DETECT PROFESSIONAL MALICIOUS USER WITH METRIC LEARNING IN RECOMMENDER SYSTEMS

##### A PROJECT REPORT

###### ***Submitted by***

***VIKRAMAN.GD [REGISTER NO: 211419104305]***

***RAMAKUMARAN.B [REGISTER NO: 211419104214]***

***SANJAY SELVAN.R [REGISTER NO: 211419104233]***

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

IN

# COMPUTER SCIENCE AND ENGINEERING



PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.

ANNA UNIVERSITY: CHENNAI 600 025

##### APRIL 2023

**BONAFIDE CERTIFICATE**

Certified that this project report **“DETECT PROFESSIONAL MALICIOUS USER WITH METRIC LEARNING IN RECOMMENDER SYSTEMS”** is the bonafide work of “**VIKRAMAN.GD(211419104305) ,RAMAKUMARAN.B (211419104214), SANJAY SELVAN.R ( 211419104233*)*”** who carried out the project work under supervision.

**SIGNATURE SIGNATURE**

**Dr.L.JABASHEELA,M.E.,Ph.D. Mrs.A.Kanchana,M.E.**

**HEAD OF THE DEPARTMENT SUPERVISOR**

DEPARTMENT OF CSE, DEPARTMENT OF CSE,

PANIMALAR ENGINEERING COLLEGE, PANIMALAR ENGINEERING COLLEGE,

NASARATHPETTAI, NASARATHPETTAI,

POONAMALLEE, POONAMALLEE,

CHENNAI-600 123. CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on **11-04-2023.**

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**DECLARATION BY THE STUDENT**

We **VIKRAMAN.GD(211419104305), RAMAKUMARAN.B(211419104214), SANJAY SELVAN.R ( 211419104233*)*** hereby declare that this project report titled “DETECT PROFESSIONAL MALICIOUS USER WITH METRIC LEARNING IN RECOMMENDER SYSTEMS” , under the guidance of **Mrs.A.Kanchana** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

**VIKRAMAN G D**

**RAMAKUMARAN B**

**SANJAYSELVAN R**

**ACKNOWLEDGEMENT**

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our beloved Directors **Tmt.C.VIJAYARAJESWARI**, **Dr.C.SAKTHI KUMAR,M.E.,Ph.D** and **Dr.SARANYASREE SAKTHI KUMAR B.E.,M.B.A.,Ph.D.,** for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. L.JABASHEELA , M.E.,Ph.D.** for the support extended throughout the project.

We would like to thank my **Project Guide- Mrs.A.Kanchana,M.E.** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

Add ACKNOWLEDGEMENT TO OTHERS WHO HELPED IN YOUR PROJECT

You are allowed to change the wordings as per your requirement

**NAME OF THE STUDENTS**

**VIKRAMAN G D(211419104305)**

**RAMAKUMARAN B(211419104214)**

**SANJAYSELVAN R(211419104233)**

**ABSTRACT**

In e-commerce, online retailers are usually suffering from professional malicious users (PMUS), who utilize negative reviews and low ratings to their consumed products on purpose to threaten the retailers for illegal profits. PMUS are difficult to be detected because they utilize masking strategies to disguise themselves as normal users. To this end, we propose an unsupervised multi-modal learning model: MMD, which employs Metric learning for professional Malicious users Detection with both ratings and reviews. MMD first utilizes a modified RNN to project the informational review into a sentiment score, which jointly considers the ratings and reviews. The extensive experiments in four datasets demonstrate that our proposed method can solve this unsupervised detection problem. Moreover, the performance of the state-of-the-art recommender models is enhanced by taking MMD as a preprocessing stage.

TABLE OF CONTENTS

| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
| --- | --- | --- |
|  | **ABSTRACT** | iii |
|  | **LIST OF TABLES** | vii |
|  | **LIST OF FIGURES** | viii |
|  | **LIST OF SYMBOLS, ABBREVIATIONS** | ix |
| **1.** | **INTRODUCTION** |  |
|  | 1.1 Organization Profile ( optional) |  |
|  | 1.2 Problem Definition |  |
| **2.** | **LITERATURE SURVEY** |  |
| **3.** | **SYSTEM ANALYSIS** |  |
|  | 2.1 Existing System |  |
|  | 2.2 Proposed system |  |
|  | 2.3 Feasibility Study |  |
|  | 2.4 Hardware Environment |  |
|  | 2.5 Software Environment |  |
| **4.** | **SYSTEM DESIGN** |  |
|  | 4.1. ER diagram |  |
|  | 4.2 Data dictionary |  |
|  | 4.3 Table Normalization |  |
|  | 4.4 Data Flow Diagram |  |
|  | 4.5 UML Diagrams |  |
| **5.** | **SYSTEM ARCHITECTURE** |  |
|  | 5.1 Module Design Specification (Description of the Modules) |  |
|  | 5.2 Algorithms |  |
| **6.** | **SYSTEM IMPLEMENTATION** |  |
|  | 6.1 Client-side coding (sample) |  |
|  | 6.2 Server-side coding(sample) |  |
| **7.** | **SYSTEM TESTING ( for Application Project) / PERFORMANCE ANALYSIS ( for IEEE Project)** |  |
|  | 7.1 Unit Testing |  |
|  | 7.2 Integration Testing |  |
|  | 7.3 Test Cases & Reports |  |
| **8.** | **CONCLUSION** |  |
|  | 8.1 Results & Discussion |  |
|  | 8.2 Conclusion and Future Enhancements |  |
|  | **APPENDICES** |  |
|  | A.1 Sample Screens |  |
|  | A.2 Publications  A.3 Plagiarism Report |  |
|  | **REFERENCES** |  |

LIST OF FIGURES

Fig 1.1 Process of Machine Learning 8

Fig 1.2 Process of dataflow diagram 23

Fig 3.1 ER Diagram 26

Fig 3.2 Data Work Flow Diagram 27

Fig 3.3 Use Case Diagram 28

Fig 3.4 Class Diagram 29

Fig 3.5 Activity Diagram 30

Fig 3.6 Sequence Diagram 31

Fig 5.1 – 5.3 Module Diagram I , II, III 35 - 37

Fig 5.4 HTML Representation 46

Fig 5.5 Accuracy of Logistic Regression 58

Fig 5.6 Accuracy of Decision tree classifier 60

Fig 5.7 Accuracy of Random forest classifier 61

Fig 5.8 Accuracy of Support vector machine 63

Fig 7.1 Classification report 74

Fig 8.1 Output page 1 77

Fig 8.2 Output page 2 78

LIST OF ABBREVATIONS

GB - Gigabyte

ERD - Entity Relationship Diagram

API – Application programming interface

URL – Uniform Resource Locater

[str](https://docs.python.org/3/library/stdtypes.html#str) – String

func – Function

HTML – HyperText Markup Language

Css – Cascading Styling Sheets

FP - False Positives

FN - False Negatives

TP - True Positives

TN - True Negatives

**1 . INTRODUCTION**

E- COMMERCE giants, such as Amazon, Jingdong, and Alibaba, have been thriving with the development of Internet technology, where millions of electronic retailers produce great wealth through selling commodities on the websites . For each day, billions of trades occur between retailers and consumers . For the sake of improving the consumers’ experience of online shopping, e-commerce websites usually allow consumers (we call them “users”) to leave reviews and rank ratings on the commodities (we call them “items”). To trade off the interests between retailers and users, e-commerce websites punish the retailers who receive a high percentage of negative reviews and low ratings from users. Being widely applied in almost all kinds of e-commerce websites, this feedback mechanism has been proved to be effective if all the users leave truthful and objective reviews or ratings. However, in practice, there exist some malicious users (MU), who leverage this feedback mechanism to gain illegal profits . For example, these malicious users first purposefully leave negative reviews and low ratings of their consumed products without any consideration of the commodities’ quality. Then they blackmail the electronic retailers to make illegal profits; otherwise, they would leave more negative feedbacks, cheating e-commerce websites to punish the electronic retailers and confuse the normal users about the items in recommendations. As a result, these malicious users undermine the fairness of e-commerce. Moreover, their negative feedbacks will confuse the recommender systems (collaborative filtering-based models or content based models), leading to a chaotic recommendation for normal users, which is also named as shilling attacks. To address the above issues, e-commerce companies usually employ statistic outlier detection or shilling attack detection models to detect MUs, i.e., finding objective users who always give negative reviews or low ratings. However, there are some restrictions for these detection models: first, these models only tackle this problem from a methodological perspective and ignore the real-world scenarios. For example, most detection models ignore that there are some professional malicious users (PMUs), who can utilize masking strategies to avoid detection; second, they usually focus on filtering either fake ratings to improve recommendation models, or negative reviews for contentbased models, which do not take both ratings and reviews into account. As a result, these models may be applied in limited application scenarios in recommender systems, but not proper for professional malicious user detections.

**PROBLEM DEFINITION :**

To tackle the professional malicious user detection, we propose MMD, an unsupervised learning model, which applies metric learning and deep learning with both reviews and ratings. The framework of MMD. At the beginning, MMD utilizes one-hot encoding to select user u and item i, then projects them into PU and qi. Latent factor model (LFM), which is the most widely used model in recommender area, is employed to get PU and qi, and calculates rating score with them. As we define professional malicious users in recommender systems, malicious user detection is a new problem, which is an issue with little attention yet. However, we can treat this detection issue as a special case of abnormal user detection, and some existing works in this area can inspire us .In e-commerce, various abnormal users (spammers, shilling group, and frauds) have greatly damaged the systems, and some abnormal user detection models are proposed to tackle this issue.

**2 . LITERATURE SURVEY**

**PAPER 1:**

**TITLE:** Influence Function based Data Poisoning Attacks to Top-N Recommender Systems

**AUTHOR:** Minghong Fang, Neil Zhenqiang Gong and Jia Liu

**YEAR:** 2020

**ABSTRACT:**

Recommender system is an essential component of web services to engage users. Popular recommender systems model user preferences and item properties using a large amount of crowd sourced user-item interaction data, e.g., rating scores; then top-N items that match the best with a user’s preference are recommended to the user. In this work, we show that an attacker can launch a data poisoning attack to a recommender system to make recommendations as the attacker desires via injecting fake users with carefully crafted user-item interaction data. Specifically, an attacker can trick a recommender system to recommend a target item to as many normal users as possible. We focus on matrix factorization based recommender systems because they have been widely deployed in industry. Given the number of fake users the attacker can inject, we formulate the crafting of rating scores for the fake users as an optimization problem. However, this optimization problem is challenging to solve as it is a non-convex integer programming problem. To address the challenge, we develop several techniques to approximately solve the optimization problem. For instance, we leverage influence function to select a subset of normal users who are influential to the recommendations and solve our formulated optimization problem based on these influential users. Our results show that our attacks are effective and outperform existing methods.

