##### A

##### Industrial-Oriented Mini Project On

**USE OF ANN TO IDENTIFY FAKE PROFILES**

(Submitted in partial fulfilment of the requirements for the award of Degree)

**BACHELOR OF TECHNOLOGY**

##### In

**COMPUTER SCIENCE AND ENGINEERING**

##### By

##### P.Naveen Reddy (227R1A05A8) Rukkaya Sulthana (217R1A05B7)

##### R.Sai Rani (217R1A05B4)

##### Under the Guidance of

#### Dr.Mantesh

(Associate Professor)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**CMR TECHNICAL CAMPUS**

###### UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi)

Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V), Medchal Road, Hyderabad-501401.

**May, 2025.**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

****

**CERTIFICATE**

This is to certify that the project entitled “**USE OF ANN TO IDENTIFY FAKE PROFILES**” being submitted by **P. Naveen Reddy (227R1A05A8), Rukkaya Sulthana (227R1A05B7), R. Sai Rani (227R1A05B4)** in partial fulfilment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, during the year 2024-25.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Dr.Suma Dr. Nuthanakanti Bhaskar Associate Professor HoD

INTERNAL GUIDE

Dr. A. Raji Reddy Signature of External Examiner DIRECTOR

Submitted for viva voice Examination held on

**ACKNOWLEDGEMENT**

We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project, we take this opportunity to express our profound gratitude and deep regard to our guide **Dr.Mantesh,Associate Professor** for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We take this opportunity to extend our heartfelt appreciation to the Project Review Committee (PRC) Coordinators—**B Sekhar, D Nageswar Rao**—for their unwavering support, insightful guidance, and valuable inputs, which played a crucial role in steering this project through its various stages.

Our sincere appreciation also goes to **Dr. Nuthanakanti Bhaskar**, Head, for his encouragement and continuous support in ensuring the successful completion of our project.

We are deeply grateful to **Dr. A. Raji Reddy**, Director, for his cooperation throughout the course of this project. Additionally, we extend our profound gratitude to Sri. **Ch. Gopal Reddy**, Chairman, Smt. **C. Vasantha Latha**, Secretary and Sri. **C. Abhinav Reddy**, Vice-Chairman, for fostering an excellent infrastructure and a conducive learning environment that greatly contributed to our progress.

We also acknowledge and appreciate the guidance and assistance provided by the faculty and staff of **CMR Technical Campus**, whose contributions have been invaluable in bringing this project to fruition.

Lastly, we sincerely thank our families for their unwavering support and encouragement. We also extend our gratitude to the teaching and non-teaching staff of CMR Technical Campus for their guidance and assistance. Their contributions, along with the support of everyone who helped directly or indirectly, have been invaluable in the successful completion of this project.

P. Naveen Reddy (22R1A05A8)

RukkayaSulthana (227R1A05B7)

R.Sai Rani (227R1A05B4)

**VISION AND MISSION**

**INSTITUTE VISION:**

##### To Impart quality education in serene atmosphere thus strive for excellence in Technology and Research.

**INSTITUTE MISSION:**

##### To create state of art facilities for effective Teaching- Learning Process.

##### Pursue and Disseminate Knowledge based research to meet the needs of Industry & Society.

##### Infuse Professional, Ethical and Societal values among Learning Community.

**DEPARTMENT VISION:**

##### To provide quality education and a conducive learning environment in computer engineering that foster critical thinking, creativity, and practical problem-solving skills.

**DEPARTMENT MISSION:**

##### To educate the students in fundamental principles of computing and induce the skills needed to solve practical problems.

##### To provide State-of-the-art computing laboratory facilities to promote industry institute interaction to enhance student’s practical knowledge.

##### To inculcate self-learning abilities, team spirit, and professional ethics among the students to serve society.

**ABSTRACT**

The widespread use of online social networks (OSNs) like Facebook, Twitter, and Instagram has revolutionized digital communication, allowing billions of users to connect, share information, and build relationships globally. However, this rapid growth has also led to significant security concerns, most notably the rise of fake profiles that pose serious threats to user privacy, trust, and platform integrity. Fake profiles are often used by malicious actors to conduct phishing attacks, identity theft, misinformation campaigns, or unauthorized data scraping through bots and automated systems. Traditional detection methods, including manual verification and rule-based filters, have proven insufficient due to the growing complexity and adaptability of such profiles. In this project, we present a machine learning-based solution that utilizes **Artificial Neural Networks (ANNs)** to accurately identify fake profiles by analyzing key profile features. Developed using Python and the Django web framework, our system extracts features such as account age, gender, status count, friend count, user age, location, and IP address to train the ANN model. The model architecture includes multiple dense layers and uses ReLU and softmax activation functions, optimized with categorical cross-entropy loss and the Adam optimizer. After training on a curated dataset, the model achieved a high prediction accuracy of **98%**, demonstrating its ability to distinguish between genuine and fake profiles effectively. The system also features an admin panel to train the model and monitor dataset performance, along with a user-friendly interface that allows users to enter profile data and receive immediate authenticity feedback. Extensive validation including unit, integration, and system testing was performed to ensure the robustness, reliability, and usability of the platform. The results confirm that our ANN-based approach is not only practical but also scalable for broader implementation across multiple social platforms.

**LIST OF FIGURES**

**FIGURE NO FIGURE NAME PAGE NO**

Figure 2.3 Process Model Used With 08

Justification

Figure 3.1.1 Class Diagram 18

Figure 3.1.2 Use case Diagram for Admin 19

and User

Figure 3.1.3 Sequence diagram 20

Figure 3.1.4 Collaboration diagram 21

Figure 3.1.5 Component Diagram for Admin 22

and User

Figure 3.1.6 Deployment Diagram 23

Figure 3.1.7 Activity Diagram for Admin 24

and User

Figure 3.2 Data Flow Diagram 25

Figure 5.1 Admin Link on Home Screen 33

Figure 5.2 Admin Login Page 34

Figure 5.3 Admin Dashboard After 34

Successful Login

Figure 5.4 Generating the ANN Train Model 35

Figure 5.5 ANN Model Achieving 35

98% Accuracy

Figure 5.6 View ANN Training Dataset Screen 36

Figure 5.7 User Screen for Profile Prediction 37

Figure 5.8 Giving Input for Genuine Profile 37

Figure 5.9 Prediction Result – Genuine 38

Figure 5.10 Giving Input for Fake Profile 38

Figure 5.11 Prediction Result - Fake Profile 39

|  |  |  |
| --- | --- | --- |
|  | **LIST OF TABLES** |  |
| **TABLE NO** | **TABLE NAME** | **PAGE NO** |
| Table 6.2 | Sample Test Results | 40 |

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **ABSTRACT** | i |
| **LIST OF FIGURES** | ii |
| **LIST OF TABLES** | iv |
| **1. INTRODUCTION** | 1 |
| 1.1 OBJECTIVE OF THE PROJECT | 2 |
| **2. LITERATURE SURVEY** | 3 |
| 2.1 EXISTING SYSTEM | 7 |
| 2.2 PROPOSED SYSTEM | 7 |
| 2.3 PROCESS MODEL USED WITH JUSTIFICATION | 8 |
| 2.4 SOFTWARE REQUIREMENT SPECIFICATION | 15 |
| 2.4.1 OVERALL DESCRIPTION  2.4.2 EXTERNAL INTERFACE REQUIREMENTS | 15  17 |
| 2.5 HARDWARE & SOFTWARE REQUIREMENTS | 17 |
| 2.5.1 HARDWARE REQUIREMENTS | 17 |
| 2.5.2 SOFTWARE REQUIREMENTS | 17 |

3.

|  |  |  |
| --- | --- | --- |
| **SYSTEM** | **ARCHITECTURE & DESIGN** | 18 |
| 3.1 | SYSTEM DESIGN | 18 |
| 3.2 | DATA FLOW DIAGRAM | 25 |

1. [IMPLEMENTATION](#_TOC_250011) 26
   1. [PYTHON 28](#_TOC_250010)
   2. [SAMPLE CODE 25](#_TOC_250009)
2. [RESULTS & DISCUSSION 33](#_TOC_250008)
3. [VALIDATION 40](#_TOC_250007)
   1. [VALIDATION METHODOLOGY 41](#_TOC_250006)
   2. SAMPLE TEST RESULTS 41
4. [CONCLUSION & FUTURE ASPECTS 42](#_TOC_250005)
   1. [CONCLUSION 42](#_TOC_250004)
   2. [FUTURE ASPECTS 42](#_TOC_250003)
5. [BIBLIOGRAPHY 44](#_TOC_250002)
   1. [REFERENCES 44](#_TOC_250001)
6. **INTRODUCTION**

**1.INTRODUCTION**

In 2017 Facebook reached a total population of 2.46 billion users making it the most popular choice of social media. Social media networks make revenues from the data provided by users. The average user does not know that their rights are given up the moment they use the social media network's service. Social media companies have a lot to gain at the expense of the user. Every time a user shares a new location, new photos, likes, dislikes, and tag other users in content posted, Facebook makes revenue via advertisements and data. More specifically, the average American user generates about $26.76 per quarter . That number adds up quickly when millions of users are involved. In today's digital age, the ever-increasing dependency on computer technology has left the average citizen vulnerable to crimes such as data breaches and possible identity theft. These attacks can occur without notice and often without notification to the victims of a data breach. At this time, there is little incentive for social networks to improve their data security. These breaches often target social media networks such as Facebook and Twitter. They can also target banks and other financial institutions. There seems to be a newsworthy issue involving social media networks getting hacked every day. Recently, Facebook had a data breach which affected about 50 million users. Facebook provides a set of clearly defined provisions that explain what they do with the user's data. The policy does very little to prevent the constant exploitation of security and privacy. Fake profiles seem to slip through Facebook's built-in security features.

The other dangers of personal data being obtained for fraudulent purposes is the presence of bots and fake profiles. Bots are programs that can gather information about the user without the user even knowing. This process is known as web scraping. What is worse, is that this action is legal. Bots can be hidden or come in the form of a fake friend request on a social network site to gain access to private information. The solution presented in this paper intends to focus on the dangers of a bot in the form of a fake profile on your social media. This solution would come in the form of an algorithm. The language that we chose to use is Python. The algorithm would be able to determine if a current friend request that a user gets online is an actual person or if it is a bot or it is a fake friend request fishing for information. Our

algorithm would work with the help of the social media companies, as we would need a training dataset from them to train our model and later verify if the profiles are fake or not. The algorithm could even work as a traditional layer on the user's web browser as a browser plug-in.

