PHISHING E-MAIL DETECTION

Submitted by

BEN K DANIEL (Reg.No:32020895015)

Under the supervision of

JOLLY SUBASH PALLICKEN

(Assistant Professor, Department of Computer Science)

COLLEGE OF APPLIED SCIENCE, PERISSERY

PROJECT REPORT

Submitted in partial fulfilment of the

requirements for the award of

B.Sc (Computer Science) degree of

University of Kerala

2023

PHISHING E-MAIL DETECTION

Submitted by

BEN K DANIEL (Reg.No:32020895015)

Under the supervision of

JOLLY SUBASH PALLICKEN

(Assistant Professor, Department of Computer Science)

COLLEGE OF APPLIED SCIENCE, PERISSERY

ELECTRONICS
manpower development

PROJECT REPORT

Submitted in partial fulfilment of the
requirements for the award of
B.Sc (Computer Science) degree of
University of Kerala

2023

COLLEGE OF APPLIED SCIENCE, PERISSERY

DEPARTMENT OF COMPUTER SCIENCE

Certificate

Certified that this report titled **PHISHING E-MAIL DETECTION** is a bonafide record of the project work done by **BEN K DANIEL (Reg.No:32020895015)** under our supervision and guidance, towards partial fulfillment of the requirements for the award of the degree of B.Sc (Computer Science) of the University of Kerala.

Project Guide	**Head of the department**	**Principal**
JOLLY SUBASH PALLICKEN	**SINDHU R NAIR**	**Dr. G SREEKUMAR**
Asst. Professor	**Asst. Professor**	
Dept. of CS	**Dept. of CS**	

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

First and foremost, i would like to acknowledge the **Almighty God** for the successful completion of my project.

I am very thankful to my guide, **Jolly Subash Pallicken**, for providing her invaluable guidance, comments and suggestions throughout this project work. I would specially thank our Head of Department, **Sindhu R Nair**, for constantly motivating me.

I would like to thank our Principal **Dr. G Sreekumar** and all the faculties of Department of Computer Science, who mentored me.

I wish to record sincere thanks to all my family members whose blessings made this task possible.

BEN K DANIEL

TABLE OF CONTENTS

LIST OF FIGURE

LIST OF TABLES

ABSTRACT

The Phishing E-Mail Detection is one of the significant threats in the world today and has caused tremendous financial losses. Although the methods of confrontation are continually being updated, the results of those methods are not very satisfactory at present. Moreover, phishing emails are growing at an alarming rate in recent years. Therefore, more effective phishing detection technology is needed to curb the threat of phishing emails. In this paper, we first analyzed the email structure. Then based on an improved Recurrent Convolutional Neural Networks (RCNN) model with multilevel vectors and attention mechanism, we proposed a new phishing email detection model named, which is used to model emails at the email header, the email body, the character level, and the word level simultaneously. To evaluate the effectiveness of, we use an unbalanced dataset that has realistic ratios of phishing and legitimate emails. Experimental results show that the. Meanwhile, the ensure that the filter can identify phishing emails with high probability and filter out legitimate emails as little as possible. This promising result is superior to the existing detection methods and verifies the effectiveness of in detecting phishing emails.

INTRODUCTION

OBJECTIVE OF THE PROJECT

The phishing email is one of the significant threats in the world today and has caused tremendous financial losses. Although the methods of confrontation are continually being updated, the results of those methods are not very satisfactory at present. Moreover, phishing emails are growing at an alarming rate in recent years. Therefore, more effective phishing detection technology is needed to curb the threat of phishing emails. In this paper, we first analyzed the email structure. Then based on an improved Recurrent Convolutional Neural Networks (RCNN) model with multilevel vectors and attention mechanism, we proposed a new phishing email detection model named, which is used to model emails at the email header, the email body, the character level, and the word level simultaneously. To evaluate the effectiveness of, we use an unbalanced dataset that has realistic ratios of phishing and legitimate emails. Experimental results show that the. Meanwhile, the ensure that the filter can identify phishing emails with high probability and filter out legitimate emails as little as possible. This promising result is superior to the existing detection methods and verifies the effectiveness of in detecting phishing emails.

SYSTEM ANALYSIS

2.1 INTRODUCTION

System analysis is the process of gathering data and facts diagnosing problem to the system. In the development of software structural analysis is required. During this analysis, information is collected in the form of answers to the question for collecting information from existing documents. Analysis specifies what the system should do.

Problem definition deals with defining the actual problem involved in the existing system or the system to be developed. Studies on various areas covered by the existing system are classified into various divisions and the actual task to be performed in the new system is determined. The project will be able to demonstrate the ideas of a website which helps the public. The website is trying to revitalize and simplify the various functions and activities and make them more people friendly. We dedicated to providing better and speedy services to the public.

System analysis is the detailed study of various operations and their relationship within and outside the system. It is the first in the developing and managing systems. System analysis is concerned with becoming aware of the problem, identifying the relevant and most decisional variables, analyzing and synthesizing the various factors and determining an optimal or at least as at is factory solution or program of action. A preliminary study was conducted in details and several fact-finding techniques like record searching, observation, comparison etc. were used to reach a better decision. The current system for this each activity was deeply studied and analyzed. All the forms and other printed or non-printed formats for data collection were checked accurately and findings were compared. Observation was done to great extend to see the difficulties of the process and time delay in findings the results. Accurate study was conducted to know the system in a much better manner.

The objectives of the system analysis are:

- Identifying the need.
- Analyzing the existing and proposed system.
- Evaluating the feasibility study.
- Perform economic and technical analysis.
- Identifying the hardware and software requirements.
- Allocating functions to the hardware and software.

2.2 IDENTIFICATION OF NEED

System analysis is the reduction of the entire system by studying various operations and their relationships with the system and the requirements of bit successor. A system can be defined as an orderly grouping of interdependent components linked together according to plan to achieve a specific objective.

The idea of the system has become most practical and necessary in conceptualizing the interrelationships and integrations of operations especially when using computers. Organizing consists of several interrelated and interacting components. Analysis is the detailed study of various operations performed by the system and their relations within and outside the system. During analysis, data are connected on the available files, decision points and is handled by the present system.

2.3 EXISTING SYSTEM

Various techniques for detecting phishing emails are mentioned in the literature. In the entire technology development process, there are mainly three types of technical methods including blacklist mechanisms, classification algorithms based on machine learning and based on deep learning. From previous work, the existing detection methods based on the blacklist mechanism mainly rely on people’s identification and reporting of phishing links requiring a large amount of manpower and time. However, applying artificial intelligence to the detection method based on a machine learning classification algorithm requires feature engineering to manually find representative features that are not conducive to the migration of application scenarios. Moreover, the current detection method based on deep learning is limited to word embedding in the content representation of the email. These methods directly transferred natural language processing (NLP) and deep learning technology, ignoring the specificity of phishing email detection so that the results were not ideal given the methods mentioned above and the corresponding problems, we set to study phishing email detection systematically based on deep learning. Specifically, this paper makes the following contributions:

2.4 PROPOSED SYSTEM

With the emergence of email, the convenience of communication has led to the problem of massive spam, especially phishing attacks through email. Various anti phishing technologies have been proposed to solve the problem of phishing attacks. Studied the effectiveness of phishing blacklists. Blacklists mainly include sender blacklists and link blacklists. This detection method extracts the sender’s address and link address in the message and checks whether it is in the blacklist to distinguish whether the email is a phishing email. The update of a blacklist is usually reported by users, and whether it is a phishing website or not is manually identified. At present, the two well-known phishing websites are Phish Tank and Open Phish. To some extent, the perfection of the blacklist determines the effectiveness of this method based on the blacklist mechanism for phishing email detection. The current situation is that new threats may not only cause severe damage to customers' computers but also aim to steal their money and identity. Among these threats, phishing is a noteworthy one and is a criminal activity that uses social engineering and technology to steal a victim's identity data and account information. According to a report from the Anti-Phishing Working compared with the fourth quarter of According to the striking data, it is clear that phishing has shown an apparent upward trend in recent years. Similarly, the harm caused by phishing can be imagined as well.

2.5 FEASIBILITY STUDY

During the system analysis study of the proposed system is carried out to see, whether it is beneficial to the organization. It is both needed and prudent to evaluate the feasibility of a project at the earliest time and minimum expenditure. Feasibility study is a test of system proposal access, its workability, impact on the organization, ability to meet the user needs, and effective use of resources. The different steps involved in feasibility analysis are.

- Formation of a project team
- Preparing the system flow chart
- Enumerating the potential candidate system
- Identifying the candidate system

The proposed system will help to solving the problem more efficiently and accurately. The reports obtained after feasibility studies are given below, they are:

- Economic Feasibility
- Technical Feasibility
- Operational Feasibility

Economic Feasibility

It will reduce expenditure and improve the quality of service. A system can develop technically and that will used if the installed must still be a good investment for the public. Financial benefits must exceed the cost. In the case of proposed system, performance of the system is effective of its accuracy, faster response and user friendly in nature. The campus-wide community for events and placements reduce unnecessary expenses and wastage of many hours by its capabilities of fast operations.

Technical Feasibility

Technical Feasibility checks the work for the project be done with current equipment, existing software technology and available personal. And if technology is required, what is the likelihood that it can develop. Also checks whether the proposed system guarantees accuracy, reliability, data security and ease of access. All the resources or implementing this software is available in this project.

Operational Feasibility

People are inherently to change, and computers have been known to facilitate chance. An estimate should be made about the reaction of the user, staff towards the development of a computerized system. Computer installations have something to do with turnover, transfer and changes in job status. Proposed projects are beneficial only if that can be turned in to information system that will meet the organizations operating requirements. In-operational feasibility study the management and users were found to have interest for a chance. Since the system is user friendly and training is less needed.

2.6 SYSTEM SPECIFICATION

The software requirements specification (SRS) is a means of translating the ideas in the minds of clients into a formal documentation. This document forms the development and software validation. The basic reason for the difficulty in software requirement specification comes from the fact that there are three interested parties-the clients, the end users and the software developer. The requirements document has to be such that the client and the user can understand easily, and the developers can use it as a basis for software development. Due to the diverse parties involved in software requirement specification, a communication gap exists. This gap arises when the client does not understand software or the software development processor when the developer does not understand the client’s problem and application area of SRS bridges this communication gap.

