AUTHENTICATION TOOL FOR RANSOMEWARE PREVENTION AND MITIGATION IN ORGANISATION BY USING MFA

A PROJECT REPORT

Submitted by

DILLIKUMAR N	**420420104010**
HARISH N	**420420104016**
KARTHIKEYAN A	**420420104025**
CHANDRASEKARAN S	**420420104301**

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

ADHIPARASAKTHI ENGINEERING COLLEGE MELMARUVATHUR

ANNA UNIVERSITY:: CHENNAI 600 025

MAY 2024

BONAFIDE CERTIFICATE

Certified that this project report **"AUTHENTICATION TOOL FOR RANSOMEWARE PREVENTION AND MITIGATION IN ORGANISATION BY USING MFA"** is the bonafide work of " DILLIKUMAR N **(420420104010)** , HARISH N **(420420104016)**, KARTHIKEYAN A **(420420104025)** , CHANDRASEKARAN S **(420420104301)** " who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. C. DHAYA, Ph.D.,	**Mr. G. SEKAR, M.E.,**
HEAD OF THE DEAPRTMENT	**SUPERVISOR**
Professor	**Assistant Professor**
Department of CSE	Department of CSE
Adhiparasakthi Engineering College	Adhiparasakthi Engineering College
Melmaruvathur 603 319.	Melmaruvathur 603 319.

Submitted for the Project work(CS8811) and Viva-voce examination held on..... at Adhiparasakthi Engineering College Melmaruvathur.. .

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

With the divine blessings of **Goddess Adhiparasakthi**, I express my deep gratitude for the spiritual blessings of His holiness **PADMA SHRI ARULTHIRU AMMA** and the divine guidance of **THIRUMATHI AMMA** thathas undoubtedly taken us to path of victory in completing this project.

The infrastructural support with all kinds of lab facilities have been a motivating factor in this completion of project work, all because of our **Correspondent Sakthi Thiru Dr. G. B. SENTHILKUMAR B.E.** with great pleasure we take this opportunity to thank him.

For the academic side the constant support from our honorable **Principal Dr. J. RAJA, Ph.D.,** as encourages us to work hard and attain this goalof completing the project.

We sincerely thank our **Head of the Department Dr. C. DHAYA, Ph.D.,** who has given us both moral and technical support adding experience to the job we have undertaken.

We thank our **Project Coordinator Mr. K. CHAIRMADURAI, M.E.,** who has led from the front questioning as at the right time and encouraging us most of the timethereby extracting quality work from us.

We sincerely thank our **Supervisor Mr. G. SEKAR, M.E.,** who helped us in crossing in the path to our glory. We also thank other staff members, librarian, and non-teaching staff member of computer center, who have their constant support and motivation in all our endeavors.

ABSTRACT

A novel aspect of this tool is the incorporation of free API services, acting as a security layer interface between raw data suppliers and end stakeholders. These APIs facilitate real-time threat intelligence and security management, reinforcing the tool's preventive and detection capabilities. This proactive approach empowers users to be the first line of defense against cyber threats. The development of this tool is a strategic objective aimed at safeguarding organizations from the disruptive and often devastating effects of ransomware. By integrating advanced detection algorithms, behavioural analytics, and strategic response protocols, the tool represents a new frontier in the fight against cybercrime, ensuring the security and resilience of digital infrastructures in an increasingly vulnerable cyber landscape In the digital battleground against cyber threats, ransomware stands as a formidable adversary. This project introduces a comprehensive tool that employs method for message authentication and data integrity within networks. The tool's architecture is designed to enhance organizational resilience against ransomware through proactive monitoring, behavioural analysis, deception techniques, isolation protocols, incident response, and user education. The integration of these services ensures a secure, user-friendly, and responsive framework, positioning the tool as an essential component in the cybersecurity infrastructure.

TABLE OF CONTENTS

LIST OF SYMBOLS

SYMBOL NAME	NOTATION	DESCRIPTION
Class	Class name Visibility attribute Type=initial value Visibility operation (arglist):returntype	Class represents a collection of similar entities grouped together
Association		Association represents a static relationship between classes.
Use case	Use Case	A use case is an interaction between the system and other external examination.
Relational		It is used for Additional Process Communication
Control flow		It represents the control flow between the state
Data process/State		A circle in DFD represent the vertical dimension the object communication

SYMBOL NAME	NOTATION	DESCRIPTION
Message		It represents the Messageexchanged
Actor		Actors are the user of the system and other external entity that react with the system

LISTS OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS	DESCRIPTION
KBA	**K**nowledge-**B**ased **A**uthentication
CQ	**C**hallenge **Q**uestion
MFA	**M**ulti-**F**actor **A**uthentication
DBMS	**D**atabase **M**anagement **S**ystem
UX	**U**ser **E**xperience
HSM	**H**ardware **S**ecurity **M**odule
HTTPS	**H**ypertext **T**ransfer **P**rotocol **S**ecure
OWASP	**O**pen **W**eb **A**pplication **S**ecurity Project

CHAPTER 1
INTRODUCTION

In the digital age, the value of information stored within computer systems is immeasurable, attracting the attention of various adversaries, including competitors, state-sponsored entities, and cybercriminals. The interconnected nature of these systems amplifies their vulnerability, making them prime targets for cyber attacks. This project is conceived against the backdrop of this digital landscape, where the need for robust cybersecurity measures is more pressing than ever. It aims to address the challenges posed by the increasing sophistication of cyber threats and the need for advanced defense mechanisms to protect critical data assets.

1.1 DOMAIN OVERVIEW

The cybersecurity domain is experiencing a rapid expansion, driven by the escalating frequency and complexity of cyber attacks. This growth is reflected in the increasing demand for sophisticated security tools capable of thwarting attacks and safeguarding enterprise networks. The market's response has been a proliferation of solutions designed to detect and neutralize threats, indicating the critical importance of cybersecurity in the contemporary business environment. This section delves into the dynamics of the cybersecurity market, highlighting the challenges faced by organizations and the innovative approaches being adopted to secure digital assets.

1.2 OVERVIEW OF THE PROJECT

This project proposes the development of an innovative interface that acts as a sentinel between data-intensive companies and their stakeholders. It is designed to preemptively defend against a spectrum of cyber threats, including malware, ransomware, and unauthorized data breaches.