**1.1 Objective of the Project**

Social media networks make revenues from the data provided by users. The average user does not know that their rights are given up the moment they use the social media network's service. Social media companies have a lot to gain at the expense of the user. Every time a user shares a new location, new photos, likes, dislikes, and tag other users in content posted, Facebook makes revenue via advertisements and data. Social media networks make revenues from the data provided by users. The average user does not know that their rights are given up the moment they use the social media network's service. Social media companies have a lot to gain at the expense of the user. Every time a user shares a new location, new photos, likes, dislikes, and tag other users in content posted, Facebook makes revenue via advertisements and data. In this paper, we use machine learning, namely an artificial neural network to determine what the chances that Facebook friend request is authentic are or not.

**2.LITERATURE SURVEY**

**2.LITERATURE SURVEY**

**Aiding the detection of fake accounts in large scale social online services**

Users increasingly rely on the trustworthiness of the information exposed on Online Social Networks (OSNs). In addition, OSN providers base their business models on the marketability of this information. However, OSNs suffer from abuse in the form of the creation of fake accounts, which do not correspond to real humans. Fakes can introduce spam, manipulate online rating, or exploit knowledge extracted from the network. OSN operators currently expend significant resources to detect, manually verify, and shut down fake accounts. Tuenti, the largest OSN in Spain, dedicates 14 full-time employees in that task alone, incurring a significant monetary cost. Such a task has yet to be successfully automated because of the difficulty in reliably capturing the diverse behavior of fake and real OSN profiles. We introduce a new tool in the hands of OSN operators, which we call SybilRank. It relies on social graph properties to rank users according to their perceived likelihood of being fake (Sybils). SybilRank is computationally efficient and can scale to graphs with hundreds of millions of nodes, as demonstrated by our Hadoop prototype. We deployed SybilRank in Tuenti's operation center. We found that ∼90% of the 200K accounts that SybilRank designated as most likely to be fake, actually warranted suspension. On the other hand, with Tuenti's current user-report-based approach only ∼5% of the inspected accounts are indeed fake.

**Audit and Analysis of Impostors: An experimental approach to detect fake profile in online social network**

In the present generation, the social life of every person has become associated with online social networks (OSN). These sites have made drastic changes in the way we socialize. Making friends and keeping in contact with them as well as being updated of their activities, has become easier. But with their rapid growth, problems like fake profiles, online impersonation have also increased. The risk lies in the fact that anybody can create a profile to impersonate a real person on the OSN. The fake profile could be exploited to build online relationship with a targeted person purely through online interactions with the friends of victim. In present work, we have proposed experimental framework with which detection of fake profile is feasible within the friend list, however this framework is restricted to a specific online social networking site namely Facebook. This framework extracts data from the friend list and uses it to classify them as real or fake by using unsupervised and supervised machine learning.

**Fake profile detection techniques in large-scale online social networks: A comprehensive review**

In the present era, online social networks are the most popular and rapid information propagation applications on the Internet. People of all ages spend most of their time on social networking sites. Huge volumes of data are being created and shared through social networks around the world. These interests have given rise to illegitimate users who engage in fraudulent activities against social network users. On social networks, fake profile creation is considered to cause more harm than any other form of cyber crime. This crime has to be detected even before the user is notified about the fake profile creation. Many algorithms and methods, most of which use the huge volume of unstructured data generated from social networks, have been proposed for the detection of fake profiles. This study presents a survey of the existing and latest technical work on fake profile detection.

**Online Social Networks: Threats and Solutions**

Many online social network (OSN) users are unaware of the numerous security risks that exist in these networks, including privacy violations, identity theft, and sexual harassment, just to name a few. According to recent studies, OSN users readily expose personal and private details about themselves, such as relationship status, date of birth, school name, email address, phone number, and even home address. This information, if put into the wrong hands, can be used to harm users both in the virtual world and in the real world. These risks become even more severe when the users are children. In this paper, we present a thorough review of the different security and privacy risks, which threaten the well-being of OSN users in general, and children in particular. In addition, we present an overview of existing solutions that can provide better protection, security, and privacy for OSN users. We also offer simple-to-implement recommendations for OSN users, which can improve their security and privacy when using these platforms. Furthermore, we suggest future research directions.

**Advanced social engineering attacks**

Social engineering has emerged as a serious threat in virtual communities and is an effective means to attack information systems. The services used by today's knowledge workers prepare the ground for sophisticated social engineering attacks. The growing trend towards BYOD (bring your own device) policies and the use of online communication and collaboration tools in private and business environments aggravate the problem. In globally acting companies, teams are no longer geographically co-located, but staffed just-in-time. The decrease in personal interaction combined with a plethora of tools used for communication (e-mail, IM, Skype, Dropbox, LinkedIn, Lync, etc.) create new attack vectors for social engineering attacks. Recent attacks on companies such as the New York Times and RSA have shown that targeted spear-phishing attacks are an effective, evolutionary step of social engineering attacks. Combined with zero-day-exploits, they become a dangerous weapon that is often used by advanced persistent threats. This paper provides a taxonomy of well-known social engineering attacks as well as a comprehensive overview of advanced social engineering attacks on the knowledge worker.

**An analysis of social network-based Sybil defenses**

Recently, there has been much excitement in the research community over using social networks to mitigate multiple identity, or Sybil, attacks. A number of schemes have been proposed, but they differ greatly in the algorithms they use and in the networks upon which they are evaluated. As a result, the research community lacks a clear understanding of how these schemes compare against each other, how well they would work on real-world social networks with different structural properties, or whether there exist other (potentially better) ways of Sybil defense. In this paper, we show that, despite their considerable differences, existing Sybil defense schemes work by detecting local communities (i.e., clusters of nodes more tightly knit than the rest of the graph) around a trusted node. Our finding has important implications for both existing and future designs of Sybil defense schemes. First, we show that there is an opportunity to leverage the substantial amount of prior work on general community detection algorithms in order to defend against Sybils. Second, our analysis reveals the fundamental limits of current social network-based Sybil defenses: We demonstrate that networks with well-defined community structure are inherently more vulnerable to Sybil attacks, and that, in such networks, Sybils can carefully target their links in order make their attacks more effective.

**Discovery of anomalous behaviour in temporal networks**

In this work we consider the problem of detecting anomalous behaviour and present a novel approach that allows ‘behaviour’ to be classified as either to be normal or abnormal by checking the p-value associated with the occurrence of the behaviour which is modelled following a binomial distribution within a discrete time model. We investigate the problem of detecting anomalous behaviour by looking at how communication evolves over time in a social network graph. Under the assumption that some nodes of the network could be labelled qualitatively, we present a novel approach that allows us to infer a subset of nodes of the social network which might share the same qualitative connotation. In other words, assuming one or more members belong to some criminal organisation, we wish to investigate how many other persons belong to the same organisation. We have tested our method in two datasets, VAST2008 and a Twitter Dataset (data collected in 2012), with encouraging results.

**Scalable community detection in massive social networks using MapReduce**

In this paper, we present a community-detection solution for massive-scale social networks using MapReduce, a parallel programming framework. We use a similarity metric to model the community probability, and the model is designed to be parallelizable and scalable in the MapReduce framework. More importantly, we propose a set of degree-based preprocessing and postprocessing techniques named DEPOLD (DElayed Processing of Large Degree nodes) that significantly improve both the community-detection accuracy and performance. With DEPOLD, delaying analysis of 1% of high-degree nodes to the postprocessing stage reduces both processing time and storage space by one order of magnitude. DEPOLD can be applied to other graph-clustering problems. Furthermore, we design and implement two similarity calculation algorithms using MapReduce with different computation and communication characteristics in order to adapt to various system configurations. Finally, we conduct experiments with publicly available datasets. Our evaluation demonstrates the effectiveness, efficiency, and scalability of the proposed solution.

**2.1 Existing System**

In today's digital age, the ever-increasing dependency on computer technology has left the average citizen vulnerable to crimes such as data breaches and possible identity theft. These attacks can occur without notice and often without notification to the victims of a data breach.

**Disadvantages of Existing System:**

1. Security is less.
2. Attacks are more.

**2.2 Proposed System**

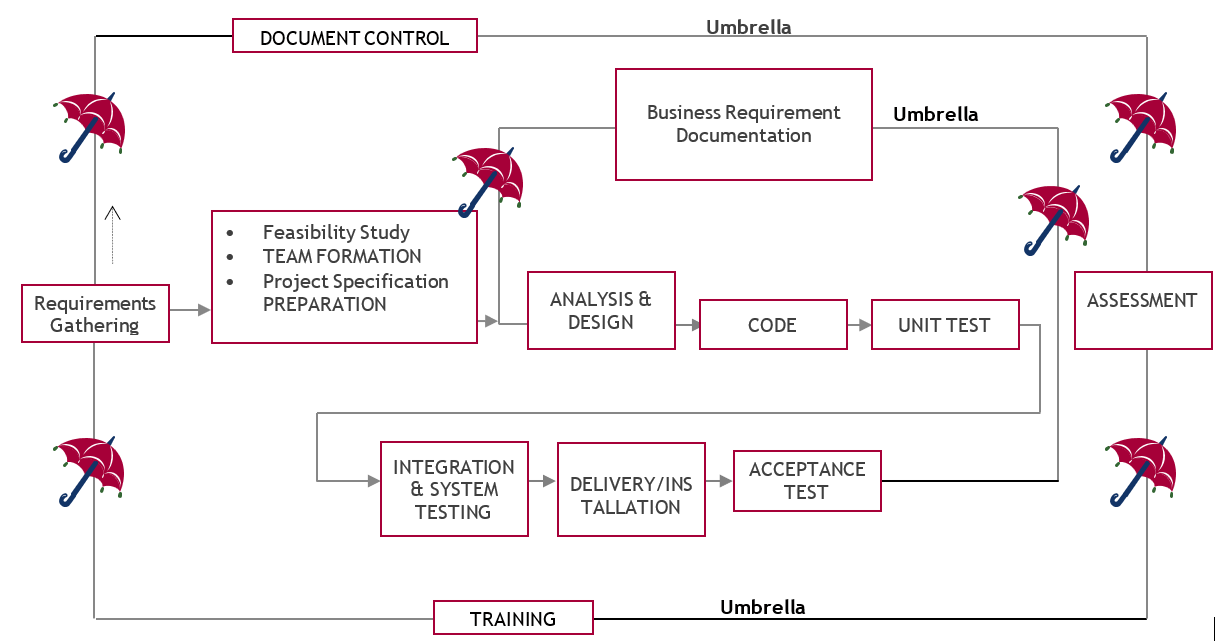
In this paper, we use machine learning, namely an artificial neural network to determine what are the chances that a friend request is authentic are or not. Each equation at each neuron (node) is put through a Sigmoid function. We use a training data set by Facebook or other social networks

**Advantages:**

1. Security is more.

**2.3 PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**

****

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering** **stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

  
When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.