Problem analysis is done to obtain a clear understanding of the needs of the clients and the users, and what exactly is desired from the software. Analysis leads to the actual specification. People performing the analysis called analysts, area also responsible for specifying the requirements.

The software project is initiated by the client’s needs. In the beginning these needs are in the minds of various people in the client organization. The requirement analyst must identify their requirements by talking to these people and understanding their needs. These people and the existing documents about the current mode of operation are the basis source of information for the analyst.

2.6.1 HARDWARE SPECIFICATION

MACHINE	: INTEL DUAL CORE
MOTHER BOARD	: INTEL 945 CHIPSET
MEMORY	: 4 GB
HARD DISK	: 500 GB
MONITOR	: 18.5” LED MONITOR
KEYBOARD	: USB/3
MOUSE	: USB/3

2.6.2 SOFTWARE SPECIFICATION

Operating system	: Windows
Web technologies	: python, Django, html, CSS
Database	: MySQL
Web browser	: Google Chrome/Mozilla Firefox

PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985-1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language. Python is a high-level, interpreted, and interactive and object- oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is a MUST for students and working professionals to become a great Software Engineer especially when they are working in Web Development Domain. I will list down some of the key advantages of learning Python. Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and UNIX shell and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.The Python 2 language, i.e. Python 2.7.x, was officially discontinued on 1 January 2020 (first planned for 2015) after which security patches and other improvements will not be released for it. With Python 2's end-of-life, only Python 3.5.x and later are supported. Python interpreters are available for many operating systems. A global community of programmers develops and maintains Python, an open source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and Python development.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one and preferably only one obvious way to do it" design philosophy. Alex Martello, a Fellow at the Python Software foundation and Python book author, writer that “To describe something as ‘clever’ is not considered a compliment in the python culture.” Python’s developers strive to avoid premature optimization, and reject patches to non-critical part of the Python reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use Pippy, a just-in-time compiler. Python is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

Django

Django is a Python based free and open-source web framework, which follows the model-template-view (MTV) architectural pattern. Django's primary goal is to ease the creation of complex, database driven websites. The framework emphasizes reusability and pluggability of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

Despite having its own nomenclature, such as naming the callable objects generating the HTTP responses views, the core Django framework can be seen as an MVC architecture. It consists of an object-relational mapper (ORM) that mediates between data models (defined as Python classes) and a relational database (Model), a system for processing HTTP requests with a web templating system (View), and a regular-expression-based URL dispatcher (Controller).

Also included in the core framework are:

- A lightweight and standalone web server for development and testing.
- A form serialization and validation system that can translate between HTML forms and values suitable for storage in the database.
- A template system that utilizes the concept of inheritance borrowed from object-oriented programming.

- A caching framework that can use any of several cache methods
- Support for middleware classes that can intervene at various stages of request processing and carry out custom functions
- An internal dispatcher system that allows components of an application to communicate events to each other via pre-defined signal.
- An internationalization system, including translations of Django's own components into a variety of languages
- A system for extending the capabilities of the template engine
- An interface to Python's built-in unit test framework
- Django REST framework is a powerful and flexible toolkit for building Web APIs

Hypertext Mark-up Language (HTML)

It is the standard mark-up language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web_browsers receive HTML documents from a web_server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML_elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive_forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by *tags*, written using angle brackets.

Tags such as <imp /> and <input /> directly introduce content into the page. Other tags such as <p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content.

JavaScript

It is a lightweight, interpreted, object-oriented language with first-class functions, and is best known as the scripting language for Web pages, but it's used in many non-browser environments as well. It is a prototype-based, multi-paradigm scripting language that is dynamic, and supports object-oriented, imperative, and functional programming styles.

JavaScript runs on the client side of the web, which can be used to design / program how the web pages behave on the occurrence of an event. JavaScript is an easy to learn and powerful scripting language, widely used for controlling web page behavior.

JavaScript can function as both a procedural and an object oriented language. Objects are created programmatically in JavaScript, by attaching methods and Properties to otherwise empty objects at run time, as opposed to the syntactic class definitions common in compiled languages like C++ and Java. Once an object has been constructed it can be used as a blueprint (or prototype) for creating similar objects.

Cascading Style Sheets (CSS)

It is a style sheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, Math or XHTML). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.

ABOUT THE BACK END

MYSQL

MySQL is an Oracle-backed open source relational database management system (RDBMS) based on Structured Query Language (SQL). MySQL runs on virtually all platforms, including Linux, UNIX and Windows. Although it can be used in a wide range of applications, MySQL is most often associated with web applications and online publishing.

MySQL is an important component of an open-source enterprise stack called WAMP. WAMP is a web development platform that uses Linux as the operating system, Apache as the web server, and MySQL as the relational database management system and PHP as the object-oriented scripting language. (Sometimes Perl or Python is used instead of PHP.)

Originally conceived by the Swedish company MySQL AB, MySQL was acquired by Sun Microsystems in 2008 and then by Oracle when it bought Sun in 2010. Developers can use MySQL under the GNU General Public License (GPL), but enterprises must obtain a commercial license from Oracle. Today, MySQL is the RDBMS behind many of the top websites in the world and countless corporate and consumer-facing web-based applications, including Facebook, Twitter and YouTube.

- MySQL is the database management system, or a database server.

How MySQL works

MySQL is based on a client-server model. The core of MySQL is MySQL server, which handles all of the database instructions (or commands). MySQL server is available as a separate program for use in a client-server networked environment and as a library that can be embedded (or linked) into separate applications.

MySQL operates along with several utility programs which support the administration of MySQL databases. Commands are sent to MySQL Server via the MySQL client, which is installed on a computer.

MySQL was originally developed to handle large databases quickly. Although MySQL is typically installed on only one machine, it is able to send the database to multiple locations, as users are able to access it via different MySQL client interfaces. These interfaces send SQL statements to the server and then display the results.

MySQL Features

- **Relational Database Management System (RDBMS):** MySQL is a relational database management system
- **Easy to use** MySQL is easy to use. You have to get only the basic knowledge of SQL. You can build and interact with MySQL with only a few simple SQL statements.
- **It is secure:** MySQL consist of a solid data security layer that protects sensitive data from intruders. Passwords are encrypted in MySQL.
- **Client/ Server Architecture:** MySQL follows a client /server architecture. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they query data, save changes, etc.
- **Free to download** MySQL is free to use and you can download it from MySQL official website.
- **It is scalable:** MySQL can handle almost any amount of data, up to as much as 50 million rows or more. The default file size limit is about 4 GB. However, you can increase this number to a theoretical limit of 8 TB of data.
- **Compatible on many operating systems:** MySQL is compatible to run on many operating systems, like Novell NetWare, Windows* Linux*, many varieties of UNIX* (such as Sun* Solaris*, AIX, and DEC* UNIX), OS/2, FreeBSD*, and others. MySQL also provides a facility that the clients can run on the same computer as the server or on another computer (communication via a local network or the Internet).

- **Allows roll-back:** MySQL allows transactions to be rolled back, commit and crash recovery.
- **High Performance:** MySQL is faster, more reliable and cheaper because of its unique storage engine architecture.
- **High Flexibility:** MySQL supports a large number of embedded applications which makes MySQL very flexible.
- **High Productivity:** MySQL uses Triggers, Stored procedures and views which allows the developer to give a higher productivity.

Core MySQL features

MySQL enables data to be stored and accessed across multiple storage engines, including Inorb, CSV, and NDB. MySQL is also capable of replicating data and partitioning tables for better performance and durability. MySQL users aren't required to learn new commands; they can access their data using standard SQL commands.

Before 2016, the main difference between MySQL and SQL was that the former could be used on multiple platforms, whereas the latter could only be used on Windows. Microsoft has since expanded SQL to support Linux, a change which went into effect in 2017. When MySQL is installed via Linux, its package Management system requires custom configuration to adjust security and optimization settings.

MySQL also allows users to choose the most effective storage engine for any given table, as the program can utilize multiple storage engines for individual tables. One of MySQL's engines is Inorb. Inorb was designed for high availability. Because of this, it is not as quick as other engines. SQL uses its own storage system, but it does maintain multiple safeguards against loss of data. Both systems are able to run in clusters for high availability.

SQL Server offers a wide variety of data analysis and reporting tools. SQL Server Reporting Services is the most popular one and is available as a free download. There are similar analysis tools for MySQL available from third-party software companies**,** such as Crystal Reports XI and Actuate BIRT.

PHP MY ADMIN

PhpMyAdmin is a (web application) client for MySQL. MySQL is server where your commands get executed and returns you data, it manages all about data while PhpMyAdmin is a web Application, with user friendly, easy to use GUI makes it easy to handle database, which is difficult to use on command line. PhpMyAdmin is the web application written primarily in PHP. It’s used for managing MySQL database.

To be more specific, here is the detailed definition:

MySQL is the world's most popular open-source database. With its proven performance, reliability, and ease-of-use, MySQL has become the leading database choice for web-based applications, used by high profile web properties including Facebook, Twitter, YouTube, and all five of the top five websites. Additionally, it is an extremely popular choice as embedded database, distributed by thousands of ISVs and OEMs.

Php My Admin is a free and open-source administration tool for MySQL and MariaDB. As a portable web application written primarily in PHP, it is one of the most popular MySQL administration tools, especially for web hosting services.

What is Windows?

Windows 10 professional integrates the strengths of windows 2008 professional such as standards-based security, manageability, and reliability, with the best business features of windows 98 and windows Millennium Edition, such as plug and play, simplified user interface, and innovative support services. This combination creates the best desktop operating system for business.

It is more users friendly and a stable operating system equipped with much more added features. The operating system supports new technologies such as Digital video disks, multiple monitors etc. along with plug and play and multi display features. It has a graphical user interface operating environment. Faster computing, easy access to remote information and control remote computers are some added features. Following are the common features of Windows 10.

Faster computing, easy access to remote information and control remote computers. Built-in networking and messaging facility.

- Easier to set up, add or remove.
- Increase system security and control.
- Support advanced networking and communication.

2.7 DATAFLOW DIAGRAM

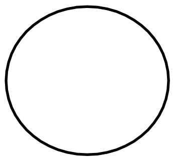
Data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. A DED shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored.

DFD is a designing tool used in the top-down approach to system Design. This context level DFD is next “exploded “, to produce a Level 1 DFD that shows some of the detail of the system being modelled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job and shows the flow of data between the various parts of the system.

