VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi-590 018

A Project Work on

"CONTROL OF CYBERBULLYING USING SEMANTIC-ENHANCED MARGINALIZED DENOISING AUTO-ENCODER"

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

Bachelor of Engineering

In

Information Science & Engineering

Submitted by

Ms. Lavanya K	**1AY14IS050**
Ms. Medha N Gudihal	**1AY14IS056**
Ms. Megha V	**1AY14IS057**
Ms. Sheethal H S	**1AY14IS095**

Under the guidance of

Prof. Sushma T M

Assistant Professor

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

ACHARYA INSTITUTE OF TECHNOLOGY

(AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI. Recognized by AICTE , NEW DELHI)

Acharya Dr. Sarvepalli Radhakrishnan Road, Soldevanahalli, Bangaluru-560107

2017-18

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

ACHARYA INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University, Belagavi, Recognized by AICTE, New Delhi)

Acharya Dr.Sarvepalli Radhakrishnan Road, Soldevanahalli, Bengaluru-560107

2017-2018

Certificate

Certified that the Project work entitled **"CONTROL OF CYBERBULLYING USING SEMANTIC-ENHANCED MARGINALIZED DENOISING AUTO-ENCODER"** is a bonafide work carried out by **Ms. Lavanya K (1AY14IS050)**, **Ms. Medha N Gudihal (1AY14IS056)**, **Ms. Megha V (1AY14IS057) and Ms. Sheethal H S (1AY14IS095)** in partial fulfillment for the award of the degree of **Bachelor of Engineering** in **Information Science and Engineering** of the **Visvesvaraya Technological University**, Belagavi during the year 2017-18. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The Project has been approved as it satisfies the academic requirements in respect of Project work prescribed for the Bachelor of Engineering Degree.

(Prof. Sushma T M) Guide	**(Dr. Mahesh G)** HOD	Principal

External Viva

Name of the Examiners **Signature with date**

1.

2.

DECLARATION

We**, Lavanya K (1AY14IS050), Medha N Gudihal (1AY14IS056), Megha V (1AY14IS057) and Sheethal H S (1AY14IS095)**, students of B.E, Information Science and Engineering, Acharya Institute of Technology, Bangalore - 107, hereby declare that the project entitled **"CONTROL OF CYBERBULLYING USING SEMANTIC-ENHANCED DENOISING AUTO-ENCODER"** is an authentic record of our own work carried out under the supervision and guidance of **Prof. SUSHMA T M**, Assistant Professor, Department of Information Science and Engineering, Acharya Institute of Technology, Bangalore. We have not submitted the matter embodied to any other university or institution for the award of any other degree.

Name	**USN**	**Signature**
Ms. Lavanya K	1AY14IS050	……...
Ms. Medha N Gudihal	1AY14IS056	……
Ms. Megha V	1AY14IS057	……
Ms. Sheethal H S	1AY14IS095	……

Place:

Date:

ACKNOWLEDGEMENT

The joy and satisfaction that comes along with the successful completion of any task would be incomplete without mentioning the people who made it possible through constant guidance, support and encouragement.

We would like to profoundly thank **Sri. B Premnath Reddy**, Chairman, Acharya Institutes and **Dr. S C Pilli,** Principal, Acharya Institute of Technology for providing the necessary infrastructure to complete this project.

It gives us immense pleasure to thank **Dr. Mahesh G**, Head of Department, Department of Information Science & Engineering, for his constant support and encouragement.

We would like to express our deepest sense of gratitude to our Project guide, **Prof. Sushma T M,** Assistant Professor, Department of Information Science & Engineering, for her constant support, encouragement and guidance throughout this Project.

We wish to express our sincere thanks to Project Coordinator **Prof. Chayapathi A R**, Assistant Professor, Department of Information Science & Engineering, for his suggestions and support.

A warm thanks to all the faculty of Department of Information Science and Engineering, who have helped us with their views and encouraging ideas.

Lavanya K (1AY14IS050)
Medha N Gudihal (1AY14IS056)
Megha V (1AY14IS057)
Sheethal H S (1AY14IS095)

ABSTRACT

Cyberbullying is defined as an aggressive, intentional act carried out by a group or individual, using electronic forms of contact, repeatedly and over time, against a victim who cannot easily defend him or herself. With the advent of social media networks such as Facebook and Twitter, cyberbullying has become more prevalent. Thus, automatic detection of cyber bullying posts is becoming an increasingly important area of research.

Machine learning techniques make automatic detection of bullying messages in social media possible, and this could help to construct a healthy and safe social media environment. We propose a new representation learning method named Semantic-Enhanced Marginalized Denoising Auto-Encoder (smSDA) is developed via semantic extension of the popular deep learning model stacked denoising autoencoder. The semantic extension consists of semantic dropout noise and sparsity constraints, where the semantic dropout noise is designed based on domain knowledge and the word embedding technique. Our proposed method is able to exploit the hidden feature structure of bullying information and learn a robust and discriminative representation of text.

The cyberbullying has developed a situation where there has been a surge of numerous bullying messages towards a specific user. This implication of cyberbullying becomes serious when the victims fail to cope with emotional strain from abusive, threatening, humiliating and aggressive messages. To deal with this issue, our proposed system is able to block the post which contains bullying words and it will also block the user with a history of constant or repetitive bullying temporarily on online social network.

TABLE OF CONTENTS

List of Figures

List of Tables

Table No. **Page No.**

CHAPTER 1
INTRODUCTION

CHAPTER 1

INTRODUCTION

Cyberbullying can be defined as aggressive, intentional actions performed by an individual or a group of people via digital communication methods such as sending messages and posting comments against a victim. Cyberbullying on social media can take place anywhere at any time. For bullies, they are free to hurt their peers' feelings because they do not need to face someone and can hide behind the Internet. Victims are easily exposed to harassment since all of us, especially youth, are constantly connected to Internet or social media.

Previous works on computational studies have shown that natural language processing and machine learning are powerful tools to study bullying. Cyberbullying detection can be named as a supervised learning problem. A classifier is first trained on cyberbullying corpus labeled by humans, and the learned classifier is then used to recognize a bullying message. Three kinds of information including text, user demography, and social network features are often used in cyberbullying detection. The focus here is on text-based cyberbullying detection.

In cyberbullying detection, the numerical representation for Internet messages should be robust and discriminative. With the knowledge of one deep learning method named stacked denoising auto encoder (SDA). We investigate a new text representation model based on SDA: marginalized stacked denoising auto encoders (mSDA), which adopts linear instead in order to learn more robust representations We utilize semantic information to expand mSDA and develop Semantic-enhanced Marginalized Stacked Denoising Autoencoders (smSDA). The semantic information consists of bullying words. An automatic extraction of bullying words based on word embeddings is proposed so that the involved human labor can be reduced. During training of smSDA, we attempt to reconstruct bullying features from other normal words, i.e. correlation, between bullying and normal words. Our proposed Semantic-enhanced Marginalized Stacked Denoising Autoencoder is able to learn robust features from BoW representation in an efficient and effective way. These robust features are learned by reconstructing original input from corrupted (i.e., missing) ones.

Semantic information is incorporated into their construction process. In our framework, high-quality semantic information, i.e., bullying words can be extracted automatically through word embeddings. During training of smSDA, we attempt to reconstruct bullying features from other normal words by discovering the latent structure, i.e. correlation, between bullying and normal words. The intuition behind this idea is that some bullying messages do not contain bullying words. The correlation information discovered by smSDA helps to reconstruct bullying features from normal words, and this in turn facilitates detection of bullying messages without containing bullying words.

1.1 Organization of Report

The technical aspect, system requirements and organization of project report are discussed as follows. The project report mainly consists of eight chapters.

CHAPTER 1: INTRODUCTION

This chapter includes the overall information about the project. The background knowledge needed for the project such as brief description on Machine Learning, Cyber Bullying and smSDA.

CHAPTER 2: LITERATURE SURVEY

This chapter mainly discusses about the papers that were referred for implementing the project. All papers provide information about cyberbullying detection and the various methods used to detect it.

CHAPTER 3: PROBLEM DEFINITION

It includes purpose of the project and the detailed study of background related to the project, objective, existing solution and proposed solution.

CHAPTER 4: SYSTEM REQUIREMENTS SPECIFICATION

Here we look into different kinds of requirements needed to successfully complete the project. It also states the functional and non-functional requirements.

CHAPTER 5: SYSTEM DESIGN

This chapter includes some basic design concepts, methodology, and some necessary diagram to outline the project easily.

CHAPTER 6: IMPLEMENTATION

This describes the implementation of different modules and snapshots depicting different pages. Introduces about the technology and methodology used to implement the project.

CHAPTER 7: TESTING

This chapter defines various testing methods used for testing the different phases in the project; they are unit testing, integration testing mainly.

CHAPTER 8: CONCLUSION AND FUTURE ENHANCEMENT

This chapter gives the summary of the work carried out and provides the conclusion and the future enhancement of the project.

CHAPTER 2
LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

This chapter mainly discusses about the papers that were referred for implementing the project. All papers provide information about cyberbullying detection and the various methods used to detect it.

2.1 Detection based on Semantic-Enhanced Marginalized Denoising Auto-Encoder

As a side effect of increasingly popular social media, cyberbullying has emerged as a serious problem afflicting children, adolescents and young adults. Machine learning techniques make automatic detection of bullying messages in social media possible, and this could help to construct a healthy and safe social media environment. In this meaningful research area, one critical issue is robust and discriminative numerical representation learning of text messages. The method named Semantic-Enhanced Marginalized Denoising Auto-Encoder (smSDA) is developed via semantic extension of the popular deep learning model stacked denoising autoencoder. The semantic extension consists of semantic dropout noise and sparsity constraints, where the semantic dropout noise is designed based on domain knowledge and the word embedding technique. This method is able to exploit the hidden feature structure of bullying information and learn a robust and discriminative representation of text. Comprehensive experiments on two public cyberbullying corpora (Twitter and MySpace) are conducted, and the results show that these approaches outperform other baseline text representation learning methods. [1]

Disadvantages: Hidden or misspelled words in textual cyberbullying was not detected.

2.2 An Effective Approach for Cyberbullying Detection

The rapid growth of social networking is supplementing the progression of cyberbullying activities. Most of the individuals involved in these activities belong to the younger generations,

especially teenagers, who in the worst scenario are at more risk of suicidal attempts. The proposal of an effective approach to detect cyberbullying messages from social media through a weighting scheme of feature selection. Presentation of a graph model to extract the cyberbullying network, which is used to identify the most active cyberbullying predators and victims through ranking algorithms. [2]

Disadvantages: Users were classified into only two categories like most active predators and most active victims.

2.3 Modeling the Detection of Textual Cyberbullying

The scourge of cyberbullying has assumed alarming proportions with an ever-increasing number of adolescents admitting to having dealt with it either as a victim or as a bystander. Anonymity and the lack of meaningful supervision in the electronic medium are two factors that have exacerbated this social menace. Comments or posts involving sensitive topics that are personal to an individual are more likely to be internalized by a victim, often resulting in tragic outcomes. The decomposition of the overall detection problem into detection of sensitive topics is done, lending itself into text classification sub-problems. The experiment with a corpus of 4500 YouTube comments, applying a range of binary and multiclass classifiers. It is found that binary classifiers for individual labels outperform multiclass classifiers. The findings show that the detection of textual cyberbullying can be tackled by building individual topic-sensitive classifiers. [3]

Disadvantages: The results were verified with standalone YouTube comments limiting itself to a particular social network and also not dynamically verified.

2.4 Detection of Harassment on Web 2.0

Web 2.0 has led to the development and evolution of web-based communities and applications. These communities provide places for information sharing and collaboration. They also open the door for inappropriate online activities, such as harassment, in which some users post messages in a virtual community that are intentionally offensive to other members of the community. It is a new and challenging task to detect online harassment; currently few systems

attempt to solve this problem. Usage of supervised learning approach for detecting harassment is helpful in this area. This technique employs content features, sentiment features, and contextual features of documents. The experimental results described herein show that our method achieves significant improvements over several baselines, including Term Frequency Inverse Document Frequency (TFIDF) approaches. Identification of online harassment is feasible when TFIDF is supplemented with sentiment and contextual feature attributes. [4]

Disadvantages: The TFIDF scheme is one of the old approaches to detect cyberbullying, more efficient approaches are being used currently.

CHAPTER 3
PROBLEM DEFINITION

CHAPTER 3

PROBLEM DEFINITION

In this section, the detailed information about the thesis is presented. First, we start discussing problem statement, existing system that has been worked on, their advantages and disadvantages and finally the proposed system.

3.1 Problem Statement

To implement a safe and secure online social networkthat is free from textual based cyberbullying. Not justwhen there are direct bullying words but also when thereare hidden features in it and thereby blocking such posts. On repetitive bullying the user is blocked permanently.

3.2 Existing System

- Previous works on computational studies of bullying have shown that natural language processing and machine learning are powerful tools to study bullying.
- Cyberbullying detection can be formulated as a supervised learning problem. A classifier is first trained on a cyberbullying corpus labeled by humans, and the learned classifier is then used to recognize a bullying message.
- Yin et.al proposed to combine BoW features, sentiment features and contextual features to train a support vector machine for online harassment detection.
- Dinakar et.al utilized label specific features to extend the general features, where the label specific features are learned by Linear Discriminative Analysis. In addition, common sense knowledge was also applied.
- Nahar et.al presented a weighted TF-IDF scheme via scaling bullying-like features by a factor of two. Besides content-based information, Maral et.al proposed to apply

users' information, such as gender and history messages, and context information as extra features

3.2.1 Disadvantages of Existing System

- The first and also critical step is the numerical representation learning for text messages.
- Secondly, cyberbullying is hard to describe and judge from a third view due to its intrinsic ambiguities.
- Thirdly, due to protection of Internet users and privacy issues, only a small portion of messages are left on the Internet, and most bullying posts are deleted.

3.3 Proposed System

- Three kinds of information including text, user demography, and social network features are often used in cyberbullying detection. Since the text content is the most reliable, our work here focuses on text-based cyberbullying detection.
- Investigation of one deep learning method named stacked denoisingautoencoder (SDA). SDA stacks several denoisingautoencoders and concatenates the output of each layer as the learned representation. Each denoisingautoencoder in SDA is trained to recover the input data from a corrupted version of it. The input is corrupted by randomly setting some of the input to zero, which is called dropout noise. This denoising process helps the autoencoders to learn robust representation.
- In addition, each autoencoder layer is intended to learn an increasingly abstract representation of the input.
- Development of a new text representation model based on a variant of SDA: marginalized stacked denoisingautoencoders (mSDA), which adopts linear instead of nonlinear projection to accelerate training and marginalizes infinite noise distribution in order to learn more robust representations.