The interface will serve as a crucial layer of protection, ensuring that the vast amounts of data generated every second are not only secure but also utilized responsibly and ethically by authorized users. The overview details the project's approach to bridging the gap between data security and accessibility, emphasizing the importance of maintaining the integrity of data flows in a highly connected world.

At the heart of the tool is a sophisticated monitoring system that continuously scans for signs of ransomware. This includes tracking file system changes, network traffic anomalies, and unusual encryption activity, which are often precursors to an attack. By employing behavior-based detection techniques, the tool can identify potential ransomware behavior patterns. This method goes beyond traditional signature-based detection, allowing for the identification of zero-day ransomware threats that have not yet been cataloged. The tool will use deception as a means of defense, creating honeypots and decoy files to lure and trap ransomware. This not only helps in identifying the attack vectors but also aids in understanding the adversary's tactics. Upon detection of a potential ransomware threat, the tool will initiate isolation protocols to contain the infection.

This includes severing network connections, disabling Wi-Fi access, and isolating affected systems to prevent lateral spread within the network. This includes instructions for system shutdowns, network disconnections, and procedures for cleaning and restoring infected systems. An integral component of the tool is a user education module that informs stakeholders about ransomware risks and best practices for prevention.

1.3 OBJECTIVE OF THE PROJECT

The project's objective is to create a state-of-the-art tool that specializes in the detection and prevention of ransomware attacks within organizational networks. By integrating signature-based, behavior-based, and deception-based detection methodologies, the tool will provide comprehensive monitoring of network traffic, file system changes, and system events. This vigilant surveillance aims to promptly identify any signs of ransomware activity, enabling organizations to swiftly respond to and neutralize threats. The objective section outlines the anticipated outcomes of the project, including enhanced network security, reduced risk of data theft, and strengthened resilience against cyber attacks.

CHAPTER 2
LITERATURE SURVEY

A Literature review is a text of a scholarly paper, which includes the current knowledge, including substantive findings, as well as theoretical and methodological contributions to a particular topic. Literature reviews use secondary sources and do not report new or original experimental work. A literature review usually precedes the methodology and results section.

2.1 REVIEW OF LITERATURE

1. **Title: "**Online Banking User Authentication Methods"

   Author: N. A. Karim

   Year: 2024

   Problem Description:

Online banking has become increasingly popular in recent years, making it a target for cyberattacks. Banks have implemented various user authentication methods to protect their customers' online accounts. This paper reviews the state-of-the-art user authentication methods used in online banking and potential cyber threats. This paper starts by exploring different user authentication methods, such as knowledge-based authentication (KBA), biometrics-based authentication (BBA), possession-based authentication (PBA), and other methods. The advantages and disadvantages of each user authentication method are then discussed. Furthermore, the paper discusses the various cyber threats that can compromise user authentication for online banking systems, such as malware attacks, social engineering, phishing attacks, man-in-the-middle (MiTM) attacks, denial of service (DoS) attacks, session hijacking, weak passwords, keyloggers, SQL injection, and replay attacks. Also, the paper explores the user authentication methods used by popular banks, which can provide insights into best practices for

safeguarding online banking accounts and future user authentication methods in online banking and cyber threats. It states that the increasing use of BBA, two-factor authentication (2FA), and multi-factor authentication (MFA) will help improve the security of online banking systems. However, the paper also warns that new cyber challenges will emerge, and banks need to be vigilant in protecting their customers' online banking accounts.

2. **Title: "**Fuzzy and Blockchain-Based Adaptive Security for Healthcare IoTs"

   Author: Z. Zulkifl

   Year: 2022

   Problem Description:

Internet of Things (IoT) is a system of interconnected devices that have the ability to monitor and transfer data to peers without human intervention. Authentication, Authorization and Audit Logs (AAA) are prime features of Network Security and easily attained in legacy systems, however, remains unachieved in IoT. The IoTs require due security considerations as the conventional security mechanisms are not optimized for such devices due to various aspects such as heterogeneity, resource constrained processing, storage and multiple factors. Additionally, the legacy systems are mostly centralized and thus introduce a single point of failure. In this research, a novel framework, FBASHI is presented that is based on fuzzy logic and blockchain technology to achieve AAA services. The proposed system is developed using Hyperledger that is a blockchain platform providing privacy and fast response capability, therefore, it is best suited for the healthcare IoT environments. This work proposes behavior driven adaptive security mechanism for healthcare IoTs and networks based on blockchain by utilizing fuzzy logic and presents a heuristic approach towards behavior driven adaptive security providing AAA services. FBASHI is implemented to analyze its security and practicality. Furthermore, a comparison is drawn with other blockchain-based solutions.

3. **Title:** "Cyber Threat Detection Based on ANN Using Event Profiles".

   Author: J. Lee

   Year: 2019

   Problem Description:

One of the major challenges in cybersecurity is the provision of an automated and effective cyber-threats detection technique. In this paper, we present an AI technique for cyber-threats detection, based on artificial neural networks. The proposed technique converts multitude of collected security events to individual event profiles and use a deep learning-based detection method for enhanced cyber-threat detection. For this work, we developed an AI-SIEM system based on a combination of event profiling for data preprocessing and different artificial neural network methods, including FCNN, CNN, and LSTM. The system focuses on discriminating between true positive and false positive alerts, thus helping security analysts to rapidly respond to cyber threats. All experiments in this study are performed by authors using two benchmark datasets (NSLKDD and CICIDS2017) and two datasets collected in the real world. To evaluate the performance comparison with existing methods, we conducted experiments using the five conventional machine-learning methods (SVM, k-NN, RF, NB, and DT). Consequently, the experimental results of this study ensure that our proposed methods are capable of being employed as learning-based models for network intrusion-detection, and show that although it is employed in the real world, the performance outperforms the conventional machine-learning methods.

4. **Title: "**Offensive Security: Proactive Threat Hunting Adversary Emulation"

Author: A. B. Ajmal

Year: 2021

Problem Description:

Attackers increasingly seek to compromise organizations and their critical data with advanced stealthy methods, often utilising legitimate tools. In the main, organisations employ reactive approaches for cyber security, focused on rectifying immediate incidents and preventing repeat attacks, through protections such as vulnerability assessment and penetration testing (VAPT) security information and event management (SIEM), firewalls, anti-spam/anti-malware solutions and system patches. Such system have weaknesses in addressing modern modern stealthy attacks. Proactive approaches, have been seen as part of the solution to this problem. However, approaches such as VAPT have limited scope and only works with threats that have already been discovered. Promising methods such as threat hunting are gaining momentum, enabling organisations to identify and rapidly respond to any potential attacks, though they have been criticised for their significant cost. In this paper, we present a novel hybrid model for uncovering tactics, techniques, and procedures (TTPs) through offensive security, specifically threat hunting via adversary emulation. The proposed technique is based on a novel approach of inducing adversary emulation (mapping each respective phase) model inside the threat hunting approach. The experimental results show that the proposed approach uses threat hunting via adversary emulation and has countervailing effects on hunting advance level threats. Moreover, the threat detection ability of the proposed approach utilizes minimum resources. The proposed approach can be used to develop the offensive security-aware environment for organizations to uncover advanced attack mechanisms and test their ability for attack detection.