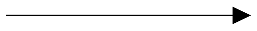
- Function- An activity or a function that is performed for some specific reason; can be manual or computerized; ultimately each process should perform only one activity.
- Data Store- collection of data that is permanently stored.
- External Entity- A person, organization or system that is external to the system but interact with it.
- Data Flow- Single piece of data or logical collection of information like a bill.

The following are some DFD symbols used in the project.

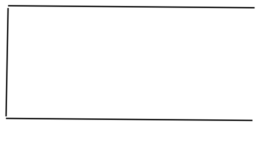
Rectangle: - It defines a source or destination of system data.



Circle: - It represents a process that transforms incoming data flow into outgoing data flow.



Arrow: - It defines data flow. It is a pipeline through which information flows.



Open rectangle: - It is used to store data or a temporary repository of data.

DATA FLOW DIAGRAM

LEVEL 0

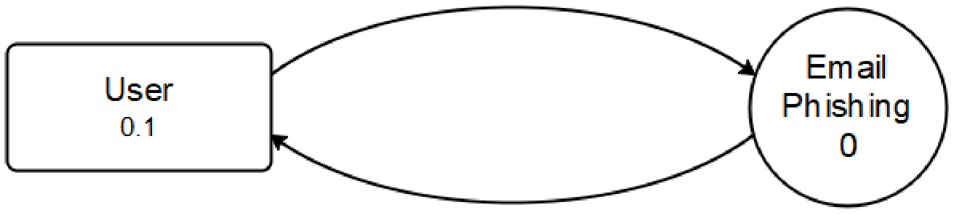


Figure 2.1: Level 0 DFD

LEVEL 1-USER

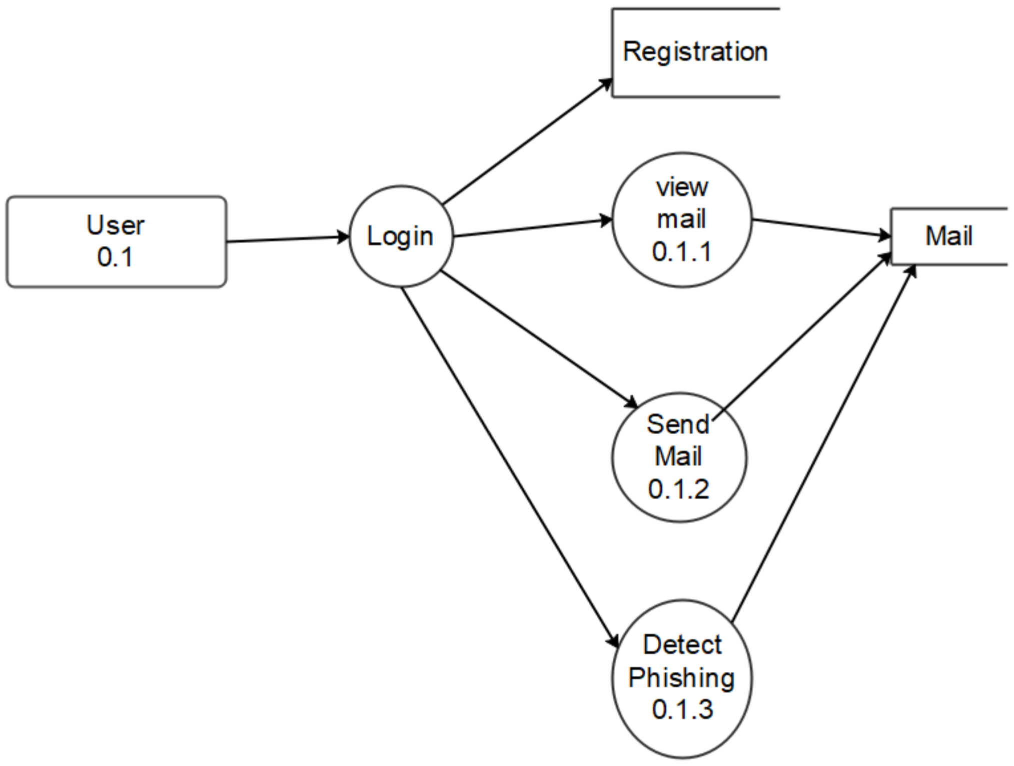


Figure 2.2: Level 1 DFD

2.8. ER DIAGRAM

E-R Diagram is a logical representation of the data for an organization and uses 3 main constraints. That is, data entities, relationships and their associated attribute.

2.8.1. Entities:

An Entity is a fundamental thing of an organization about which data may be maintained. An entity has its own identity, which distinguishes it from each other entity. An entity type is the description of all entities to which a common definition and common relationships attributes applied.

2.8.2. Relationship:

A relationship is a reason for associating two entity types. These relationships are sometimes called binary relationships. Because they involve two entity types. Some forms of data model allow more than two entity types to be associated.

2.8.3. Degree of Relationship:

The degree of a relationship is a number of entities that participate in that relationship. The 3 most common relationships in E-R models are:

1. Unary (Degree 1)
2. Binary (Degree 2)
3. Ternary (Degree 3)

Higher degree relationships are possible but they are rarely encountered in practice.

1. Unary Relationship

This is also called recursive relationship. It is a relationship between the instances of one entity type. An instance is a single occurrence of an entity type. There may be many instances of an entity type.

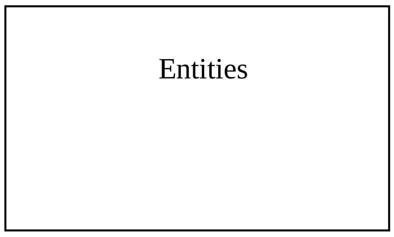
2. Binary Relationship

It is a relationship between instances of two entity types and is a most common type of relationship encountered in E-R diagram.

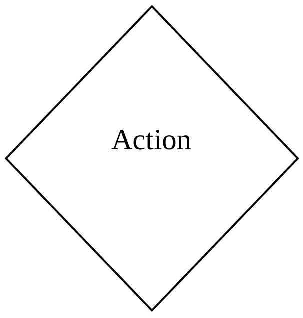
3. Ternary Relationship

It is a simultaneous relationship between instances of 3 entity types. Relationship may have provided the association of 3 entities. All the 3 entities are many to many participants. There may be one or many participants in a ternary relationship.

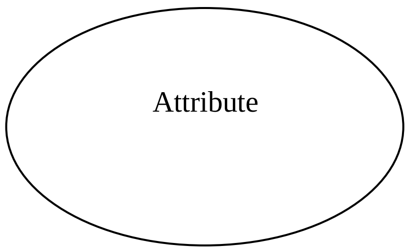
• **Entities** are represented by rectangles. An entity is an object or concept about which you want to store information.



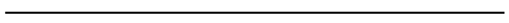
• **Actions** are represented by diamond shapes, show how two entities share information in the database.



• **Attributes** are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee's social security number mightbe the employee's key attribute



• Connecting lines are solid lines that connect attributes to show the relationships of entities in the diagram.



ER DIAGRAM

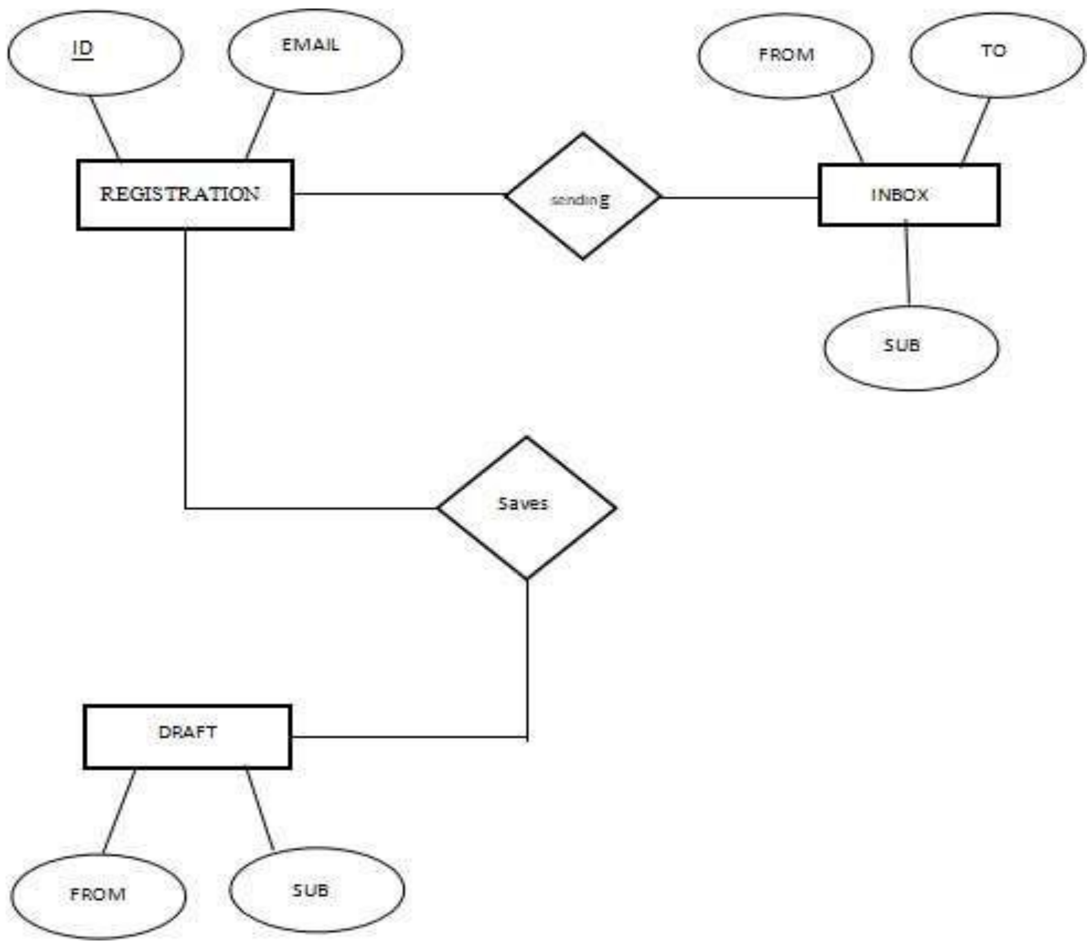


Figure 2.3: ER Diagram

SYSTEM DESIGN

3.1 INTRODUCTION

System design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. System design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of system analysis, systems architecture and systems engineering.