- Utilization of semantic information to expand mSDA and develop Semantic-enhanced Marginalized Stacked DenoisingAutoencoders (smSDA). The semantic information consists of bullying words. An automatic extraction of bullying words based on word embeddings is proposed so that the involved human labor can be reduced. During training of smSDA, we attempt to reconstruct bullying features from other normal words by discovering the latent structure, i.e. correlation, between bullying and normal words. The intuition behind this idea is that some bullying messages do not contain bullying words. The correlation information discovered by smSDA helps to reconstruct bullying features from normal words, and this in turn facilitates detection of bullying messages without containing bullying words.

3.3.1 Advantages of Proposed System

- The Semantic-enhanced Marginalized Stacked DenoisingAutoencoder is able to learn robust features from BoW representation in an efficient and effective way. These robust features are learned by reconstructing original input from corrupted (i.e., missing) ones. The new feature space can improve the performance of cyberbullying detection even with a small labeled training corpus.
- Semantic information is incorporated into the reconstruction process via the designing of semantic dropout noises and imposing sparsity constraints on mapping matrix. In this framework, high-quality semantic information, i.e., bullying words, can be extracted automatically through word embeddings.
- Finally, these specialized modifications make the new feature space more discriminative and this in turn facilitates bullying detection.
- Comprehensive experiments on real-data sets have verified the performance of our proposed model.

CHAPTER 4

SYSTEM REQUIREMENTS SPECIFICATION

CHAPTER 4

SYSTEM REQUIRMENTS SPECIFICATION

4.1 Introduction to System Requirements Specification

A software requirements specification (SRS) is a comprehensive description of the intended purpose and environment for software under development. The software requirements specification fully describes what the software will do and how it will be expected to perform.

A software requirements specification minimizes the time and effort required by developers to achieve desired goals and also minimizes the development cost. A good software requirements specification defines how an application will interact with system hardware, other programs and human users in a wide variety of real-world situations. Parameters such as operating speed, response time, availability, portability, maintainability, footprint, security and speed of recovery from adverse events are evaluated.

4.2 Functional Requirements

Most requirements definition focuses mainly on functional requirements, which are based upon the expected functioning of the product or system to be created. Functioning typically is equated with product/system features for which you might have a menu or button choice.

All things considered, requirements definers probably are best at identifying functional requirements, although they often overlook and get wrong more of the functional requirements than they ordinarily recognize. On hindsight reflection, they frequently do realize that many of the problems which surface later, and thus are harder and more expensive to fix, are attributable to inadequately addressed non-functional requirements.

This section describes the functional requirements of the system for those requirements which are expressed in the natural language style.

REQ 1: Install our Framework

REQ 2: Create online social network which contains User, Admin.

REQ 3: User posts on his/her friends' wall.

REQ 4: Reconstruct corrupted input using smSDA.

REQ 5: Apply word embedding technique to detect bullying words.

REQ 6: Application should securely filter bulling words.

4.3 Non-functional Requirements

Non-functional requirements refer to a whole slewof attributes including performance levels, security, and the various "ileitis," such as usability, reliability, and availability. Invariably, requirements definers get wrapped up in how the product/system is expected to function and lose sight of these added elements.

When such factors are not addressed adequately, seemingly proper product/system functioning in fact fails to function. For example, a system may identify customers in such a slow, insecure, and difficult to use manner that it can cause mistakes which make data unreliable, provoke frustration-based attempted work-around that can create further problems, and ultimately lead to abandonment. That's the recognized way in which non-functional requirements impact product/system success. Other often unrecognized issues also need to be appreciated.

4.4 Software Requirements

Software Requirements is a field within software engineering that deals with establishing the needs of stakeholders that are to be solved by software. The IEEE Standard Glossary of Software Engineering Terminology defines a requirement as:

A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposeddocument.

Some of the important Software requirements required in our project are:

4.4.1 MySQL Database

MySQL is a popular choice of database for use in web applications, and is a central component of the widely used XAMPP open-source web application software stack (and other "AMP" stacks). Free-software open-source projects that require a full-featured database management system often use MySQL.

Figure 4.1: MySQL

4.4.2 Apache Server

The Apache HTTP Server, colloquially called Apache, is the world's most used web server software. Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. Most commonly used on a Unix-like system (usually Linux), the software is available for a wide variety of operating systems besides Unix, including eComStation, Microsoft Windows, NetWare, OpenVMS, OS/2, and TPF. Released under the Apache License, Apache is free and open-source software.

Figure 4.2: Apache Server

4.4.3 Java (jdk 1.8 version)

Java is a computer programming language that is concurrent, class-based and object-oriented. It was originally developed by James Gosling at Sun Microsystems. Java applications are compiled to bytecode (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture.

In order to be able to compile Java code, we need the Java Development Kit (JDK) package that comes with a Java compiler. The JDK package also comes with a Java runtime environment (JRE) that is needed to run compiled Java code.

Figure 4.3: Java

4.4.4 JDBC(Java Database Connectivity)

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database, and is oriented towards relational databases. A JDBC-to-ODBC bridge enables connections to any ODBC-accessible data source in the Java virtual machine (JVM) host environment.

Figure 4.4: JDBC connecting MySQL

4.4.5 NetBeans IDE(Version 8.0.1)

NetBeans is an integrated development environment (IDE) for Java. NetBeans allows applications to be developed from a set of modular software components called modules. NetBeans runs on Microsoft Windows, macOS, Linux and Solaris. In addition to Java development, it has extensions for other languages like PHP, C, C++, HTML5,Javadoc, and Javascript. Applications based on NetBeans, including the NetBeans IDE, can be extended by third party developers.

Figure 4.5: NetBeans

4.4.6 PHP (Hypertext Pre-processor)

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. PHP originally stood for Personal Home Page, but it now stands for the recursive backronym PHP: Hypertext Pre-processor. PHP code may be

embedded into HTML code, or it can be used in combination with various web template systems, web content management system and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and can be used to implement standalone graphical applications.

CHAPTER 5
SYSTEM DESIGN

CHAPTER 5

SYSTEM DESIGN

Once the requirements document for the software to be developed is available, the software design phase begins. While the requirement specification activity deals entirely with the problem domain, design is the first phase of transforming the problem into a solution. In the design phase, the customer and business requirements and technical considerations all come together to formulate a product or a system.

The design process comprises a set of principles, concepts and practices, which allow a software engineer to model the system or product that is to be built. This model, known as design model, is assessed for quality and reviewed before a code is generated and tests are conducted. The design model provides details about software data structures, architecture, interfaces and components which are required to implement the system. This chapter discusses the design elements that are required to develop a software design model. It also discusses the design patterns and various software design notations used to represent a software design.

5.1 Principles of Design

Developing design is a cumbersome process as most expansive errors are often introduced in this phase. Moreover, if these errors get unnoticed till later phases, it becomes more difficult to correct them. Therefore, a number of principles are followed while designing the software. These principles act as a framework for the designers to follow a good design practice as described in Figure 5.1.

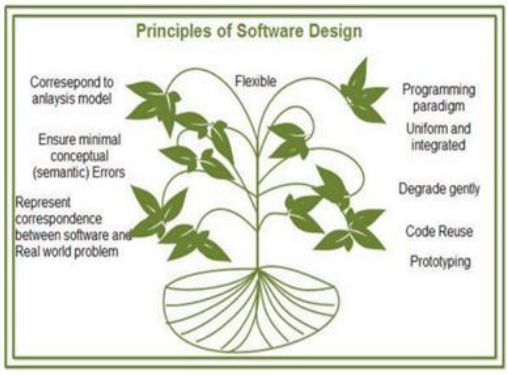

Figure 5.1: Principles of Software Design

Some of the commonly followed design principles are as following:

- **Software design should correspond to the analysis model:** Often a design elementcorresponds to, many requirements; therefore, we must know how the design model satisfies all the requirements represented by the analysis model.
- **Choose the right programming paradigm:** A programming paradigm describes the structureof the software system. Depending on the nature and type of application, different programming paradigms such as procedure oriented, object-oriented, and prototyping paradigms can be used. The paradigm should be chosen keeping constraints in mind such as time, availability of resources and nature of user's requirements.
- **Software design should be uniform and integrated:** Software design is considered uniformand integrated, if the interfaces are properly defined among the design components. For this, rules, format, and styles are established before the design team starts designing the software.
- **Software design should be flexible:** Software design should be flexible enough to adaptchanges easily. To achieve the flexibility, the basic design concepts such as abstraction, refinement, and modularity should be applied effectively.
- **Software design should ensure minimal conceptual (semantic) errors:** The design teammust ensure that major conceptual errors of design such as ambiguousness and consistency are addressed in advance before dealing with the syntactical errors present in the design model.

- **Software design should be structured to degrade gently:** Software should be designed tohandle unusual changes and circumstances, and if the need arises for termination, it must do so in a proper manner so that functionality of the software is not affected.
- **Software design should represent correspondence between the software and real-world problem:** The software design should be structured in such a way that it always relates with thereal-world problem.
- **Software reuse:** Software engineers believe on the phrase: 'do not reinvent thewheel'. Therefore, software components should be designed in such a way that they can be effectively reused to increase the productivity.
- **Designing for testability:** A common practice that has been followed is to keep the testingphase separate from the design and implementation phases. That is, first the software is developed (designed and implemented) and then handed over to the testers who subsequently determine whether the software is fit for distribution and subsequent use by the customer. However, it has become apparent that the process of separating testing is seriously flawed, as if any type of design or implementation errors are found after implementation, then the entire or a substantial part of the software requires to be redone. Thus, the test engineers should be involved from the initial stages. For example, they should be involved with analysts to prepare tests for determining whether the user requirements are being met.
- **Prototyping:** Prototyping should be used when the requirements are not completely defined inthe beginning. The user interacts with the developer to expand and refine the requirements as the development proceeds. Using prototyping, a quick 'mock-up' of the system can be developed. This mock-up can be used as an effective means to give the users a feel of what the system will look like and demonstrate functions that will be included in the developed system. Prototyping also helps in reducing risks of designing software that is not in accordance with the customer's requirements.

5.2 High Level Design

High-level design (HLD) explains the architecture that would be used for developing a software product.

The diagram provides an overview of an entire system, identifying the main components that would be developed for the product and their interfaces. The HLD uses possibly nontechnical to mildly technical terms that should be understandable to the administrators of the system. In contrast, low-level design further exposes the logical detailed design of each of these elements for programmers.

5.2.1 Data Flow Diagram

A data flow diagram is a graphic depiction of how the information moves through the system and how it is modified by a series of transformations by the elements of a system.

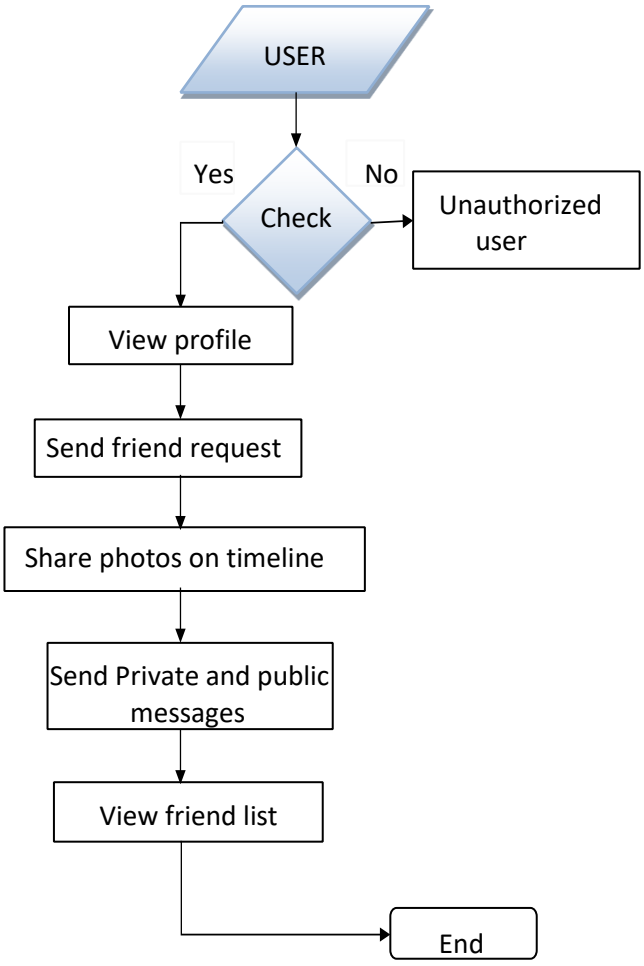

Figure 5.2: Data flow diagram for user activities

User activities Data Flow Diagram

The Figure 5.2 depicts the various user activities. Firstly, the user is authenticated and if the user is an authorized one then he is allowed to perform various activities like viewing profile, sending friend request, sharing photos on timeline, sending messages and viewing and accepting friend requests.

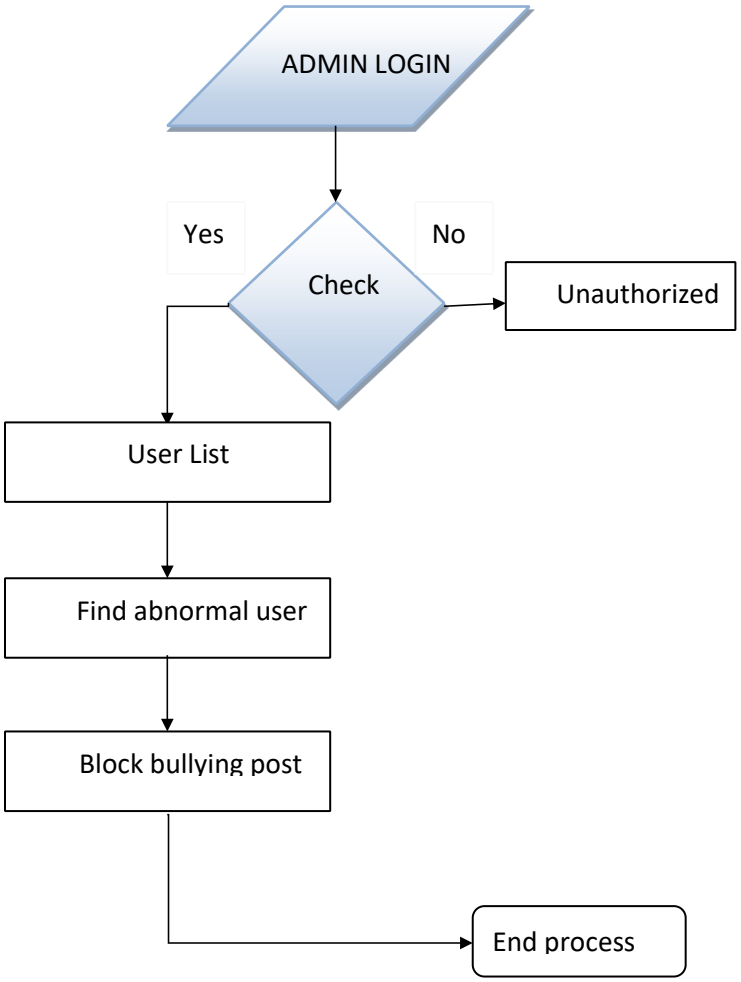

Figure 5.3: Data flow diagram for admin activities

Admin activities Data Flow Diagram

The Figure 5.3 depicts the various admin activities. Firstly, the admin is authenticated and if the admin is an authorized one then he is allowed to view users list, find abnormal user and also block bullying posts.