5. **Title:** "Know your customer (KYC) based authentication method for financial services through the internet,"

Author: P. C. Mondal, R. Deb and M. N. Huda,

Year: 2016.

Problem Description:

In this research paper, Know Your Customer (KYC) information verification technique has been introduced as Challenge Question (CQ) during login using user ID and Password in order to verify user more intensively. In that case KYC must be privatized with widespread dynamic user input. The KYC database enriches from account opening initial data, user interaction and dynamic update through the application; on the other hand user can add more confidential information or random question/questions with answer/answers to the KYC database to make the authentication process much stronger and secured. Ranking on the KYC information will also be considered to be used as CQ; CQ will be asked to the user during login after success in user ID and Password verification. One or more CQ will be assigned to ask the user based on the risk factors assessment result. Top ranked CQ will be asked to the user when the risk assessment result is comparatively higher; on the other hand low ranked CQ will be asked for lower risk.

2.2 PROBLEM STATEMENT

How can we develop a tool that evaluates an organization's preparedness for ransomware threats? This tool would focus on proactive monitoring, behavioral analysis, deception techniques, isolation protocols, incident response, and user education to bolster defenses against cyber attacks and how the tool will assess an organization's ransomware defense, focusing on early detection, user training,

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

There are several existing statements and approaches to implementing Multifactor authentications in login site for the admin/user pievileges. Some examples include in this existing system:

Knowledge Based Authentication (KBA):

This is a method of authentication that requires the knowledge of private information from the individual to prove their identity. It's commonly used by financial institutions or websites. KBA can be static, based on pre-agreed shared secrets, or dynamic, based on questions generated from a wider base of personal information.

The **CQ(Challenge Question)** based authentication applies immediately after login and before performing and committing a transaction to verify the user rigorously. This CQ replaces the 2FA or traditional question and answers mechanism from some other existing authentication models. The brief idea of the model is the initial step is login of the user with user ID and Password verification like other online applications. The forwarding stage is risk analysis for the login succeeded user. Next stage assign one or more CQ based on risk level formed by the result of prior stage. Final stage of the verification is OTP / EMAIL / OTP & Email confirmation if it is indicated by the result of risk analysis. In some other cases confirmation stage may not be applicable where CQ is in final stage before performing and committing a transaction.

Hidden Markov Model in Machine learning:

A statistical model called a Hidden Markov Model (HMM) is used to describe systems with changing unobservable states over time. It is predicated on the idea that there is an underlying process with concealed states, each of which has a known result. Probabilities for switching between concealed states and emitting observable symbols are defined by the model. Because of their superior ability to capture uncertainty and temporal dependencies, HMMs are used in a wide range of industries, including finance, bioinformatics, and speech recognition. HMMs are useful for modelling dynamic systems and forecasting future states based on sequences that have been seen because of their flexibility.

Viterbi algorithm:

The Viterbi algorithm is used to calculate the most likely sequence of hidden states that generated the observations using the decode method of the model. The method returns the log probability of the most likely sequence of hidden states and the sequence of hidden states itself

Limitations:

HMMs are primarily used for modeling and prediction tasks, not for security purposes. They are not designed to provide cryptographic strength or protect against attacks.

3.2 PROPOSED SYSTEM

" To make an authentication by using **Subjective Based Questions** to the User in Order to Get reasource from site or Online Access as finanitial resources. That Subject Based Questions performes in after the login into site by providing credentials and before action of resource getting from the site. That Question and answer only known and felt by the user, Once user registered the Question and Answer the the Questions Raised whenever the user login activity successfully completed ... the Question will be as in the place Holding for entering user's subjective answer" .

Subjective questions are questions that require answers in the form of explanations. Unlike objective questions that have clear-cut answers, subjective questions involve personal feelings, opinions, and interpretations. Here are some common types of subjective questions:

Essay Questions: These prompts ask you to provide a detailed response, often requiring you to analyze, evaluate, or discuss a topic. Essay questions allow you to express your understanding and critical thinking skills.

Short Answer Questions: Similar to essay questions but more concise, short answer questions still require you to explain your reasoning. You might need to provide examples or evidence to support your answer.

Definitions: Subjective definitions go beyond dictionary-style explanations. They ask to elaborate on the meaning of a term, concept, or idea. Our response should demonstrate a deep understanding.

Scenario Questions: These present a hypothetical situation or case study. We'll need to apply our knowledge to analyze the scenario, make decisions, and justify your choices.

Opinion Questions: Opinion questions ask for your personal viewpoint. They're common in surveys, polls, and discussions. For example, "What's your favorite book?" or "How do you feel about climate change?"

3.2.1 ADVANTAGES

In subjective-based questions offer improved security, customization, and engagement compared to existing methods. They empower users to authenticate based on their personal experiences, making the authentication process more robust and user-friendly, the advantages are.,

1.Enhanced Security:

- Subjective-based questions provide an additional layer of security beyond traditional methods like passwords or knowledge-based authentication (KBA).
- Since the answers are unique to each user and not easily guessable, it becomes harder for unauthorized individuals to gain access.

2. Reduced Vulnerability to Social Engineering:

- KBA often relies on personal information that can be obtained through social engineering or data breaches.
- Subjective questions, on the other hand, require answers that are not publicly available, making them less susceptible to social engineering attacks.

3. Customizable and Contextual:

- Subjective questions can be tailored to the user's preferences or context.
- For example, a user might choose questions related to their favorite book, childhood memory, or personal experiences.

4. User Engagement and Satisfaction:

- Subjective questions add an element of personalization to the authentication process.
- Users may find it more engaging and satisfying to answer questions that resonate with their own experiences.