**PAPER 2:**

**TITLE:** Fast Matrix Factorization With Nonuniform Weights on Missing Data

**AUTHOR:**  Xiangnan He and Jinzhou Tang ,

**YEAR:** 2019

**ABSTRACT:**

Matrix factorization (MF) has been widely used to discover the low-rank structure and to predict the missing entries of data matrix. In many real-world learning systems, the data matrix can be very high dimensional but sparse. This poses an imbalanced learning problem since the scale of missing entries is usually much larger than that of the observed entries, but they cannot be ignored due to the valuable negative signal. For efficiency concern, existing work typically applies a uniform weight on missing entries to allow a fast learning algorithm. However, this simplification will decrease modeling fidelity, resulting in suboptimal performance for downstream applications. In this paper, we weight the missing data nonuniformly, and more generically, we allow any weighting strategy on the missing data. To address the efficiency challenge, we propose a fast learning method, for which the time complexity is determined by the number of observed entries in the data matrix rather than the matrix size. The key idea is twofold: 1) we apply truncated singular value decomposition on the weight matrix to get a more compact representation of the weights and 2) we learn MF parameters with elementwise alternating least squares (eALS) and memorize the key intermediate variables to avoid repeating computations that are unnecessary. We conduct extensive experiments on two recommendation benchmarks, demonstrating the correctness, efficiency, and effectiveness of our fast eALS method

**PAPER 3:**

**TITLE:** Detecting shilling attacks in social recommender systems based on time series analysis and trust features

**AUTHOR:** Yishu Xu and Fuzhi Zhang

**YEAR:** 2019

**ABSTRACT:**

In social recommender systems or trust-based recommender systems, malicious users can bias the recommendations by injecting a large number of fake profiles and by building bogus trust relationships. The existing shilling attack detection methods suffer from low precision when detecting attacks in social recommender systems because they focus mainly on the rating pattern differences between attack profiles and genuine ones and ignore the trust relationships between users. In this paper, we propose an approach for detecting shilling attacks in social recommender systems based on time series analysis and trust features (TSA–TF). Firstly, we construct rating distribution time series for items and propose a dynamic rating distribution prediction model to detect suspicious items by using a single exponential smoothing method. Then, we filter out a part of genuine user profiles by analyzing suspicious items and obtain the set of suspicious user profiles. Secondly, we propose four features by combining rating patterns and trust relationships and train a support vector machine (SVM) classifier to discriminate attack profiles in the set of suspicious user profiles. Experiments on the CiaoDVD dataset and Epinions dataset show that the proposed approach can improve the detection precision while maintaining a high recall.

**PAPER 4:**

**TITLE:** Exploiting the sentimental bias between ratings and reviews with neural networks

**AUTHOR:** Yuanbo Xu , Yongjian Yang , Jiayu Han , En Wang , Fuzhen Zhuang and Yang d

**YEAR:** 2019

**ABSTRACT:**

Traditional recommender systems rely on user profiling based on either user ratings or reviews through bi-sentimental analysis. However, in real-world scenarios, there are two common phenomena users only provide ratings for items but without detailed review comments. As a result, the historical transaction data available for recommender systems are usually unbalanced and sparse in many cases, users opinions can be better grasped in their reviews than ratings. For the reason that there is always a bias between ratings and reviews, it is really important that users ratings and reviews should be mutually reinforced to grasp the users’ true opinions. To this end, in this paper, we develop an opinion mining model based on convolutional neural networks for enhancing recommendation. Specifically, we exploit two-step training neural networks, which utilize both reviews and ratings to grasp users’ true opinions in unbalanced data. Moreover, we propose a Sentiment Classification scoring (SC) method, which employs dual attention vectors to predict the users’ sentiment scores of their reviews rather than using bisentiment analysis. Next, a combination function is designed to use the results of SC and user–item rating matrix to catch the opinion bias. It can filter the reviews and users, and build an enhanced user– item matrix. Finally, a Multilayer perceptron based Matrix Factorization (MMF) method is proposed to make recommendations with the enhanced user–item matrix. Extensive experiments on several realworld datasets (Yelp, Amazon, Taobao and Jingdong) demonstrate that our approach can achieve a superior performance over state-of-the-art baselines; our approach is able to tackle unbalanced data and achieve stable performances.

**PAPER 5:**

**TITLE:** Characterizing and Detecting Malicious Accounts in Privacy-Centric Mobile Social Networks

**AUTHOR:** Zenghua Xia, Chang Liu and Yong Cui

**YEAR:** 2019

**ABSTRACT:**

Malicious accounts are one of the biggest threats to the security and privacy of online social networks (OSNs). In this work, we study a new type of OSN, called privacy-centric mobile social network (PCMSN), such as KakaoTalk and LINE, which has attracted billions of users recently. The design of PC-MSN is inspired to protect their users’ privacy from strangers: a stranger is not easy to send a friend request to a user who does not want to make friends with strangers; and strangers cannot view a user’s post. Such a design mitigates the security issue of malicious accounts. At the same time, it also brings the battleground between attackers and defenders to an earlier stage, i.e., making friendship, than the one studied in previous works. Also, previous defense proposals mostly rely on certain assumptions on the attacker, which may not be robust in the new PC-MSNs. As a result, previous malicious accounts detection approaches are less effective on a PC-MSN. To mitigate this issue, we study the patterns in friend requests to distinguish malicious accounts, and perform a systematic study over 1 million labeled data from WLink, a real PC-MSN with billions of users, to confirm our hypothesis. Based on the results, we propose dozens of new features and leverage machine learning to detect malicious accounts. We evaluate our method and compare it with existing methods, and the results show that our method achieves a precision of 99.5% and a recall of 98.4%, which significantly outperform previous state-of-the-art methods.

**3 . SYSTEM ANALYSIS**

**3.1 Existing System**

As we define professional malicious users in recommender systems, malicious user detection is a new problem, which is an issue with little attention yet. However, we can treat this detection issue as a special case of abnormal user detection, and some existing works in this area can inspire us. In e-commerce, various abnormal users have greatly damaged the systems, and some abnormal user detection models are proposed to tackle this issue. As we define professional malicious users in recommender systems, malicious user detection is a new problem, which is an issue with little attention yet. However, we can treat this detection issue as a special case of abnormal user detection, and some existing works in this area can inspire us. In e-commerce, various abnormal users have greatly damaged the systems, and some abnormal user detection models are proposed to tackle this issue.

**DISADVANTAGE:**

* Not utilize the abundant information hidden in reviews to detect professional malicious users.
* There is a noticeable sentiment gap between ratings and reviews for professional malicious users, while for normal users, the sentiment gap is small.

**3.2 Proposed system**

This is the first work focusing on solving the professional malicious user detection issue utilizing both users’ ratings and reviews to enhance the state-of the-art recommender systems. A novel multi-modal unsupervised method-MMD-is proposed to detect professional malicious users with the modified RNN and attention metric learning based clustering. Extensive experiments are conducted on four real world e-commerce datasets to verify our proposed method. Moreover, by filtering professional malicious users, some state-of-the-art models are enhanced.

**ADVANTAGE:**

* Detect malicious users with different levels of information, one needs to be able to detect malicious users even when there is minimum information available.
* Humans detect malicious users on social media by their characteristics

**3.4 HARDWARE ENVIRONMENT**

**H/W SYSTEM CONFIGURATION:**

* Processor – I3, i5,i7
* RAM - 8 Gb
* Hard Disk - 500 GB

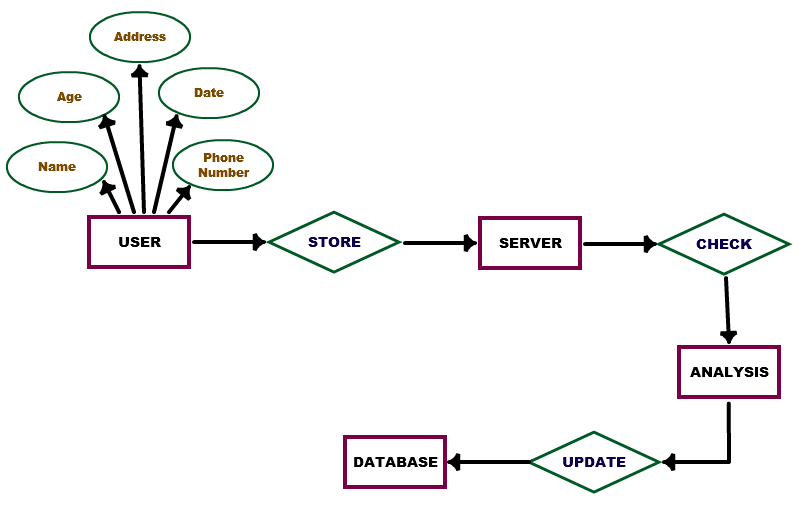
**3.5 SOFTWARE ENVIRONMENT**

**S/W SYSTEM CONFIGURATION:**

* Operating System - Windows 7/8/10
* Front End - Html,Css
* Scripts – python language
* Tool – Python idle

**4 . SYSTEM DESIGN**

**4.1.** **ER diagram**



**4.2 Data flow diagram**



**Data Flow Diagram :**

Data Flow Diagram --1:

User Interface

Mobile App web portal

User

Request

v response

Data Flow Diagram --2:

**Rating based evidences:**

App’s database

App Rating

User

Voting store in database

v

Data Flow Diagram --3:

