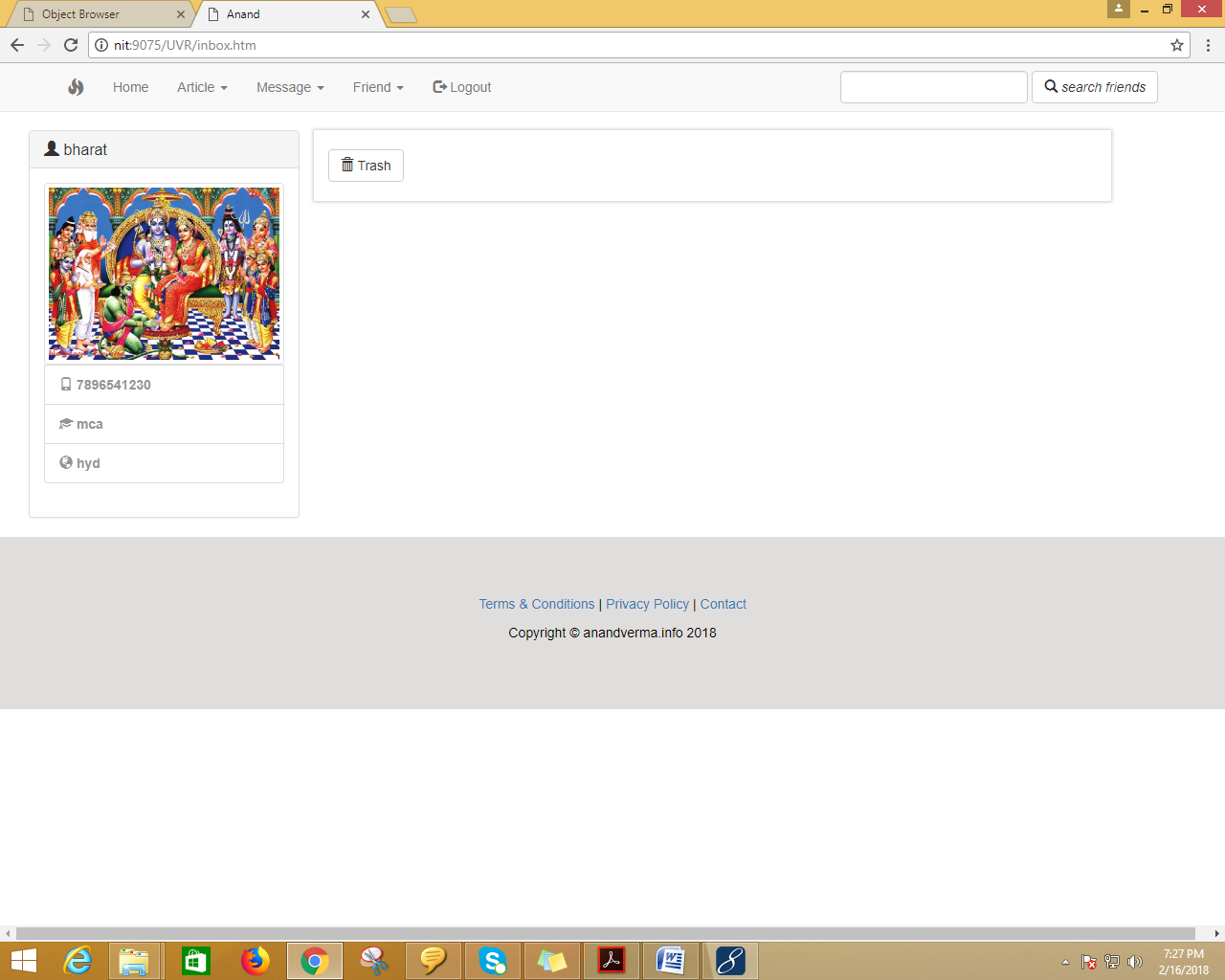
User post new artcles