5.2.2 Use Case Diagram

A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements.

Use Case Diagram for Admin and Users

The Figure 5.4 depicts the role of Administrator and User. The administrator's responsibilities are to monitor user logins, find abnormal users and to view the blocked bullying posts. He is also responsible for maintaining the system. The User's activities are to register to the Online Social Network, to view profile, to find friends, to share photos, to send and view friend requests, to view and send messages and to search friends.

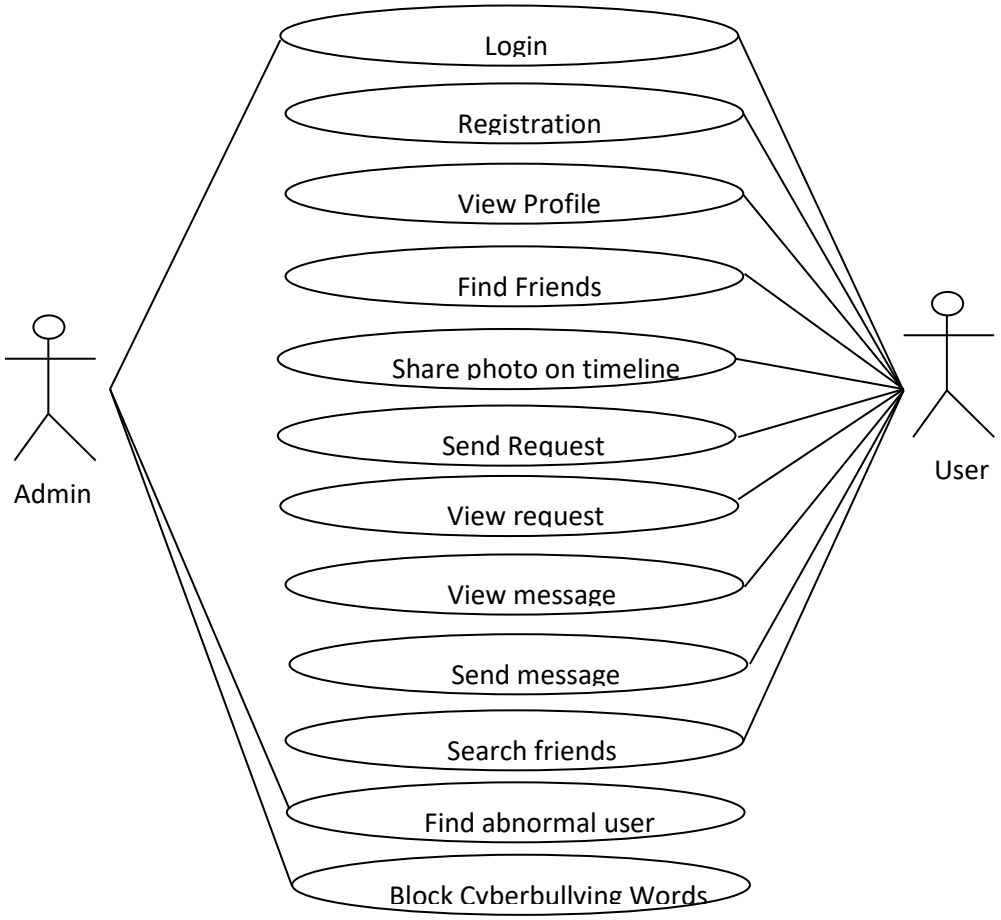

Figure 5.4: Use Case Diagram for Users and Admin

5.2.3 Class Diagram

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

Class Diagram for user, admin and registrations

The user class has operations like to login, to view profile, to find friends, to share photos, to send and view friend requests, to send private and public posts, to view and send messages and to search friends. The admin has operations like to monitor user logins, find abnormal users and to view the blocked bullying posts. The registration class has attributes like username, password, profile images and user details. The Figure 5.5 describes the user, admin and registration activities.

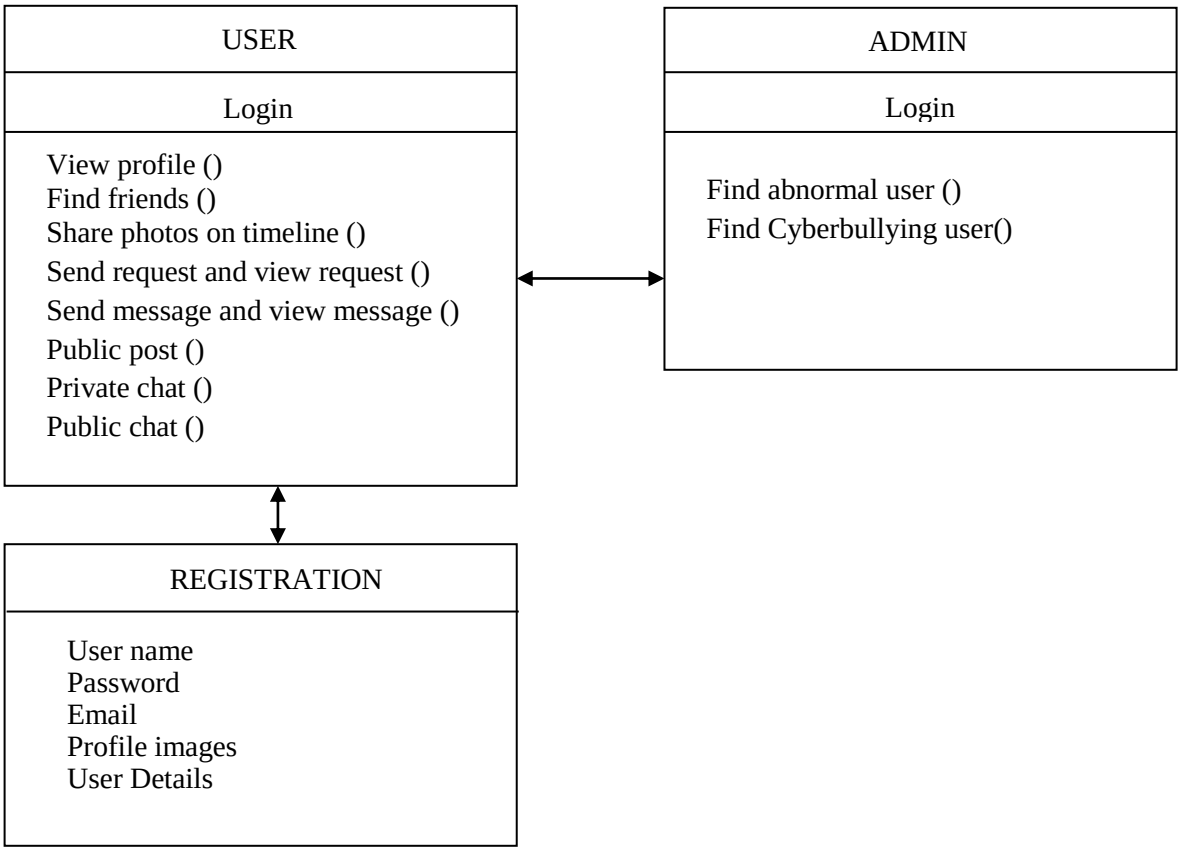

Figure 5.5: Class diagram for user, admin and registrations

5.2.4 Sequence Diagram

A Sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence.

Sequence diagram for user, admin and registrations

The user creates an account in the database and with the creation of the account on social network he is allowed to perform various activities. The admin monitors all the user's activities throughout the session and finds abnormal users and blocks bullying posts. The Figure 5.6 describes the sequence diagram for user, admin and registration activities.

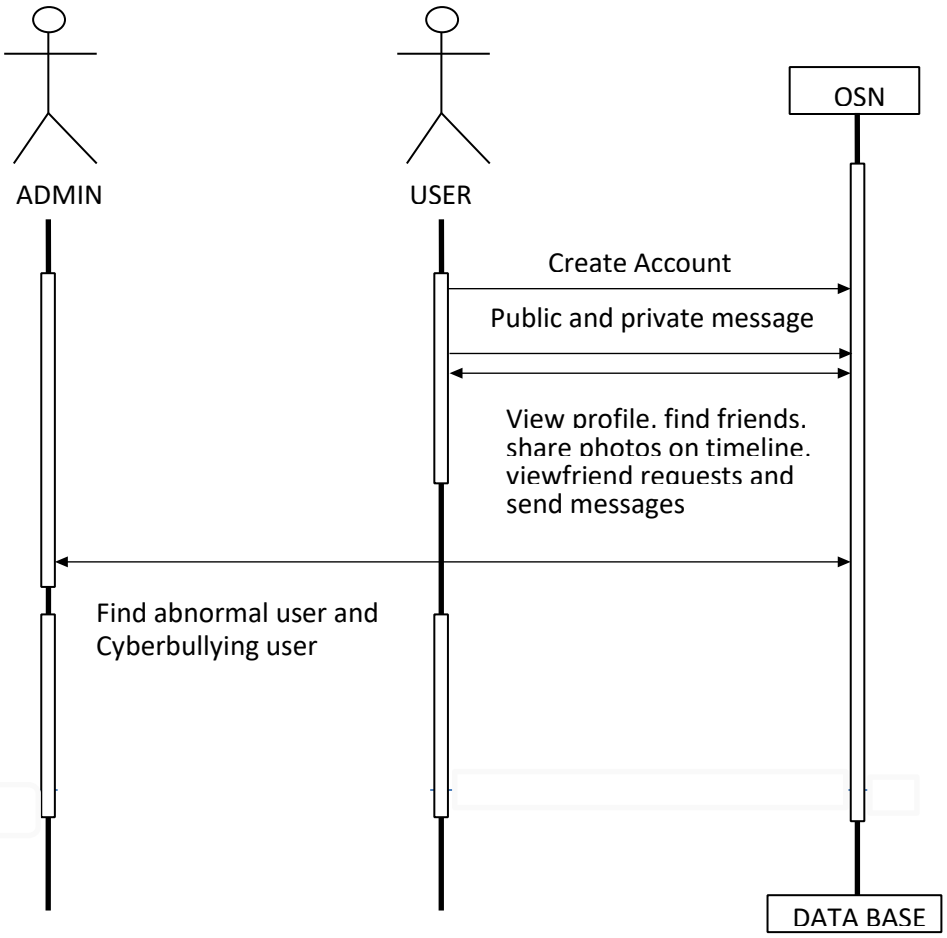

Figure 5.6: Sequence diagram for user, admin and registrations

5.2.5 Activity Diagram

The Figure 5.7 describes the operational step-by-step workflows of components in the system. It shows the overall flow of control in the system.

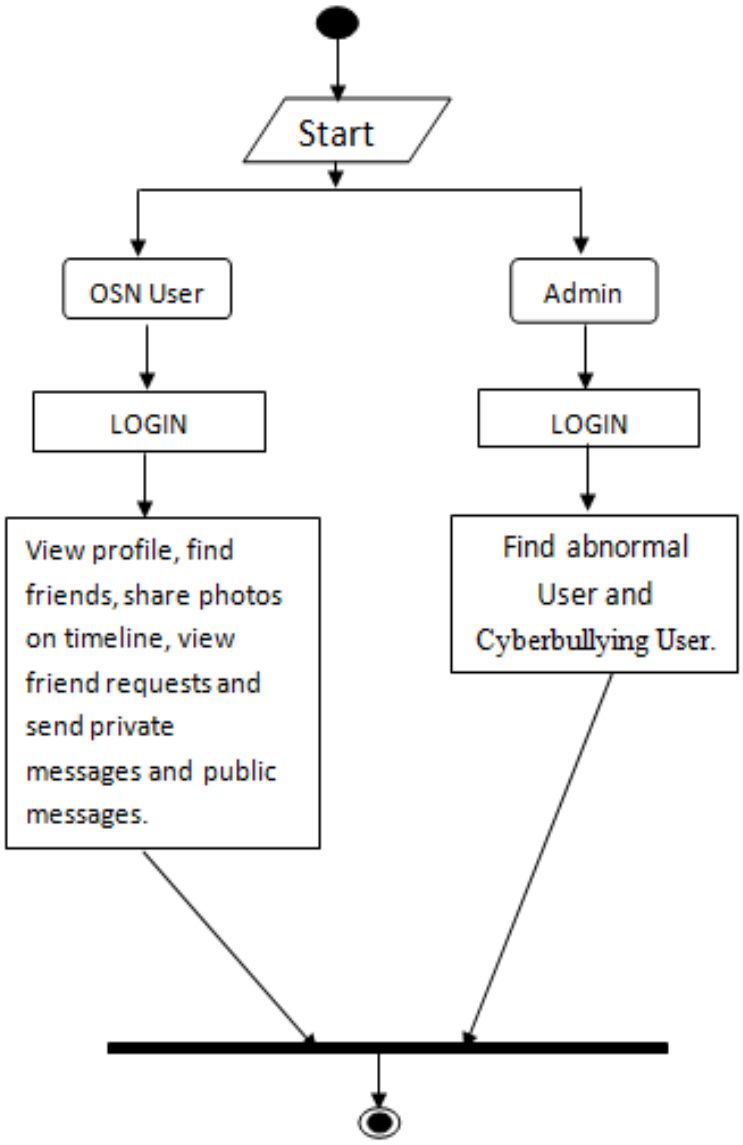

Figure 5.7: Activity diagram for user and admin

5.3 Algorithms Used

An algorithm is a procedure or step-by-step instruction for solving a problem. They form the foundation of writing a program.

5.3.1 Construction of OSN

Procedure ***Construction_of_OSN***(User and admin)

Input: n users and admin

Output: Basic OSN

Begin

Step1: Create the web application like a facebook with basic features.

Step2: User should be able to register, login and send/accept friend requests.

Step3: User can post on their walls and their friend's walls as well.

Step4: Store all the user information.

End

5.3.2 Construction of bullying features

Procedure ***Construction_of_bullyingfeatures***(Bow, Dict)

Input: Bag of words (BOW) and dict

Output: Construction of bullying features

Begin

Step1: Add the bullying words to our own corpus with word embedding technique.

Step2: Assign the weights to the words.