**Review Based Evidences:**

App’s database

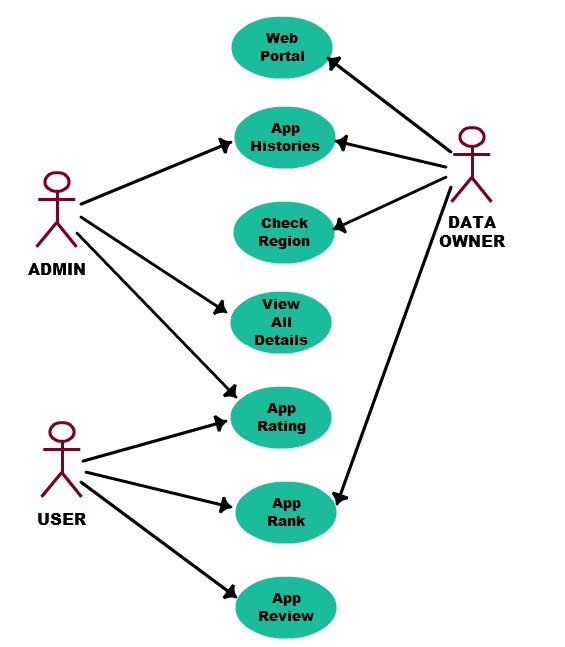
App Review

User

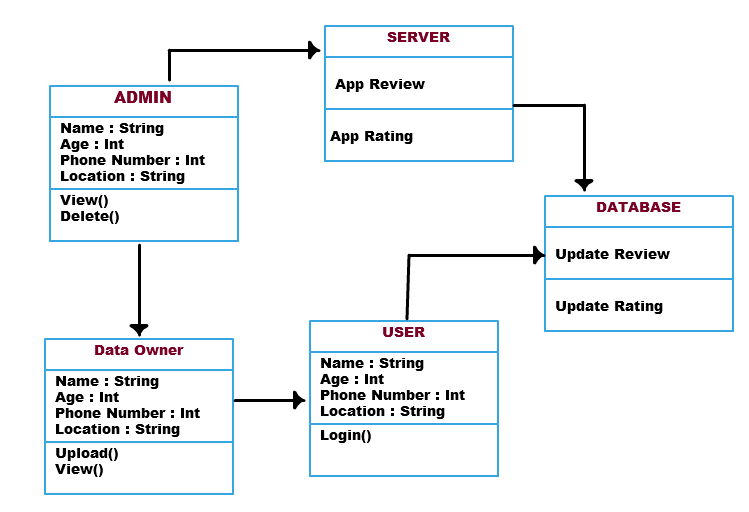
Voting store in database

v

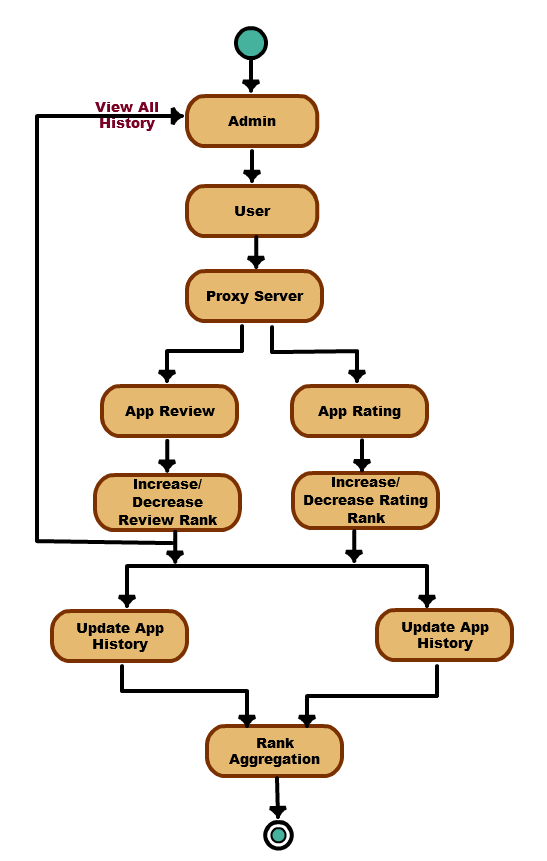
**4.3 USE CASE DIAGRAM**



**4.4 CLASS DIAGRAM :**



**4.5 ACTIVITY DIAGRAM**



**5 . SYSTEM ARCHITECTURE**

**5.1 MODULE DESCRIPTION:**

**MODULE DESCRIPTION:**

* **GEOLOCATION BASED TWEETS**

In this modules, specific applications of a real-time, country-level tweet geolocation system. Our methodology enables us to perform a thorough analysis of tweet geolocation, revealing insights into the best approaches for an accurate country-level location classifier for geolocation based tweets.

* **PHONY NEWS CLASSIFIER**

In this module, the user analyze the tweet’s nature like positive negative or neutral by analyzing the containing words in the tweet and the each word we use in a regular basis comprises of words of positive and negative as well as neutral words. These words are initially inserted into the database for analysis.

* **SENTIWORD CALL**

In this module the data to be analyzed for analysis is gets the score by calling the sentiword file because of the generation of score for individual words and this score is used for generation of analysis result

* **ANALYSIS RESULT COMPARISON**

The final score is generated by the result of the sentiword and this is useful in comparison of scores and this comparison results are kept some threshold to state the tweet as negative, positive or neutral one.

## INTRODUCTION:

Ranking fraud in the mobile App market rulers to fraudulent or deceptive activities which have a purpose of bumping up the Apps in the popularity list. Indeed, it becomes more and more frequent for App developers to user shady means, such as inflating their Apps’ sales or’ posting phony App ratings, to commit ranking fraud. While the

importance of preventing ranking fraud has been widely recognized, there is limited understand and research in this area. To this end, in this paper, we used Rank Aggregation Algorithm for better way to Evaluate the Ranking system .

Specifically, we first propose to accurately locate the ranking fraud by active periods, namely leading sessions, of mobile Apps. Such leading sessions can be leveraged for detecting the local anomaly instead of global anomaly of App rankings.

Furthermore, we investigate three types of evidences, i .é. Ranking based evidences, Rating based evidences and review based evidences, by modeling Apps’ ranking, rating and review behaviors through statistical hypotheses tests. In addition, we propose an optimization based aggregation method to integrate all the evidences for fraud detection.

CHAPTER 3

**REQUIREMENTS ENGINEERING:**

**GENERAL**

In this Paper we are developing the P2P infrastructure, all the nodes are interlinked with each other so the all the nodes are need the best configuration.

**HARDWARE REQUIREMEN:**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shouls what the system do and not how it should be implemented.

**3.3 SOFTWARE REQUIREMENTS**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

CHAPTER 5

**DEVELOPMENT TOOLS:**

**GENERAL**

This chapter is about the software language and the tools used in the development of the project. The platform used here is JAVA. The Primary languages are JAVA,J2EE and J2ME. In this project J2EE is chosen for implementation.

**FEATURES OF JAVA**

**THE JAVA FRAMEWORK**

Java is a programming language originally developed by James Gosling at Microsystems and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is general-purpose, concurrent, class-based, and object-oriented, and is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere".

Java is considered by many as one of the most influential programming languages of the 20th century, and is widely used from application software to web applications the java framework is a new platform independent that simplifies application development internet. Java technology's versatility, efficiency, platform portability, and security make it the ideal technology for network computing. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere!

**OBJECTIVES OF JAVA**

To see places of Java in Action in our daily life, explore java.com.

## Why Software Developers Choose Java

Java has been tested, refined, extended, and proven by a dedicated community. And numbering more than 6.5 million developers, it's the largest and most active on the planet. With its versatility, efficiency, and portability, Java has become invaluable to developers by enabling them to:

* Write software on one platform and run it on virtually any other platform
* Create programs to run within a Web browser and Web services
* Develop server-side applications for online forums, stores, polls, HTML forms processing, and more
* Combine applications or services using the Java language to create highly customized applications or services
* Write powerful and efficient applications for mobile phones, remote processors, low-cost consumer products, and practically any other device with a digital heartbeat Some Ways Software Developers Learn Java

Today, many colleges and universities offer courses in programming for the Java platform. In addition, developers can also enhance their Java programming skills by reading Sun's java.sun.com Web site, subscribing to Java technology-focused newsletters, using the Java Tutorial and the New to Java Programming Center, and signing up for Web, virtual, or instructor-led courses.

Object-oriented

To be an Object Oriented language, any language must follow at least the four characteristics.

1. Inheritance   :It is the process of creating the new classes and using the behavior of the existing classes by extending them just to reuse  the existing code and adding addition a features as needed.

2. Encapsulation: It is the mechanism of combining the information and providing the abstraction.

3. Polymorphism: As the name suggest one name multiple form, Polymorphism is the way of providing the different functionality by the functions having the same name based on the signatures of the  methods.

4. Dynamic binding: Sometimes we don't have the knowledge of objects about their specific types while writing our code. It is the way of providing the maximum functionality to a program about the specific type at runtime.

**JAVA SWING OVERVIEW**

Abstract Window Toolkit (AWT) is cross-platform

Swing provides many controls and widgets to build user interfaces with. Swing class names typically begin with a J such as JButton, JList, JFrame. This is mainly to differentiate them from their AWT counterparts and in general is one-to-one replacements. Swing is built on the concept of Lightweight components vs AWT and SWT's concept of Heavyweight components. The difference between the two is that the Lightweight components are rendered (drawn) using purely Java code, such as drawLine and drawImage, whereas Heavyweight components use the native operating system to render the components.

Some components in Swing are actually heavyweight components. The top-level classes and any derived from them are heavyweight as they extend the AWT versions. This is needed because at the root of the UI, the parent windows need to be provided by the OS. These top-level classes include JWindow, JFrame, JDialog and JApplet. All Swing components to be rendered to the screen must be able to trace their way to a root window of one of those classes.

Note: It generally it is not a good idea to mix heavyweight components with lightweight components (other than as previously mentioned) as you will encounter layering issues, e.g., a lightweight component that should appear "on top" ends up being obscured by a heavyweight component. The few exceptions to this include using heavyweight components as the root pane and for popup windows. Generally speaking, heavyweight components will render on top of lightweight components and will not be consistent with the look and feel being used in Swing. There are exceptions, but that is an advanced topic. The truly adventurous may want to consider reading this article from Sun on mixing heavyweight and lightweight components.

**Evolution of Collection Framework:**

Almost all collections in Java are derived from the java.util.Collection interface. Collection defines the basic parts of all collections. The interface states the add() and remove() methods for adding to and removing from a collection respectively. Also required is the toArray() method, which converts the collection into a simple array of all the elements in the collection. Finally, the contains() method checks if a specified element is in the collection. The Collection interface is a subinterface of java.util.Iterable, so the iterator() method is also provided. All collections have an iterator that goes through all of the elements in the collection. Additionally, Collection is a generic. Any collection can be written to store any class. For example, Collection<String> can hold strings, and the elements from the collection can be used as strings without any casting required.