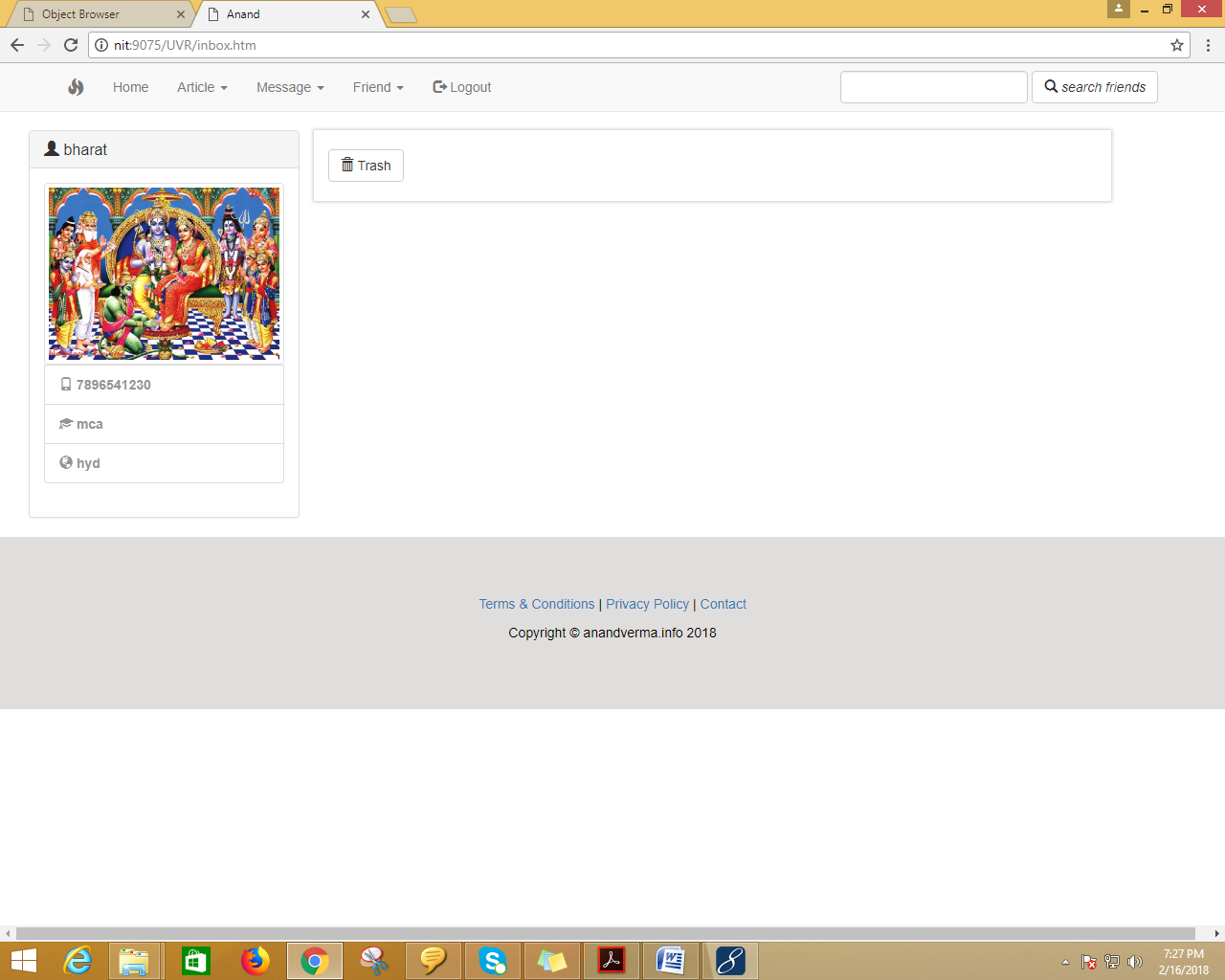
User compose mail