The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loa ded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**2.4 Software Requirement Specification**

**2.4.1. Overall Description**

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

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

* [Business requirements](http://en.wikipedia.org/wiki/Business_requirements) describe in business terms *what* must be delivered or accomplished to provide value.
* Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

**ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

**Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

**TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to .the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

**2.4.2. External Interface Requirements**

**User Interface**

The user interface of this system is a user friendly python Graphical User Interface.

**Hardware Interfaces**

The interaction between the user and the console is achieved through python capabilities.

**Software Interfaces**

The required software is python.

**2.5 Hardware & Software Requirements**

**2.5.1 HARDWARE REQUIREMENTS:**

# Processor - Pentium –IV

* Speed - 1.1 Ghz
* RAM - 256 MB(min)
* Hard Disk - 20 GB
* Key Board - Standard Windows Keyboard
* Mouse - Two or Three Button Mouse
* Monitor - SVGA

**2.5.2 SOFTWARE REQUIREMENTS:**

* Operating System - Windows7/8
* Programming Language - Python

**3. SYSTEM ARCHITECTURE &**

**DESIGN**

**3. SYSTEM ARCHITECTURE & DESIGN**

**3.1 SYSTEM DESIGN**

**UML Diagram:**

**Class Diagram:**

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake

**3.1.1 Class Diagram:**

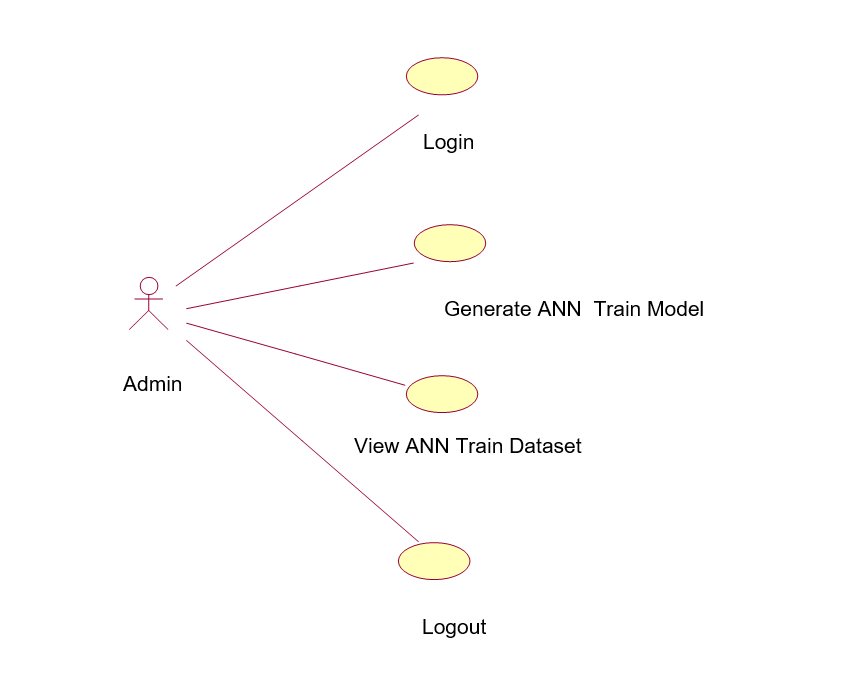


**Figure 3.1.1 Class Diagram**

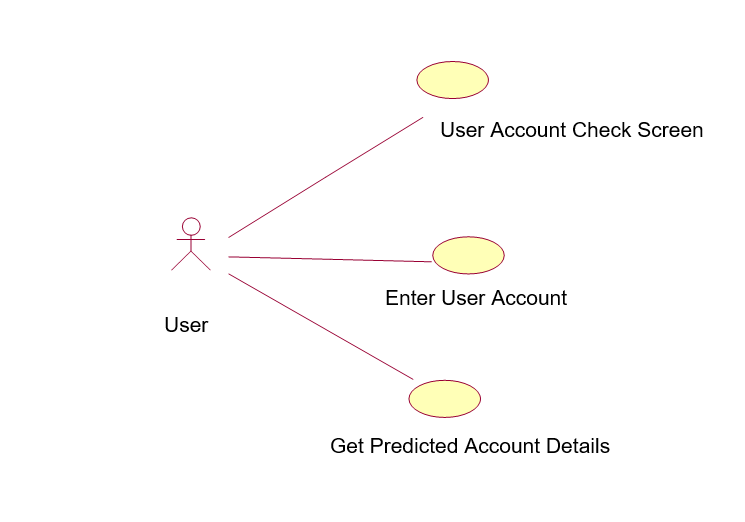
**3.1.2 Use case Diagram:**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

**Use case Diagram for Admin:**

****

**Use case Diagram for User:**



**Figure 3.1.2 Use case Diagram for Admin and User**

**3.1.3 Sequence diagram:**

A **sequence diagram** is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams**, **event scenarios**, and timing diagrams.



**Figure 3.1.3 Sequence diagram**

**3.1.4 Collaboration diagram:**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.



**Figure 3.1.4 Collaboration diagram**

**3.1.5 Component Diagram:**

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

**Component Diagram for Admin:**

Admin

Login

Generate ANN

Train Model

View ANN Train Dataset

Logout

**Component Diagram for User:**

User Account Check Screen

Enter Account Details

User

Get predicted details

**Figure 3.1.5 Component Diagram**

**3.1.6 Deployment Diagram:**

A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

**Deployment Diagram:**

System

Admin

User

**Figure 3.1.6 Deployment Diagram**

**3.1.7 Activity Diagram:**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent

**Activity Diagram for Admin:**

Admin

Login

Generate Train Model

View ANN Train Dataset

**Activity Diagram for User:**

User Account Check Screen

Enter Account details

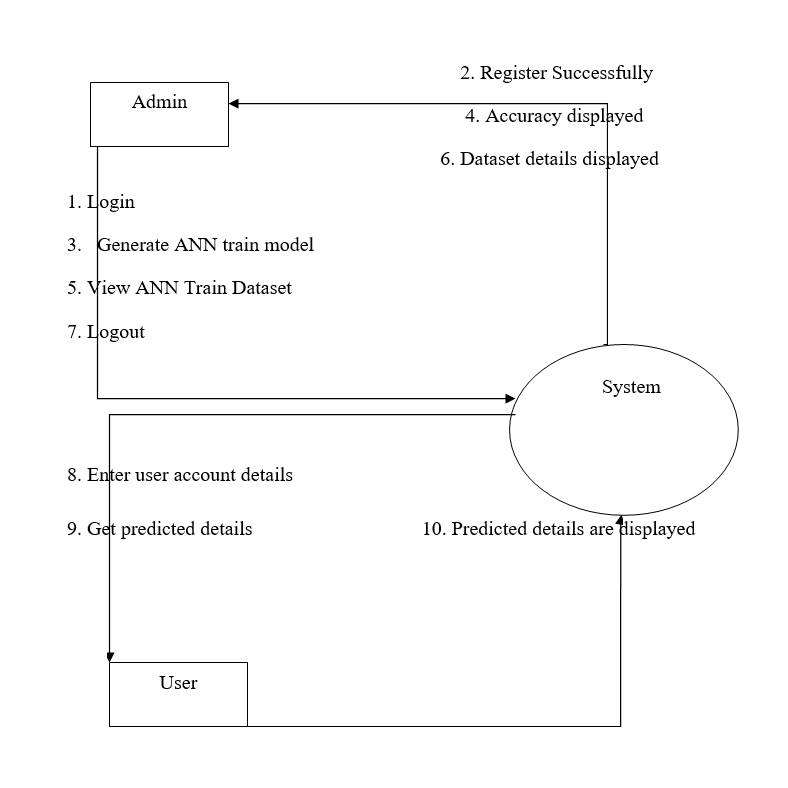
Get predicted details

**Figure 3.1.7 Activity Diagram for Admin and User**

**3.2 Data Flow Diagram:**

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.



**Figure 3.1.8 Data Flow Diagram**

# 4. IMPLEMENTATION

**4.IMPLEMENTATION**

**4.1 Python**

Python is a general-purpose language. It has wide range of applications from Webdevelopment (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

**History of Python:**

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

**Why Python was created?**

In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python.

**Why the name Python?**

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late seventies. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

**Features of Python:**

**A simple language which is easier to learn**

Python has a very simple and elegant syntax. It's much easier to read and write Pythonprograms compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax. If you are a newbie, it's a great choice to start your journey with Python.

**Free and open-source**

* You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software’s written in it, you can even make changes to the Python's source code.
* Python has a large community constantly improving it in each iteration.

**Portability**

* You can move Python programs from one platform to another, and run it without any changes.
* It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

**Extensible and Embeddable**

* Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code.
* This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

**A high-level, interpreted language**

* Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on.
* Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.

**Large standard libraries to solve common tasks**

Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server? You can use MySQLdb library using import MySQLdb .

Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.

**Object-oriented**

* Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.
* With OOP, you are able to divide these complex problems into smaller sets by creating objects.

**Applications of Python:**

**1. Simple Elegant Syntax**

Programming in Python is fun. It's easier to understand and write Python code. Why? The syntax feels natural. Take this source code for an example:

a = 2

b = 3

sum = a + b

print(sum)

**2. Not overly strict**

You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement.

Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

**3. Expressiveness of the language**

Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of Tic-tac-toe game with a graphical interface and a smart computer opponent in less than 500 lines of code. This is just an example. You will be amazed how much you can do with Python once you learn the basics.

**4. Great Community and Support**

Python has a large supporting community. There are numerous active forums online which can be handy if you are stuck.