There are three main types of collections:

* Lists: always ordered, may contain duplicates and can be handled the same way as usual arrays
* Sets: cannot contain duplicates and provide random access to their elements
* Maps: connect unique keys with values, provide random access to its keys and may host duplicate values

**LIST**

Lists are implemented in the JCF via the java.util.List interface. It defines a list as essentially a more flexible version of an array. Elements have a specific order, and duplicate elements are allowed. Elements can be placed in a specific position. They can also be searched for within the list. Two concrete classes implement List. The first is java.util.ArrayList, which implements the list as an array. Whenever functions specific to a list are required, the class moves the elements around within the array in order to do it. The other implementation is java.util.LinkedList. This class stores the elements in nodes that each have a pointer to the previous and next nodes in the list. The list can be traversed by following the pointers, and elements can be added or removed simply by changing the pointers around to place the node in its proper place.

**SET:**

Java's java.util.Set interface defines the set. A set can't have any duplicate elements in it. Additionally, the set has no set order. As such, elements can't be found by index. Set is implemented by java.util.HashSet,java.util.LinkedHashSet, and java.util.TreeSet. HashSet uses a hash table. More specifically, it uses a java.util.HashMap to store the hashes and elements and to prevent duplicates. Java.util.LinkedHashSet extends this by creating a doubly linked list that links all of the elements by their insertion order. This ensures that the iteration order over the set is predictable. java.util.TreeSet uses a red-black tree implemented by a java.util.TreeMap. The red-black tree makes sure that there are no duplicates. Additionally, it allows Tree Set to implement java.util.SortedSet.

The java.util.Set interface is extended by the java.util.SortedSet interface. Unlike a regular set, the elements in a sorted set are sorted, either by the element's compareTo() method, or a method provided to the constructor of the sorted set. The first and last elements of the sorted set can be retrieved, and subsets can be created via minimum and maximum values, as well as beginning or ending at the beginning or ending of the sorted set. The SortedSet interface is implemented by java.util.TreeSet

Java.util.SortedSet is extended further via the java.util.NavigableSet interface. It's similar to SortedSet, but there are a few additional methods. The floor(), ceiling(), lower(), and higher() methods find an element in the set that's close to the parameter. Additionally, a descending iterator over the items in the set is provided. As with SortedSet, java.util.TreeSet implements NavigableSet.

**MAP:**

Maps are defined by the java.util.Map interface in Java. Maps are simple data structures that associate a key with a value. The element is the value. This lets the map be very flexible. If the key is the hash code of the element, the map is essentially a set. If it's just an increasing number, it becomes a list. Maps are implemented by java.util.HashMap, java.util.LinkedHashMap, and java.util.TreeMap. HashMap uses a hash table. The hashes of the keys are used to find the values in various buckets. LinkedHashMap extends this by creating a doubly linked list between the elements. This allows the elements to be accessed in the order in which they were inserted into the map. TreeMap, in contrast to HashMap and LinkedHashMap, uses a red-black tree. The keys are used as the values for the nodes in the tree, and the nodes point to the values in the map

**Thread:**

Simply put, a thread *i*s a program's path of execution. Most programs written today run as a single thread, causing problems when multiple events or actions need to occur at the same time. Let's say, for example, a program is not capable of drawing pictures while reading keystrokes. The program must give its full attention to the keyboard input lacking the ability to handle more than one event at a time. The ideal solution to this problem is the seamless execution of two or more sections of a program at the same time.

## Creating threads

Java's creators have graciously designed two ways of creating threads: implementing an interface and extending a class. Extending a class is the way Java inherits methods and variables from a parent class. In this case, one can only extend or inherit from a single parent class. This limitation within Java can be overcome by implementing interfaces, which is the most common way to create threads. (Note that the act of inheriting merely allows the class to be run as a thread. It is up to the class to start() execution, etc.)

Interfaces provide a way for programmers to lay the groundwork of a class. They are used to design the requirements for a set of classes to implement. The interface sets everything up, and the class or classes that implement the interface do all the work. The different set of classes that implement the interface have to follow the same rules.

**The Java Platform**

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

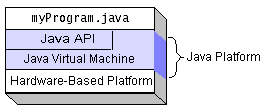
The Java platform has two components:

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

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

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

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



Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

**What Can Java Technology Do?**

The most common types of programs written in the Java programming language are applets and applications. If you’ve surfed the Web, you’re probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

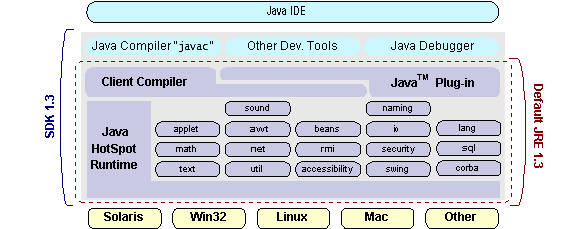
However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

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

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

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

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



**How Will Java Technology Change My Life?**

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

* **Get Started Quickly**: Although the Java programming language is a powerful object-oriented language, it’s easy to learn, especially for programmers already familiar with C or C++.
* **Write Less Code**: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
* **Write Better Code**: The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people’s tested code and introduce fewer bugs.
* **Develop Programs More Quickly**: Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
* **Avoid Platform Dependencies With 100% Pure Java**: You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure JavaTM Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.
* **Write Once, Run Anywhere**: Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.
* **Distribute Software More Easily**: You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded “on the fly,” without recompiling the entire program.

**ODBC**

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

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN. The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources.

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

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn’t as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

**JDBC**

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

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

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

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

**JDBC Goals**

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

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

1. **SQL Level API**

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

1. **SQL Conformance**

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

1. **JDBC must be implemental on top of common database interfaces**   
    The JDBC SQL API must “sit” on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.
2. **Provide a Java interface that is consistent with the rest of the Java system**

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

1. **Keep it simple**

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

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

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

1. **Keep the common cases simple**

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

Finally we decided to precede the implementation using JSP.

And for dynamically updating the cache table we go for MySQL database.

Java ha two things: a programming language and a platform.

Java is a high-level programming language that is all of the following

Simple Architecture-neutral

Object-oriented Portable

Distributed High-performance

Interpreted Multithreaded

Robust Dynamic

Secure

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

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

**Conclusion**

Swing's high level of flexibility is reflected in its inherent ability to override the native host operating system (OS)'s GUI controls for displaying itself. Swing "paints" its controls using the Java 2D APIs, rather than calling a native user interface toolkit. The Java thread scheduler is very simple. All threads have a priority value which can be changed dynamically by calls to the threads setPriority() method . Implementing the above concepts in our project to do the efficient work among the Server.

**SOFTWARE TESTING**

**GENERAL**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**DEVELOPING METHODOLOGIES**

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used.

The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

**Types of Tests**

**Unit testing**

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

Functional test

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

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

**System Test**

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

**Performance Test**

The Performance test ensures that the output is produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

**Integration Testing**

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

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

*Acceptance Testing*

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

**Acceptance testing for Data Synchronization:**

* The Acknowledge will be received by the Sender Node after the Packets are received by the Destination Node
* The Route add operation is done only when there is a Route request in need
* The Status of Nodes information is done automatically in the Cache Updating process

**Build the test plan**

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

**6 . SYSTEM IMPLEMENTATION**

**6.1 Client-side coding**

**User\_Register.java:**

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.Statement;

import java.util.Iterator;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import org.apache.tomcat.util.http.fileupload.FileItem;

import org.apache.tomcat.util.http.fileupload.FileUploadException;

import org.apache.tomcat.util.http.fileupload.disk.DiskFileItemFactory;

import org.apache.tomcat.util.http.fileupload.servlet.ServletFileUpload;

/\*\*

\*

\* @author mvinoth

\*/

@WebServlet(urlPatterns = {"/User\_register"})

public class User\_register extends HttpServlet {

/\*\*

\* Processes requests for both HTTP

\* <code>GET</code> and

\* <code>POST</code> methods.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

HttpSession session=request.getSession(true);

try {

String password="",F\_Name="",L\_Name="",U\_Ph="",gender="",file1="";

String saveFile="",l="",ln="",mname="",dob="",celno="",typ="";

String sex="",voterno="",email="",bg="",pstreet="",cstreet="",parea="",carea="",pcity="",ccity="";

String ppinno="",cpinno="",pass="",repass="";

int fileidnum=0,downloadcount=0,vc=0;

// String contentType = request.getContentType();

// Create a factory for disk-based file items

DiskFileItemFactory factory = new DiskFileItemFactory();

// Set factory constraints

factory.setSizeThreshold(4012);

//factory.setRepository("c:");

// Create a new file upload handler

ServletFileUpload upload = new ServletFileUpload(factory);

// Set overall request size constraint

//upload.setSizeMax(10024);

// Parse the request

List items = null;

try {

items = upload.parseRequest(request);

} catch (FileUploadException e) {

e.printStackTrace();

}

byte[] data = null;

String fileName = null;

// Process the uploaded items

Iterator iter = items.iterator();

while (iter.hasNext()) {

FileItem item = (FileItem) iter.next();

if (item.isFormField()) {

//processFormField(item);

String name = item.getFieldName();

String value = item.getString();

if (name.equalsIgnoreCase("email")) {

email = value;

System.out.println("email" + email);

}

else if (name.equalsIgnoreCase("password")) {

password = value;

System.out.println("password" + password);

}

else if (name.equalsIgnoreCase("F\_Name")) {

F\_Name = value;

System.out.println("F\_Name" + F\_Name);

}

else if (name.equalsIgnoreCase("L\_Name")) {

L\_Name = value;

System.out.println("L\_Name" + L\_Name);

}

else if (name.equalsIgnoreCase("U\_Ph")) {

U\_Ph = value;

System.out.println("U\_Ph" + U\_Ph);

}

else if (name.equalsIgnoreCase("gender")) {

gender = value;

System.out.println("gender" + gender);

}

else if (name.equalsIgnoreCase("dob")) {

dob = value;

System.out.println("Mail" + dob);

}

else {

System.out.println("ERROR");

}

} else {

data = item.get();

fileName = item.getName();

}

}

saveFile = fileName;

String path1 = request.getSession().getServletContext().getRealPath("/");

// String patt=path.replace("\\build", "");