5. Less Dependency on Shared Secrets:

- KBA often relies on shared secrets (e.g., mother's maiden name) that can be compromised.
- Subjective questions do not rely on pre-agreed secrets, reducing the risk associated with shared information.

6. Adaptive Risk Assessment:

- The proposed system's risk analysis assigns questions based on the user's risk level.
- If a user has a higher risk profile (e.g., suspicious login behavior), more challenging questions can be posed.
- This adaptive approach enhances security while minimizing inconvenience for low-risk users.

3.3 LIST OF ALGORITHMS

3.3.1 HMAC (Hash-based Message Authentication Code)

It is a type of a message authentication code (MAC) that is acquired by executing a cryptographic hash function on the data (that is) to be authenticated and a secret shared key. Like any of the MAC, it is used for both data integrity and authentication. Checking data integrity is necessary for the parties involved in communication.

HTTPS, SFTP, FTPS, and other transfer protocols use HMAC. The cryptographic hash function may be MD-5, SHA-1, or SHA-256. Digital signatures are nearly similar to HMACs i.e they both employ a hash function and a shared key. The difference lies in the keys i.e HMACs use symmetric key(same copy) while Signatures use asymmetric (two different keys). HMAC stands for Hashed or Hash-based Message Authentication Code.

It is a result of work done on developing a MAC (Message Authentication Code) derived from cryptographic hash functions. HMAC is designed to provide resistance against cryptanalysis attacks by using the hashing concept twice. It combines the benefits of both hashing and MAC, making it more secure than other authentication codes. RFC 2104 has standardized HMAC, and it is compulsory to implement in IP security. The NIST standard (FIPS 198) also includes HMAC.

How HMAC Works:

HMAC starts with a message M containing blocks of length b bits. An input signature is padded to the left of the message, and the entire input is given to a hash function, resulting in a temporary message digest MD'.MD' is then appended to an output signature, and the entire input is processed by the hash function again, resulting in the final message digest MD.

The structure of HMAC is as follows:

$$H(M) = H((K \oplus opad) \,||\, H((K \oplus ipad) \,||\, M))$$

Here,

H stands for the hashing function (e.g., MD5, SHA-1, SHA-256, etc.).

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 REQUIREMENT ANALYSIS

4.1.1 FUNCTIONAL REQUIREMENTS

Functional requirements describe what the system should do. Here are some functional requirements for the proposed system:

1. User Authentication:

The system must allow users to log in using their credentials (e.g., username and password). After successful login, the system should prompt users with subjective-based questions.

2. Subjective Question Management:

The system needs a mechanism to manage subjective questions. Users can able to set their own questions during registration. Administrators can configure question pools and assign them to users based on risk levels.

3. Risk Assessment and Question Assignment:

The system analyze user behavior and assign risk levels (low, medium, high).Based on the risk level, appropriate subjective questions should be posed during login.

4. Frequent BackUps:

To Use isolated Server from the Internet for Storing the Data Retrive if when after the Ransomeware attack Happen we can use that date to avoid lot money and confidential data being destroyed.

4.1.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements focus on system qualities such as performance, security, and usability. Here are some non-functional requirements:

1. Security:

The system must ensure the confidentiality of subjective answers. Strong encryption should protect user data during transmission and storage.

2. Usability:

The user interface should be intuitive and user-friendly. Subjective questions should be clear and easy to understand.

3. Performance:

The system should respond quickly during login and question presentation. Scalability is essential to handle a large number of users.

4.1.3 HARDWARE REQUIREMENTS

The proposed system requires suitable hardware resources. These should include:

1. Server Infrastructure:

Sufficient processing power, memory, and storage to handle user requests. Load balancers for distributing traffic.

2. Database Server:

A database server to store user profiles, questions, and risk assessment data. Adequate storage capacity and backup mechanisms.

3. Network Infrastructure:

Reliable network connectivity to serve users across different locations.

4.1.4 SOFTWARE REQUIREMENTS

The software components needed for the proposed system include:

1. Web Application Framework:

- Choose a web framework (Flask - python 3) for building the application. Integrated Development Environment Application(Pycharm) we used in this project)

2. Database Management System (DBMS):

Select a DBMS (SQLite) for storing user data and question details.

3. Authentication Libraries:

We Use authentication libraries or modules to handle user login and session management is CSRF.

4. Encryption Tools:

Implement encryption algorithms (HMAC) for securing data.

5. Web Server:

To Deploy a web server (e.g., Apache, Nginx) to serve the application.

CHAPTER 5

HARDWARE AND SOFTWARE ANALYSIS

5.1. HARDWARE ANALYSIS:

When designing the hardware infrastructure for the proposed authentication system, several critical considerations we faced :

5.1.1. Server Infrastructure:

- Evaluate the server requirements based on expected user load and scalability.
- Consider factors such as processing power, memory, storage capacity, and network bandwidth.
- Choose reliable servers or cloud-based solutions to handle concurrent user requests.

5.1.2. Database Server:

- Select an appropriate database management system (DBMS) for storing user profiles, question data, and risk assessment information.
- Optimize the database schema design to ensure efficient data retrieval and storage.
- Implement backup and disaster recovery mechanisms.

5.1.3. Network Infrastructure:

- Assess network connectivity to ensure seamless communication between components.
- Implement secure protocols (e.g., HTTPS) for data transmission.
- Consider load balancers for distributing traffic and ensuring high availability.

5.1.4. Scalability and Redundancy:

- Plan for system growth by assessing scalability options.
- Consider redundancy (e.g., failover servers, load balancing) to ensure uninterrupted service.

5.1.5. Budget Constraints:

- Balance hardware capabilities with budget limitations.
- Prioritize critical components while ensuring cost-effectiveness.

5.2. SOFTWARE ANALYSIS:

The software components play a crucial role in system functionality and security:

5.2.1. Authentication Software and IDEA (Pycharm community edition):

- Develop authentication modules that handle user login, session management, and subjective question validation for we need IDEA.
- we Ensures robust encryption for sensitive data (e.g., user answers) by hashing algorithm HMAC-sha-256.
- We used the Pycharm for development of whole components in a single place by using lot of dependencies and plugins are easy to handles.