TYPES OF SYSTEM DESIGN

Logical Design

Logical design pertains to an abstract representation of the data flow, inputs and outputs of the system. It describes the inputs (source), outputs (destinations), databases (data stores), procedures (data flows) all in a format that meets the user requirements.

Physical Design

Physical design relates to the actual input and outputs processes of the system. It focuses on how data is entered into a system, verified, processed, and displayed as output.

Architectural Design

It is also known as high level design that focuses on the design of system architecture. It describes the structure and behavior of the system. It defines the structure and relationship between various modules of system development process.

Detailed Design

It follows Architectural design and focuses on development of each module.

Conceptual Data Modelling

It is representation of organization data which includes all the major entities and relationship. System analyses develop a conceptual data model for the current system that supports the scope and requirement for the proposed system.

3.2 INPUT DESIGN

Input design is a part of the overall design. The input methods can be broadly classified. Internal controls must be established for monitoring the number of inputs and for ensuring that the data are valid. The basic steps involved in input design are:

- Review input requirements.
- Decide how the input data flow will be implemented.
- Decide the source document.

- Prototype online input screens.
- Design the input screens.

The quality of the system input determines the quality of the system output. Input specifications describe the way data enter the system for processing. Input design features can ensure the reliability of the system and produce result.

3.3 OUTPUT DESIGN

A quality is one, which meets the requirements of end user and present the information clearly. In any system results of processing are communicated to the user and to the other system through outputs. In the outputs design it is determined how the information is to be displayed for immediate need.

It is the most important and direct source information is to user. Efficient and intelligent output design improves the system’s relationships with the user and helps in decision making. The objectives of the output design are to convey the information of all the past activities, current status and to emphasis important events. The output generally refers to the results and information that is generated from the system. Outputs from computes are required primarily to communicate the result of processing to the users.

Output also provides a means of storage by coping the results for later reference in consolation. There is a chance that some of the end users will not actually operate the input data or information through workstations but will see the output from the system.

Two phases of the output design are:

1. Output Definitions
2. Output Specification

Output definitions considers the type of outputs contents, its frequency and its volume, the appropriate outputs media is determined for output. Once the media is chosen, the details specification of output documents are carried out. The nature of output required from the proposed system is determined during logical design stage. It takes the outline of the output from the logical design and produces output as specified during the logical design phase. In a project, when designing the output, the system must accomplish the following:

- Determine the information to present.
- Decide whether to display, speak the information and select the output medium.
- Arrange the information in acceptable format.
- Decide how to distribute the output to the intended receipt.

Thus, by following the above specification, a high-quality output can be generated. Outputs from compute system are required primarily to communicate thee result of processing to users. Computer output is the most important and direct source of information to the user. Efficiency, intelligible output should improve the system's relationship with the user and help in decision making. The output devices to consider depend on factors as compatibility of the device with the system, response time requirements, expected print quality, number of copies needed etc.

3.4 DATA BASE DESIGN

A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and efficiently. The general objective is to make information access easy, quick, inexpensive and flexible for users. The general theme behind a database is to integrate all information. Database design is recognized as a standard of management information system and is available virtually for every computer system. In database design several specific objectives are considered:

- Ease of learning and use
- Controlled redundancy
- Data independence
- More information at low cost
- Accuracy and integrity
- Recovery from failure
- Privacy and security

NORMALIZATION

Designing a database is complete task and the normalization theory is a useful aid in the design process. The process of normalization is concerned with transformation of conceptual schema into computer representation form. There will be need for most databases to grow by adding new attributes and new relations. The data will be used new ways. Tuple will be added and deleted. Information stored may undergo updating also. New association may also be added. In such situation the performance of a database is entirely depend upon its design. A bad database design may lead to certain undesirable things like:

1. Repetition of information
2. Inability to represent certain information.
3. Loss of information

To minimize these anomalies, normalization may be used. If the database is in a normalized from, the data can be growing without, in most cases, forcing the rewriting application programs. This is important because of the excessive and growing cost of maintaining an organization’s application programs and its data from the disrupting effects of database growth. As the quality of application programs increase, the cost of maintaining the without normalization will rise to prohibitive levels, A normalized database can also encompass many related activities of an organization thereby minimizing the need for rewriting the applications of programs. Thus, normalization helps one attain a good database design and there by ensures continued efficiency of database.

We can define the procedure as the successive reduction of a given collection of relations to some more desirable from. This procedure is reversible. That is, it is always possible to take the output from the procedure and convert them back into input. In this process, no information is lost. So, it is also called “no lose decomposition”.

First normal form: A relation is first normal form (1NF) if and all its attributes are based on single domain. The objective of normalization a table is to remove its repeating groups and ensure that all entries of the resulting table have at most single value.

Second normal form: A table is said to be second normal form (2NF), when it is in 1NF and every attribute in the record is functionally dependent upon the whole key, and not just a part of the key.

Third normal form: A table is in third normal form (3NF). When it is in 2NF and every non-key attribute is functionally dependent on just the primary key.

Fourth normal form: Fourth Normal Form, is a database normalization technique used in relational database design. Normalization is the process of organizing data in a database to eliminate redundancy and ensure data integrity.

Fifth normal form: Fifth Normal Form, is the highest level of normalization in relational database design. It builds upon the concepts of previous normal forms, such as 1NF, 2NF, 3NF, and 4NF, and aims to eliminate a specific type of data anomaly known as a join dependency.

SYSTEM DEVELOPMENT

4.1 MODULE DESCRIPTION

1. DATASET

The dataset has been divided into a training set and testing set. Both the training set and the testing set contain emails without header and emails with header. In this paper, we only focus on email data with the header. Due to the irrationality of the segmentation of the training set and the testing set in the original dataset, after merging the two datasets, the training-validation set and the testing set are redevised. The dataset is divided by stratified random sampling; that is, random samples are taken from legitimate email and phishing email at the same proportion. This ensures that the two datasets used in training and testing phases are well.

2. USER QUERIES

Users can have queries about the process. This part of project is dedicated to make and get response for queries that are needed to answerable. The major part of the modules is making project as interactive one, queries have been very normally arise to users regarding different details about the process.

3. GRAPH ANALYSIS

Graph analysis is the part where admin can knows the statistics about process of details. The data are taken from the project flow and it shows until updated value. The data are gives clear solution to admin that part of improvement and user satisfaction and other factors.

4. ANALYSIS

Analysis of email structure. A circle represents a character, and a rectangle represents a word. A rectangle is filled with an indefinite number of circles, indicating that the word consists of an indefinite number of characters.