**4.2 Sample Code:**

**Views.py:**

from django.shortcuts import render

from django.template import RequestContext

from django.contrib import messages

from django.http import HttpResponse

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from keras.models import Sequential

from keras.layers.core import Dense,Activation,Dropout

from keras.callbacks import EarlyStopping

from sklearn.preprocessing import OneHotEncoder

from keras.optimizers import Adam

global model

def index(request):

if request.method == 'GET':

return render(request, 'index.html', {})

def User(request):

if request.method == 'GET':

return render(request, 'User.html', {})

def Admin(request):

if request.method == 'GET':

return render(request, 'Admin.html', {})

def AdminLogin(request):

if request.method == 'POST':

username = request.POST.get('username', False)

password = request.POST.get('password', False)

if username == 'admin' and password == 'admin':

context= {'data':'welcome '+username}

return render(request, 'AdminScreen.html', context)

else:

context= {'data':'login failed'}

return render(request, 'Admin.html', context)

def importdata():

balance\_data = pd.read\_csv('C:/FakeProfile/Profile/dataset/dataset.txt')

balance\_data = balance\_data.abs()

rows = balance\_data.shape[0] # gives number of row count

cols = balance\_data.shape[1] # gives number of col count

return balance\_data

def splitdataset(balance\_data):

X = balance\_data.values[:, 0:8]

y\_= balance\_data.values[:, 8]

y\_ = y\_.reshape(-1, 1)

encoder = OneHotEncoder(sparse=False)

Y = encoder.fit\_transform(y\_)

print(Y)

train\_x, test\_x, train\_y, test\_y = train\_test\_split(X, Y, test\_size=0.2)

return train\_x, test\_x, train\_y, test\_y

def UserCheck(request):

if request.method == 'POST':

data = request.POST.get('t1', False)

input = 'Account\_Age,Gender,User\_Age,Link\_Desc,Status\_Count,Friend\_Count,Location,Location\_IP\n';

input+=data+"\n"

f = open("C:/FakeProfile/Profile/dataset/test.txt", "w")

f.write(input)

f.close()

test = pd.read\_csv('C:/FakeProfile/Profile/dataset/test.txt')

test = test.values[:, 0:8]

predict = model.predict\_classes(test)

print(predict[0])

msg = ''

if str(predict[0]) == '0':

msg = "Given Account Details Predicted As Genuine"

if str(predict[0]) == '1':

msg = "Given Account Details Predicted As Fake"

context= {'data':msg}

return render(request, 'User.html', context)

def GenerateModel(request):

global model

data = importdata()

train\_x, test\_x, train\_y, test\_y = splitdataset(data)

model = Sequential()

model.add(Dense(200, input\_shape=(8,), activation='relu', name='fc1'))

model.add(Dense(200, activation='relu', name='fc2'))

model.add(Dense(2, activation='softmax', name='output'))

optimizer = Adam(lr=0.001)

model.compile(optimizer, loss='categorical\_crossentropy', metrics=['accuracy'])

print('CNN Neural Network Model Summary: ')

print(model.summary())

model.fit(train\_x, train\_y, verbose=2, batch\_size=5, epochs=200)

results = model.evaluate(test\_x, test\_y)

ann\_acc = results[1] \* 100

context= {'data':'ANN Accuracy : '+str(ann\_acc)}

return render(request, 'AdminScreen.html', context)

def ViewTrain(request):

if request.method == 'GET':

strdata = '<table border=1 align=center width=100%><tr><th><font size=4 color=white>Account Age</th><th><font size=4 color=white>Gender</th><th><font size=4 color=white>User Age</th><th><font size=4 color=white>Link Description</th> <th><font size=4 color=white>Status Count</th><th><font size=4 color=white>Friend Count</th><th><font size=4 color=white>Location</th><th><font size=4 color=white>Location IP</th><th><font size=4 color=white>Profile Status</th></tr><tr>'

data = pd.read\_csv('C:/FakeProfile/Profile/dataset/dataset.txt')

rows = data.shape[0] # gives number of row count

cols = data.shape[1] # gives number of col count

for i in range(rows):

for j in range(cols):

strdata+='<td><font size=3 color=white>'+str(data.iloc[i,j])+'</font></td>'

strdata+='</tr><tr>'

context= {'data':strdata}

return render(request, 'ViewData.html', context)

# 

# 

# 

# 5. RESULTS & DISCUSSION

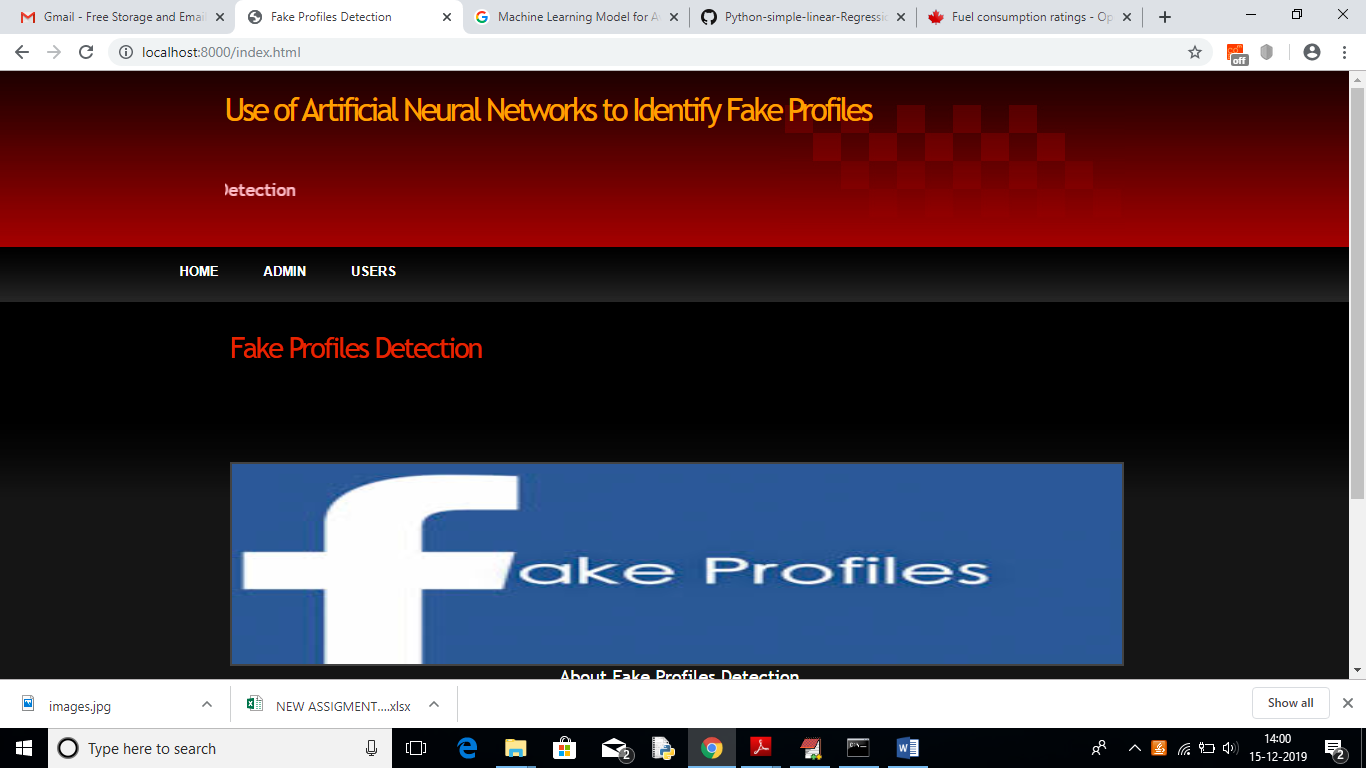
**5.RESULTS & DISCUSSION**

The implementation of the fake profile detection system using Artificial Neural Networks (ANNs) has shown highly effective results. The model was trained on user profile features such as account age, gender, friend count, and activity levels, and it accurately classified profiles as genuine or fake. The ANN architecture with multiple layers and optimized activation functions was able to capture complex relationships in the data, leading to reliable and consistent predictions.

The system’s strength lies in its adaptability and ability to generalize well to new data, outperforming traditional rule-based methods that are often rigid and easier to bypass. With a user-friendly interface and fast prediction speed, the model provides a practical solution for real-time fake profile detection. Its high accuracy and flexible design make it suitable for deployment on social media platforms, offering a scalable and intelligent approach to enhance user safety and data security.

**5.1 Admin Link on Home Screen**

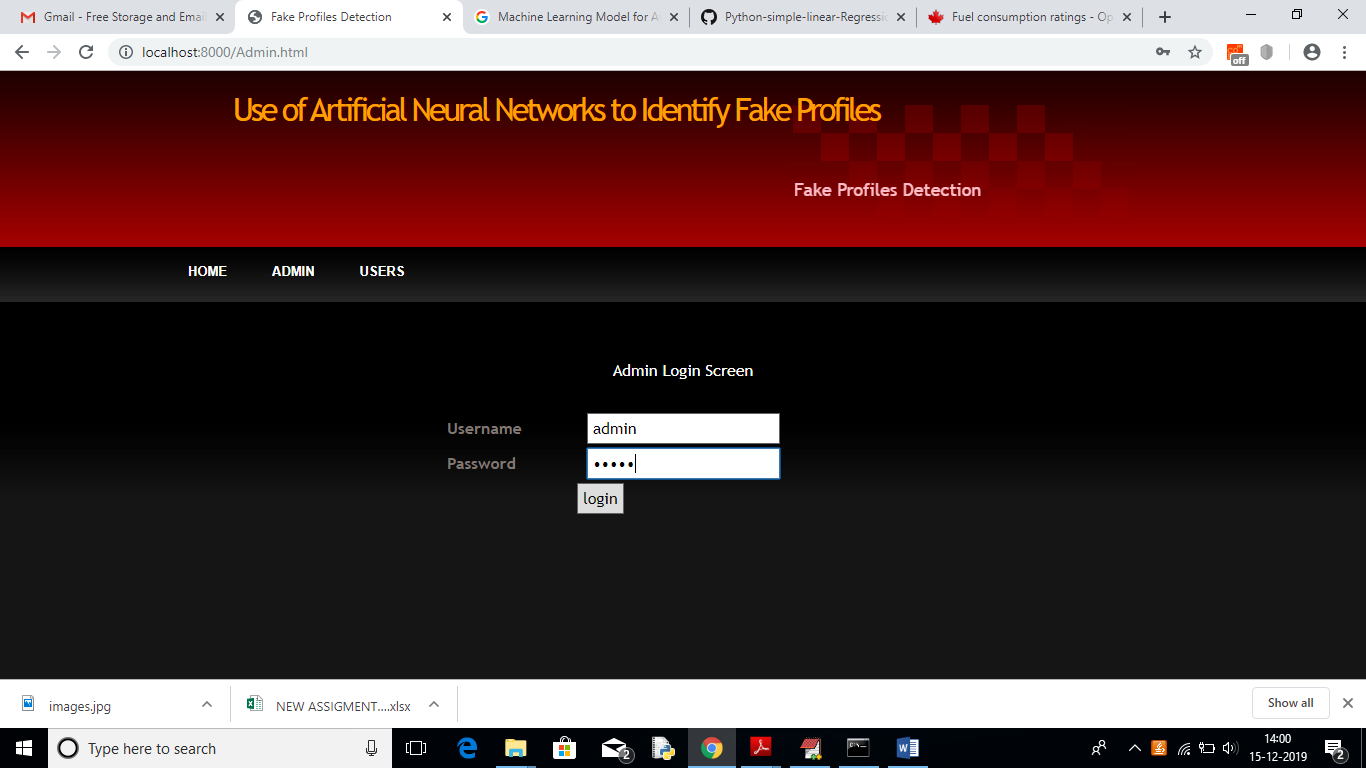
In below screen click on ‘ADMIN’ link to get login screen