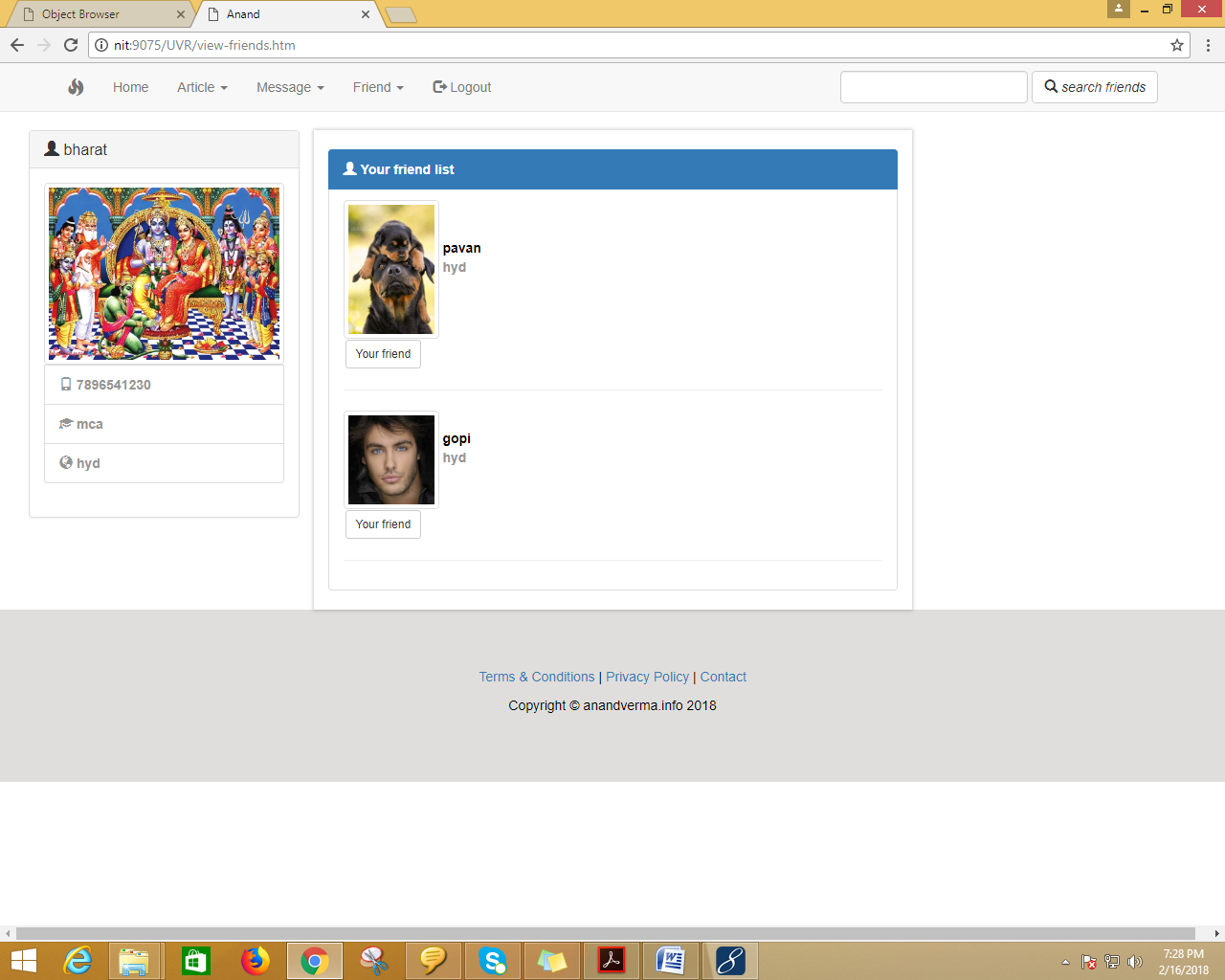
User check user mail inbox