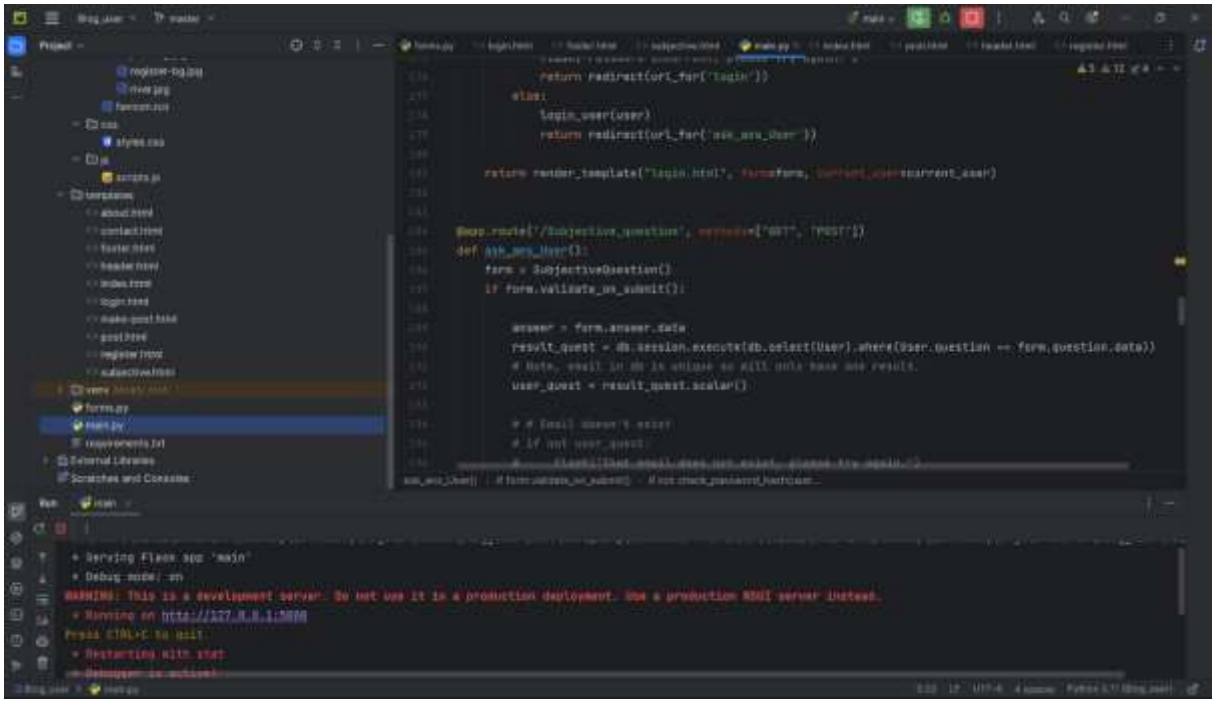

Fig.5.1 IDEA for integration and development

5.2.2. Database Management System (DBMS):

- We used the DBMS(SQLite) that aligns with the hardware infrastructure.
- Optimize database queries and indexing for efficient data retrieval.

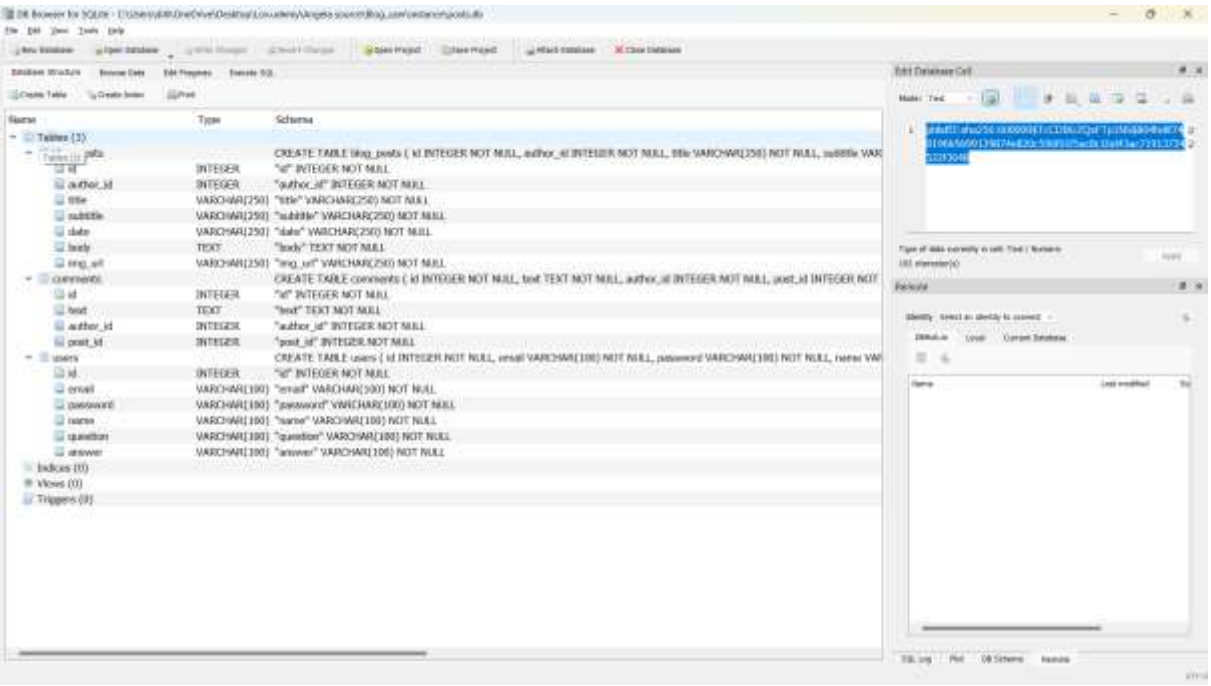

Fig.5.2 SQLite for Database Management.

5.2.3. Web Application Framework:

- Decide on a suitable web framework (Flask) for building the user interface.
- Implement user registration, login, and question management features.

5.2.4. Security Libraries and Practices:

- Integrate security librarie (OWASP as csrf) to prevent common vulnerabilities (e.g., SQL injection, cross-site scripting).
- Follow best practices for secure coding and session management.

5.2.5. User Experience (UX) Design:

- Design an intuitive and user-friendly interface for users to set up their subjective questions. Implement unit testing, integration testing, and security testing. We Ensured smooth navigation and clear instructions.

CHAPTER 6

DESIGN ENGINEERING

6.1 ARCHITECTURE DIAGRAM

The system Architecture ensures secure login and registration, incorporating multi-factor authentication (MFA) for enhanced security. Users gain regular access to isolated server features, while strict adherence to specific protocols maintains data integrity. Frequent backups and maintenance further enhance reliability which are mojar important in this system.