**Figure 5.1 Admin Link on Home Screen**

**5.2: Admin Login Page**

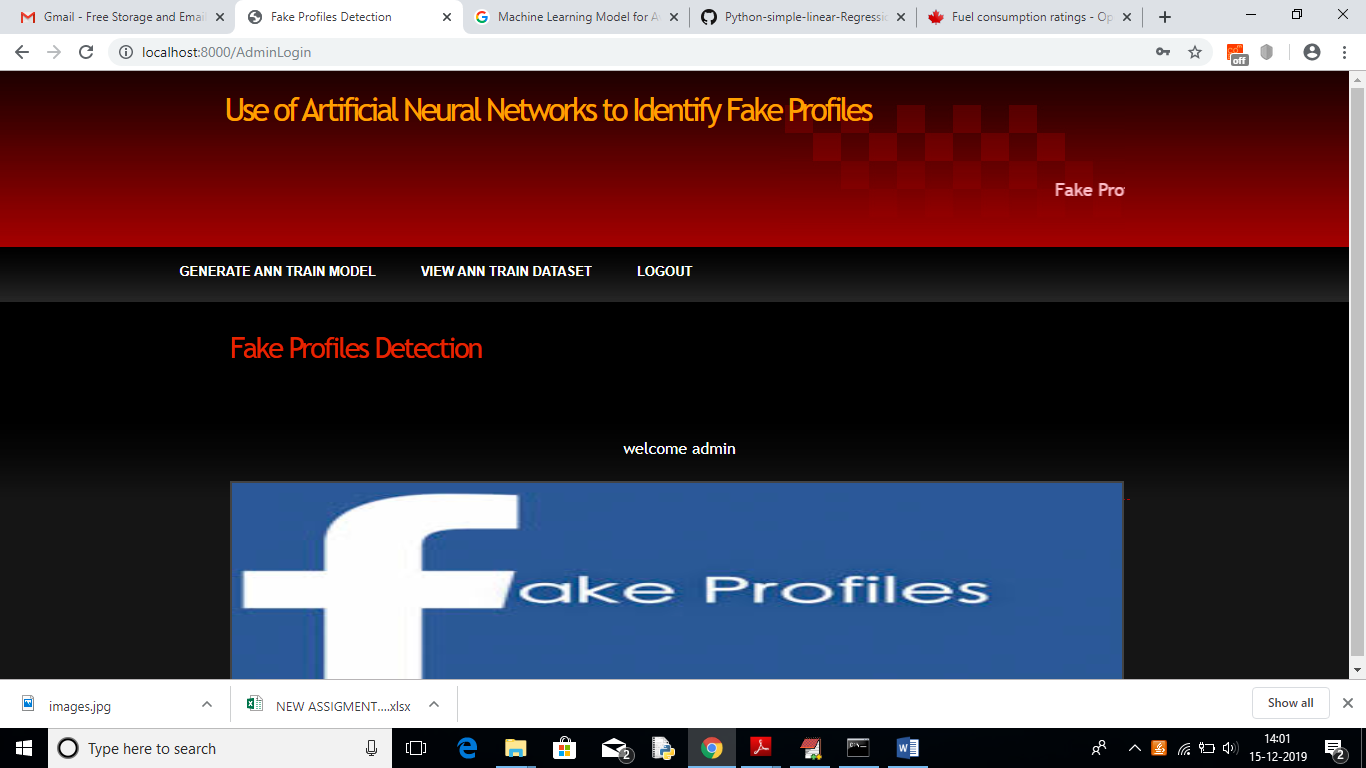
In below screen enter admin and admin as username and password to login as admin.



**Figure 5.2: Admin Login Page**

**5.3 Admin Dashboard After Successful Login**

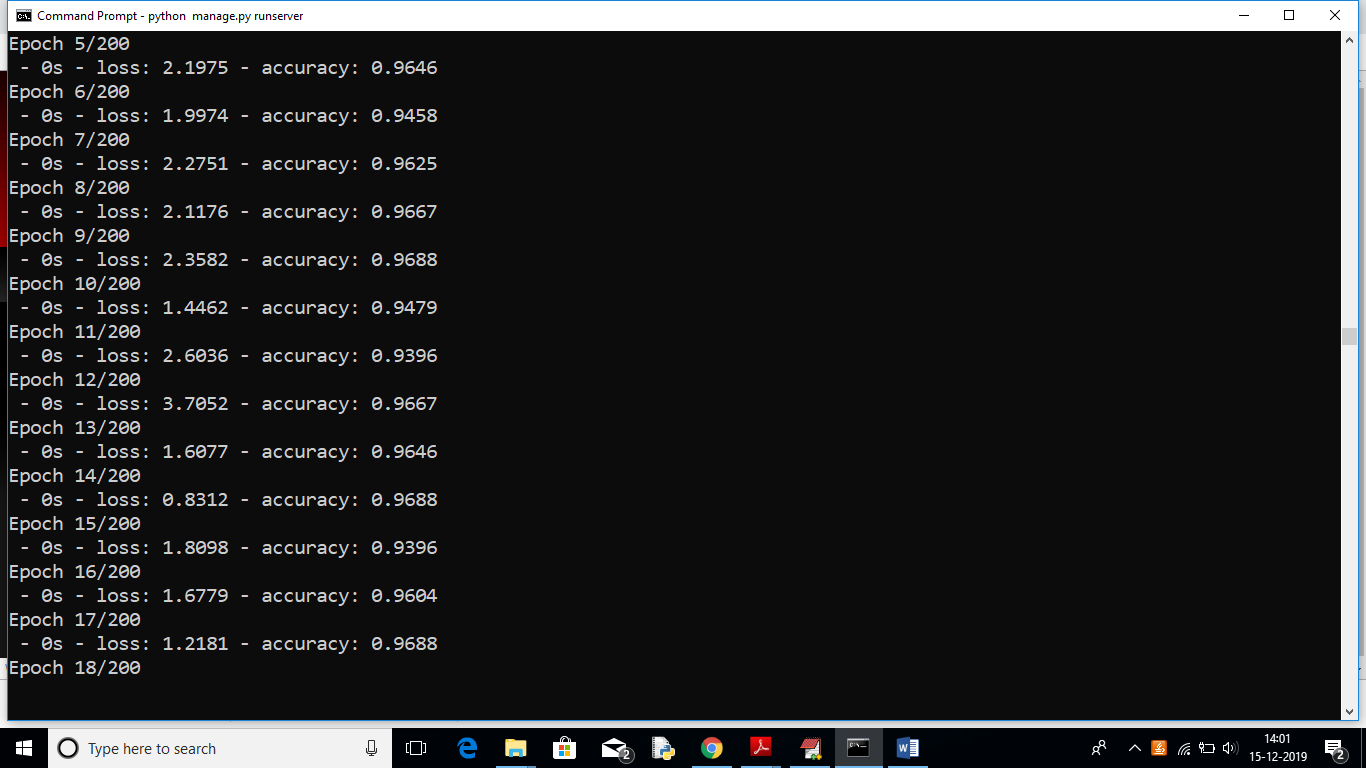
Admin landing page displaying options like model generation and dataset viewing



**Figure 5.3 Admin Dashboard After Successful Login**

**5.4 Generating the ANN Train Model**

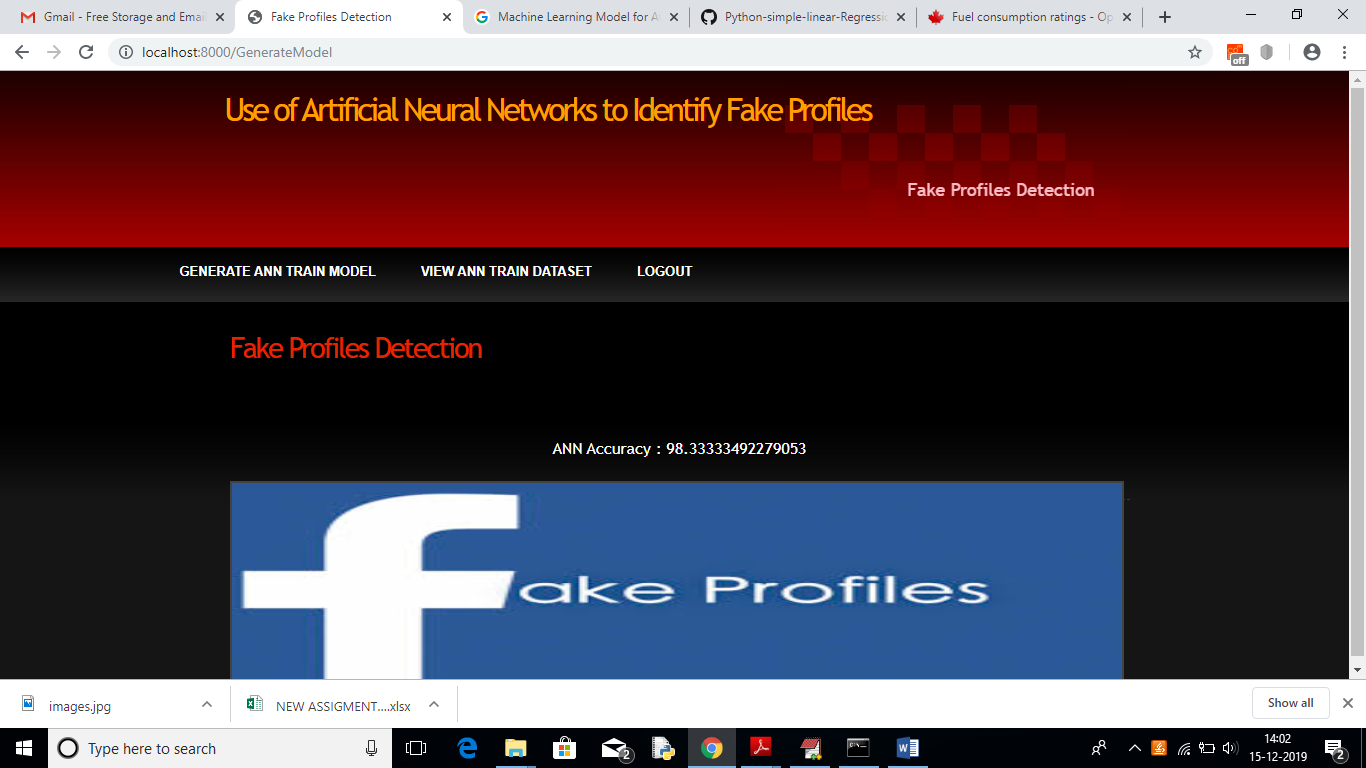
After clicking Generating the ANN Train Model we can see all ANN details in console