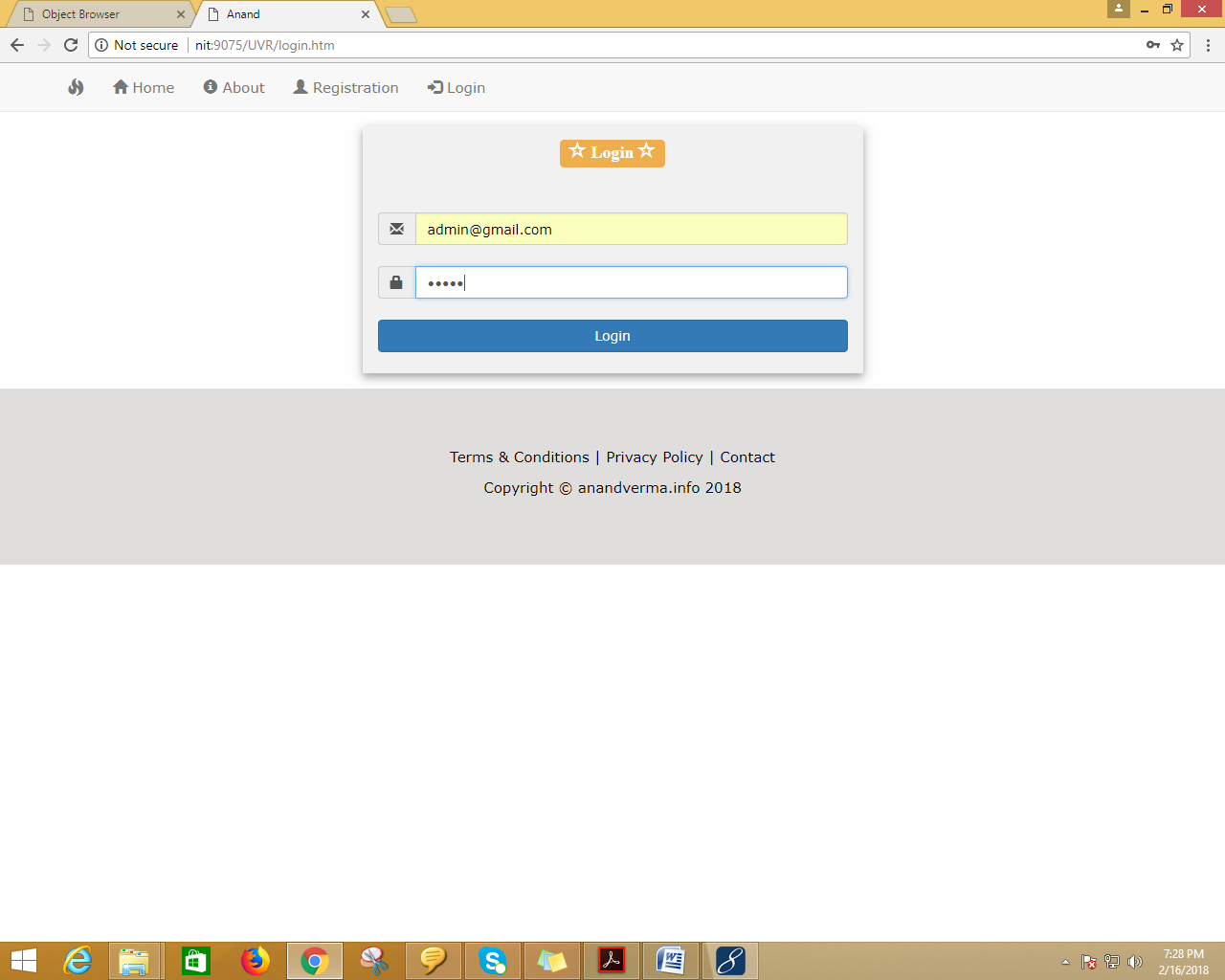
User see any friend requests