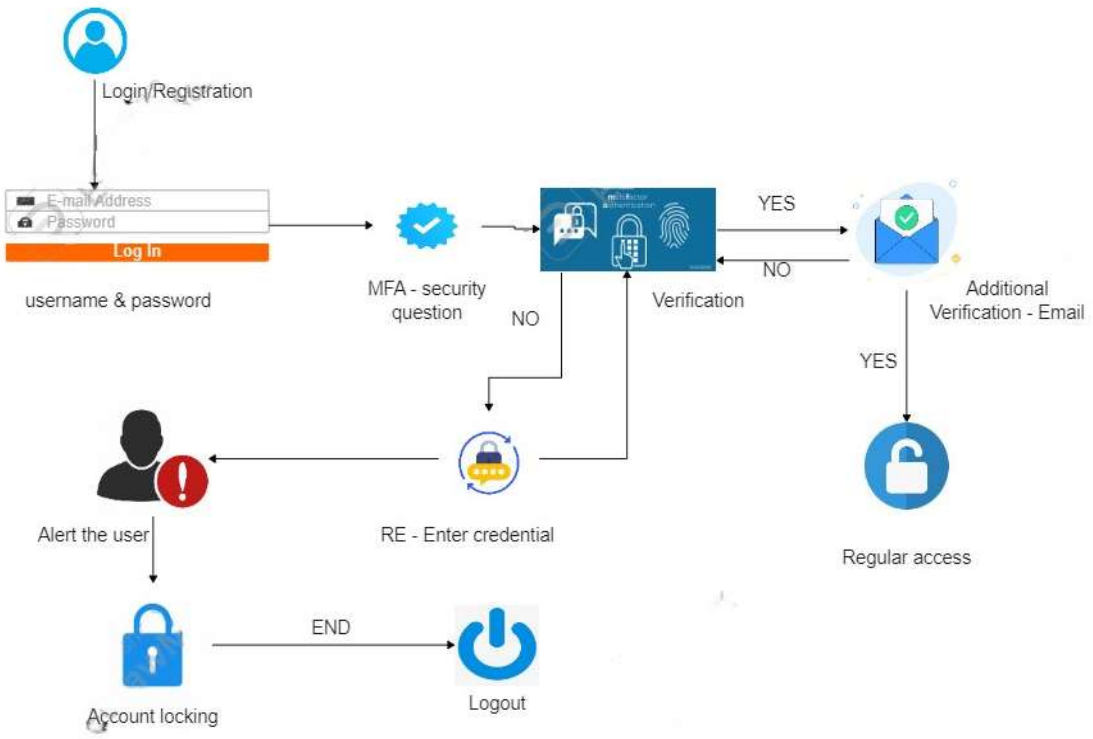

Figure 6.1 System Architecture Diagram for Ransomware prevention tool.

6.2 FLOW DIAGRAM

The process begins at the “LOGIN/REGISTRATION” phase, where users are prompted to enter their “Username and Password.” Upon successful entry, they gain “REGULAR ACCESS” to the system’s features and can choose to “LOGOUT” when their session is complete. If additional verification is required, the user is presented with an “MFA - Security Question.” A correct response (YES) leads to “Additional Verification - Email,” where an email is sent to the user for further confirmation. If the answer is incorrect (NO), the user must “RE-ENTER CREDENTIALS.” After three unsuccessful attempts, an “ALERT THE USER” notification is triggered, which may result in “Account Locking” to protect the user’s security.

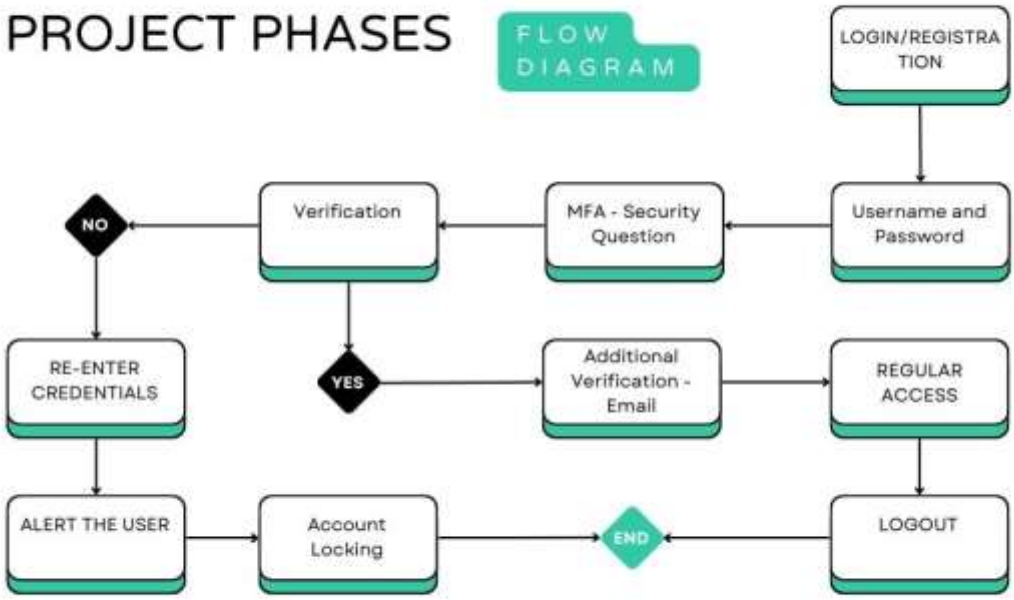

Figure 6.2 Flow Diagram for Ransomware prevention tool.

6.3 USE CASE

The process begins with a secure login by the client, ensuring that access is safeguarded through robust authentication mechanisms. This is followed by factorized authentication, which adds an additional layer of security to verify the client's identity. Once authenticated, the client can engage in regular access to the server, which is designed to be isolated to prevent unauthorized entry and enhance overall security. The server undergoes frequent backup and maintenance to maintain data integrity and ensure that all information is up-to-date and available when needed. Throughout this process, there is a strict adherence to specific protocols or standards, guaranteeing that data handling is consistent, reliable, and conforms to the highest security standards.