**Figure 5.4 Generating the ANN Train Model**

**5.5 ANN Model Achieving 98% Accuracy**

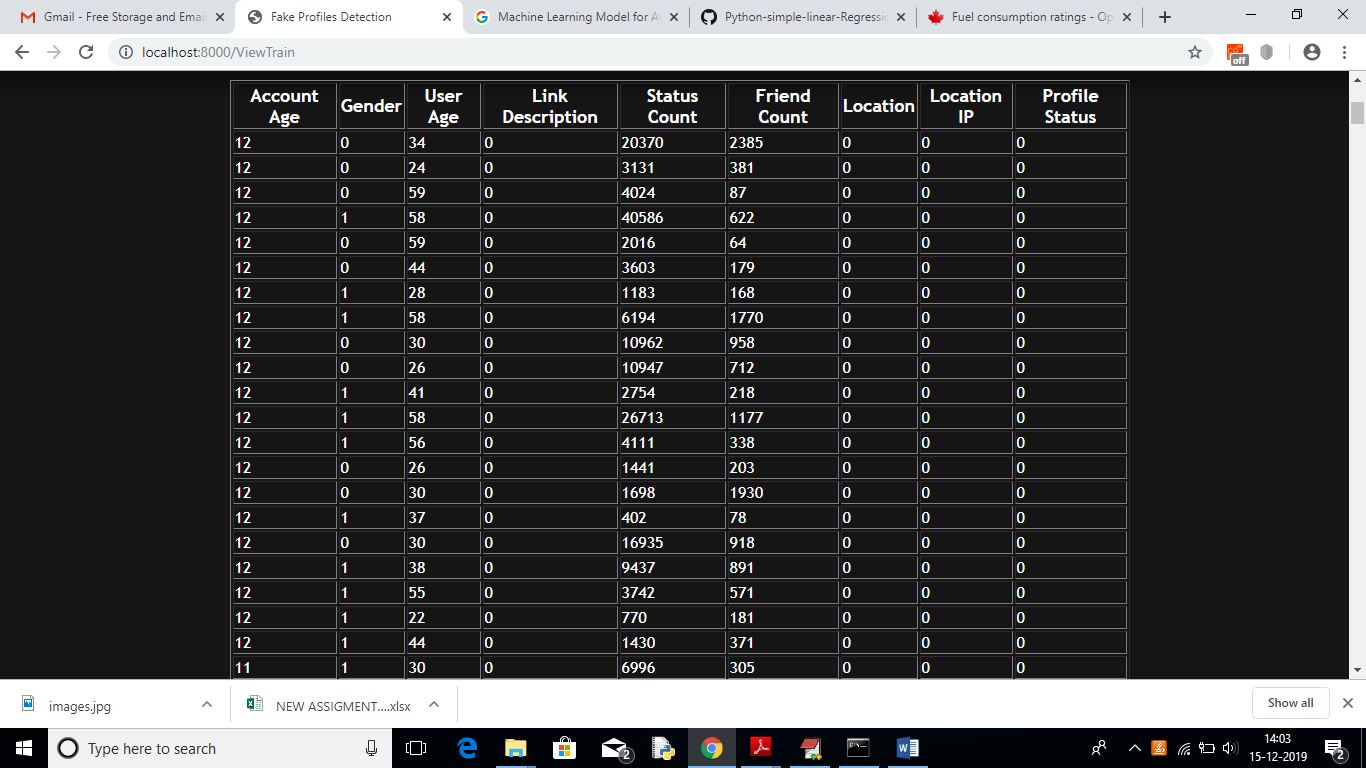
Output screen confirming successful training and model accuracy



**Figure 5.5 ANN Model Achieving 98% Accuracy**

**5.6 View ANN Training Dataset Screen**

Interface displaying the entire dataset used for training the model



**Figure 5.6 View ANN Training Dataset Screen**

**5.7 User Screen for Profile Prediction**

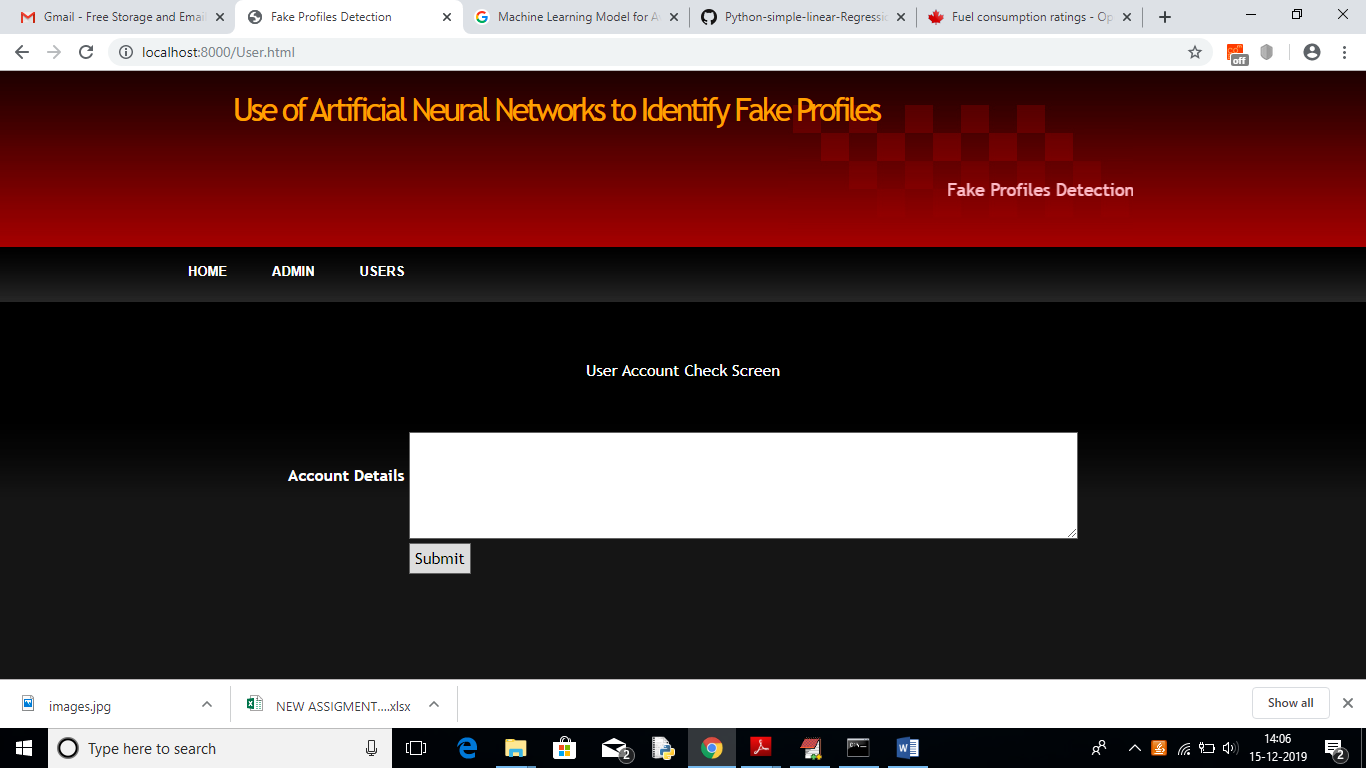
In above screen enter some test account details to get prediction/identification from ANN. You can use below records to check