SOURCE CODE

```
from django.shortcuts import render
from django.http import HttpResponse,HttpResponseRedirect
import MySQLdb
import datetime
import subprocess
from django.core.files.storage import FileSystemStorage
import os
db=MySQLdb.connect("localhost","root","","dataexchange")
c=db.cursor()
def login(request):

    error=""
    request.session['username']=""
    password=""
    if(request.POST):
        username=request.POST.get("uname")
        request.session['username']=username
        password=request.POST.get("password")
        if((username=='admin@gmail.com') and (password=='admin')):
             return HttpResponseRedirect("/adminhome/")
        else:

            c.execute("select count('"+ username +"') from registration where email_id='"+ username +"' and 
password='"+password+"'" )
            data=c.fetchone()
            if data[0]==1:
                c.execute("select status from registration where email_id='"+username+"' ")
                data1=c.fetchone()
                #if data1:
                if (data[0]==1 and data1[0]=="approved"):
                        # subprocess.call("E:\\windapp\\windapp\\bin\\Debug\\windapp.exe")
                        # f=open("E:\\face.txt","r")
                        # data=f.read()
                        # z=len(data)
                        # f.close()
                        # if(data[0:z-1]==username):
                        return HttpResponseRedirect("/userhome/")
                else:
                        if(data1[0]=="rejected"):
                            error="you have been rejected from ADMIN"
                        else:
                            error="enter valid email"
            else:
                  return HttpResponseRedirect("/")


            #db.commit()
    return render(request,"login.html",{"error":error})
def adminlogin(request):
    error=""
    request.session['username']=""
    if(request.POST):
        username=request.POST.get("uname")
        request.session['username']=username
        password=request.POST.get("password")
        if((username=='admin') and (password=='admin')):
            return HttpResponseRedirect("/adminhome/")
        else:
            error="enter valid email"
    return render(request,"adminlogin.html",{"error":error})
def forgot(request):
    error=""
    if(request.POST):
        username=request.POST.get("uname")
        mobile=request.POST.get("mobile")
        request.session['username']=username
        c.execute("select count('"+ username +"') from registration where email_id='"+ username +"' and
mobile='"+ mobile +"'")
        data=c.fetchone()
        if (data[0]==1):
            return HttpResponseRedirect("/security/")
        else:
            error="enter valid email"

    return render(request,"forgot.html",{"error":error})

def security(request):
    error=""
    if(request.POST):
        answer=request.POST.get("answer")
        c.execute("select count('"+answer+"') from registration where answer='"+answer+"'")
        data=c.fetchone()
        if (data[0]==1):
            return HttpResponseRedirect("/newpass/")
        else:
            error="enter correct answer"
    return render(request,"security.html",{"error":error})

def newpass(request):
    error=""
    unam=request.session['username']
    if(request.POST):
        password=request.POST.get("password")
        cpassword=request.POST.get("cpassword")
        if(password==cpassword):
            c.execute("update registration set password='"+password+"' where email_id='"+unam+"'")
            db.commit()
            return HttpResponseRedirect("/login/")
        else:
            error="password mismatch"
    return render(request,"newpass.html",{"error":error})

def reg(request):
    error=""
    msg=""
    err=""
    msg1=""
    if(request.POST):
        # subprocess.call('E:\\windapp\\windapp\\bin\\Debug\\windapp.exe')
        name=request.POST.get("name")
        address=request.POST.get("address")
        dob=request.POST.get("dob")
        gender=request.POST.get("gender")
        email=request.POST.get("email")
        mobile=request.POST.get("mobile")
        password=request.POST.get("password")
        cpassword=request.POST.get("cpassword")
        answer=request.POST.get("answer")

        if(request.FILES['img']):
            myfile=request.FILES['img']
            fs=FileSystemStorage()
            filename=fs.save(myfile.name,myfile)
            fileurl=fs.url(filename)
        else:
            fileurl="/static/media/a.png"
        if(password==cpassword):
            c.execute("select count('"+email+"') from registration where email_id='"+email+"'")
            data=c.fetchone()
            c.execute("select count('"+mobile+"') from registration where mobile='"+mobile+"'")
            mob=c.fetchone()
            if (data[0]==0):
                if (mob[0]==0):
                    c.execute("insert into
registration(name,address,dob,gender,email_id,mobile,image,password,answer)
values('"+name+"','"+address+"','"+dob+"','"+gender+"','"+email+"','"+str(mobile)+"','"+fileurl+"','"+pas
sword+"','"+answer+"')")
                    db.commit()
                    msg="wait for admin to approve"
                else:
                    msg1="existing mobile number"


            else:
                error="USERNAME ALREDY EXISTED"
        else:
            err="password and confirm password donot match"

    return render(request,"reg.html",{"error":error,"msg":msg,"err":err,"msg1":msg1})

def save(request):
    if(request.session['username']):
        unam=request.session['username']
        c.execute("select * from registration where email_id='"+unam+"'")
        data2=c.fetchall()
        content="no data"
        s=""
        if(request.GET.get("msg")):
            content=request.GET.get("msg")
            sendto=request.GET.get("to")
            date=datetime.date.today()
            subject=request.GET.get("sub")
            status="draft"
            unam=request.session['username']
            s="insert into message(`from`,sendto,date,subject,content,status) values('"+ str(unam) +"','"+
str(sendto) +"','"+ str(date)+"','"+ str(subject) +"','"+ str(content) +"','"+ str(status) +"')"
        # c.execute("insert into message(from,sendto,subject,content)
values('"+unam+"','"+sendto+"','"+subject+"','"+content+"')")
            c.execute(s)
            db.commit()
    else:
        return HttpResponseRedirect("/login/")

    return render(request,"message.html",{"data":s,"msg":request.POST.get("content"),"data2":data2})


def message(request):
    det=[]
    if(request.session['username']):
        unam=request.session['username']
        c.execute("select * from registration where email_id='"+unam+"'")
        data2=c.fetchall()
        content="no data"
        s="sent"
        c.execute("select * from(select * from message where sendto='"+unam+"' and status='"+s+"' order
by m_id desc limit 2)as r order by m_id")
        count2=c.fetchall()
        for i in count2:
            c.execute("select name,image from registration where email_id='"+ str(i[2]) +"'")
            count3=c.fetchone()
            det.append(count3)
        c.execute("select complaint from feedback order by f_id desc limit 3")
        feed=c.fetchall()
        if(request.GET.get("msg")):
            content=request.GET.get("msg")
            sendto=request.GET.get("to")
            date=datetime.date.today()
            subject=request.GET.get("sub")
            status="sent"
            unam=request.session['username']
            s="insert into message(`from`,sendto,date,subject,content,status) values('"+ str(unam) +"','"+
str(sendto) +"','"+ str(date)+"','"+ str(subject) +"','"+ str(content) +"','"+ str(status) +"')"
        # c.execute("insert into message(from,sendto,subject,content)
values('"+unam+"','"+sendto+"','"+subject+"','"+content+"')")
            c.execute(s)
            db.commit()

        if("send" in request.POST):
            if(request.GET.get("content")==""):
                content=request.POST.get("content")
            else:
                content=request.POST.get("content")

            sendto=request.POST.get("sendto")
            date=datetime.date.today()
            subject=request.POST.get("subject")
            unam=request.session['username']
            status="sent"
            import main
            s="insert into message(`from`,sendto,date,subject,content,status)
values('"+str(unam)+"','"+str(sendto)+"','"+ str(date)+"','"+str(subject)+"','"+str(content)+"','"+status+"')"
            #c.execute("insert into message(from,sendto,subject,content)
values('"+unam+"','"+sendto+"','"+subject+"','"+content+"')")
            c.execute(s)
            db.commit()

        if("draft" in request.POST):
            if(request.GET.get("content")==""):
                content=request.POST.get("content")
            else:
                content=request.POST.get("content")

            sendto=request.POST.get("sendto")
            date=datetime.date.today()
            subject=request.POST.get("subject")
            unam=request.session['username']
            status="Draft"
            s="insert into message(`from`,sendto,date,subject,content,status)
values('"+str(unam)+"','"+str(sendto)+"','"+ str(date)+"','"+str(subject)+"','"+str(content)+"','"+status+"')"
            #c.execute("insert into message(from,sendto,subject,content)
values('"+unam+"','"+sendto+"','"+subject+"','"+content+"')")
            c.execute(s)
            db.commit()
    else:
        return HttpResponseRedirect("/login/")

    return
render(request,"message.html",{"data":s,"msg":request.POST.get("content"),"data2":data2,"det":det,"fee
d":feed})
def compose(request):
    if(request.session['username']):
        unam=request.session['username']
        c.execute("select * from registration where email_id='"+unam+"'")
        data2=c.fetchall()
        frm=request.GET.get("count")
        s="select `from`,subject,content from message where m_id='"+frm+"'"
        print(s)
        c.execute(s)
        data=c.fetchall()
        frm1=data[0][0]
        sub=data[0][1]
        con=data[0][2]
        request.session['frm1']=frm1
        if request.POST:
                return HttpResponseRedirect("/message1/")
    else:
        return HttpResponseRedirect("/login/")
    return render(request,"compose.html",{"frm1":frm1,"sub":sub,"con":con,"s":s,"data2":data2})
def draft1(request):
    det=[]
    if(request.session['username']):
        s1="sent"
        unam=request.session['username']
        c.execute("select * from(select * from message where sendto='"+unam+"' and status='"+s1+"' order
by m_id desc limit 2)as r order by m_id")
        count2=c.fetchall()
        for i in count2:
            c.execute("select name,image from registration where email_id='"+ str(i[2]) +"'")
            count3=c.fetchone()
            det.append(count3)
        c.execute("select complaint from feedback order by f_id desc limit 3")
        feed=c.fetchall()


        c.execute("select * from registration where email_id='"+unam+"'")
        data2=c.fetchall()
        frm=request.GET.get("count")
        request.session['did']=frm
        s="select sendto,subject,content from message where m_id='"+frm+"'"
        print(s)
        c.execute(s)
        data=c.fetchall()
        frm1=data[0][0]
        sub=data[0][1]
        con=data[0][2]
        request.session['frm1']=frm1
        request.session['sub']=sub
        request.session['con']=con

        if request.POST:
                return HttpResponseRedirect("/draft2/")
    else:
        return HttpResponseRedirect("/login/")
    return render(request,"draft1.html",{"frm1":frm1,"sub":sub,"con":con,"s":s,"data2":data2})
def inbox(request):
    if(request.session['username']):
        data3=""
        s="sent"
        c1="draft"
        unam=request.session['username']
        c.execute("select * from registration where email_id='"+unam+"'")
        data2=c.fetchall()
        c.execute("select count(*) from message where sendto='"+unam+"' and `status`='"+s+"'")
        data1=c.fetchone()
        c.execute("select count(*) from message where `from`='"+unam+"' and `status`='"+c1+"'")
        c1=c.fetchone()
        c.execute("select `from`,date,subject,content,sendto,m_id from message where sendto='"+unam+"'
and `status`='"+s+"'")
        data=c.fetchall()
        if(request.POST):
            request.session["z"]=request.POST.get("se")

            return HttpResponseRedirect("/search/")
            return render(request,"search.html",{"data3":data3})
    else:
        return HttpResponseRedirect("/login/")
    return render(request,"inbox.html",{"data":data,"data1":data1[0],"data2":data2,"c1":c1[0]})
```

```
def search(request):
    if(request.session['username']):
        z=request.session["z"]
        unam=request.session['username']
        y=z+'%'
        s="select * from message where content like '"+z+"%' and sendto ='"+unam+"'"
        c.execute(s)
        data3=c.fetchall()
        unam=request.session['username']
        c.execute("select * from registration where email_id='"+unam+"'")
        data2=c.fetchall()
    else:
        return HttpResponseRedirect("/login/")
    return render(request,"search.html",{"data2":data2,"data3":data3})

def sent(request):
    det=[]
    c1="draft"
    if(request.session['username']):

        s1="sent"
        unam=request.session['username']
        c.execute("select * from(select * from message where sendto='"+unam+"' and status='"+s1+"' order
by m_id desc limit 2)as r order by m_id")
        count2=c.fetchall()
        for i in count2:
            c.execute("select name,image from registration where email_id='"+ str(i[2]) +"'")
            count3=c.fetchone()
            det.append(count3)
        c.execute("select complaint from feedback order by f_id desc limit 3")
        feed=c.fetchall()

        s="sent"
        c.execute("select count(*) from message where sendto='"+unam+"' and `status`='"+s+"'")
        data1=c.fetchone()
        c.execute("select count(*) from message where `from`='"+unam+"' and `status`='"+c1+"'")
        c1=c.fetchone()
        c.execute("select * from registration where email_id='"+unam+"'")
        data2=c.fetchall()
        c.execute("select * from message where `from`='"+unam+"' and `status`='"+s+"'")
        data=c.fetchall()
        print("select * from message where `from`='"+unam+"' and `status`='"+s+"'")
    else:
        return HttpResponseRedirect("/login/")
    return
render(request,"sent.html",{"data":data,"data2":data2,"feed":feed,"det":det,"c1":c1[0],"data1":data1[0]})

def draft(request):
    det=[]
    if(request.session['username']):
        s1="sent"
        unam=request.session['username']
        c.execute("select * from(select * from message where sendto='"+unam+"' and status='"+s1+"' order
by m_id desc limit 2)as r order by m_id")
        count2=c.fetchall()
        for i in count2:
            c.execute("select name,image from registration where email_id='"+ str(i[2]) +"'")
            count3=c.fetchone()
            det.append(count3)
        c.execute("select complaint from feedback order by f_id desc limit 3")
        feed=c.fetchall()

        s="draft"
        c1="sent"

        c.execute("select * from registration where email_id='"+unam+"'")
        data2=c.fetchall()
        c.execute("select count(*) from message where sendto='"+unam+"' and `status`='"+c1+"'")
        data1=c.fetchone()
        c.execute("select count(*) from message where `from`='"+unam+"' and `status`='"+s+"'")
        data4=c.fetchone()
        c.execute("select sendto,date,subject,content,m_id from message where `from`='"+unam+"' and
`status`='"+s+"'")
        data=c.fetchall()
    else:
        return HttpResponseRedirect("/login/")
    return
render(request,"draft.html",{"data":data,"data1":data1[0],"data2":data2,"data4":data4[0],"feed":feed,"det
":det})
def message1(request):
    if(request.session['username']):
        c.execute("select * from registration")
        data=c.fetchall()
        frm1=request.session['frm1']
        if(request.POST):
            if(request.GET.get("content")==""):
                content=request.POST.get("content")
            else:
                content=request.POST.get("content")

            sendto=request.POST.get("sendto")
            date=datetime.date.today()
            subject=request.POST.get("subject")
            unam=request.session['username']
            status="sent"
            s="insert into message(`from`,sendto,date,subject,content,status)
values('"+str(unam)+"','"+str(sendto)+"','"+ str(date)+"','"+str(subject)+"','"+str(content)+"','"+status+"')"
            #c.execute("insert into message(from,sendto,subject,content)
values('"+unam+"','"+sendto+"','"+subject+"','"+content+"')")
            c.execute(s)
            db.commit()
            return HttpResponseRedirect("/inbox/")

    return render(request,"message1.html",{"frm1":frm1,"data":data})
def draft2(request):
    if(request.session['username']):
        c.execute("select * from registration")
        data=c.fetchall()
        frm1=request.session['frm1']
        sub=request.session['sub']
        con=request.session['con']
        if(request.POST):
            if(request.GET.get("content")==""):
                content=request.POST.get("content")
            else:
                content=request.POST.get("content")

            sendto=request.POST.get("sendto")
            date=datetime.date.today()
            subject=request.POST.get("subject")
            unam=request.session['username']
            status="sent"
            s="insert into message(`from`,sendto,date,subject,content,status)
values('"+str(unam)+"','"+str(sendto)+"','"+ str(date)+"','"+str(subject)+"','"+str(content)+"','"+status+"')"
            #c.execute("insert into message(from,sendto,subject,content)
values('"+unam+"','"+sendto+"','"+subject+"','"+content+"')")
            c.execute(s)
            db.commit()
            # c.execute("select max(m_id) from message where `from`='"+ str(unam) +"'")
            # did=c.fetchone()

            #draft
            c.execute("delete from message where m_id='"+ str(request.session['did']) +"'")
            db.commit()

            return HttpResponseRedirect("/inbox/")

    return render(request,"draft2.html",{"frm1":frm1,"sub":sub,"con":con,"data":data})

def userview(request):
    if(request.session['username']):
        c.execute("select * from registration")
        data=c.fetchall()
        id=request.GET.get("id")
        status=request.GET.get("status")
        if(id):
            c.execute("update registration set status='"+status+"' where u_id='"+id+"';")
            db.commit()
    else:
        return HttpResponseRedirect("/login/")
    return render(request,"userview.html",{"data":data})
def profile(request):
    det=[]
    if(request.session['username']):
        s1="sent"
        unam=request.session['username']
        c.execute("select * from(select * from message where sendto='"+unam+"' and status='"+s1+"' order
by m_id desc limit 2)as r order by m_id")
        count2=c.fetchall()
        for i in count2:
            c.execute("select name,image from registration where email_id='"+ str(i[2]) +"'")
            count3=c.fetchone()
            det.append(count3)
        c.execute("select complaint from feedback order by f_id desc limit 3")
        feed=c.fetchall()

        c.execute("select * from registration where email_id='"+unam+"'")
        data=c.fetchall()
        if(request.POST):
            return HttpResponseRedirect("/editprofile/")
    else:
        return HttpResponseRedirect("/login/")
    return render(request,"profile.html",{"data":data,"feed":feed,"det":det})
def editprofile(request):
    det=[]
    if(request.session['username']):
        s1="sent"
        unam=request.session['username']
        c.execute("select * from(select * from message where sendto='"+unam+"' and status='"+s1+"' order 
by m_id desc limit 2)as r order by m_id")
        count2=c.fetchall()
        for i in count2:
            c.execute("select name,image from registration where email_id='"+ str(i[2]) +"'")
            count3=c.fetchone()
            det.append(count3)
        c.execute("select complaint from feedback order by f_id desc limit 3")
        feed=c.fetchall()

        c.execute("select * from registration where email_id='"+unam+"'")
        data=c.fetchall()
        for d in data:
                uid=d[0]
        if(request.POST):
            name=request.POST.get("name")
            address=request.POST.get("address")
            dob=request.POST.get("dob")
            #gender=request.POST.get("gender")
            email=request.POST.get("email")
            mobile=request.POST.get("mobile")
            #password=request.POST.get("password")
            c.execute("update registration set 
name='"+name+"',address='"+address+"',dob='"+str(dob)+"',email_id='"+email+"',mobile='"+str(mobile
)+"' where u_id='"+str(uid)+"'")
            db.commit()
            return HttpResponseRedirect("/profile/")
    else:
        return HttpResponseRedirect("/login/")
    return render(request,"editprofile.html",{"data":data,"feed":feed,"det":det})
def adminhome(request):
    if request.session["username"]:
       return render(request,"adminhome.html")
    else:
         return HttpResponseRedirect("/login")




def voice(request):
    return render(request,"voice.html")
def commonhome(request):
    return render(request,"commonhome.html")

def userhome(request):
    det=[]
    if(request.session['username']):
        unam=request.session['username']
        c.execute("select * from registration where email_id='"+unam+"'")
        data=c.fetchall()
        c.execute("select count(content) from message where sendto='"+unam+"'")
        count=c.fetchone()
        c.execute("select count(*) from message")
        count1=c.fetchone()
        c.execute("select count(*) from registration")
        data1=c.fetchone()
        s="sent"
        c.execute("select * from(select * from message where sendto='"+unam+"' and status='"+s+"' order
by m_id desc limit 2)as r order by m_id")
        count2=c.fetchall()
        for i in count2:
            c.execute("select name,image from registration where email_id='"+ str(i[2]) +"'")
            count3=c.fetchone()
            det.append(count3)
        c.execute("select complaint from feedback order by f_id desc limit 3")
        feed=c.fetchall()
    else:
        return HttpResponseRedirect("/login/")
    return
render(request,"userhome.html",{"data":data,"count":count[0],"data1":data1[0],"count1":count1[0],"det"
:det,"feed":feed})

def feedback(request):
    det=[]
    if(request.session['username']):
        s1="sent"
        unam=request.session['username']
        c.execute("select * from(select * from message where sendto='"+unam+"' and status='"+s1+"' order
by m_id desc limit 2)as r order by m_id")
        count2=c.fetchall()
        for i in count2:
            c.execute("select name,image from registration where email_id='"+ str(i[2]) +"'")
            count3=c.fetchone()
            det.append(count3)
        c.execute("select complaint from feedback order by f_id desc limit 3")
        feed=c.fetchall()

        c.execute("select * from registration where email_id='"+unam+"'")
        data=c.fetchall()
        content="no data"
        s=""
        if(request.GET.get("msg")):
            content=request.GET.get("msg")
            msgto="admin"

            date=datetime.date.today()
            subject=request.GET.get("sub")
            status="sent"
            unam=request.session['username']
            s="insert into feedback(`from`,`to`,`date`,sub,complaint) values('"+unam+"','"+msgto+"','"+
str(date)+"',"+subject+","+content+"')"
        # c.execute("insert into message(from,sendto,subject,content)
values('"+unam+"','"+sendto+"','"+subject+"','"+content+"')")
            c.execute(s)
            db.commit()
        if(request.POST):
            unam=request.session['username']
            msgto="admin"
            date=datetime.date.today()
            subject=request.POST.get("subject")
            feedback=request.POST.get("mycontent")
            c.execute("insert into feedback(`from`,`to`,`date`,sub,complaint)
values('"+unam+"','"+msgto+"','"+ str(date)+"','"+subject+"','"+feedback+"')")
            db.commit()
    else:
        return HttpResponseRedirect("/login/")

    return render(request,"feedback.html",{"data":data,"det":det,"feed":feed})

def viewfeedback(request):
    if(request.session['username']):
        c.execute("select * from feedback")
        data=c.fetchall()
    else:
        return HttpResponseRedirect("/login/")
    return render(request,"viewfeedback.html",{"data":data})
def changeimage(request):
    if(request.session['username']):
        unam=request.session['username']
        c.execute("select * from registration where email_id='"+unam+"'")
        data=c.fetchall()
        if(request.POST):
            if(request.FILES['img']):
                    myfile=request.FILES['img']
                    fs=FileSystemStorage()
                    filename=fs.save(myfile.name,myfile)
                    fileurl=fs.url(filename)
            c.execute("update registration set image='"+fileurl+"' where email_id='"+unam+"'")
            db.commit()
            return HttpResponseRedirect("/profile/")
    else:
        return HttpResponseRedirect("/login/")
    return render(request,"changeimage.html",{"data":data})

# Create your views here.
```