Step3: Find the co-relation between words.

i)if co-related words

then add into the bullying features

ii)else

repeat step2

End

5.3.3 smSDA

Procedure ***smSDA***(corrupted words, bullying features)

Input: corrupted words, bullying features

Output: Reconstruction of corrupted words and block the bullying messages.

Begin

Step1: Read corrupted words.

Step2: Find the co-relation between the corrupted words and normal words.

Step3: Reconstruct the corrupted words.

Step4: Detect if the reconstructed word is bullying word or normal word.

Step5: if it is a bullying word

Block the post

else

Post the message on the user's wall

End

5.3.4 LCH Algorithm

LCH (word1, word2) (Leacock and Chodorow measure)

Construct T1 which contains co-related words to word1

Construct T2 which contains co-related words to word2

Lowest Common Subsumer (s) = argmin (length (subsume (T1, T2)))

= {subsume (T1[1], T2[2]), subsume (T1[1], T2[3])} = {word1}

Length(word1) = 12

Max Depth(n) = 20 (from parameter)

Score = -log(length(LCS) / (2*max_depth (LCS.pos)) = -log (12/ (2*20))

return score;

CHAPTER 6
IMPLEMENTATION

CHAPTER 6

IMPLEMENTATION

The implementation consists of four modules. They are as follows:

1. OSN System Construction Module
2. Construction of Bullying Feature Set
3. Cyberbullying Detection
4. Semantic-Enhanced Marginalized Denoising Auto-Encoder

6.1 Modules Description

6.1.1 OSN System Construction Module

- In the first module, we develop the Online Social Networking (OSN) system module. We build up the system with the feature of Online Social Networking. Where, this module is used for new user registrations and after registrations the users can login with their authentication.
- Where after the existing users can send messages to privately and publicly, options are built. Users can also share post with others. The user can able to search the other user profiles and public posts. In this module users can also accept and send friend requests.
- With all the basic feature of Online Social Networking System modules is build up in the initial module, to prove and evaluate our system features.

6.1.2 Construction of Bullying Feature Set

- The bullying features play an important role and should be chosen properly. In the following, the steps for constructing bullying feature set Zb are given, in which the first layer and the other layers are addressed separately.

- For the first layer, expert knowledge and word embeddings are used. For the other layers, discriminative feature selection is conducted.
- In this module firstly, we build a list of words with negative affective, including swear words and dirty words. Then, we compare the word list with the BoW features of our own corpus and regard the intersections as bullying features.
- Finally, the constructed bullying features are used to train the first layer in our proposed smSDA. It includes two parts: one is the original insulting seeds based on domain knowledge and the other is the extended bullying words via word embeddings
- Observe Attentively Over A Period of Time.

6.1.3 Cyberbullying Detection

- In this module we propose the Semantic-enhanced Marginalized Stacked Denoising Auto-encoder (smSDA). In this module, we describe how to leverage it for cyberbullying detection. smSDA provides robust and discriminative representations The learned numerical representations can then be fed into our system.
- In the new space, due to the captured feature correlation and semantic information, even trained in a small size of training corpus, is able to achieve a good performance on testing documents.
- Based on word embeddings, bullying features can be extracted automatically. In addition, the possible limitation of expert knowledge can be alleviated by the use of word embedding
- BLOCK THE ACCOUNTS:
  Abnormal user
  Cyber-Crime user

6.1.4 Semantic-Enhanced Marginalized Denoising Auto-Encoder

- An automatic extraction of bullying words based on word embeddings is proposed so that the involved human labor can be reduced. During training of smSDA, we attempt

to reconstruct bullying features from other normal words by discovering the latent structure, i.e. correlation, between bullying and normal words. The intuition behind this idea is that some bullying messages do not contain bullying words.

- The correlation information discovered by smSDA helps to reconstruct bullying features from normal words, and this in turn facilitates detection of bullying messages without containing bullying words. For example, there is a strong correlation between bullying word fuck and normal word off since they often occur together.
- If bullying messages do not contain such obvious bullying features, such as fuck is often misspelled as fck, the correlation may help to reconstruct the bullying features from normal ones so that the bullying message can be detected. It should be noted that introducing dropout noise has the effects of enlarging the size of the dataset, including training data size, which helps alleviate the data sparsity problem.

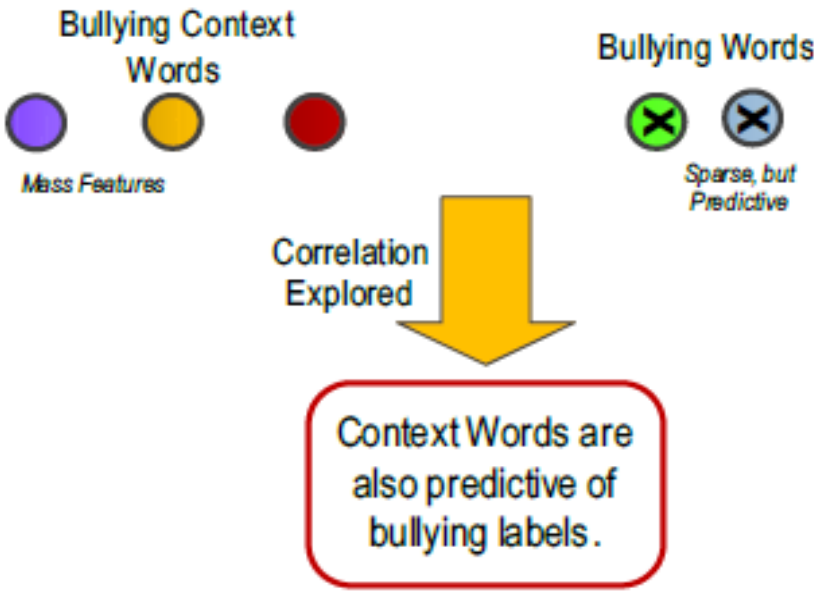

Figure 6.1: Exploring Correlation between bullying words and bullying context words

Jazzy Spell Checker – LCH Algorithm:

```
public class Spelling {
        private final HashMap<String, Integer> nWords = new HashMap<String, Integer>();
        public Spelling(String file) throws IOException {
                BufferedReader in = new BufferedReader(new FileReader(file));
                Pattern p = Pattern.compile("\\w+");
                for(String temp = ""; temp != null; temp = in.readLine()){
                        Matcher m = p.matcher(temp.toLowerCase());
                        while(m.find()) nWords.put((temp = m.group()), nWords.containsKey(temp) ?
                        nWords.get(temp) + 1 : 1);
                }
                in.close();
        }
        private final ArrayList<String> edits(String word) {
        ArrayList<String> result = new ArrayList<String>();
         Pattern p = Pattern.compile("[-+_.^:~@#$%*]");
        Matcher m = p.matcher(word);
        boolean b=m.find();
        if(b)
        {
                String word1=word. replaceAll("[-+_.^:~@#$%*]", "");
                for(int i=0; i < word.length(); ++i) {
                        result.add(word.substring(0, i) + word.substring(i+1));
                 }
                for(int i=0; i < word.length()-1; ++i)
                {
                        result.add(word.substring(0, i) + word.substring(i+1, i+2) +
                word.substring(i,i+1)      +word.substring(i+2));
                }
                for(int i=0; i < word.length(); ++i)
                {
                 for(char c='a'; c <= 'z'; ++c)
                {
                        result.add(word.substring(0, i) + String.valueOf(c) + word.substring(i+1));
                 }
         }
                for(int i=0; i <= word.length(); ++i) {
                for(char c='a'; c <= 'z'; ++c){
              result.add(word.substring(0, i) + String.valueOf(c) + word.substring(i));
         }
```

```
            }
    }
    else
    {
       System.out.println("else loooo");
       for(int i=0; i < word.length(); ++i)
       {
              result.add(word.substring(0, i) + word.substring(i+1));
        }
                    for(int i=0; i < word.length()-1; ++i)
        {
               result.add(word.substring(0, i) + word.substring(i+1, i+2) + word.substring(i, i+1) +
word.substring(i+2))
         }
                    for(int i=0; i < word.length(); ++i)
         {
             for(char c='a'; c <= 'z'; ++c)
              {
                 result.add(word.substring(0, i) + String.valueOf(c) + word.substring(i+1));
                     System.out.println("second for lop==========\n"+result.add(word.substring(0, i) +
          String.valueOf(c) + word.substring(i+1))+"\n===");
       }
          for(int i=0; i <= word.length(); ++i) {
          for(char c='a'; c <= 'z'; ++c){
                 result.add(word.substring(0, i) + String.valueOf(c) + word.substring(i));
                System.out.println("theard for loppp========\n"+result.add(word.substring(0, i) +
                String.valueOf(c) + word.substring(i))+"\n=======");
           }
           }
    }
           return result;
    }
          public final String correct(String word) {
                    if(nWords.containsKey(word)) return word;
                    ArrayList<String> list = edits(word);
                    HashMap<Integer, String> candidates = new HashMap<Integer, String>();
                    for(String s : list){
               if(nWords.containsKey(s))
                  candidates.put(nWords.get(s),s);}
                    if(candidates.size() > 0)
           {
```

```
                System.out.println("if size is
        >0=============\n"+candidates.get(Collections.max(candidates.keySet()))+"\n=======");
            return candidates.get(Collections.max(candidates.keySet()));
          }
                for(String s : list){
            for(String w : edits(s)){
            if(nWords.containsKey(w))
            {
            candidates.put(nWords.get(w),w);
           System.out.println("keyyyyyyyyyyyy========\n"+candidates+"\n=========");
            }
    }
    }
         String word1=candidates.size() > 0 ? candidates.get(Collections.max(candidates.keySet())) :
                word;
         System.out.println("final word issssssss==========\n"+word1+"\n=============");
                return candidates.size() > 0 ? candidates.get(Collections.max(candidates.keySet())) :
                word;
        }

      public static void main(String args[]) throws IOException {
       System.out.println("enter the string");
       Scanner sc=new Scanner(System.in);
       String str=sc.nextLine();
       System.out.println("enbtered string is===="+str);
       String cline[]="";
       StringBuffer sb=new StringBuffer();
       for(int i=0;i<mwords.length;i++)
       {
                if(mwords.length > 0)
                        System.out.println
                ((new Spelling("C:\\Documents and
         Settings\\Shekar\\Desktop\\words.txt")).correct(mwords[i])+"======\n");
            sb.append(new Spelling("D:\\new code\\find.txt").correct(mwords[i])+" ");
       }
       System.out.println("recunstructed line isss=====\n"+sb.toString()+"\n==========");
    }

}
```

Spelling.java - Reconstruction of words:

```
public class Spelling {
        private final HashMap<String, Integer> nWords = new HashMap<String, Integer>();
        public Spelling(String file) throws IOException {
                BufferedReader in = new BufferedReader(new FileReader(file));
                Pattern p = Pattern.compile("\\w+");
                for(String temp = ""; temp != null; temp = in.readLine()){
                        Matcher m = p.matcher(temp.toLowerCase());
                        while(m.find()) nWords.put((temp = m.group()), nWords.containsKey(temp) ?
                        nWords.get(temp) + 1 : 1);
                }
                in.close();
        }
        private final ArrayList<String> edits(String word) {
        ArrayList<String> result = new ArrayList<String>();
   Pattern p = Pattern.compile("[-+_.^:~@#$%*]");
    Matcher m = p.matcher(word);
    boolean b=m.find();
    if(b)
    {
    String word1=word. replaceAll("[-+_.^:~@#$%*]", "");
                for(int i=0; i < word.length(); ++i) {
                result.add(word.substring(0, i) + word.substring(i+1));
         }
                for(int i=0; i < word.length()-1; ++i)
         {
                result.add(word.substring(0, i) + word.substring(i+1, i+2) + word.substring(i, i+1)
        +word.substring(i+2));
          }
                for(int i=0; i < word.length(); ++i)
         {
                 for(char c='a'; c <= 'z'; ++c)
                {
                        result.add(word.substring(0, i) + String.valueOf(c) + word.substring(i+1));
                 }
         }
                for(int i=0; i <= word.length(); ++i) {
                for(char c='a'; c <= 'z'; ++c){
              result.add(word.substring(0, i) + String.valueOf(c) + word.substring(i));
         }
```

```
            }
    }
    else
    {
       System.out.println("else loooo");
       for(int i=0; i < word.length(); ++i)
       {
              result.add(word.substring(0, i) + word.substring(i+1));
        }
                    for(int i=0; i < word.length()-1; ++i)
        {
               result.add(word.substring(0, i) + word.substring(i+1, i+2) + word.substring(i, i+1) +
word.substring(i+2))
         }
                    for(int i=0; i < word.length(); ++i)
         {
             for(char c='a'; c <= 'z'; ++c)
              {
                 result.add(word.substring(0, i) + String.valueOf(c) + word.substring(i+1));
                     System.out.println("second for lop==========\n"+result.add(word.substring(0, i) +
           String.valueOf(c) + word.substring(i+1))+"\n===");
       }
           for(int i=0; i <= word.length(); ++i) {
           for(char c='a'; c <= 'z'; ++c){
                  result.add(word.substring(0, i) + String.valueOf(c) + word.substring(i));
                 System.out.println("theard for loppp========\n"+result.add(word.substring(0, i) +
                 String.valueOf(c) + word.substring(i))+"\n=======");
            }
            }
    }
           return result;
    }
          public final String correct(String word) {
                    if(nWords.containsKey(word)) return word;
                    ArrayList<String> list = edits(word);
                    HashMap<Integer, String> candidates = new HashMap<Integer, String>();
                    for(String s : list){
               if(nWords.containsKey(s))
                  candidates.put(nWords.get(s),s);}
                    if(candidates.size() > 0)
            {
```

```
                System.out.println("if size is
        >0=============\n"+candidates.get(Collections.max(candidates.keySet()))+"\n=======");
            return candidates.get(Collections.max(candidates.keySet()));
          }
                for(String s : list){
            for(String w : edits(s)){
            if(nWords.containsKey(w))
            {
            candidates.put(nWords.get(w),w);
           System.out.println("keyyyyyyyyyyyy========\n"+candidates+"\n=========");
            }
    }
    }
          String word1=candidates.size() > 0 ? candidates.get(Collections.max(candidates.keySet())) :
word;
          System.out.println("final word issssssss==========\n"+word1+"\n=============");
                return candidates.size() > 0 ? candidates.get(Collections.max(candidates.keySet())) :
word;
         }

       public static void main(String args[]) throws IOException {
        System.out.println("enter the string");
        Scanner sc=new Scanner(System.in);
        String str=sc.nextLine();
        System.out.println("enbtered string is===="+str);
        String cline[]="";
        StringBuffer sb=new StringBuffer();
        for(int i=0;i<mwords.length;i++)
        {
                if(mwords.length > 0)
                        System.out.println
                ((new Spelling("C:\\Documents and
          Settings\\Shekar\\Desktop\\words.txt")).correct(mwords[i])+"======\n");
            sb.append(new Spelling("D:\\new code\\find.txt").correct(mwords[i])+" ");
        }
        System.out.println("recunstructed line isss=====\n"+sb.toString()+"\n==========");
    }

}
```

6.2 Snapshots of the Implemented Module

The snapshots of the implemented module are:

6.2.1 Login Page

The html page opens the login page through which existing users and administrators can login with their userdId, password and non-existing users can sign-up.