String strPath1 = path1+"\\"+saveFile;

File ff1 = new File(strPath1);

FileOutputStream fileOut1 = new FileOutputStream(ff1);

fileOut1.write(data, 0, data.length);

fileOut1.flush();

fileOut1.close();

out.println(saveFile);

/////////////////////////////////////////////////////////

FileInputStream fis11 = null;

File image1 = null;

//FileInputStream fis11 = null;

File image11 = null;

Connection con7=null;

PreparedStatement st7=null;

PreparedStatement st11=null;

image1 = new File(strPath1);

fis11 = new FileInputStream(image1);

String dd="0";

Class.forName("com.mysql.jdbc.Driver");

con7 = DriverManager.getConnection("jdbc:mysql://localhost:3306/mobile\_apps","root","admin");

String sname=F\_Name+" "+L\_Name;

String query="Select \* from User\_register where Email\_Id='"+email+"'";

System.out.println(query);

Statement st=con7.createStatement();

ResultSet rs=st.executeQuery(query);

if(rs.next())

{

session.setAttribute("msg","Already exist Please Check Values");

response.sendRedirect("User\_Reg.jsp");

}

else

{

st7 =con7.prepareStatement("insert into User\_register values (?,?,?,?,?,?,?,?,?)");

st7.setInt(1,0);

st7.setString(2,email);

st7.setString(3,password);

st7.setString(4,sname);

st7.setString(5,gender);

st7.setString(6,U\_Ph);

st7.setString(7,dob);

st7.setString(9,dd);

if(fileName != "")

st7.setBinaryStream(8, (InputStream)fis11, (int)(image1.length()));

else

st7.setBinaryStream(8, null);

int i =st7.executeUpdate();

session.setAttribute("msg","Successfully Register");

response.sendRedirect("User\_Reg.jsp");

}

}

catch(Exception e)

{

out.println(e);

} finally {

out.close();

}

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP

\* <code>GET</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Handles the HTTP

\* <code>POST</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Returns a short description of the servlet.

\*

\* @return a String containing servlet description

\*/

@Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}

**Upload.java:**

import Connection.DbConnection;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.PrintWriter;

import java.net.InetAddress;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.Statement;

import java.text.SimpleDateFormat;

import java.util.Iterator;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import org.apache.commons.fileupload.FileItem;

import org.apache.commons.fileupload.FileUploadException;

import org.apache.commons.fileupload.disk.DiskFileItemFactory;

import org.apache.commons.fileupload.servlet.ServletFileUpload;

import org.apache.commons.io.FilenameUtils;

/\*\*

\*

\* @author mvinoth

\*/

@WebServlet(urlPatterns = {"/Upload"})

public class Upload extends HttpServlet {

/\*\*

\* Processes requests for both HTTP

\* <code>GET</code> and

\* <code>POST</code> methods.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

HttpSession session=request.getSession(true);

try {

byte[] data1 = null;

String apk=null;

String Apps=null,AppsD=null,AppsF=null,date1=null;

String saveFile="",sn="",un="";

int fileidnum=0,downloadcount=0,vc=0;

sn=(String)session.getAttribute("Sn");

un=(String)session.getAttribute("un");

int ud1=(Integer) session.getAttribute("id");

String mail = (String) session.getAttribute("mail");

String contentType = request.getContentType();

// Create a factory for disk-based file items

DiskFileItemFactory factory = new DiskFileItemFactory();

// Set factory constraints

factory.setSizeThreshold(4012);

//factory.setRepository("c:");

// Create a new file upload handler

ServletFileUpload upload = new ServletFileUpload(factory);

// Set overall request size constraint

//upload.setSizeMax(10024);

// Parse the request

List items = null;

try {

items = upload.parseRequest(request);

} catch (FileUploadException e) {

e.printStackTrace();

}

byte[] data = null;

String fileName = null;

// Process the uploaded items

Iterator iter = items.iterator();

while (iter.hasNext()) {

FileItem item = (FileItem) iter.next();

if (item.isFormField()) {

//processFormField(item);

String name = item.getFieldName();

String value = item.getString();

if(name.equals("Apps"))

{

Apps=value;

}

if(name.equals("AppsD"))

{

AppsD=value;

}

if(name.equals("AppsF"))

{

AppsF=value;

}

if(name.equals("date1"))

{

date1=value;

}

} else {

String ext = FilenameUtils.getExtension(item.getName());

if(ext.equals("apk"))

{

apk=item.getName();

data1 = item.get();

//fileName = item.getName();

String path1 = request.getSession().getServletContext().getRealPath("/");

// String patt=path.replace("\\build", "");

String strPath1 = path1+"Apk\_File\\"+apk;

out.println(strPath1);

File ff1 = new File(strPath1);

FileOutputStream fileOut1 = new FileOutputStream(ff1);

fileOut1.write(data1, 0, data1.length);

fileOut1.flush();

fileOut1.close();

}

else if(!ext.equals("apk"))

{

data = item.get();

fileName = item.getName();

System.out.println("fn:" + fileName);

}

}

}

String extension = "";

int i = fileName.lastIndexOf('.');

if (i > 0) {

extension = fileName.substring(i+1);

}

System.out.println("--"+extension);

saveFile = fileName;

String path = request.getSession().getServletContext().getRealPath("/");

System.out.println(path);

String strPath = path+"\\"+saveFile;

System.out.println(strPath);

File ff = new File(strPath);

FileOutputStream fileOut = new FileOutputStream(ff);

fileOut.write(data, 0, data.length);

fileOut.flush();

fileOut.close();

System.out.println(strPath);

System.out.println("Thrid");

Connection con = null;

PreparedStatement psmnt = null;

FileInputStream fis;

InputStream sImage;

java.util.Date now = new java.util.Date();

String DATE\_FORMAT = "yyyy-MM-dd";

SimpleDateFormat sdf = new SimpleDateFormat(DATE\_FORMAT);

String strDateNew = sdf.format(now);

System.out.println(strDateNew);

String ip1=InetAddress.getLocalHost().getHostAddress();

try {

DbConnection Db=new DbConnection();

File f = new File(strPath);

double cost=0.0;

long length = f.length();

if(String.valueOf(session.getAttribute("Sn")).compareToIgnoreCase("Not")!=0)

cost=(length\*(0.02/100));

else

cost=(length\*(0.04/100));

System.out.println("length " + length);

String ip=request.getRemoteAddr();int id=0;

ResultSet rs=Db.Select("select max(Fid) from upload");

if(rs.next())

{

id=rs.getInt(1);

}

id=id+1;

con=Db.con;

int id1=id;

String dd="select \* from Upload where Filename='"+saveFile+"' or Appl\_name='"+Apps+"' ";

System.out.println(dd+","+apk);

Statement st=con.createStatement();

ResultSet rs2=st.executeQuery(dd);

if(rs2.next())

{

if(rs2.getString("Filename").equalsIgnoreCase(saveFile))

{

session.setAttribute("msg", "File Name Already Exit !!!");

session.setAttribute("color", "red");

}

if(rs2.getString("Appl\_name").equalsIgnoreCase(Apps))

{

session.setAttribute("msg","Application name Already Exit !!!");

session.setAttribute("color", "red");

// response.sendRedirect("Upload\_Application.jsp");

}

response.sendRedirect("Upload\_Application.jsp");

}

else

{

String queryString = "insert into Upload(Un,Email, Fid, Filename, Extension,Appl\_name,Content,App\_dec,Date,App\_F,req,Apk\_File) values (?,?,?,?,?,?,?,?,?,?,?,?)";

System.out.println("four" + un);

psmnt = con.prepareStatement(queryString);

fis = new FileInputStream(f);

psmnt.setInt(1, ud1);

psmnt.setString(2, mail);

psmnt.setInt(3, id1);

psmnt.setString(4, saveFile);

psmnt.setString(5, extension);

psmnt.setString(6,Apps);

psmnt.setBinaryStream(7, (InputStream) fis, (int) (f.length()));

psmnt.setString(8,AppsD);

psmnt.setString(9, strDateNew);

psmnt.setString(10, AppsF);

psmnt.setString(11, "");

psmnt.setString(12, apk);

int s = psmnt.executeUpdate();

int Flength=0,Dcount=0;

if(s>0)

{

if(String.valueOf(session.getAttribute("Sn")).compareToIgnoreCase("Not")!=0)

session.setAttribute("msg", "'"+Apps+"' is Successfully Stored");

else

session.setAttribute("msg", "'"+Apps+"' is Successfully Stored");

session.setAttribute("color", "green");

response.sendRedirect("Upload\_Application.jsp");

}

else

{

session.setAttribute("msg", "Failed !!!");

session.setAttribute("color", "red");

response.sendRedirect("Upload\_Application.jsp");

}

}

} catch (Exception ex) {

ex.printStackTrace();

}

} catch (Exception e) {

e.printStackTrace();

}

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP

\* <code>GET</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Handles the HTTP

\* <code>POST</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Returns a short description of the servlet.

\*

\* @return a String containing servlet description

\*/

@Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}

**Login\_check.java:**

import Connection.DbConnection;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.ResultSet;

import java.sql.SQLException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

/\*\*

\*

\* @author mvinoth

\*/

@WebServlet(urlPatterns = {"/logincheck\_a"})

public class logincheck\_a extends HttpServlet {

/\*\*

\* Processes requests for both HTTP

\* <code>GET</code> and

\* <code>POST</code> methods.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

HttpSession session=request.getSession(true);

try {

String name=request.getParameter("A\_Name");

String pass=request.getParameter("A\_Pass");

if(name!=null && pass!=null)

{

if(name.equals("admin")&&pass.equals("admin"))

{

int id=1;

session.setAttribute("name", name);

session.setAttribute("mail", pass);

session.setAttribute("id", id);

session.setAttribute("msg", "Successfully Login");

String ff="1";

response.sendRedirect("Admin\_Home.jsp");

}

else

{

String dd= "Invalid Username and Password";

session.setAttribute("msg", dd);

response.sendRedirect("Admin.jsp");

}

}

}finally {

out.close();

}

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP

\* <code>GET</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Handles the HTTP

\* <code>POST</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Returns a short description of the servlet.