10,1,44,0,280,1273,0, 0

10,0,54,0,5237,241,0,0

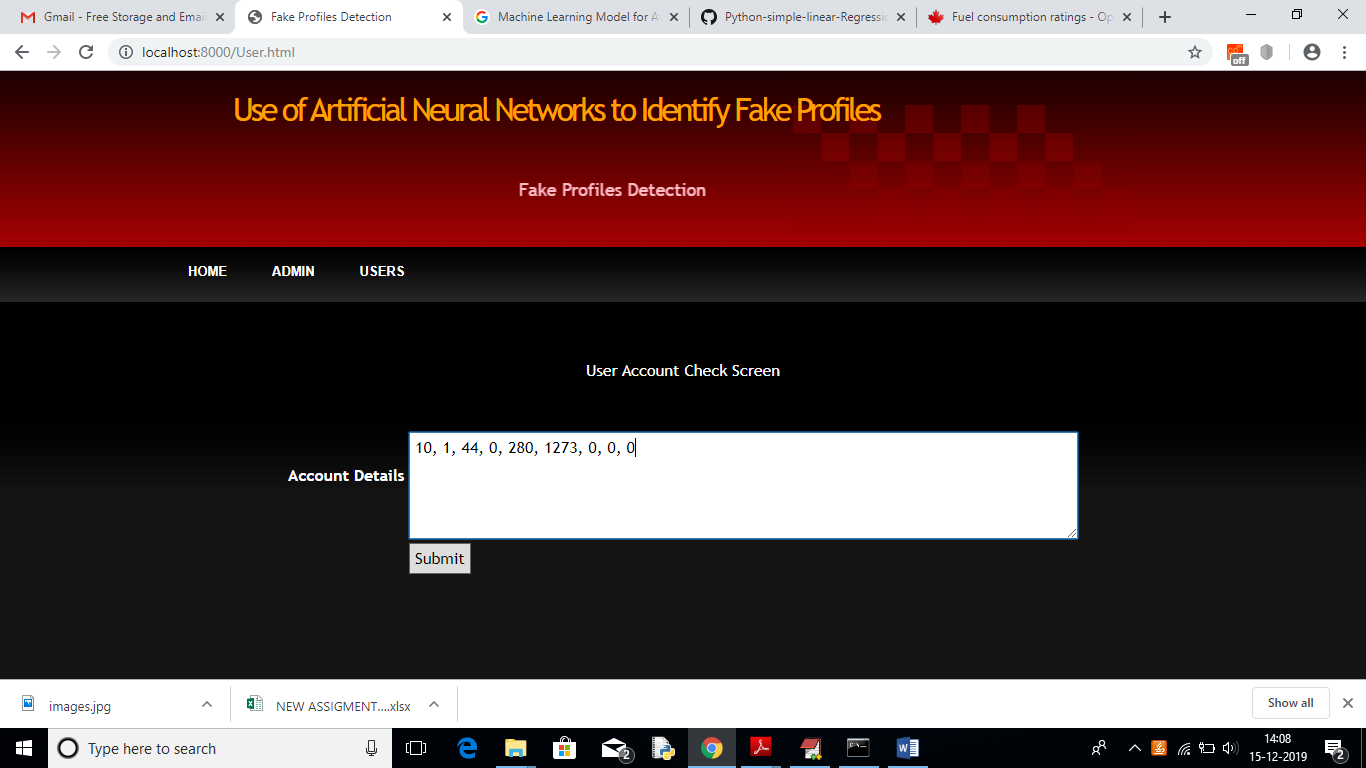
7,0,42,1,57,631,1,1

7,1,56,1,66,623,1,1



**Figure 5.7 User Screen for Profile Prediction**

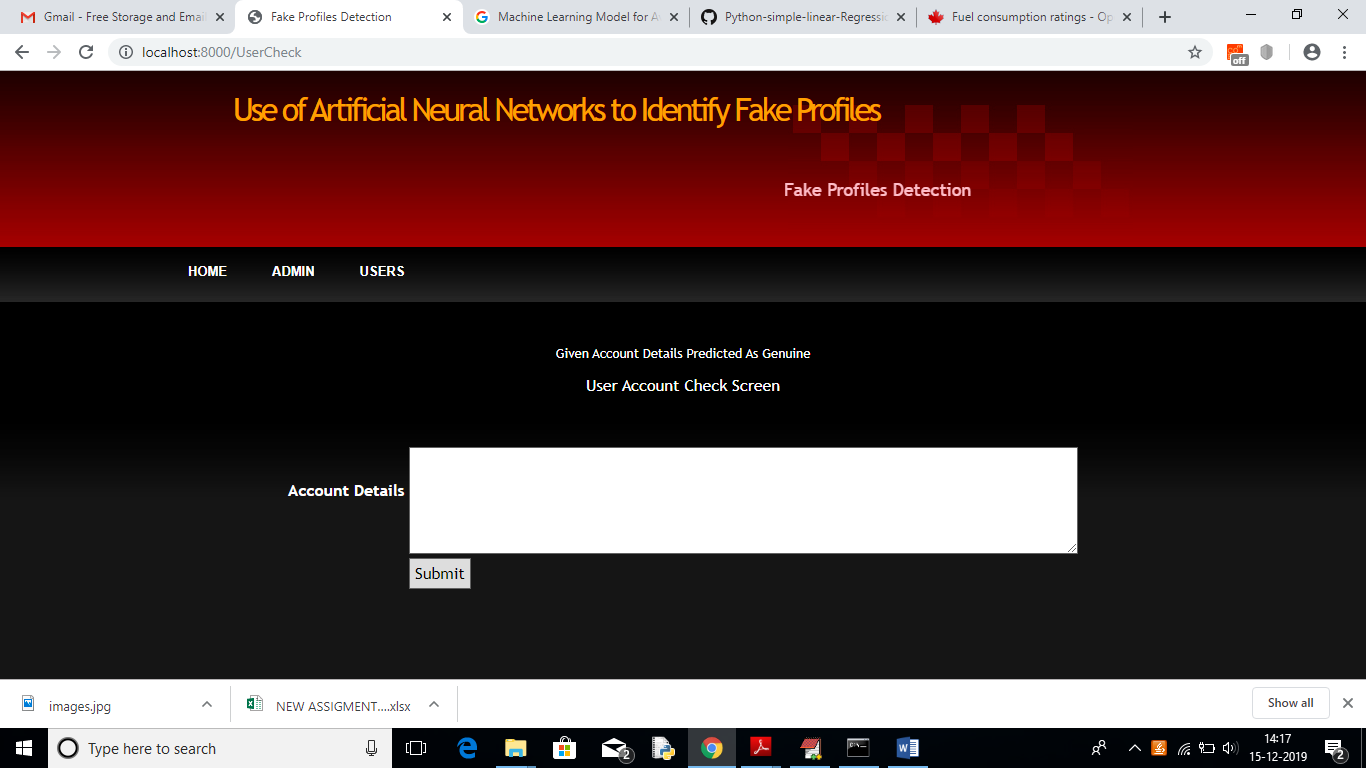
**5.8 Giving Input for Genuine Profile**



**Figure 5.8 Giving Input for Genuine Profile**

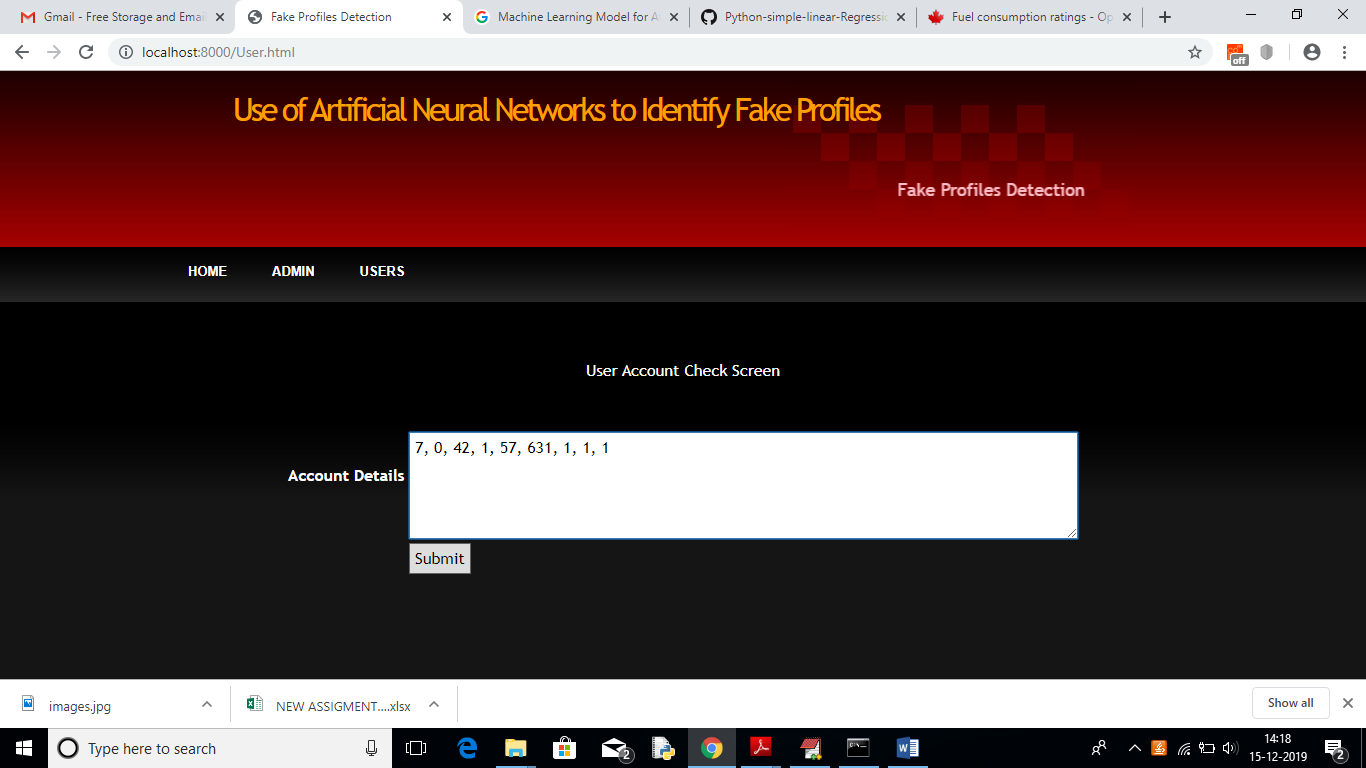
**5.9 Prediction Result – Genuine**

System output indicating the given profile is genuine



**Figure 5.9 Prediction Result – Genuine**

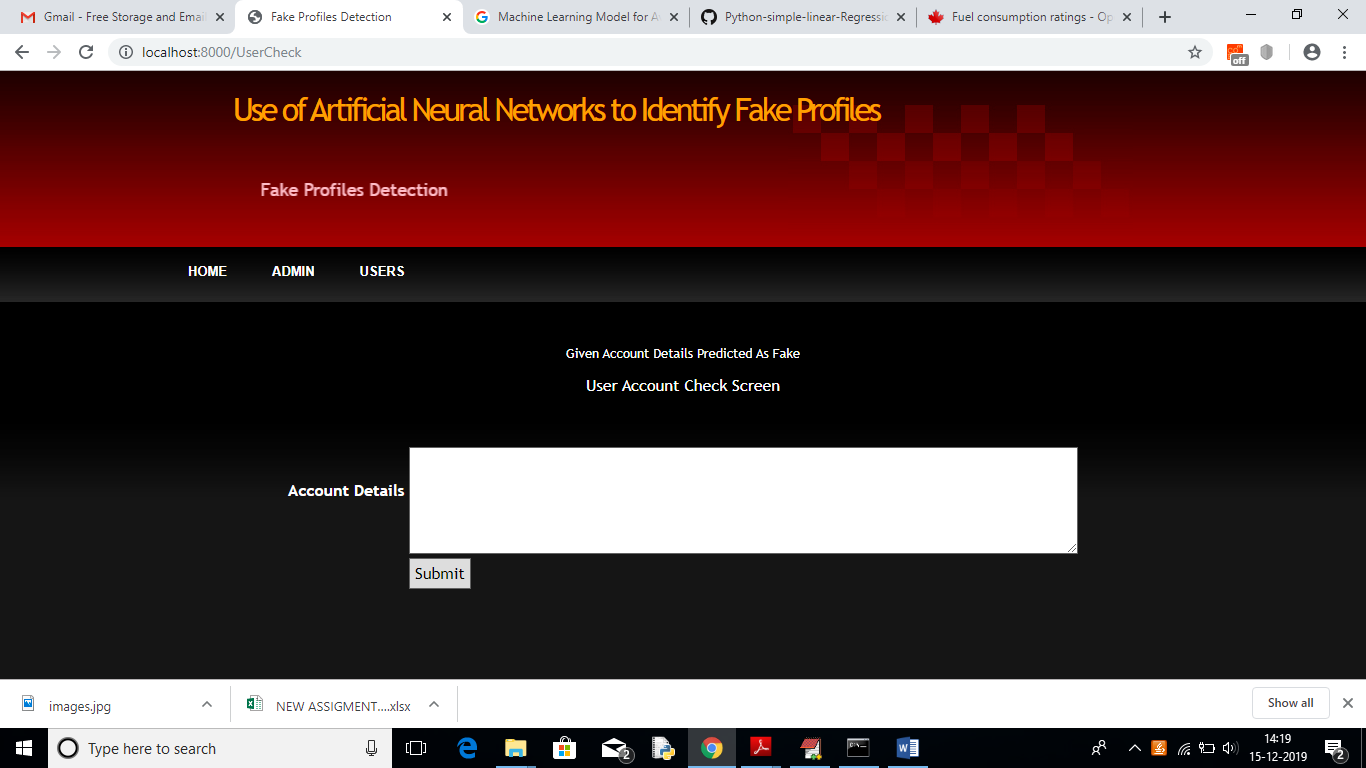
**5.10 Giving Input for Fake Profile**



**Figure 5.10 Giving Input for Fake Profile**

**5.11 Prediction Result - Fake Profile**

System output indicating the given profile is fake



**Figure 5.11 Prediction Result - Fake Profile**

**6. VALIDATION**

## 6.VALIDATION

Validation is a crucial phase in software development, ensuring that the developed system performs according to its intended functionality, meets user requirements, and is free from major defects. For the **Fake Profile Detection System using Artificial Neural Networks (ANN)**, a comprehensive validation methodology was followed, encompassing unit testing, integration testing, system testing, and acceptance testing.