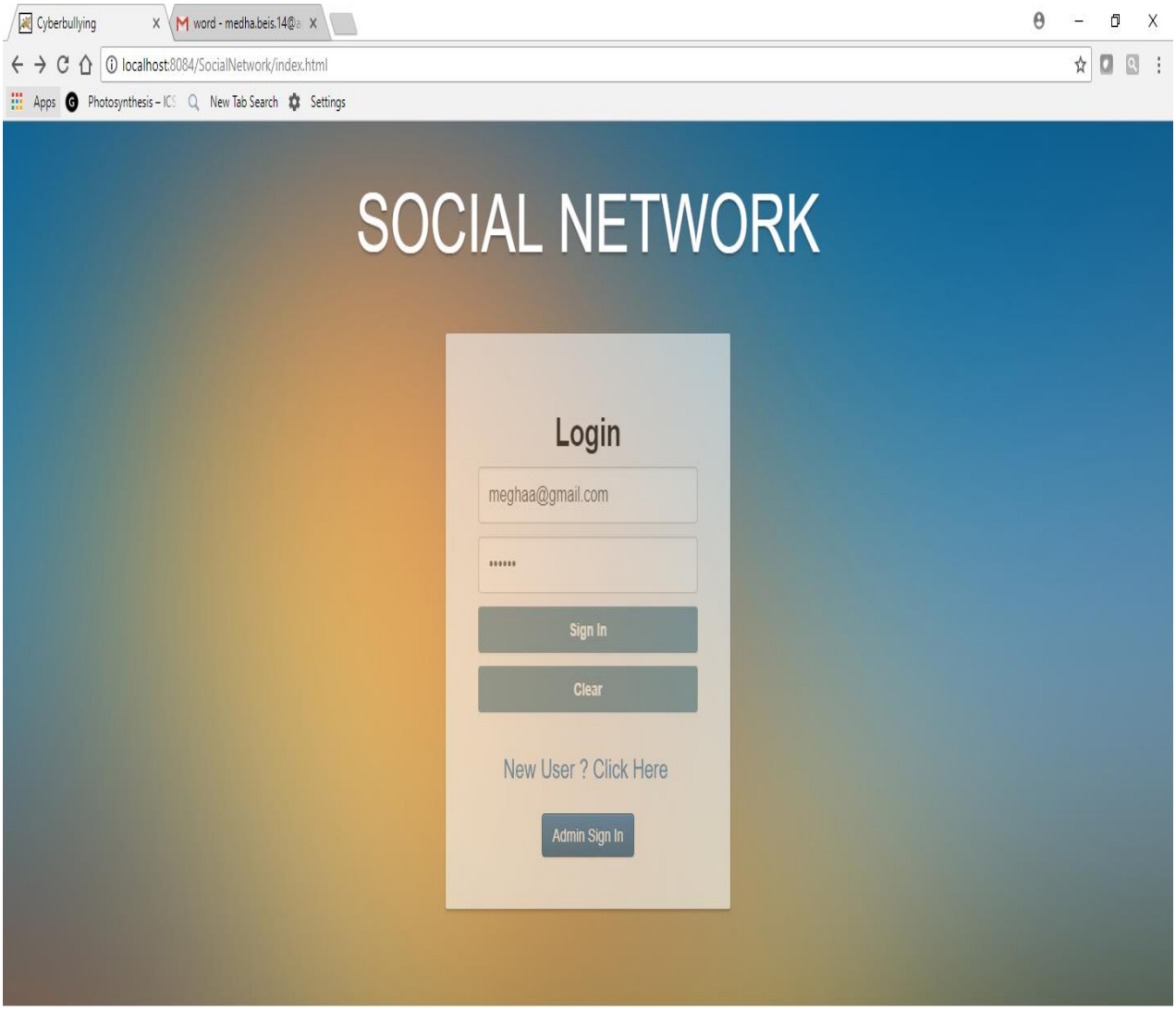

Figure 6.2: Login Page

6.2.2 New User Sign-up

The login page redirects to new user sign-up page whenever a non-existing user needs to sign-up. This page takes certain details to create a new account, validates the entered details, stores the information in the database and creates their profile.

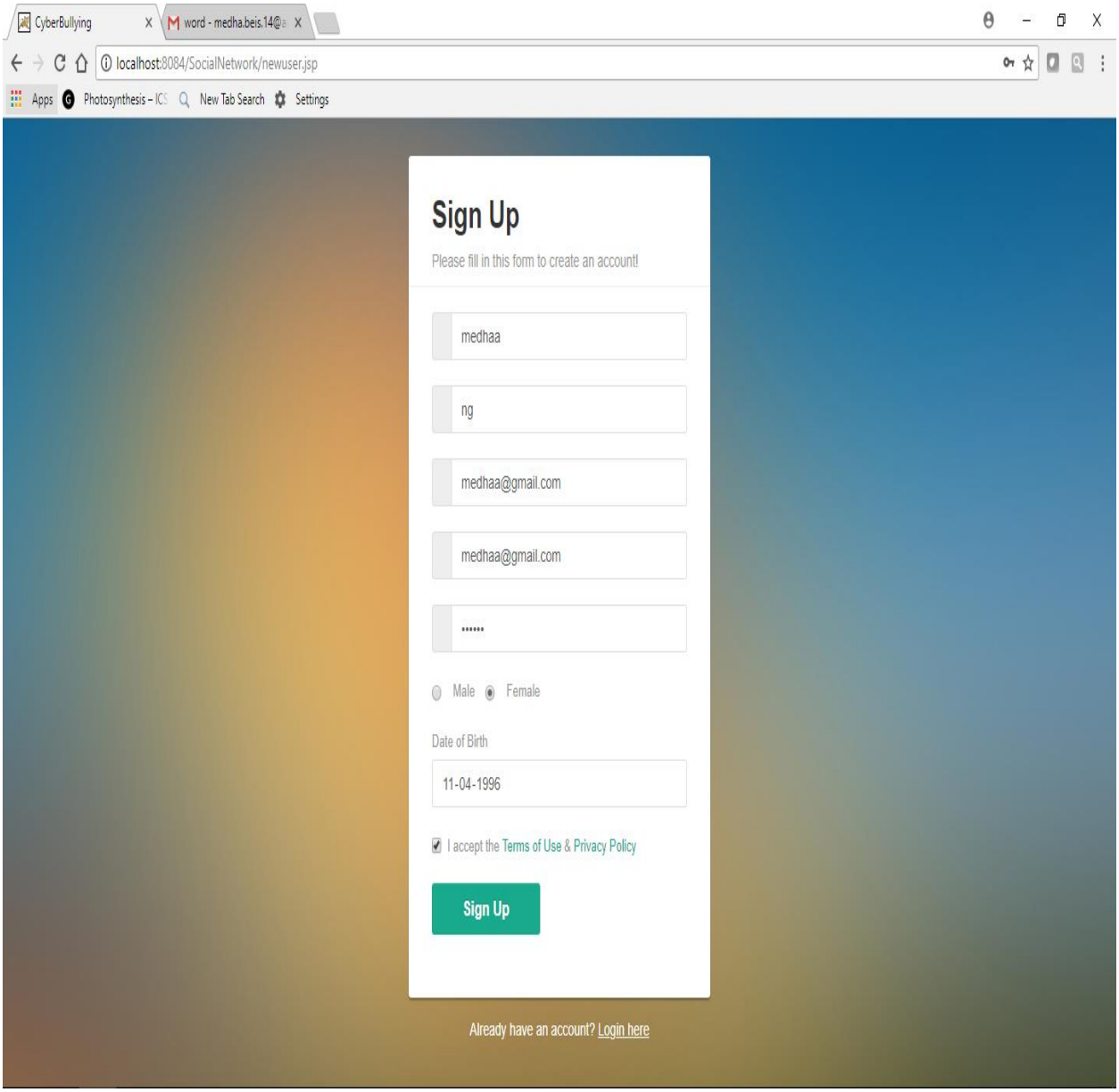

Figure 6.3: New User Sign-up

6.2.3 Add Profile Information

Once sign-up is complete through new user sign-up page, this page is used to collect further details of the user like Profile Information which is stored in the database and is used to update their profile.

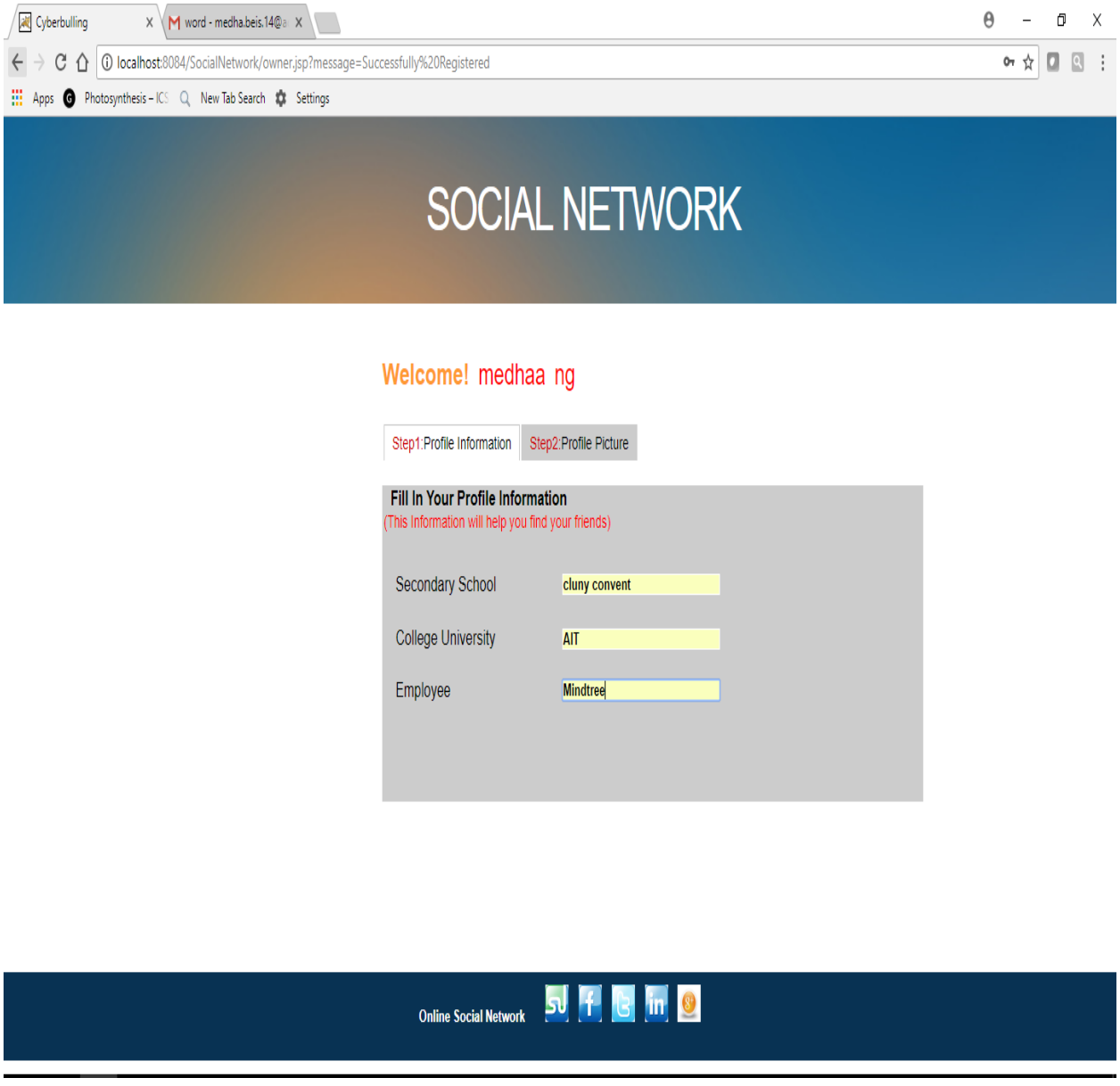

Figure 6.4: Add Profile Information

6.2.4 Add Profile Picture

Once sign-up is complete through new user sign-up page, this page is used to add Profile Picture by choosing an image after details of the user like Profile Information which is entered.

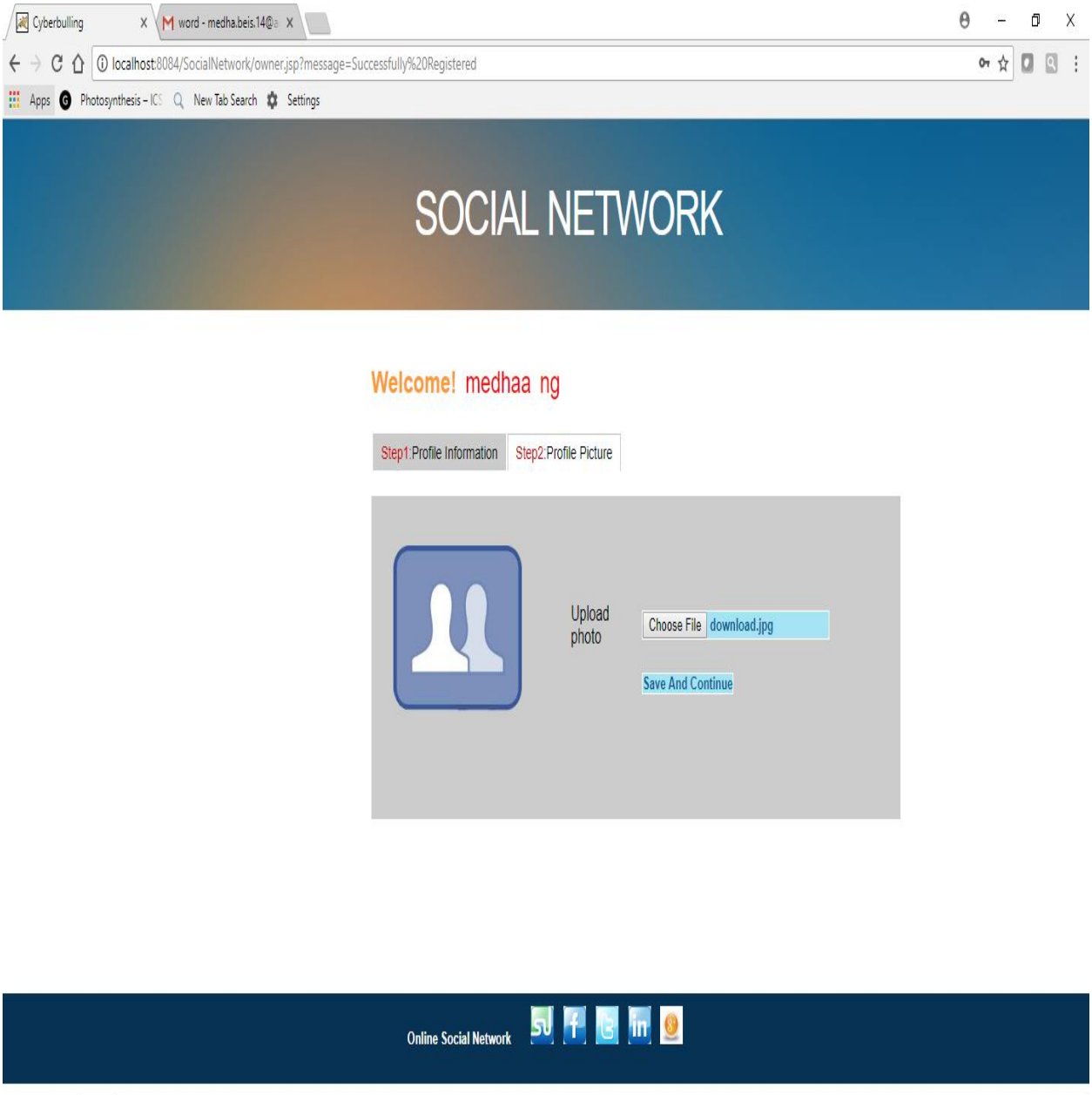

Figure 6.5: Add Profile Picture

6.2.5 User Page

The User page is used to retrieve the user's profile from the database and display it after a successful login.