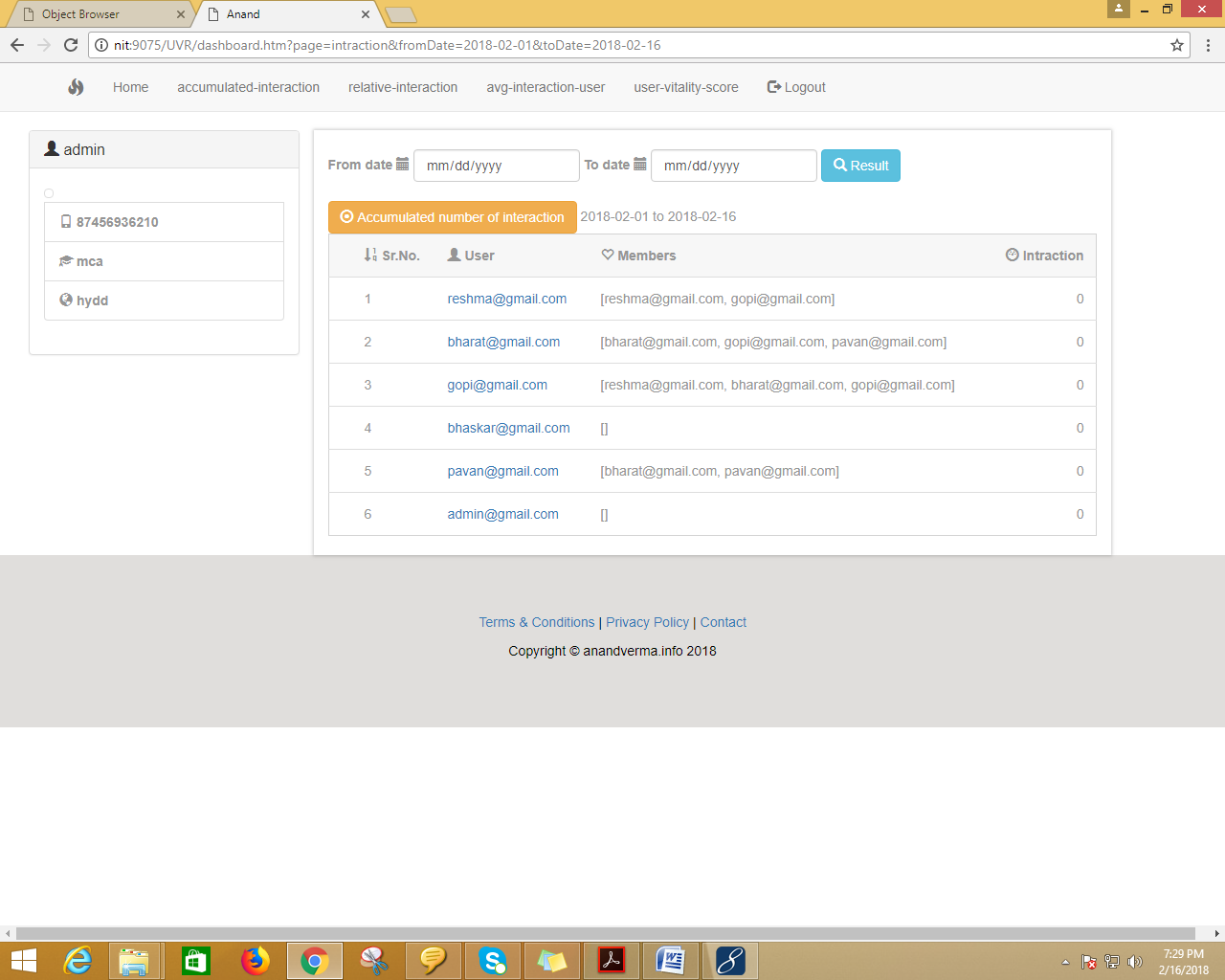
User all friends



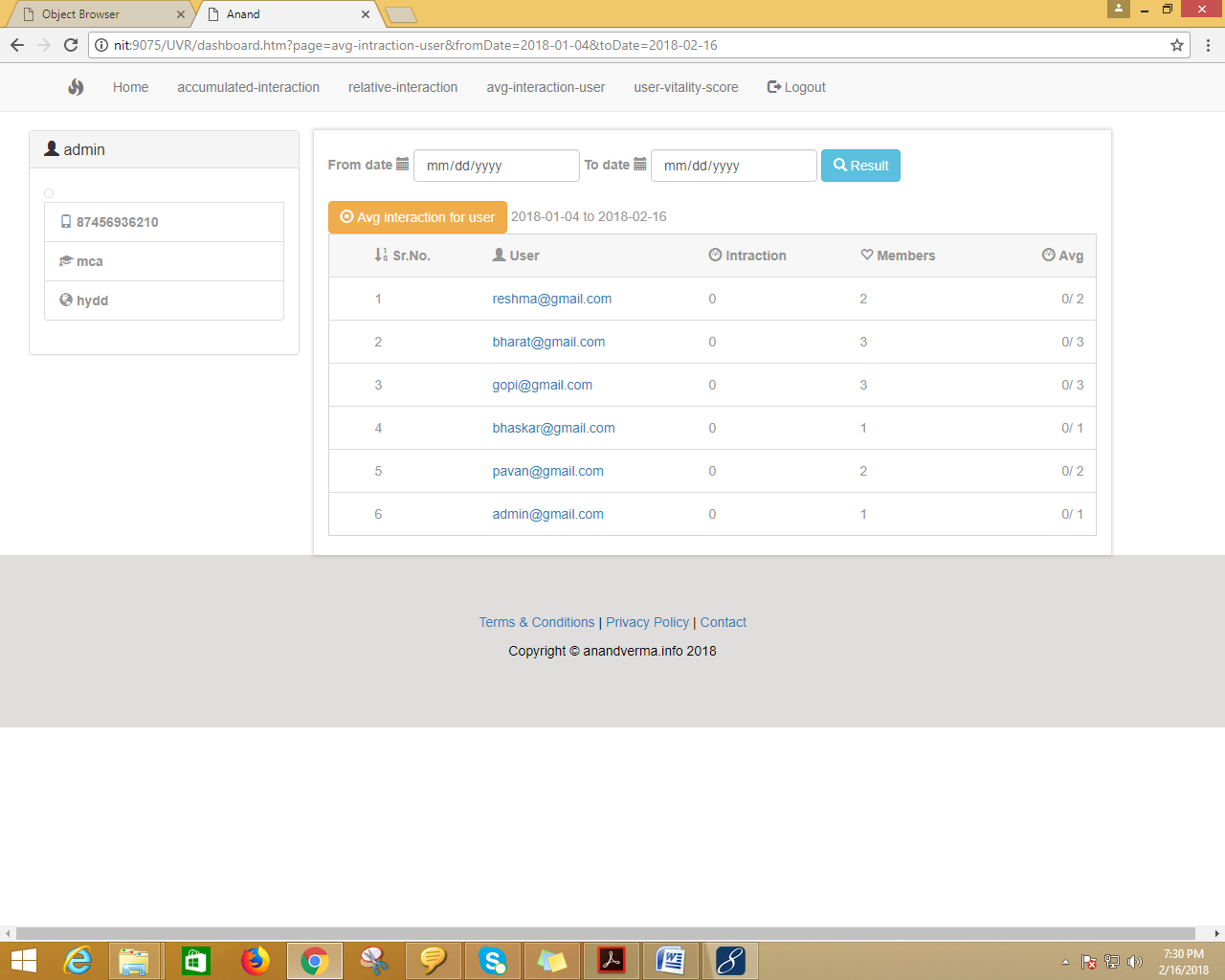
After completion of user activity then login admin