\*

\* @return a String containing servlet description

\*/

@Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}

**Download.java:**

import Connection.DbConnection;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.SimpleDateFormat;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

/\*\*

\*

\* @author Dhamo

\*/

@WebServlet(urlPatterns = {"/downlaod"})

public class downlaod extends HttpServlet {

/\*\*

\* Processes requests for both HTTP

\* <code>GET</code> and

\* <code>POST</code> methods.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

HttpSession session=request.getSession(true);

try

{

String name = (String) session.getAttribute("name");

String id1=request.getParameter("id1");

String name1=request.getParameter("name");

int count=1;

//response.sendRedirect("http://localhost:8084/Moblie\_Apps/Apk\_File/"+id1);

DbConnection db=new DbConnection();

String ss1="select \* from download where user\_id='"+name+"'";

System.out.println(ss1);

ResultSet rs1=db.Select(ss1);

int i=0;

try {

if(rs1.next())

{

i=i+1;

}

String str="select max(count\_id) as n from download where application\_id='"+name1+"'";

ResultSet rs=db.Select(str);

try {

if(rs.next())

{

String c=rs.getString("n");

count=count+Integer.parseInt(c);

if(i>0)

{

count=0;

count=Integer.parseInt(c);

}

}

} catch (SQLException ex) {

Logger.getLogger(downlaod.class.getName()).log(Level.SEVERE, null, ex);

}

java.util.Date now = new java.util.Date();

String DATE\_FORMAT = "yyyy-MM-dd";

SimpleDateFormat sdf = new SimpleDateFormat(DATE\_FORMAT);

String strDateNew = sdf.format(now);

System.out.println(strDateNew);

String ss="insert into download values('"+name1+"','"+count+"','"+strDateNew+"','"+name+"')";

int ij=db.Insert(ss);

response.sendRedirect("http://localhost:8084/Moblie\_Apps/Apk\_File/"+id1);

} catch (Exception ex) {

Logger.getLogger(downlaod.class.getName()).log(Level.SEVERE, null, ex);

}

} finally {

out.close();

}

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP

\* <code>GET</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Handles the HTTP

\* <code>POST</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Returns a short description of the servlet.

\*

\* @return a String containing servlet description

\*/

@Override

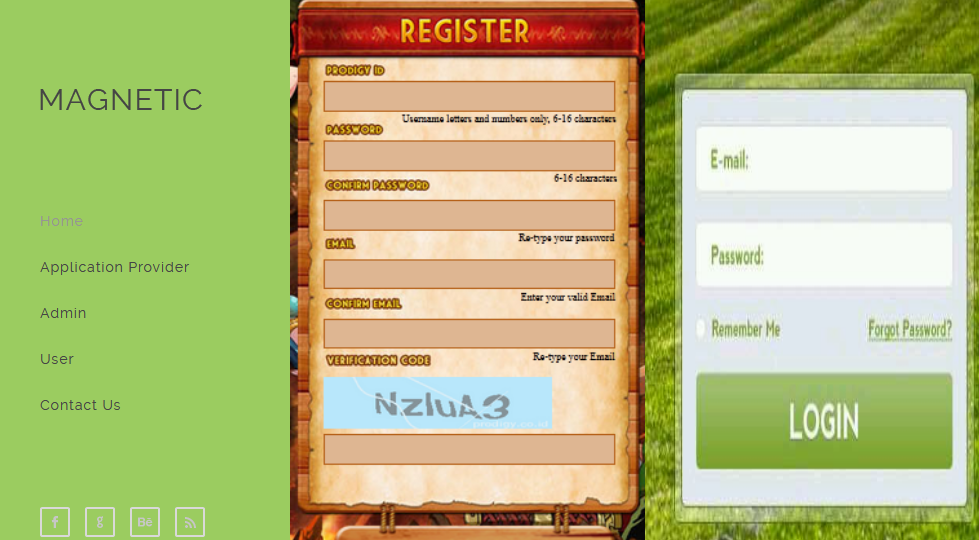
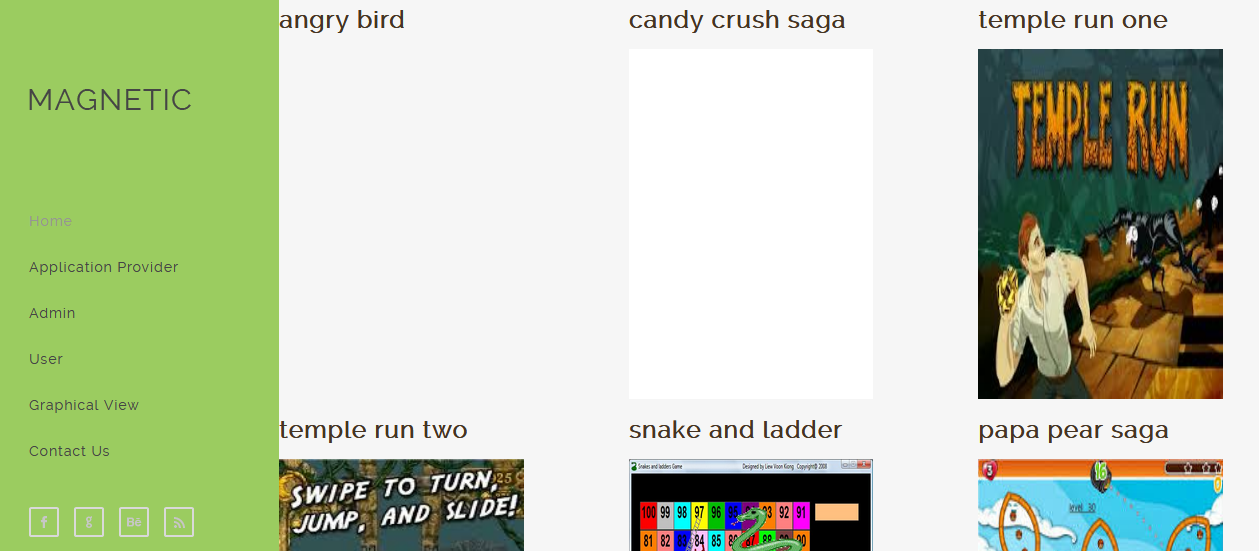
public String getServletInfo() {

return "Short description";

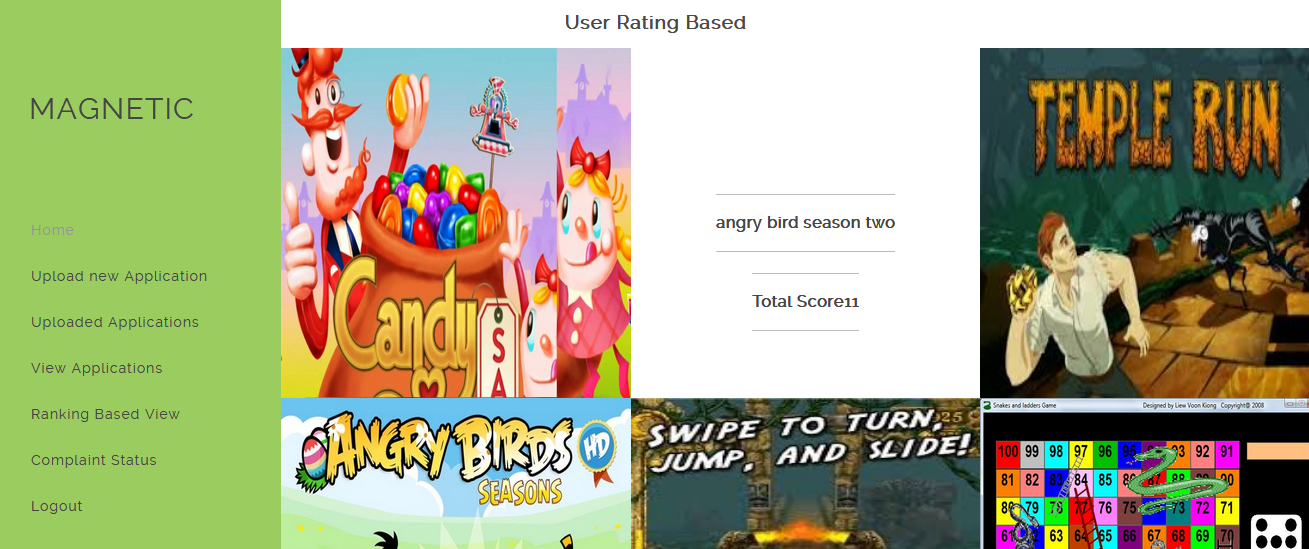
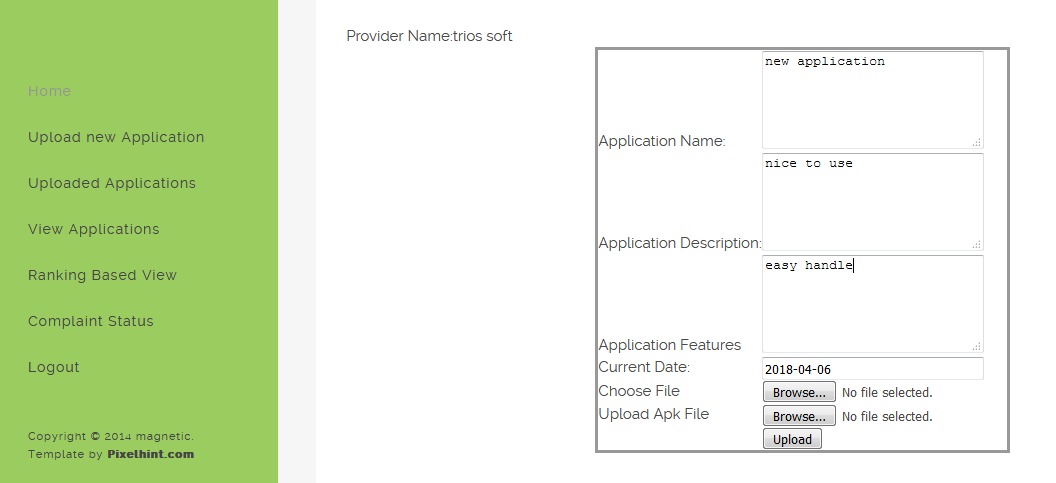
}// </editor-fold>