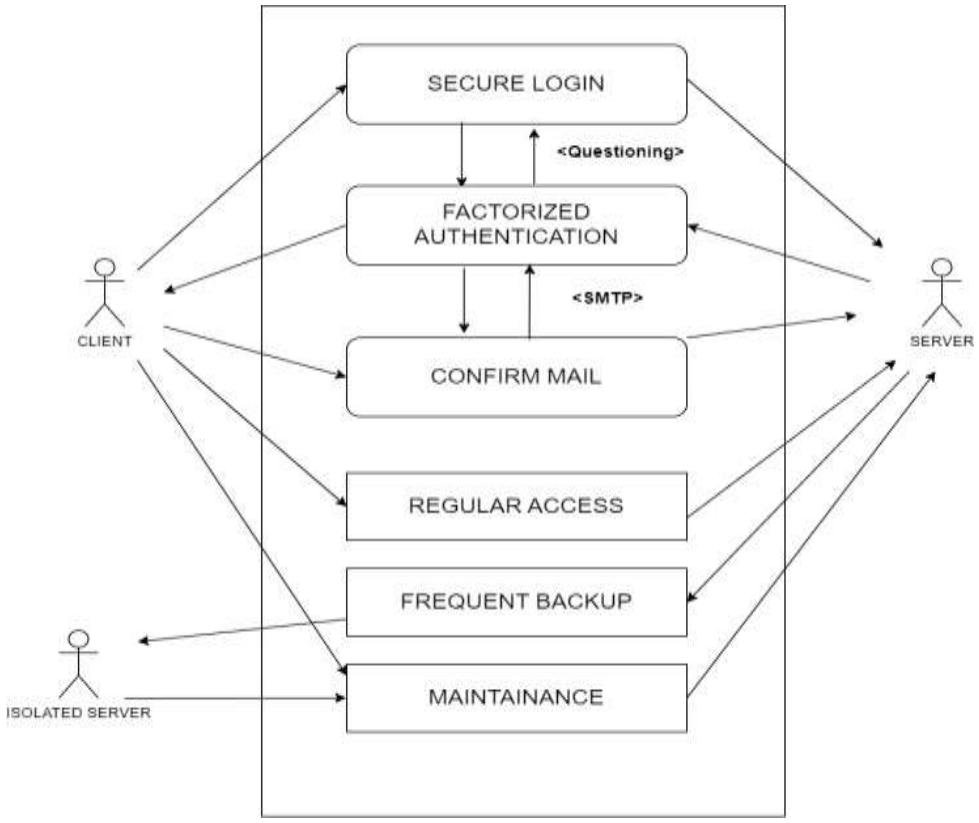

Figure 6.3 Use Case Diagram for Ransomware prevention tool.

CHAPTER 7

IMPLEMENTATION OF MODULES

We are used several modules in Our prject modules structure will be look like,

/Blog_user

/app

/venv

/main.py

/forms.py

/templates

/header.html, /footer.html,/ register.html,/ subjective.html,/ makepost.html, / about.html,/ contact.html,/index.html,/base.html

/static

/css

/main.css

/js

/app.js

/images

#images used in this project#

#idea, instances and other stuffs#

While considering project there are core modules involving action and behaviour of the projects are,

Templates: This folder contains the HTML files that define the structure and layout of your web pages. These templates are used to render the dynamic content served by your web application.

Static: This directory holds all the static files of your project, such as CSS for styling, JavaScript for interactivity, and images. These files don't change often and are sent to the user's browser as-is.