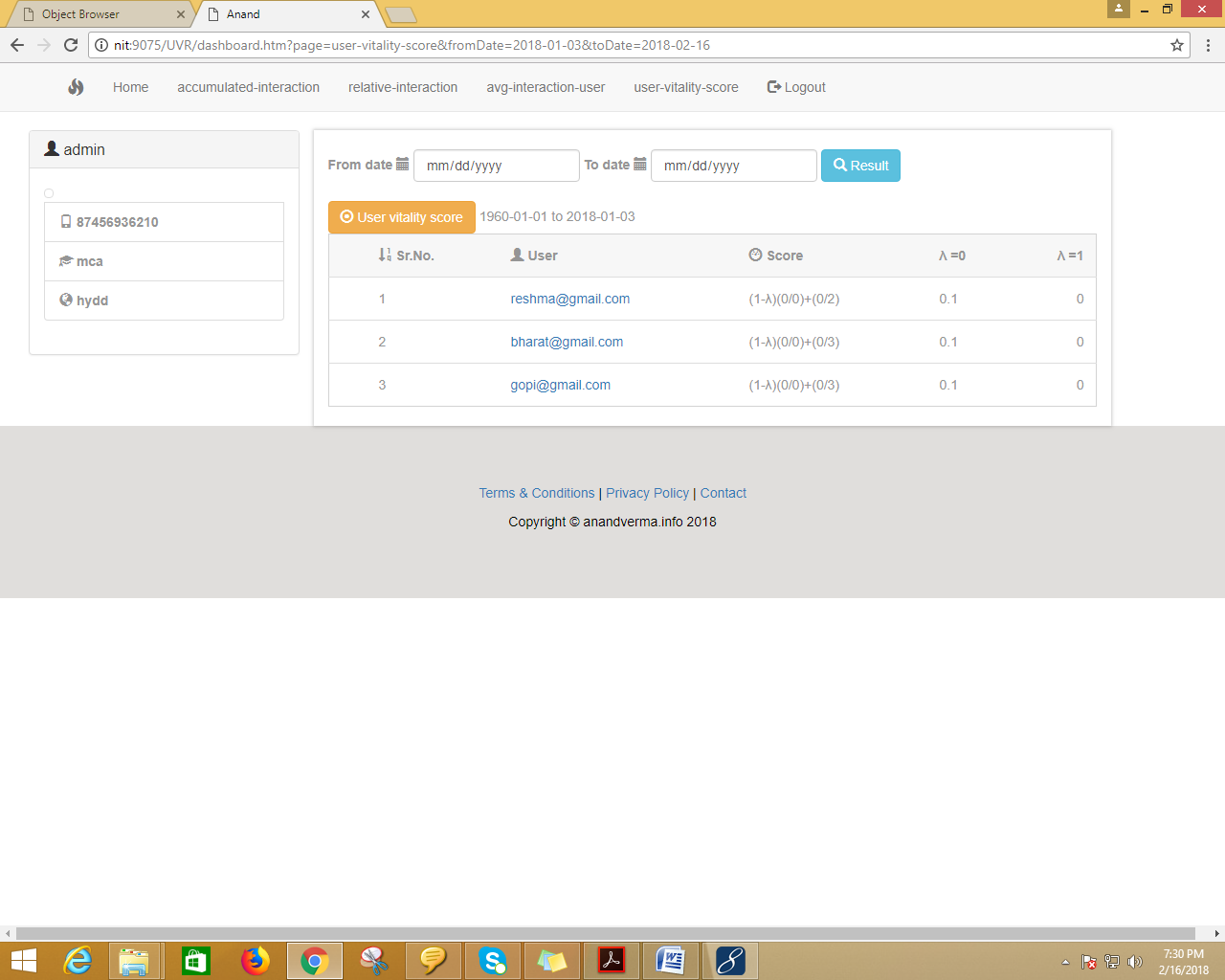
Admin see the acumulated interaction:



Admin see the average interaction for the user



Admin see the user vitality score



Finaly logout

**11. Reports**

The accurate results of both user vitality ranking and prediction could benefit many parties in different social networking services, e.g., a user vitality ranking list could help ads providers to better display their ads to active users and reach more audiences.

**12. Conclusion**

In this paper, we presented a study on user vitality ranking and prediction in social networking services such as microblog application. Specifically, we first introduced a user vitality ranking problem, which is based on dynamic interactions between users on social networks. To solve this problem, we developed two algorithms to rank users based on vitality. While the first algorithm works based on the developed two user vitality measurements, the second algorithm further takes into account the mutual influence among users while computing the vitality measurements. Then we presented a user vitality prediction problem and introduced a regressionbased method for the prediction task. Intensive experiments on two real-world data sets that are collected from different domains clearly demonstrate the effectiveness of our ranking and prediction methods. The accurate results of both user vitality ranking and prediction could benefit many parties in different social networking services, e.g., a user vitality ranking list could help ads providers to better display their ads to active users and reach more audiences..

**13. Future Enhancements**

**Future Enhancements:**

It is not possible to develop a system that makes all the requirements of the user. User requirements keep changing as the system is being used. Some of the future enhancements that can be done to this system are:

* As the technology emerges, it is possible to upgrade the system and can be adaptable to desired environment.
* Because it is based on object-oriented design, any further changes can be easily adaptable.
* Based on the future security issues, security can be improved using emerging technologies.
* Attendance module can be added
* sub admin module can be added

**14. References**

[1] Eytan Bakshy, Jake M Hofman, Winter A Mason, and Duncan J Watts. Everyone’s an influencer: quantifying influence on twitter. In *Proceedings of the fourth ACM international conference on Web search* *and data mining*, pages 65–74. ACM, 2011.