SYSTEM IMPLEMENTATION

5.1 TESTING

The term software testing is defined as to find for the errors in the application that might lead to fault or failure of the whole application. There are testing conditions that the system must pass to says that it is tested and working properly. The quality and reliability are also attained by going through the process of testing.

- **UNIT TESTING**

Unit testing is a level of a software testing where individual units/components of software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.) Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing.

- **INTEGRATION TESTING**

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of the level of testing is to expose faults in the interaction between integrated units. The purpose of this level of testing is to expose faults in the intersection between integrated units. The drivers and test stubs are used to assist in Integration Testing.

- **SYSTEMS TESTING**

System testing is a level of software testing where complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements.

- **ACCEPTANCE TESTING**

Acceptance testing is performed to ensure that the functional, behavioral, and performance requirements of the software are met IEEE defines acceptance testing as a 'formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system.'

During acceptance testing, the software is tested and evaluated by a group of users either at the developer's site or user's site. This enables the users to site. The enables the users to test the software themselves and analyses whether it is meeting their requirements. To perform acceptance testing, a predetermined set of data is given to the software as input. It is important to know the expected output before performing acceptance testing so that outputs produced by the software as a result of testing can be compared with them. Based on the results of tests, users decide whether to accept or reject the software is correct and is accepted; otherwise, it is rejected.