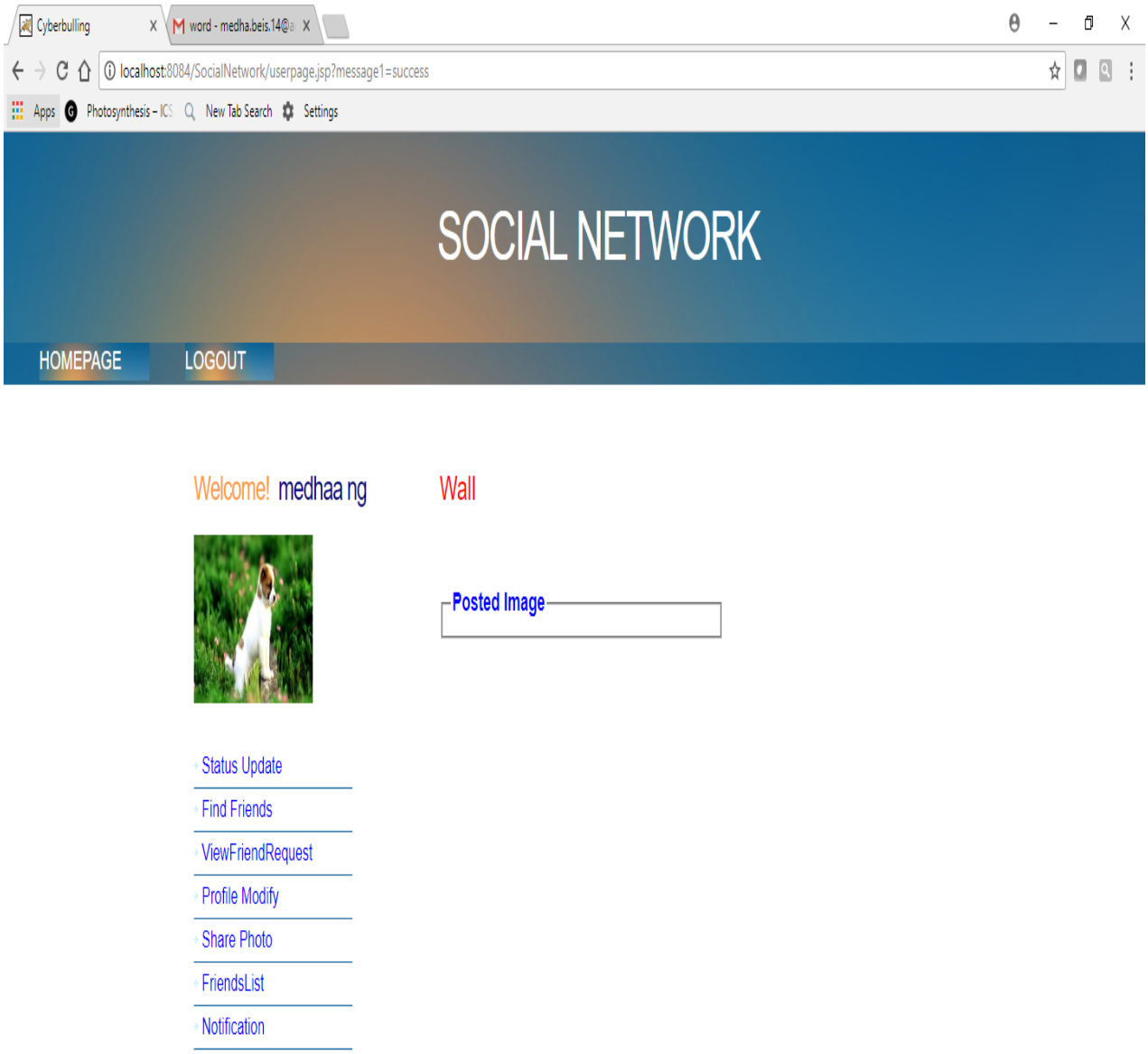

Figure 6.6: User Page

6.2.6 Status Update

The Status Update page is used to update a status which can be viewed by the other users of the social network.

Figure 6.7: Status Update

6.2.7 Image View

The Image View page is used to view the image uploaded by the user and also provides information about the count of the user's friends/ family members who can view this image.

Figure 6.8: Image View

6.2.8 Find Friends

The Find Friends page is used to view the friends list and also recommendations of people the users might know, through which the users can view their friend's profile or send friend requests.

Figure 6.9: Find Friends

6.2.9 View Friend Request

The View Friend Request page is used to view the persons who have sent friend requests to the user and the user can confirm or ignore these requests provided in the view request box.

Figure 6.10: View Friend Request

6.2.10 View Friend Profile

The user can view this page to know more details about their friends like date of birth, work place, etc. The user can also send private messages to their friends through the chat box provided.

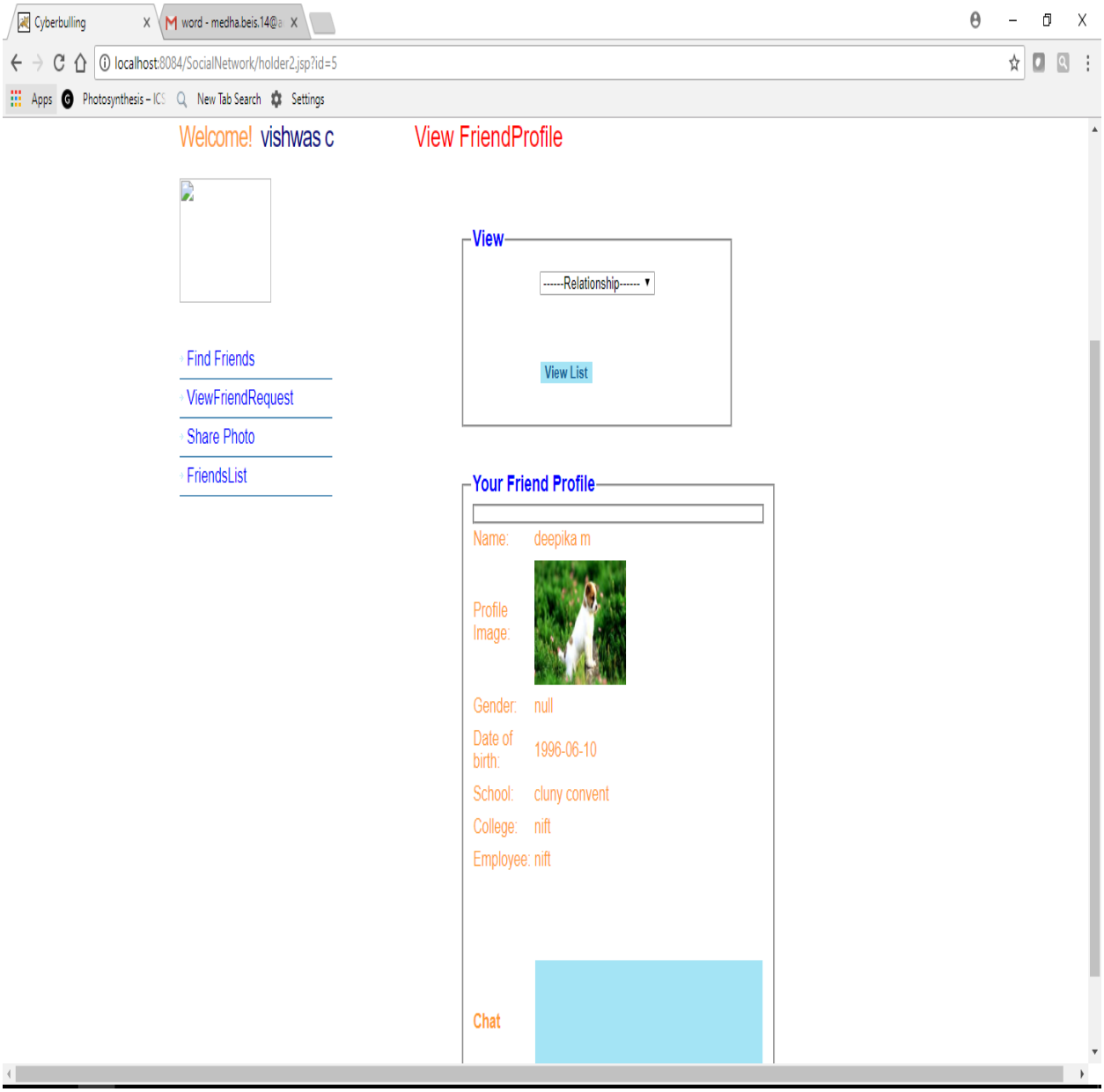

Figure 6.11: View Friend Profile

6.2.11 A post containing directbullying message

If the user posts any text message containing direct bullying words, then such messages will be detected using the deep learning model smSDA. These bullying messages will be blocked.

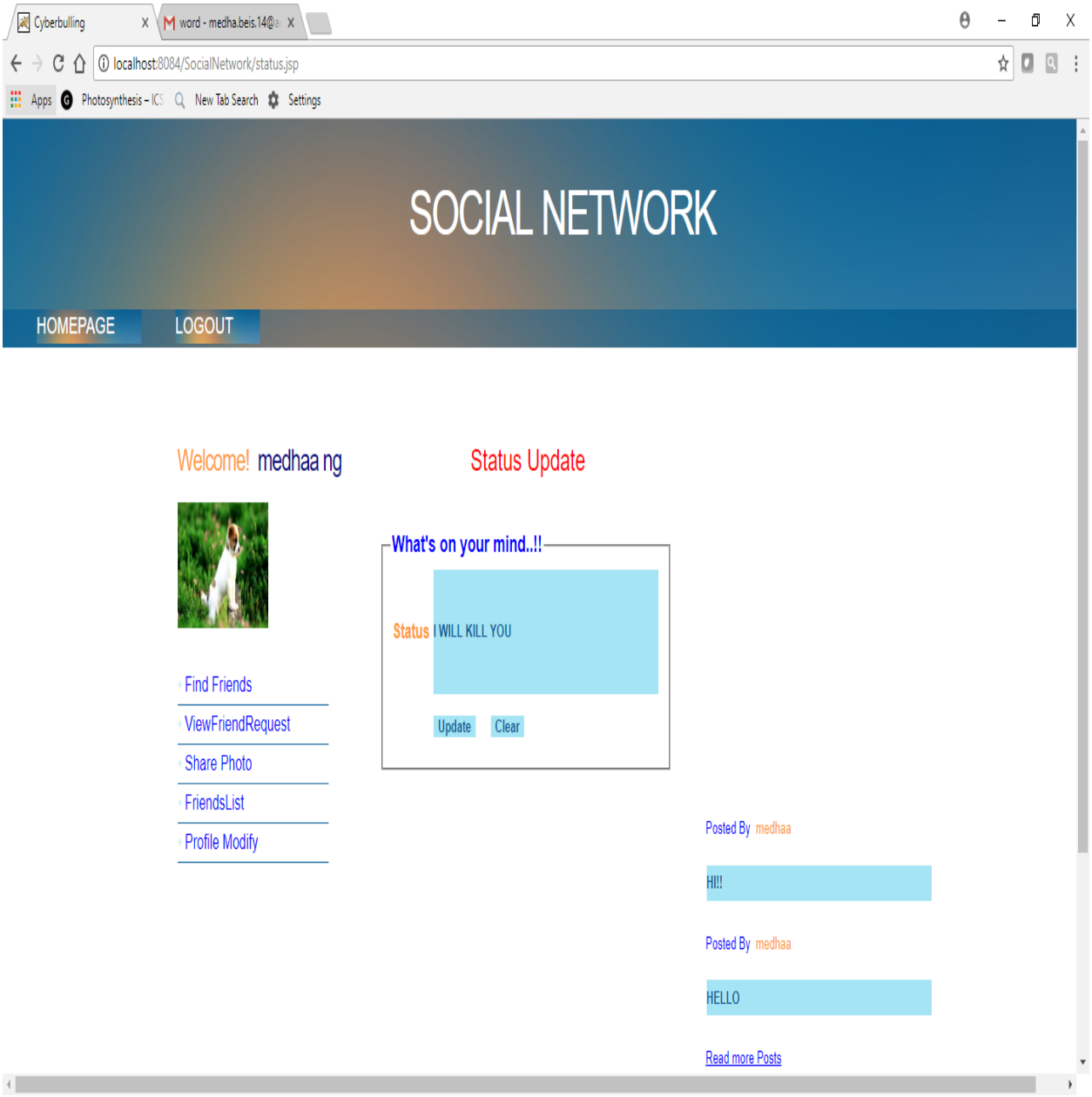

Figure 6.12: Status that contains direct bullying message

6.2.12 A post containing message with hidden characters

If the user posts any text message containing hidden characters instead of direct bullying words, then such messages will be reconstructed to obtain actual bullying word using the deep learning model smSDA. These bullying messages will then be blocked.