[2] Sergey Brin and Lawrence Page. Reprint of: The anatomy of a largescale hypertextual web search engine. *Computer networks*, 56(18):3825– 3833, 2012.

[3] Robert Goodell Brown. *Smoothing, forecasting and prediction of discrete time series*. Courier Corporation, 2004.

[4] Christopher S Campbell, Paul P Maglio, Alex Cozzi, and Byron Dom. Expertise identification using email communications. In *Proceedings* *of the twelfth international conference on Information and knowledge* *management*, pages 528–531. ACM, 2003.

[5] Meeyoung Cha, Hamed Haddadi, Fabricio Benevenuto, and P Krishna Gummadi. Measuring user influence in twitter: The million follower fallacy. *ICWSM*, 10(10-17):30, 2010.

[6] Pedro Domingos and Matt Richardson. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international* *conference on Knowledge discovery and data mining*, pages 57–66.

ACM, 2001.

[7] Philip E Brown Junlan Feng. Measuring user influence on twitter using modified k-shell decomposition. 2011.

[8] Jian Jiao, Jun Yan, Haibei Zhao, and Weiguo Fan. Expertrank: An expert user ranking algorithm in online communities. In *New Trends in Information* *and Service Science, 2009. NISS’09. International Conference* *on*, pages 674–679. IEEE, 2009.

[9] Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[10] Shamanth Kumar, Fred Morstatter, and Huan Liu. *Twitter data analytics*. Springer, 2014.

[11] Wenting Liu, Guangxia Li, and James Cheng. Fast pagerank approximation by adaptive sampling. *Knowledge and Information Systems*, 42(1):127–146, 2015.

[12] Rada Mihalcea. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the ACL 2004 on* *Interactive poster and demonstration sessions*, page 20. Association for Computational Linguistics, 2004.

[13] Amin Omidvar, Mehdi Garakani, and Hamid R Safarpour. Context based user ranking in forums for expert finding using wordnet dictionary and social network analysis. *Information Technology and Management*, 15(1):51–63, 2014.

[14] Tore Opsahl, Filip Agneessens, and John Skvoretz. Node centrality in weighted networks: Generalizing degree and shortest paths. *Social* *networks*, 32(3):245–251, 2010.

[15] Daniel M Romero, Wojciech Galuba, Sitaram Asur, and Bernardo A Huberman. Influence and passivity in social media. In *Machine learning* *and knowledge discovery in databases*, pages 18–33. Springer, 2011.

[16] Yuanfeng Song, Wilfred Ng, Kenneth Wai-Ting Leung, and Qiong Fang. Sfp-rank: significant frequent pattern analysis for effective ranking. *Knowledge and Information Systems*, 43(3):529–553, 2015.

[17] Stanley Wasserman and Philippa Pattison. Logit models and logistic regressions for social networks: I. an introduction to markov graphs andp. *Psychometrika*, 61(3):401–425, 1996.

[18] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of the third ACM* *international conference on Web search and data mining*, pages 261–

270. ACM, 2010.

[19] Yuto Yamaguchi, Tsubasa Takahashi, Toshiyuki Amagasa, and Hiroyuki Kitagawa. Turank: Twitter user ranking based on user-tweet graph analysis. In *Web Information Systems Engineering–WISE 2010*, pages

240–253. Springer, 2010.

[20] Yang Yang, Ryan N Lichtenwalter, and Nitesh V Chawla. Evaluating link prediction methods. *Knowledge and Information Systems*, 45(3):751–782, 2015.

[21] Kang Zhao, John Yen, Greta Greer, Baojun Qiu, Prasenjit Mitra, and Kenneth Portier. Finding influential users of online health communities: a new metric based on sentiment influence. *Journal of the American* *Medical Informatics Association*, 21(e2):e212–e218, 2014.

**15. Bibliography**

(1) Java Complete Reference by Herbert Shield

(2) Database Programming with JDBC and Java by George Reese

(3) Java and XML By Brett McLaughlin

(4) Wikipedia, URL: <http://www.wikipedia.org>.

(5) Answers.com, Online Dictionary, Encyclopedia and much more, URL: <http://www.answers.com>

(6) Google, URL: <http://www.google.co.in>

(7)Project Management URL: http://www.startwright.com/project.htm