- ❖ **REGRESSION TESTING**

Regression testing is the retesting of a software system to confirm that changes made to few parts of the codes has not any side effects on existing system functionalities. It is to ensure that old codes are still working as they were before introduction of the new change. The ideal process would be to create an extensive test suite and run it after each and every change.

5.2 VALIDATION CHECKS

A validation check ascertains that the value (or data) input into a computer is valid. Validation checks are performed automatically by computer to ensure that entered data is correct and reasonable. Validation means check the input submitted by the user. There are two types of validation are available in PHP. They are as follows −

- **Client-Side Validation** − Validation is performed on the client machine web browsers.
- **Server-Side Validation** – After submitted by data, the data has sent to a server and perform validation checks in server machine.

Server-Side Validation

In the Server-Side Validation, the input submitted by the user is being sent to the server and validated using one of server-side scripting languages such as ASP.Net, PHP etc. After the validation process on the Server Side, the feedback is sent back to the client by a new dynamically generated web page. It is better to validate user input on Server Side because you can protect against the malicious users, who can easily bypass your Client-Side scripting language and submit dangerous input to the server.

Client-Side Validation

In the Client-Side Validation, you can provide a better user experience by responding quickly at the browser level. When you perform a Client-Side validation, all the user inputs validated in the user's browser itself. Client-Side validation does not require a round trip to the server, so the network traffic which will help your server perform better. This type of validation is done on the browser side using script languages such as JavaScript, VBScript or HTML5 attributes.

5.3 IMPLEMENTATION

Implementation includes placing the system into operation and providing the users and operation personnel with the necessary documentation to use and maintain the new system. Implementation includes all those activities that take place to convert from the old system. Proper implementation is essential to provide are liable system to meet the organizational requirements. Successful implementation may not guarantee improvement in the organization using the new system, as well as improper installation will prevent. There are four methods,

- **Parallel approach:** The old system is operated with the new system.
- **Direct cut over method:** The old system is replaced with the new system.
- **Pilot approach:** Working version of the system is implemented in one part of the organization based on the feedback, changes are made, and the system is installed in the rest of the organization by one the other methods.
- **Phase-in-method:** Gradually implements the system across all users.

5.4 SECURITY

The protection of computer-based resources that includes hardware, software, data, procedures and against unauthorized use or natural.

- Integrity
- Privacy
- Disaster is known as system security.
- System security can be divided into four related issues.
- Confidentiality
- Security

Data security is the protection of data from loss, disclosure, modification and destruction.

System Integrity refers to the power functioning of hardware and programs, appropriate physical security, and safety against external threats such as eavesdropping and writes tapping.

5.5 SYSTEM MAINTENANCE

Maintenance means restoring something to its original conditions. Enhancement means adding, modifying the code to support the changes in the user specification. System maintenance conforms the system to its original requirements and enhancement adds to system capability by incorporating new requirements.

Thus, maintenance changes the existing system, enhancement adds features to the existing system, and development replaces the existing system. It is an important part of system development that includes the activities which corrects errors in system design and implementation, updates the documents, and tests the data.

Maintenance Types

System maintenance can be classified into four types –

- Corrective Maintenance
- Adaptive Maintenance
- Perfective Maintenance
- Preventive Maintenance

Corrective Maintenance

Corrective Maintenance deals with the repair of faults or defects found in day-today system functions. A defect can result due to errors in software design, logic and coding. Design errors occur when changes made to the software are incorrect, incomplete, wrongly communicated, or the change request is misunderstood. Logical errors result from invalid tests and conclusions, incorrect implementation of design specifications, faulty logic flow, or incomplete implementation of design specifications, faulty logic flow, or incomplete test of data. All these errors, referred to as residual errors, prevent the software from confirming to its agreed specifications. Note that the need for corrective maintenance is usually initiated by big reports drawn by the users.

Adaptive Maintenance

Adaptive Maintenance is the implementation of changes in a part of the system, which has been affected by a change that occurred in some other part of the system. Adaptive Maintenance consists of adapting software to changes in the environment such as the hardware or the operating system. The term environment in this context refers to the conditions and the influences which act (from outside) on the system. For example, business rules, work patterns and government policies have a significant impact on the software system.

Perfective Maintenance

Perfective Maintenance mainly deals with implementing new or changed user requirements. Perfective Maintenance involves making functional enhancements to the system in addition to the activities to increase the system's performance even when the changes have not been suggested by faults. This includes enhancing both the function and efficiency of the code and changing the functionalities of the system as per the users' changing needs.

Preventive Maintenance

Preventive Maintenance involves performing activities to prevent the occurrence of errors. It tends to reduce the software complexity thereby improving program understand ability and increasing software maintainability. It comprises documentation updating, code optimization and code restructuring. Documentation updating involves modifying the documents affected by the changes in order to correspond to the present state of the system. Code optimization involves modifying the programs for faster execution or efficient use of storage space. Code restructuring involves transforming the program structure for reducing the complexity in source code and making it easier to understand.

FUTURE SCOPE OF PROJECT

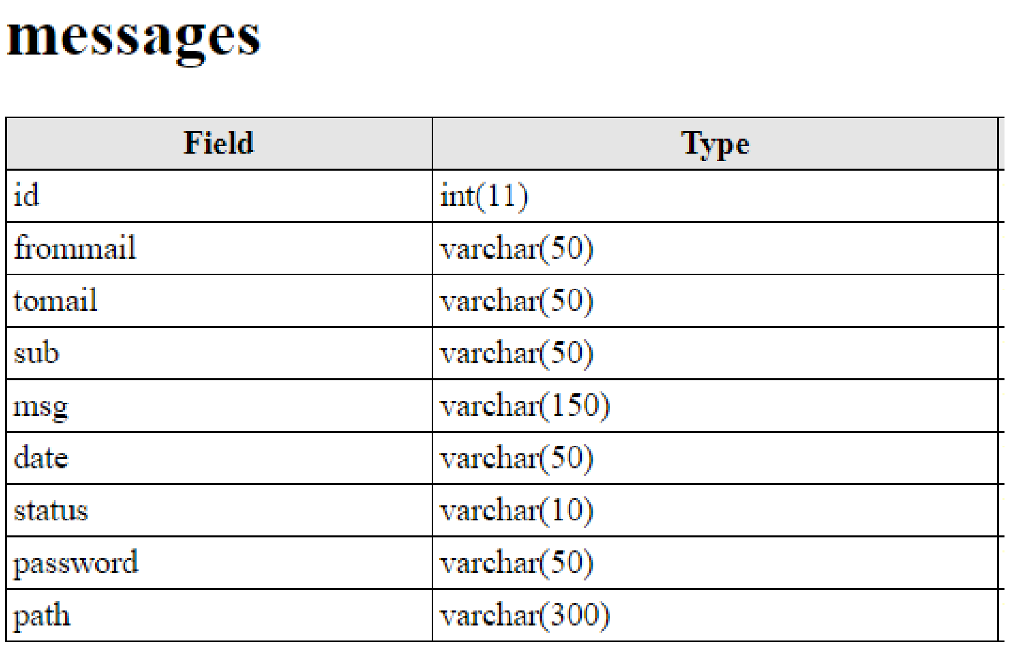
6.1 FUTURE ENHANCEMENT

Almost every project is subjected to change on depending on the client requirements. Since this system is subjected to change for each client, there is always a scope for further enhancement. The system and the architecture of the assessment system is a compatible one, so addition of new modules can be done without much difficulty. The software is developed in visual basic which makes the system more reliable and compatible with other environments. The application proves better extensibility and flexibility for future enhancements. Any further requirement application is possible with the same feature guaranteed. It is a user – friendly system, which is very easy and convenient to use. The system is complete in the sense that it is operational, and it is tested by entering data and getting reports in proper order. During the development of this project coding standards are followed for easy maintainability and extensibility. Though the new system provides a base for improving the efficiency of operations, there are lots of further enhancement that can be added to this project. Keeping this in view, provision has been made in the system to facilitate easy modification updating in future. Any modification will not affect the normal working of the system.

APPENDIX

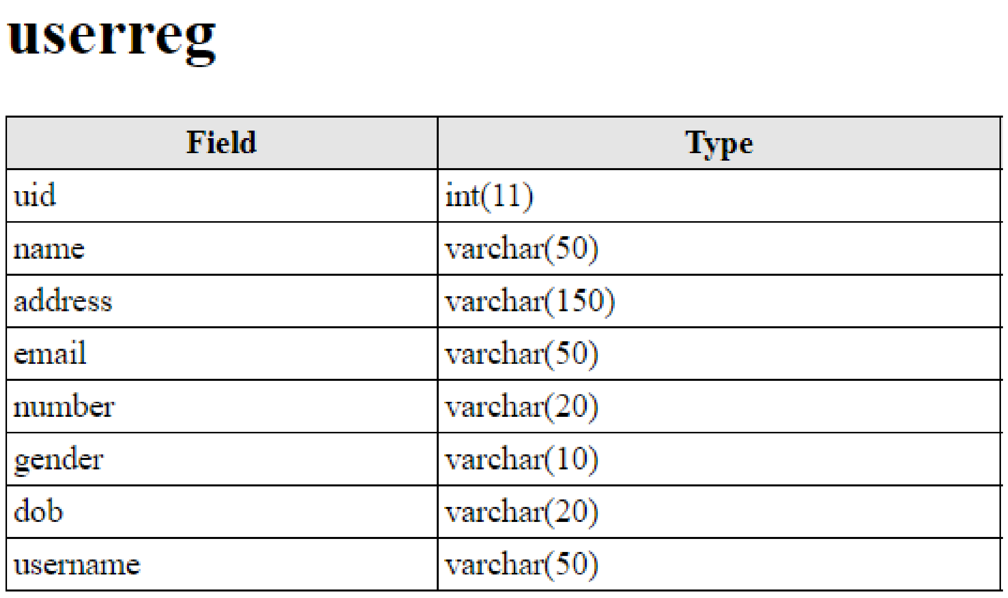
7.1 TABLE DESIGN

The design of the tables in the database is done according to the rules specified for database. In the proposed project, 11 tables are used and some of them are connected using foreign keys. Insertion and retrieval of values are easy by designing the database in this way.