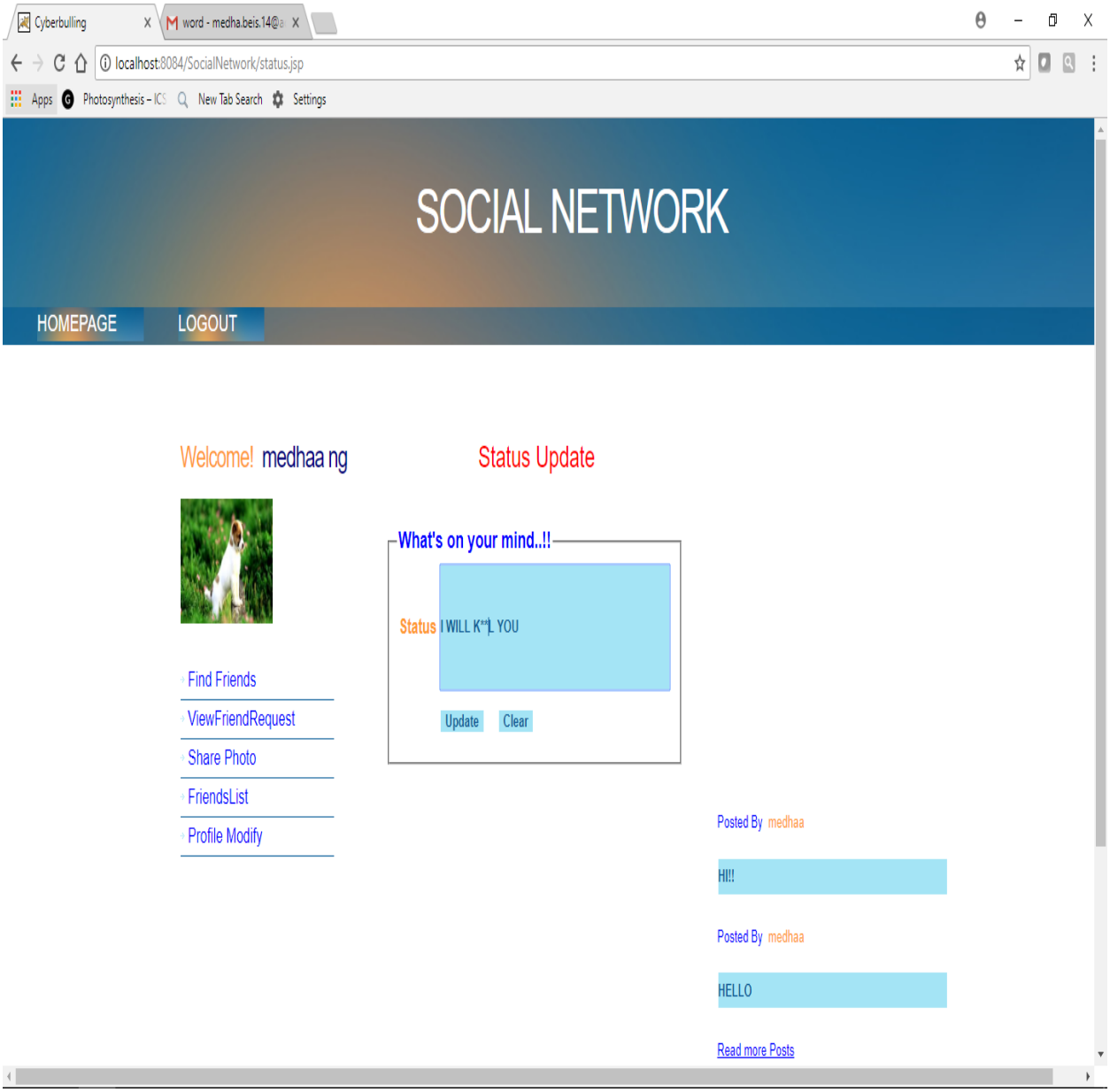

Figure 6.13: Status that contains message with hidden characters

6.2.13 A post containing message with misspelled words

If the user posts any text message containing misspelled words instead of direct bullying words, then such messages will be reconstructed to obtain actual bullying word using the deep learning model smSDA. These bullying messages will then be blocked.

Figure 6.14: Status that contains message with misspelled words

6.2.14 Blocking of a bullying user temporarily

A user who posts a bullying message will not be allowed to post any other message temporarily irrespective of whether the message is bullying or not. The user has to logout and login to post again.

Figure 6.15: Blocking of a bullying user temporarily

6.2.15 Blocking of a bullying user permanently

A user who posts bullying messages more than ten times will not be allowed to post any other message. This user will be blocked permanently from the social network.

Figure 6.16: Blocking of a bullying user permanently

6.2.16 Admin Login Page

The Admin Login page displays the login page for the administrator. This page is used to validate the entered userId and password entered by the administrator.

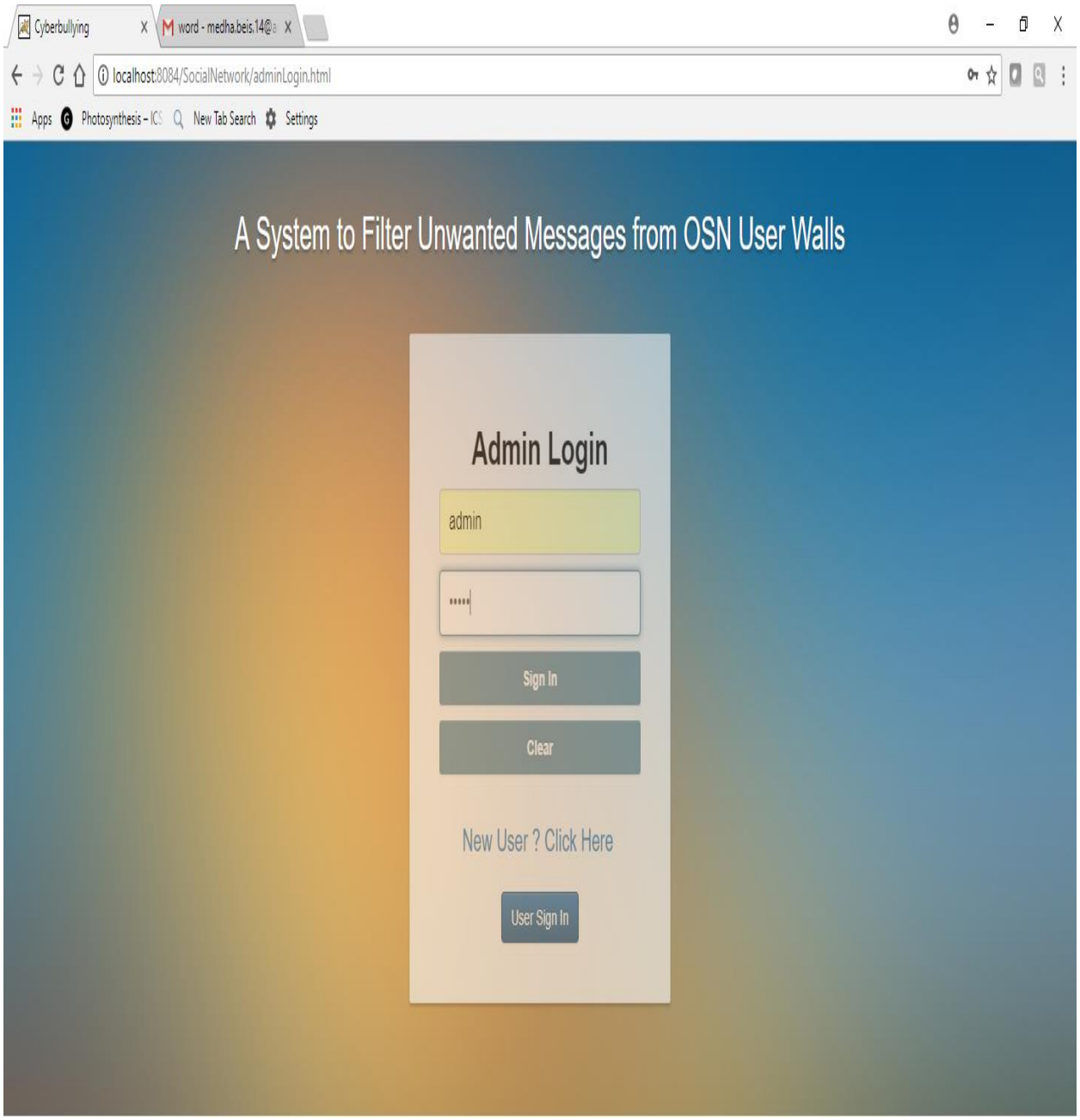

Figure 6.17: Admin Login Page

6.2.17 Admin User Page

The Admin Login Page re-directs to this JSP page once the administrator enters the details in the login html page. This page contains information like User list, Blocked users and Filter Performance.

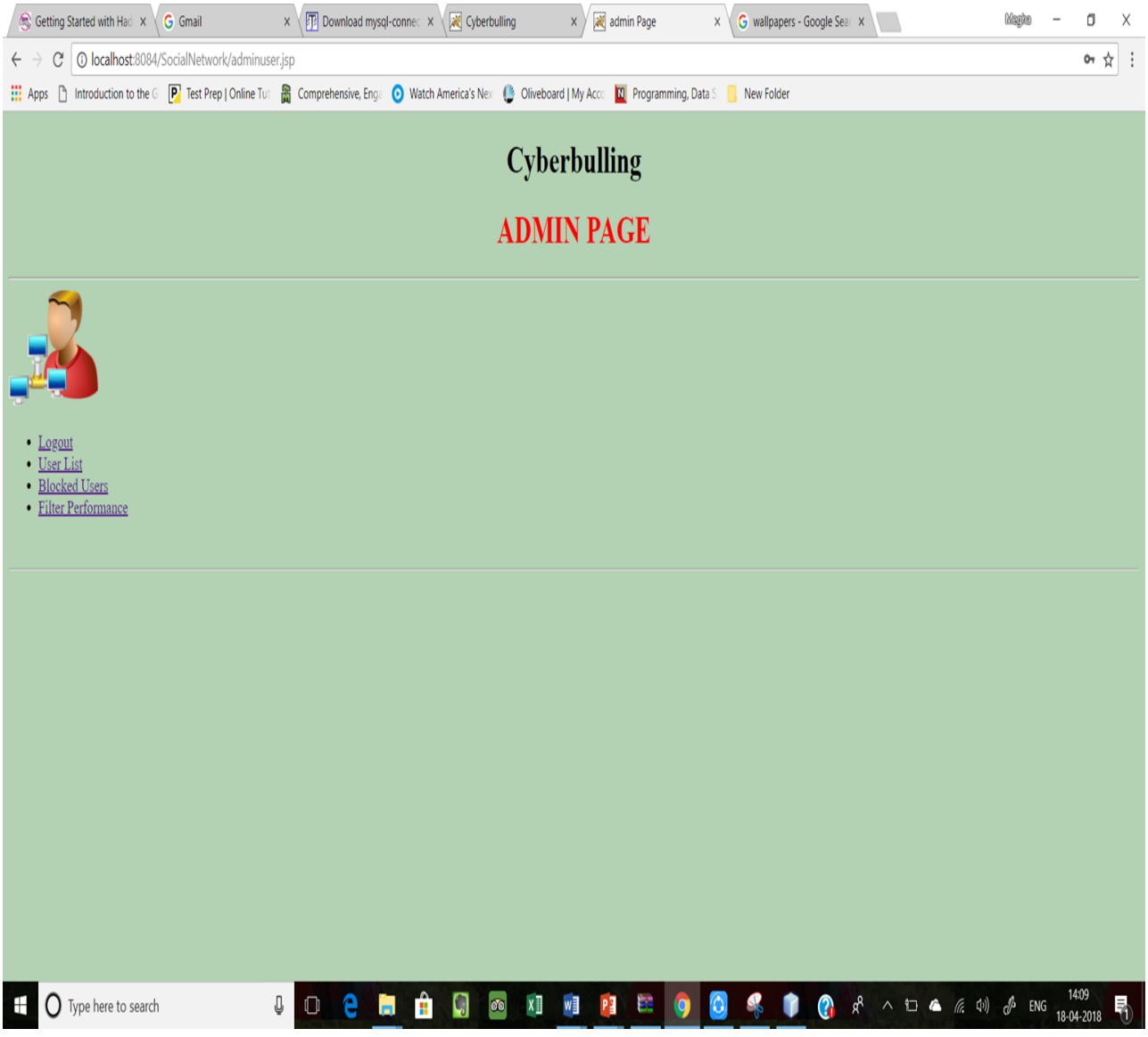

Figure 6.18: Admin User Page

6.2.18 User Details

This page contains information about the users of the Social Network including both the existing users and blocked users.

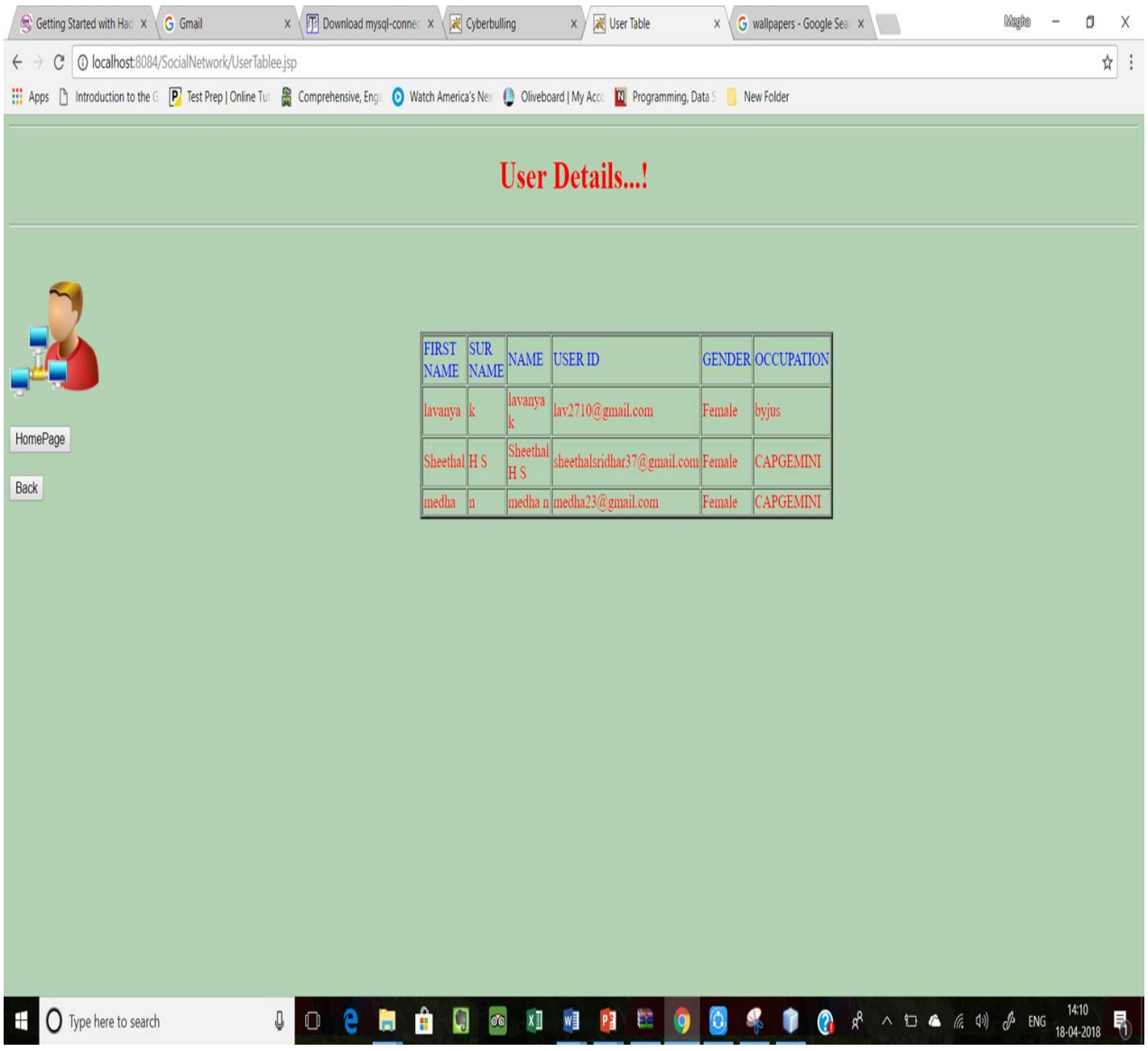

Figure 6.19: User Details

6.2.19 Blocked Messages List

The Blocked user List page contains a list of all the bullying messages posted on the Social Network which were blocked.