}

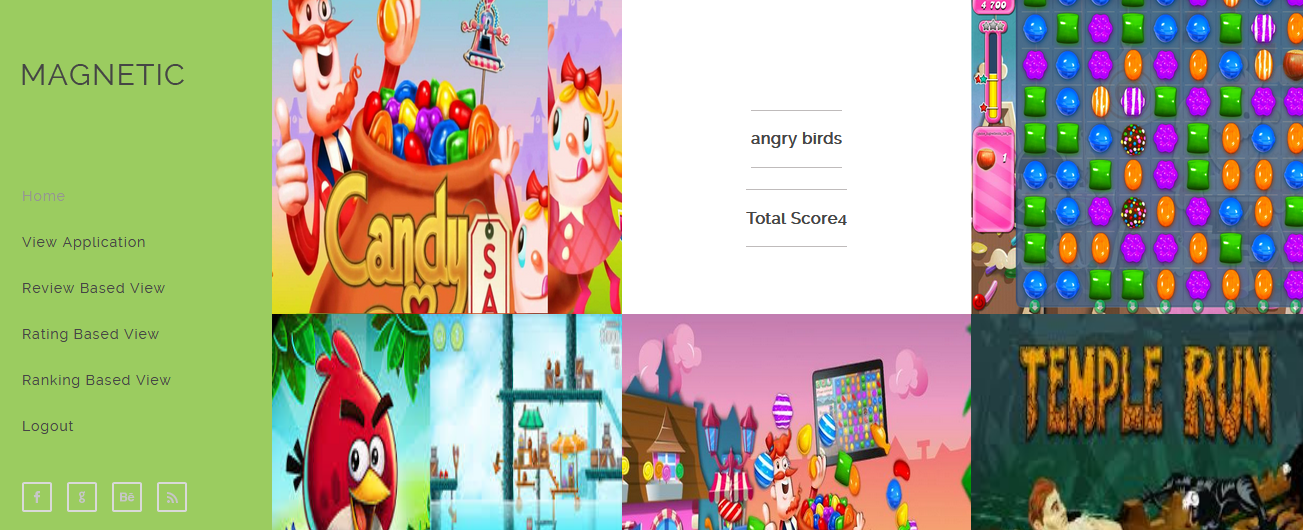
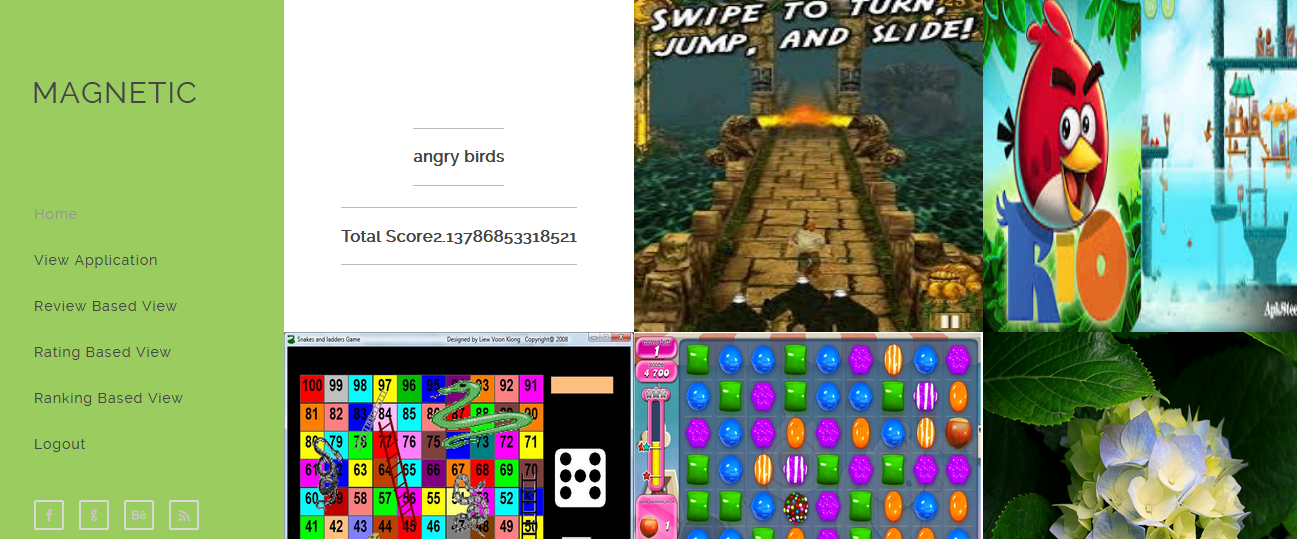
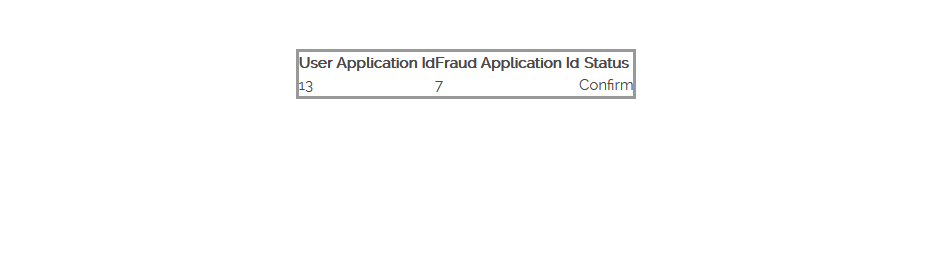
**7. SCREENSHOTS**



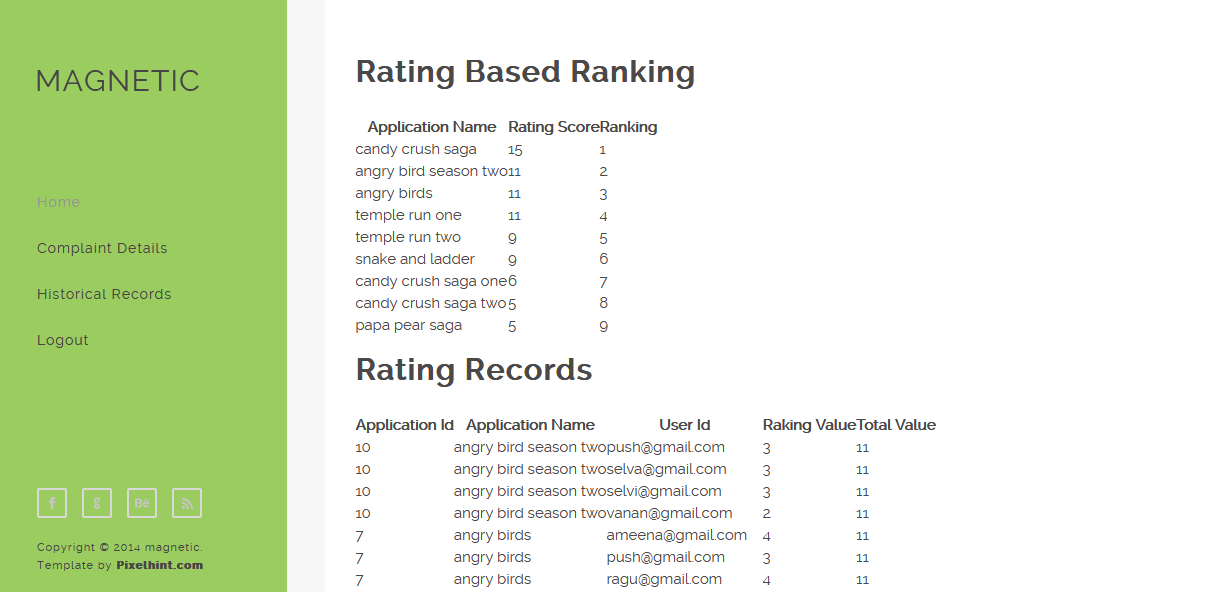
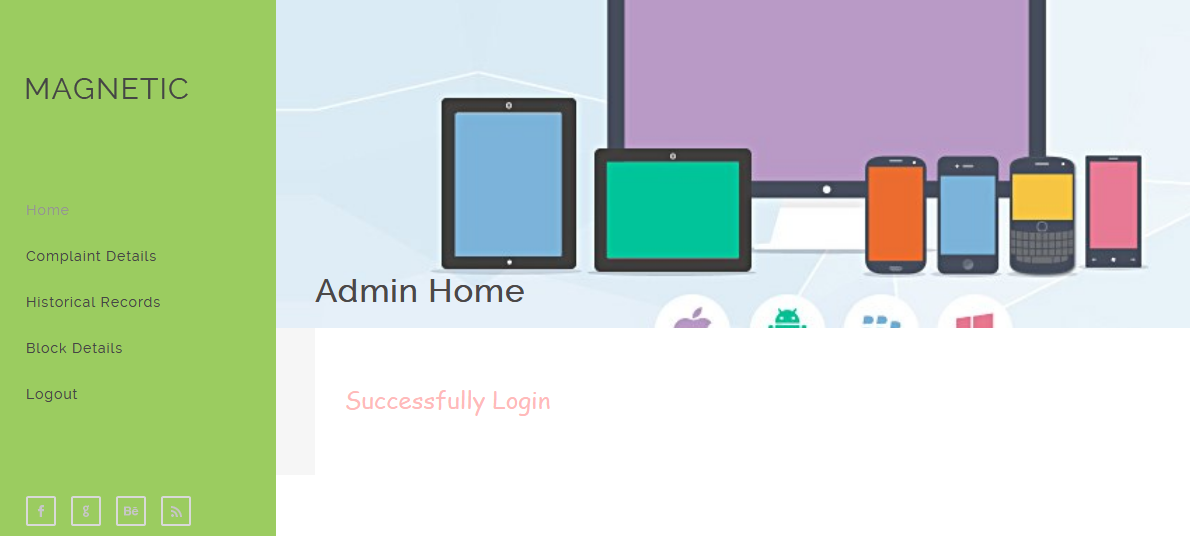
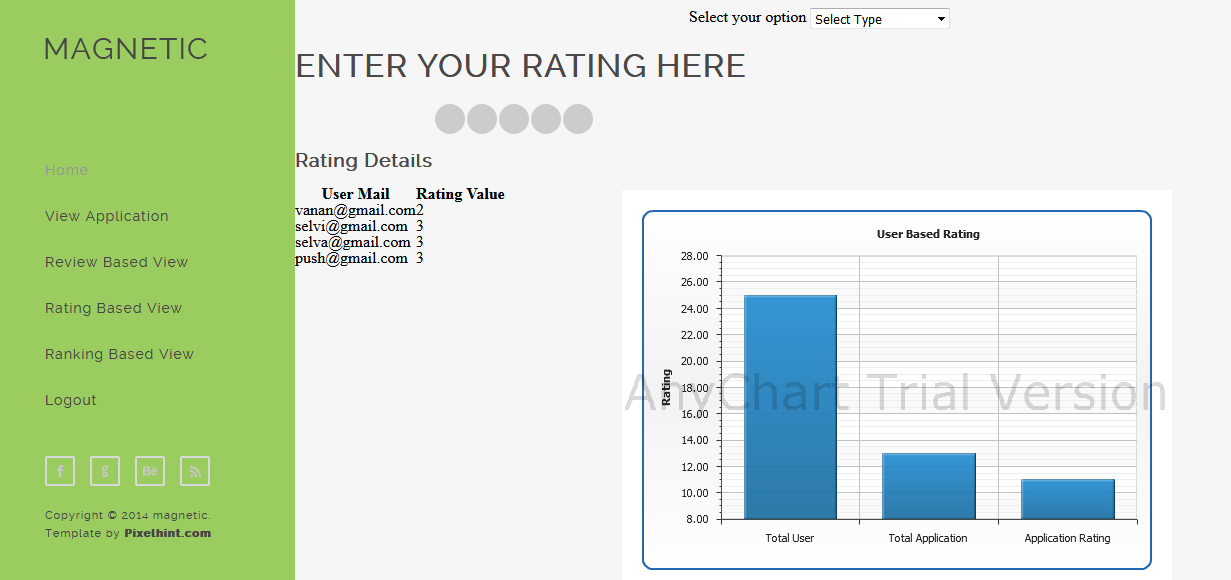
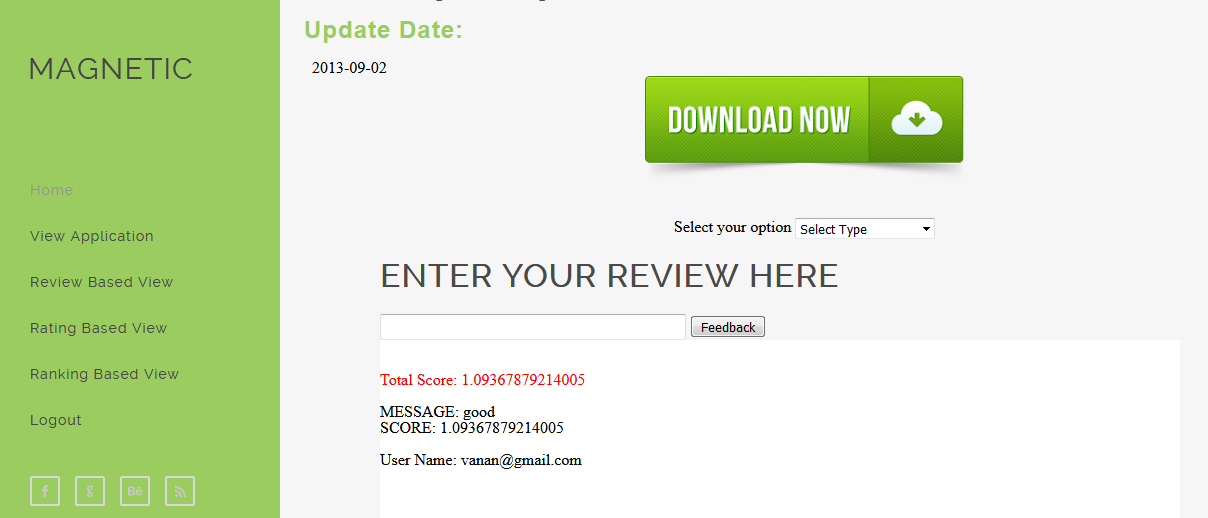
Application provider new application upload:



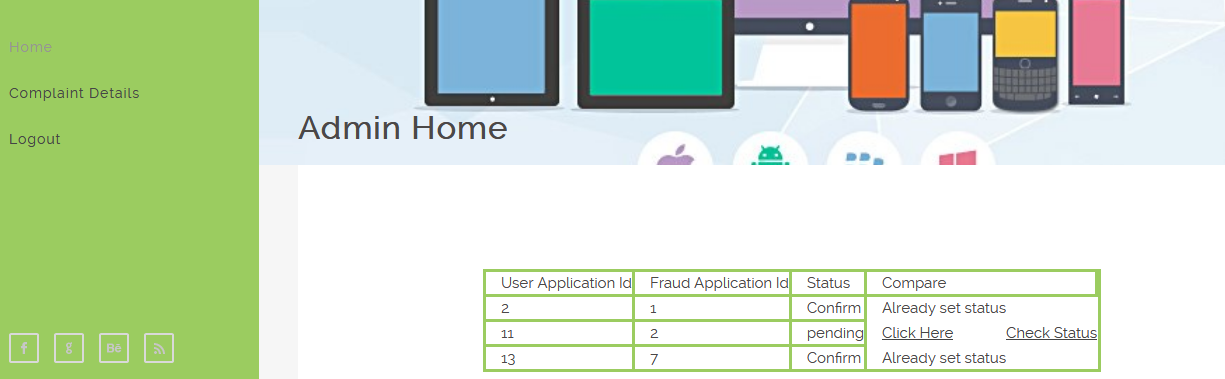
Complaint Status:



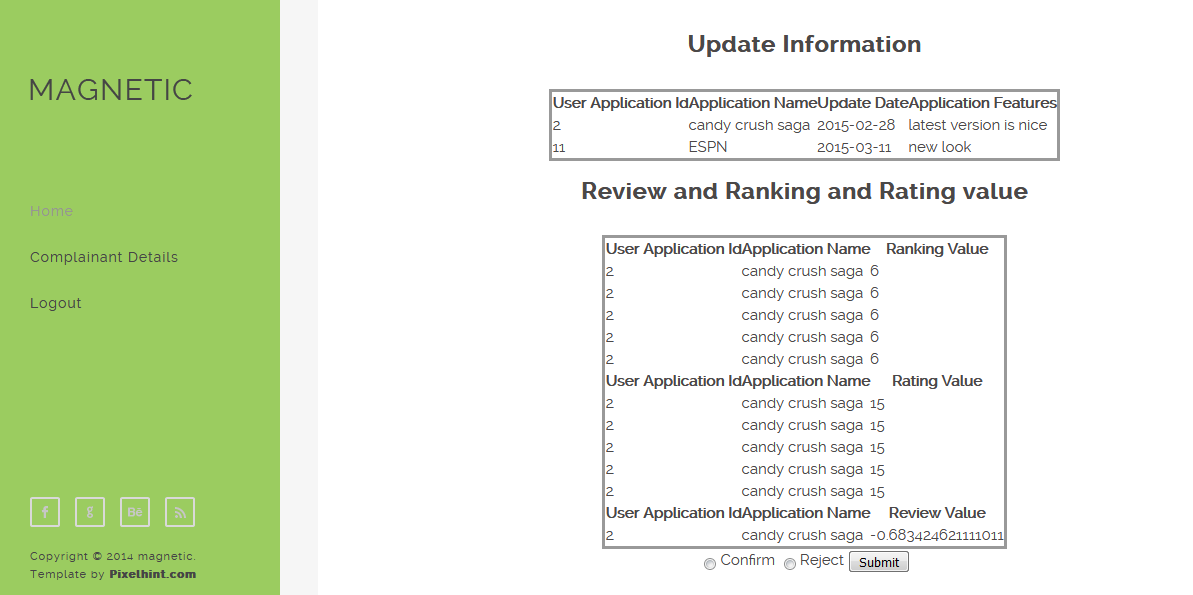
User review:



Admin complaint view:



Admin approval:



**8.CONCLUSION**

In this work, we first defined the professional malicious users (PMUS), who give fake feedbacks to confuse the normal users, hurt the recommender systems, and make illegal profits. We noticed that the traditional outlier detections could not be applied in the recommender system area to detect these professional malicious users because of their professional masking strategies. Also, supervised detection models could not work well on PMU detection for the lack of labels. To address the professional malicious user detection issue, we presented a new unsupervised multimodal learning model named MMD. In essence, MMD is a generic solution, which can not only detect the professional malicious users that are explored in this paper but also serve as a general foundation for malicious user detections. At last, we are very interested in building an online professional malicious user detection model that utilizes the recent advances in human-machine interactions.

**9. REFERENCES :**

[1] Z. Xia, C. Liu, N. Z. Gong, Q. Li, Y. Cui, and D. Song, “Characterizing and detecting malicious accounts in privacy-centric mobile social networks: A case study,” in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019. ACM, 2019, pp. 2012–2022.

[2] M. Fang, N. Z. Gong, and J. Liu, “Influence function based data poisoning attacks to top-n recommender systems,” in WWW ’20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020. ACM / IW3C2, 2020, pp. 3019–3025.

[3] Y. Cai and D. Zhu, “Trustworthy and profit: A new value-based neighbor selection method in recommender systems under shilling attacks,” Decision Support Systems, vol. 124, p. 113112, 2019.

[4] Y. Xu and F. Zhang, “Detecting shilling attacks in social recommender systems based on time series analysis and trust features,” Knowledge-Based Systems, vol. 178, pp. 25–47, 2019.

[5] R. A. Maronna, R. D. Martin, V. J. Yohai, and M. Salibian-Barrera, ´ Robust statistics: theory and methods (with R). John Wiley & Sons, 2019.

[6] Y. Xu, Y. Yang, J. Han, E. Wang, F. Zhuang, J. Yang, and H. Xiong, “Neuo: Exploiting the sentimental bias between ratings and reviews with neural networks,” Neural Networks, vol. 111, pp. 77–88, 2019.

[7] X. He, J. Tang, X. Du, R. Hong, T. Ren, and T.-S. Chua, “Fast matrix factorization with nonuniform weights on missing data,” IEEE transactions on neural networks and learning systems, 2019.

[8] B. Wang, J. Jia, and N. Z. Gong, “Graph-based security and privacy analytics via collective classification with joint weight learning and propagation,” 2019.

[9] Binghui Wang and Neil Zhenqiang Gong. 2019. Attacking Graph-based Classification via Manipulating the Graph Structure.

[10] Haibo Yang, Xin Zhang, Minghong Fang, and Jia Liu. 2019. Byzantine-Resilient Stochastic Gradient Descent for Distributed Learning: A Lipschitz-Inspired Coordinate-wise Median Approach. In Conference on Decision and Control (CDC).