messages

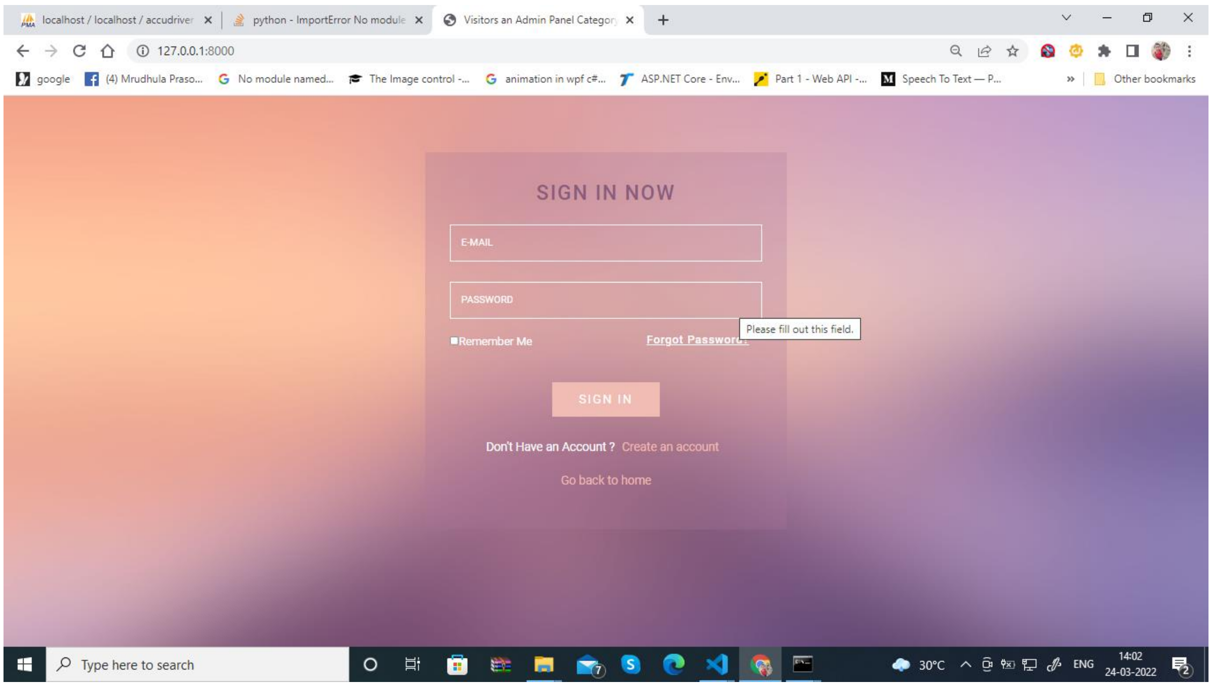
Field	Type
id	int(11)
frommail	varchar(50)
tomail	varchar(50)
sub	varchar(50)
msg	varchar(150)
date	varchar(50)
status	varchar(10)
password	varchar(50)
path	varchar(300)

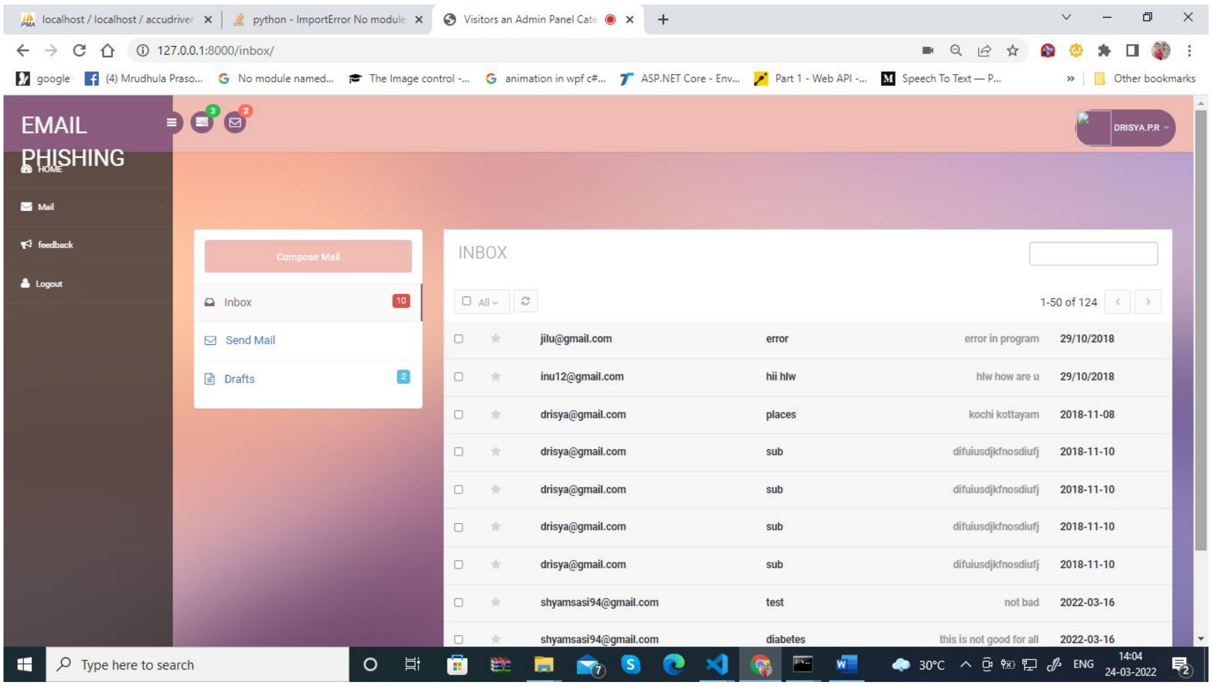


userreg

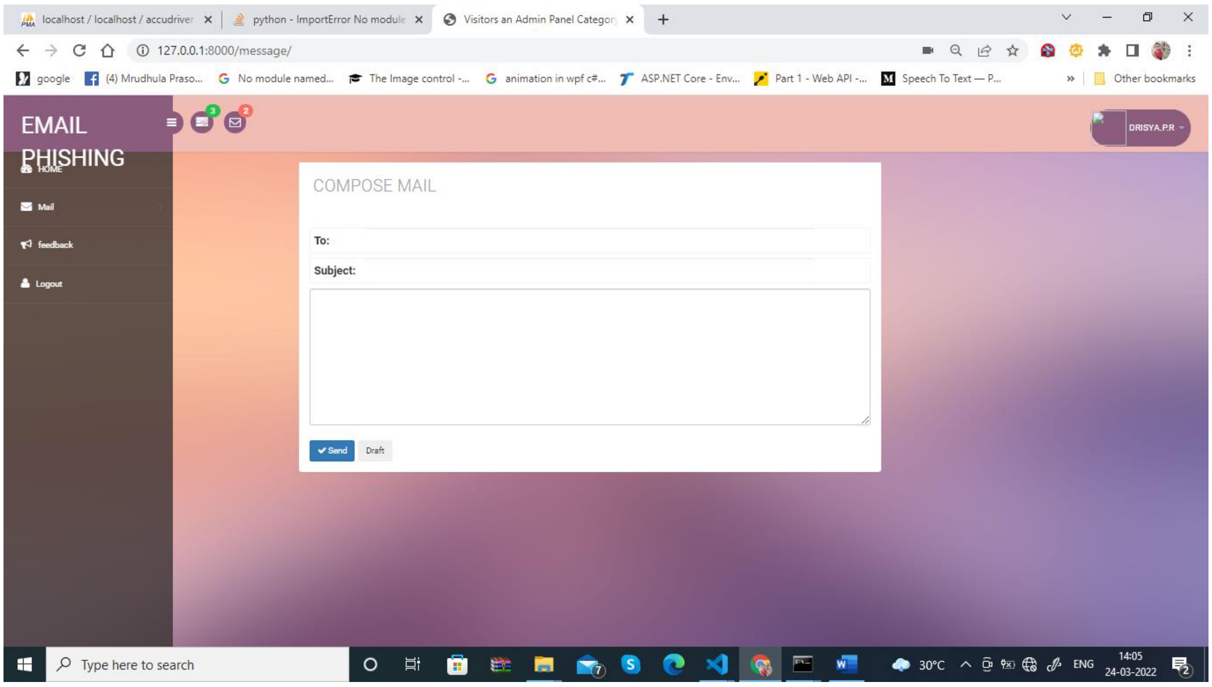
Field	Type
uid	int(11)
name	varchar(50)
address	varchar(150)
email	varchar(50)
number	varchar(20)
gender	varchar(10)
dob	varchar(20)
username	varchar(50)

7.1 SAMPLE INPUT SCREENS





jilu@gmail.com	error	error in program	29/10/2018
inu12@gmail.com	hii hlw	hlw how are u	29/10/2018
drisya@gmail.com	places	kochi kottayam	2018-11-08
drisya@gmail.com	sub	difuiusdjkfnosdiufj	2018-11-10
drisya@gmail.com	sub	difuiusdjkfnosdiufj	2018-11-10
drisya@gmail.com	sub	difuiusdjkfnosdiufj	2018-11-10
drisya@gmail.com	sub	difuiusdjkfnosdiufj	2018-11-10
shyamsasi94@gmail.com	test	not bad	2022-03-16
shyamsasi94@gmail.com	diabetes	this is not good for all	2022-03-16


EMAIL PHISHING
HOME
Mail
feedback
Logout
DRISYA.P.R
COMPOSE MAIL
To:
Subject:
Send
Draft

CONCLUSION

CONCLUSION

We use a new deep learning model named to detect phishing emails. The model employs an improved RCNN to model the email header and the email body at both the character level and the word level. Therefore, the noise is introduced into the model minimally. In the model, we use the attention mechanism in the header and the body, making the model pay more attention to the more valuable information between them. We use the unbalanced dataset closer to the real-world situation to conduct experiments and evaluate the model. The model obtains a promising result. Several experiments are performed to demonstrate the benefits of the proposed model. For future work, we will focus on how to improve our model for detecting phishing emails with no email header and only an email body.

BIBLIOGRAPHY

REFERENCES

1.	Python	- Guido van Rossum
2.	HTML	- Ivan Barros
3.	JavaScript	- Ivan Barros
4.	CSS	- Thomas A. Powell
5.	MySQL	- Paul DuBois

WEBSITES VISITED

www.google.com

www.w3schools.com

www.wikipedia.com