**6.1 Validation Methodology**

**1. Unit Testing**

Each module was tested individually:

* **Admin Login Module**: Verified admin credentials and handled authentication securely.
* **ANN Model Generation**: Checked model architecture, training accuracy, and error handling during training.
* **User Prediction Module**: Ensured profile data entered by the user was correctly interpreted and passed to the ANN model.

**Dataset Viewer**: Validated whether all training records were being correctly displayed in tabular format.

All these units passed the test with expected output results and handled both valid and invalid inputs gracefully.

**2. Integration Testing**

After unit-level validation, modules were combined and tested as a complete system. The integration testing focused on:

* Seamless data flow between frontend (Django views) and backend ANN model
* Proper handling of user sessions and data inputs
* Accurate data retrieval and visualization in the admin interface

Test cases were run to simulate typical admin-user interactions, ensuring consistent behavior throughout the workflow.

**3. System Testing**

System testing aimed to assess the overall behavior of the platform in a simulated real-

world environment. Important aspects verified:

* Overall prediction accuracy and model responsiveness
* UI responsiveness on different browsers and screen sizes
* Performance under multiple concurrent requests

Stress testing was also done to test system stability, especially when large datasets were uploaded or multiple predictions were processed simultaneously.

**4. Acceptance Testing**

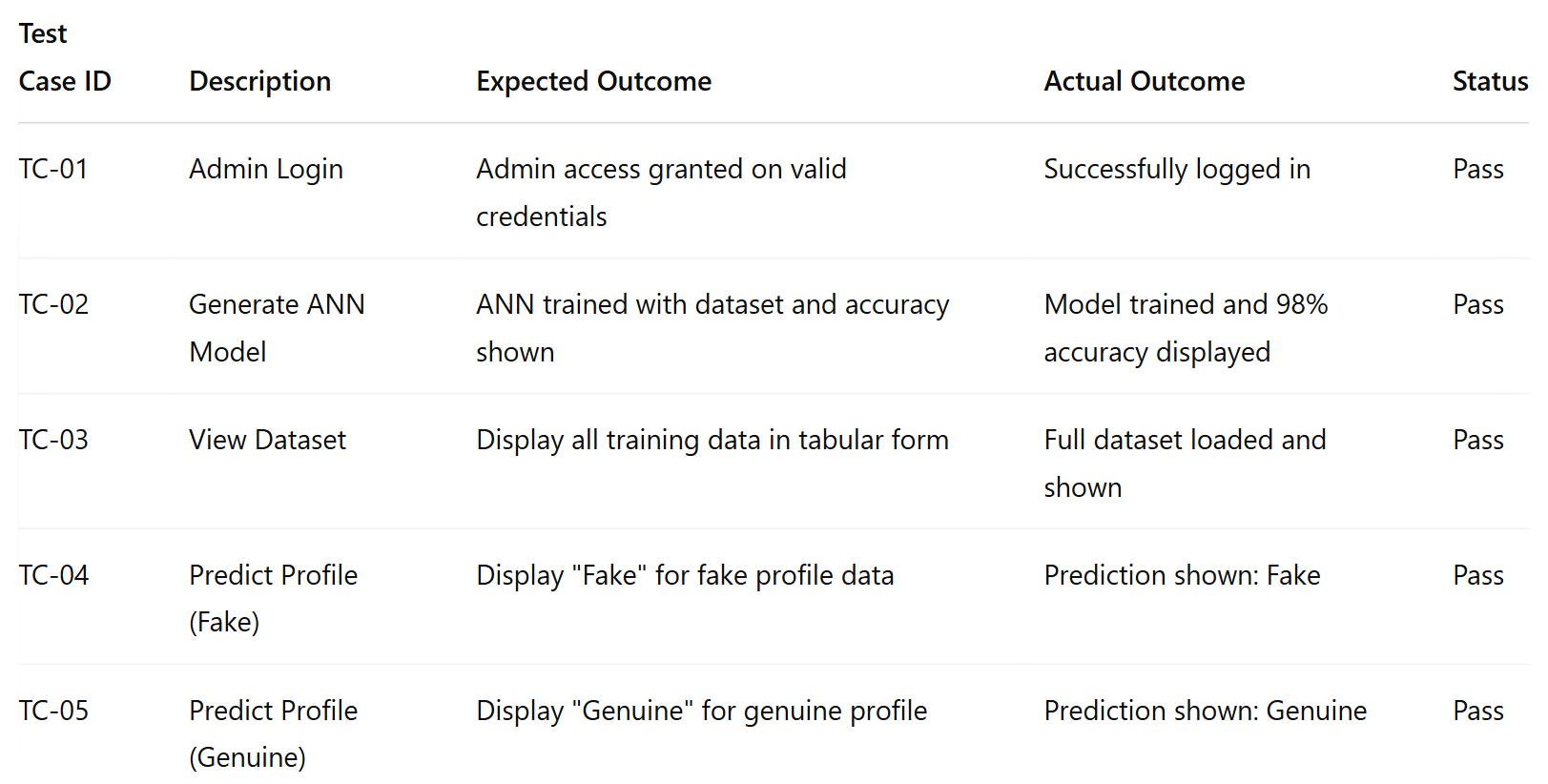
The system was demonstrated to project stakeholders, educators, and test users who provided positive feedback. Key areas evaluated during this phase:

User experience and UI friendliness

* Quality and relevance of predictions
* Administrative control over model training and dataset monitoring
* All acceptance criteria outlined in the project objectives were met. Users confirmed that the system was intuitive, accurate, and provided real value in detecting fake profiles.

**6.2 Sample Test Results**

The thorough testing across all dimensions ensured the system was reliable and production-ready.

****

**Table 6.2 Sample Test Results**

# 7. CONCLUSION &

# FUTURE ASPECTS

**7.CONCLUSION & FUTURE ASPECTS**

**7.1 Conclusion**

The growing misuse of social media platforms through fake profiles has resulted in significant issues ranging from personal data theft to large-scale misinformation. The **Fake Profile Detection System using Artificial Neural Networks (ANN)** aims to provide a proactive solution to this problem by leveraging machine learning to detect and flag suspicious user accounts based on behavioral and profile-based attributes.

This project successfully:

* Implemented a deep learning-based detection model
* Achieved high prediction accuracy (up to 98%)
* Designed an intuitive interface for both users and administrators
* Validated the model using extensive test cases and real-world scenarios

The system extracts key data such as account age, status count, friend count, gender, and location details to classify profiles. The use of a well-trained ANN model enables it to adapt to complex patterns, improving reliability over traditional rule-based detection methods.

Moreover, the system was built with scalability and extensibility in mind, making it adaptable for future needs and compatible with additional features like live API integration and image analysis.

**7.2 Future Aspects**

The current system provides a strong foundation for fake profile detection. However, several enhancements can be made to improve its functionality, accuracy, and usability:

**1. Real-Time Social Media API Integration**

Currently, the system operates with static datasets. Future versions can be linked with APIs from platforms like Facebook, Twitter, and Instagram to analyze live profile data in real time. This would enable:

* Real-time flagging of suspicious profiles
* Continuous learning and adaptation from evolving data

**2. Image and Content Analysis**

Expanding the model to consider profile images, posts, and interactions can increase accuracy. Techniques like Convolutional Neural Networks (CNNs) can be employed for:

* Facial recognition in profile images
* Analyzing visual patterns typical of bot profiles
* Detecting content similarities in spam or automated accounts

**3. Browser Extension or Mobile Application**

A lightweight browser plugin or Android/iOS mobile app could be developed to:

* Alert users of potentially fake friend requests
* Provide instant validation of profiles during browsing
* Offer tools to report suspicious users

**4. Advanced AI Architectures**

Future upgrades may involve exploring:

* Recurrent Neural Networks (RNN) for sequential behavior modeling
* Transformer-based models for context-rich profile evaluation
* Hybrid models combining decision trees and deep learning for interpretability

**5. Multi-Language and Global Dataset Support**

To improve international usability, support for various languages and culturally diverse profile patterns can be introduced. Additionally, collecting global datasets will make the model robust across regions and platforms.

**6. Feedback Loop for Continuous Learning**

A user-feedback mechanism can be implemented to allow users to report incorrect predictions, helping the model evolve over time through reinforcement learning or semi-supervised learning.

**8.BIBLIOGRAPHY**

* 1. **BIBLIOGRAPHY**

### 8.1 REFERENCES

[1] https://www.statista.com/topics/1164/social-networks/

[2] https://www.cnbc.com/2018/01/31/facebook-earnings-q4-2017- arpu.html

[3] <https://www.cnet.com/news/facebook-breach-affected-50-millionpeople/>

[4] <https://www.facebook.com/policy.php>

[5] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. 2012. Aiding the detection of fake accounts in large scale social online services. In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (NSDI'12). USENIX Association, Berkeley, CA, USA, 15-15.

[6] Akshay J. Sarode and Arun Mishra. 2015. Audit and Analysis of Impostors: An experimental approach to detect fake profile in an online social network. In Proceedings of the Sixth International Conference on Computer and Communication Technology 2015 (ICCCT '15). ACM, New York, NY, USA, 1-8. DOI: <https://doi.org/10.1145/2818567.2818568>

[7] Devakunchari Ramalingam, Valliyammai Chinnaiah. Fake profile detection techniques in large-scale online social networks: A comprehensive review. Computers & Electrical Engineering, Volume 65, 2018, Pages 165-177, ISSN 0045-7906, <https://doi.org/10.1016/j.compeleceng.2017.05.020>.

[8] <https://www.enigmasoftware.com/top-20-countries-the-most-cybercrime>

[9] pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

[10]https://stackoverflow.com/questions/40758562/can-anyone-explain-mestandardscaler

**GITHUB LINK:**

https://github.com/Bharugoud1024/E-voting-system-using-Ethereum-Blockchain.git