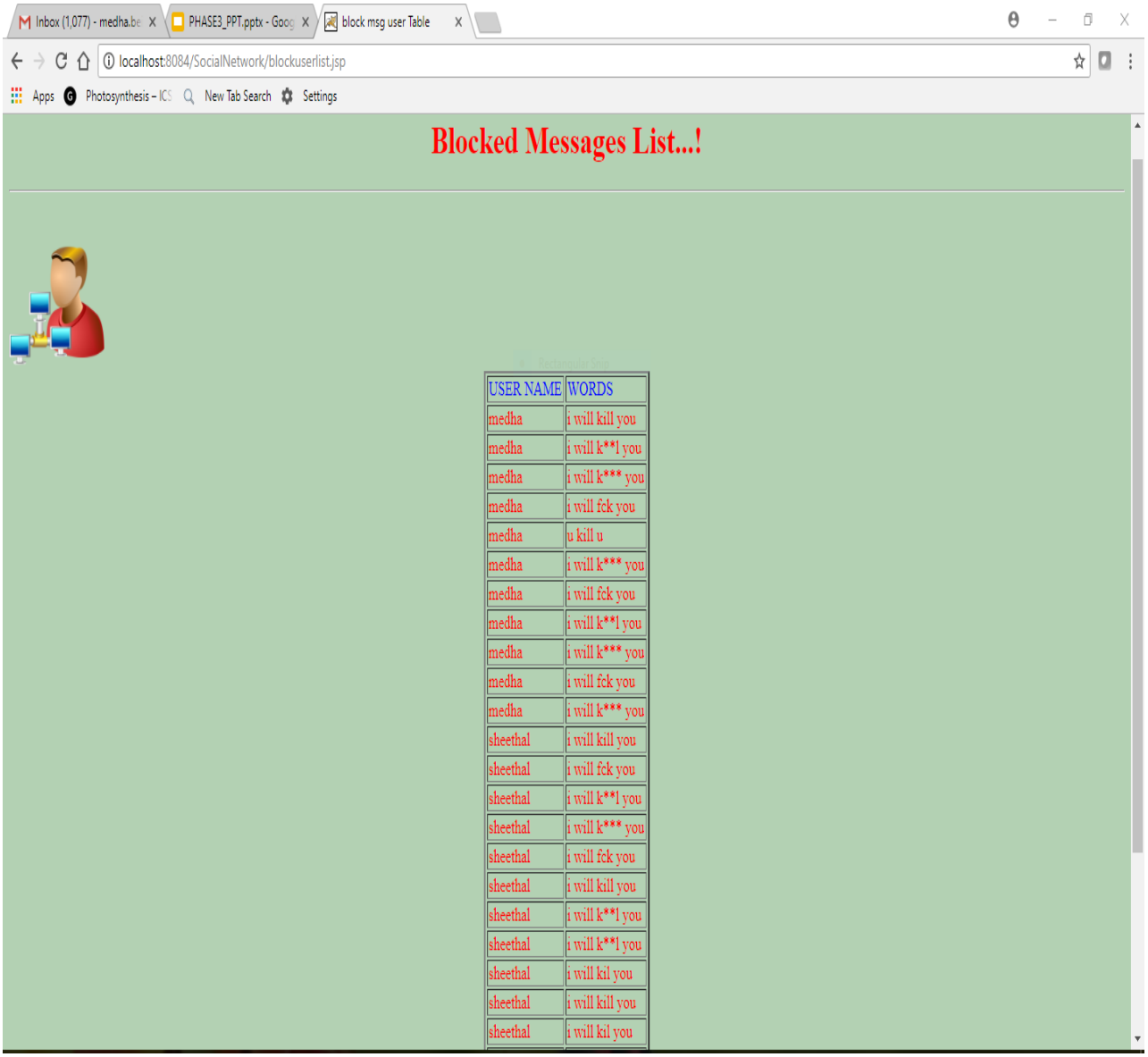

Figure 6.20: Blocked Messages List

6.2.20 Filter Performance

The Filter Performance page contains a pie chart which provides information about the percentage of Registered users and Blocked users of the Social Network.

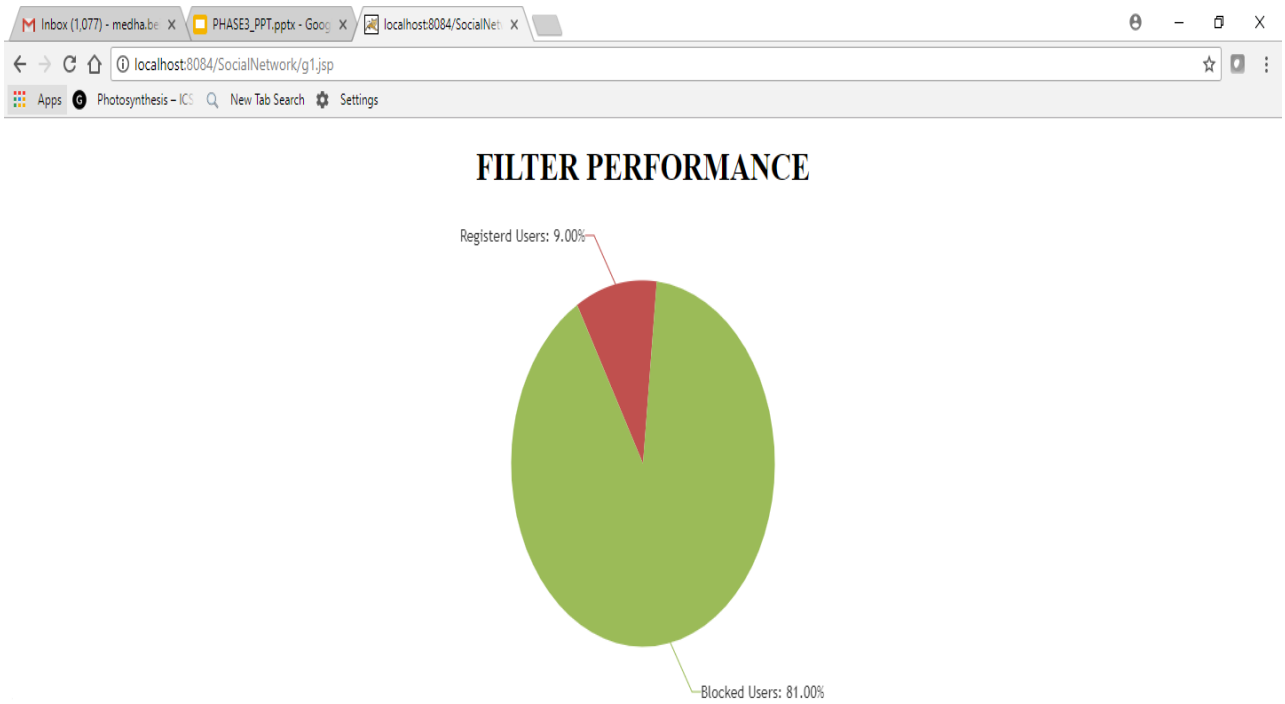

Figure 6.21: Filter Performance

CHAPTER 7
TESTING

CHAPTER 7

TESTING

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

7.1 Testing Principle

Before applying methods to design effective test cases, a software engineer must understand the basic principle that guides software testing. All the tests should be traceable to customer requirements.

7.2 Testing Methods

There are different methods that can be used for software testing. They are:

1. **Black-Box Testing**
2. **White-Box Testing**

Black-Box Testing

The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

White-Box Testing

White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called glass testing or open-box testing. In order to perform white-box testing on an application, a tester needs to know the internal workings of the code. The tester

needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

7.3 Levels of Testing

There are different levels during the process of testing. Levels of testing include different methodologies that can be used while conducting software testing. The main levels of software testing are:

1. **Functional Testing**
2. **Non-functional Testing**

Functional Testing:

This is a type of black-box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. Functional testing of software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. There are five steps that are involved while testing an application for functionality.

- The determination of the functionality that the intended application is meant to perform.
- The creation of test data based on the specifications of the application.
- The output based on the test data and the specifications of the application.
- The writing of test scenarios and the execution of test cases.
- The comparison of actual and expected results based on the executed test cases.

Non-functional Testing

This section is based upon testing an application from its non-functional attributes. Non-functional testing involves testing software from the requirements which are non-functional in nature but important such as performance, security, user interface, etc. Testing can be done in different levels of SDLC.

7.3.1 Unit Testing

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unit testing is often automated but it can also be done manually. The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality. Test cases and results are shown in the Table 7.1.

Serial Number	Components Tested	Method Used	Description	Status	Remarks
1	Login Page	Unit Testing	Enter the user name and the password of the registered user.	Pass	Verification takes 2-3 seconds
2	Signup Page	Unit Testing	Enter the details of the new user and thereby setup the profile for the first time.	Pass	A new profile for the user is created.
3	Status update	Unit Testing	Enter the status to be posted.	Pass	The post is allowed if there are no bullying words or else it is blocked.
4	Find friends	Unit Testing	View the list of registered users on the online social network.	Pass	Send friend requests to the registered users.
5	View friend request	Unit Testing	View the list of friend requests to be confirmed or rejected.	Pass	Confirm or reject the request .
6	Profile modify	Unit Testing	Enter the changes to be made to the profile information and change profile picture.	Pass	Changes are reflected on the users profile.
7	Share photo	Unit Testing	Choose the image to be uploaded.	Pass	The image appears on the friends wall based on the privacy.

Table 7.1: Test Cases for Unit Testing

7.3.2 Integration Testing

Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. It occurs after unit testing and before validation testing. Integration testing can be done in two ways: Bottom-up integration testing and Top-down integration testing.

Bottom-up Integration

This testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds.

Top-down Integration

In this testing, the highest-level modules are tested first and progressively, lower-level modules are tested thereafter.

Serial Number	Components Tested	Method Used	Description	Status	Remark
1	Login Phase	Integration Testing	Check if the user is able to login or not	Pass	Server verifies the user and sends a push notification on successful login.
2	Signup Phase	Integration Testing	Check if the user is already existing user or not create a new profile for the user.	Pass	On successful signup the user is requested to update his profile picture and other details.
3	Blocking Phase	Integration Testing	Check if the posts contains any bullying words.	Pass	These posts are blocked from being posted and the user is temporarily is unable to post any status updates. On persistent attempts to post the user is blocked permanently.
4	Admin Phase	Integration Testing	Check if the admin is able to view all the user profiles.	Pass	The admin will be able to view the no of users and posts which are blocked along with the performance rate.

Table 7.2: Test Cases for Integration Testing

In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic actual situations. Table 7.2 shows the test cases for integration testing and their results.

CONCLUSION AND FUTURE ENHANCEMENT

Conclusion

The proposed system addresses the text-based cyberbullying detection problem, where robust and discriminative representations of messages are critical for an effective detection system. By designing semantic dropout noise and enforcing sparsity constraints, semantic-enhanced marginalized denoisingautoencoder as a specialized representation learning model for cyberbullying detection has been developed. In addition, word embeddings have been used to automatically expand and refine bullying wordlists that is initialized by domain knowledge. The performance of our approaches has been experimentally verified through an online social network designed for this project. As a next step we are planning to further improve the robustness of the learned representation by considering word order in messages.

Future Enhancement

- The whole focus of the project is based on automatic detection of textual-based cyberbullying and the same has been verified with the help of a sample OSN.
- The major components of today's Social media platform is images, videos, GIF's and various other visual posts through which bullying can occur.
- These neglected mediums can be focused on in the coming days.
- Technology like image processing to extract the words from an image can be very useful.

REFERENCES

[1] G Netaji,"Detection based on Semantic-Enhanced Marginalized DenoisingAutoEncoder", International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 2017.

[2] D. Yin, Z. Xue, L. Hong, B. D. Davison, A. Kontostathis, and L. Edwards, "Detection of harassment on web 2.0," Proceedings of the Content Analysis in the WEB, vol. 2, pp. 1–7, 2009.

[3] K. Dinakar, R. Reichart, and H. Lieberman, "Modeling the detection of textual cyberbullying." in The Social Mobile Web, 2011.

[4] V. Nahar, X. Li, and C. Pang, "An effective approach for cyberbullying detection", Communications in Information Science and Management Engineering, 2012.

[5] Rui Zhao and Kezhi Mao, "Cyber Bullying Detection based on Semantic-Enhanced Marginalized Denoising Auto-Encoder", IEEE Transactions on Affective Computing, 2016.

[6] S. ShaibaHajira Begum, "Enhanced Auto Learning Encoder for Cyber Bullying Detection" in IJCSEST, Dec 2017.

[7] Veeramallu Naga Srinivas, "Detection of text based Cyberbullying using semantic enhanced denoising auto-encoder learning model" in IJCSMC, Aug 2017.

[8] https://www.kaggle.com/

[9] https://www.coursera.org/learn/machine-learning

[10] https://goo.gl/forms/U6w7F03iLyh4XzRM2 - A Survey on Cyberbullying.