Python Files: These are the actual script files written in Python that make up the backend of your project. They include views, models, forms, and other utility scripts that handle the logic and database interactions of your application.

```
##################### main.py ####################################

from datetime import date
from flask import Flask, abort, render_template, redirect, url_for, flash
from flask_bootstrap import Bootstrap5
from flask_ckeditor import CKEditor
from flask_gravatar import Gravatar
from flask_login import UserMixin, login_user, LoginManager, current_user,
logout_user
from flask_sqlalchemy import SQLAlchemy
from sqlalchemy.orm import relationship, DeclarativeBase, Mapped,
mapped_column
from sqlalchemy import Integer, String, Text
from functools import wraps
from werkzeug.security import generate_password_hash, check_password_hash
# Import your forms from the forms.py
from forms import CreatePostForm, RegisterForm, LoginForm, CommentForm,
SubjectiveQuestion
app = Flask(__name__)
app.config['SECRET_KEY'] = '8BYkEfBA6O6donzWlSihBXox7C0sKR6b'
ckeditor = CKEditor(app)
Bootstrap5(app)
```

```
#  Configure Flask-Login
login_manager = LoginManager()
login_manager.init_app(app)

@login_manager.user_loader
def load_user(user_id):
    return db.get_or_404(User, user_id)

# For adding profile images to the comment section
gravatar = Gravatar(app,
                    size=100,
                    rating='g',
                    default='retro',
                    force_default=False,
                    force_lower=False,
                    use_ssl=False,
                    base_url=None)

# CREATE DATABASE
class Base(DeclarativeBase):
    pass
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///posts.db'
db = SQLAlchemy(model_class=Base)
db.init_app(app)

# CONFIGURE TABLES
class BlogPost(db.Model):
    __tablename__ = "blog_posts"
    id: Mapped[int] = mapped_column(Integer, primary_key=True)
```

```
    author_id: Mapped[int] = mapped_column(Integer,
db.ForeignKey("users.id"))
    author = relationship("User", back_populates="posts")
    title: Mapped[str] = mapped_column(String(250), unique=True,
nullable=False)
    subtitle: Mapped[str] = mapped_column(String(250), nullable=False)
    date: Mapped[str] = mapped_column(String(250), nullable=False)
    body: Mapped[str] = mapped_column(Text, nullable=False)
    img_url: Mapped[str] = mapped_column(String(250), nullable=False)

    # Parent relationship to the comments
    comments = relationship("Comment", back_populates="parent_post")

class User(UserMixin, db.Model):
    __tablename__ = "users"
    id: Mapped[int] = mapped_column(Integer, primary_key=True)
    email: Mapped[str] = mapped_column(String(100), unique=True)
    password: Mapped[str] = mapped_column(String(100))
    name: Mapped[str] = mapped_column(String(100))
    question: Mapped[str] = mapped_column(String(100), unique=True)
    answer: Mapped[str] = mapped_column(String(100))
    posts = relationship("BlogPost", back_populates="author")
    comments = relationship("Comment", back_populates="comment_author")

class Comment(db.Model):
    __tablename__ = "comments"
    id: Mapped[int] = mapped_column(Integer, primary_key=True)
    text: Mapped[str] = mapped_column(Text, nullable=False)
    author_id: Mapped[int] = mapped_column(Integer,
db.ForeignKey("users.id"))
```

```
    comment_author = relationship("User", back_populates="comments")


    # Child Relationship to the BlogPosts
    post_id: Mapped[int] = mapped_column(Integer,
db.ForeignKey("blog_posts.id"))
    parent_post = relationship("BlogPost", back_populates="comments")


with app.app_context():
    db.create_all()


# Create admin-only decorator
def admin_only(f):
    @wraps(f)
    def decorator_function(*args,**kwargs):
        # If id is not 1 then return abort with 403 error
        if (current_user.is_authenticated and current_user.id != 1) or (not
current_user.is_authenticated):
            return abort(403)
        # Otherwise continue with the route function
        return f(*args, **kwargs)
    return decorator_function


# Register new users into the User database
@app.route('/register', methods=["GET", "POST"])
def register():
    form = RegisterForm()
    if form.validate_on_submit():


        # Check if user email is already present in the database.
```

```
    result = db.session.execute(db.select(User).where(User.email ==
form.email.data))
    user = result.scalar()

    if user:
        # User already exists
        flash("You've already signed up with that email, log in instead!")
        return redirect(url_for('login'))

    hash_and_salted_password = generate_password_hash(
        form.password.data,
        method='pbkdf2:sha256',
        salt_length=16
    )
    # Check if user Question is already present in the database.
    result_quest = db.session.execute(db.select(User).where(User.question ==
form.question.data))
    user_quest = result_quest.scalar()

    if user_quest:
        # User already exists
        flash("You've already signed up with that question, log in instead!")
        return redirect(url_for('login'))

    hash_and_salted_answer = generate_password_hash(
        form.answer.data,
        method='pbkdf2:sha256',
        salt_length=16
    )
    new_user = User(
```

```
            email=form.email.data,
            name=form.name.data,
            password=hash_and_salted_password,
            question=form.question.data,
            answer=hash_and_salted_answer,
        )

        db.session.add(new_user)
        db.session.commit()
        # This line will authenticate the user with Flask-Login
        login_user(new_user)
        return redirect(url_for("get_all_posts"))
    return render_template("register.html", form=form,
current_user=current_user)

#  Retrieve a user from the database based on their email.

@app.route('/login', methods=["GET", "POST"])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        password = form.password.data
        result = db.session.execute(db.select(User).where(User.email ==
form.email.data))

        # Note, email in db is unique so will only have one result.
        user = result.scalar()
        # Email doesn't exist
        if not user:
            flash("That email does not exist, please try again.")
```

```
            return redirect(url_for('login'))
        # Password incorrect
        elif not check_password_hash(user.password, password):
            flash('Password incorrect, please try again.')
            return redirect(url_for('login'))
        else:
            login_user(user)
            return redirect(url_for('ask_ans_User'))

    return render_template("login.html", form=form, current_user=current_user)

@app.route('/Subjective_question', methods=["GET", "POST"])
def ask_ans_User():
    form = SubjectiveQuestion()
    if form.validate_on_submit():
        answer = form.answer.data
        result_quest = db.session.execute(db.select(User).where(User.question ==
form.question.data))
        user_quest = result_quest.scalar()
user_mail = result_mail.scalar()
        if not user_quest or not check_password_hash(user_quest.answer, answer):
            Mail.connection.sendmail(from_addr="traialmakinginfo@gmail.com",
to_addrs=user_mail, msg="some one who trying to access your blog site
account!")
            Mail.connection.quit()        if not user_quest:
            flash(f"Please Answer Your Subjective Question.")
            return redirect(url_for('login'))
        # Password incorrect
        if not check_password_hash(user_quest.answer, answer):
            flash('Subjective Answer incorrect, please try again.')
```

```
            return redirect(url_for('login'), )
        else:
            login_user(user_quest)
            return redirect(url_for('get_all_posts'))

    return render_template("subjective.html", form=form,
current_user=current_user)



@app.route('/logout')
def logout():
    logout_user()
    return redirect(url_for('get_all_posts'))

@app.route('/')
def get_all_posts():
    result = db.session.execute(db.select(BlogPost))
    posts = result.scalars().all()
    return render_template("index.html", all_posts=posts,
current_user=current_user)



# Add a POST method to be able to post comments
@app.route("/post/<int:post_id>", methods=["GET", "POST"])
def show_post(post_id):
    requested_post = db.get_or_404(BlogPost, post_id)
    # Add the CommentForm to the route
    comment_form = CommentForm()
    # Only allow logged-in users to comment on posts
    if comment_form.validate_on_submit():
```

```
        if not current_user.is_authenticated:
            flash("You need to login or register to comment.")
            return redirect(url_for("login"))

        new_comment = Comment(
            text=comment_form.comment_text.data,
            comment_author=current_user,
            parent_post=requested_post
        )
        db.session.add(new_comment)
        db.session.commit()
    return render_template("post.html", post=requested_post,
current_user=current_user, form=comment_form)

# Use a decorator so only an admin user can create new posts
@app.route("/new-post", methods=["GET", "POST"])
@admin_only
def add_new_post():
    form = CreatePostForm()
    if form.validate_on_submit():
        new_post = BlogPost(
            title=form.title.data,
            subtitle=form.subtitle.data,
            body=form.body.data,
            img_url=form.img_url.data,
            author=current_user,
            date=date.today().strftime("%B %d, %Y")
        )
        db.session.add(new_post)
        db.session.commit()
```

```
        return redirect(url_for("get_all_posts"))
    return render_template("make-post.html", form=form,
current_user=current_user)


# Use a decorator so only an admin user can edit a post
@app.route("/edit-post/<int:post_id>", methods=["GET", "POST"])
def edit_post(post_id):
    post = db.get_or_404(BlogPost, post_id)
    edit_form = CreatePostForm(
        title=post.title,
        subtitle=post.subtitle,
        img_url=post.img_url,
        author=post.author,
        body=post.body
    )
    if edit_form.validate_on_submit():
        post.title = edit_form.title.data
        post.subtitle = edit_form.subtitle.data
        post.img_url = edit_form.img_url.data
        post.author = current_user
        post.body = edit_form.body.data
        db.session.commit()
        return redirect(url_for("show_post", post_id=post.id))
    return render_template("make-post.html", form=edit_form, is_edit=True,
current_user=current_user)


@app.route("/delete/<int:post_id>")
@admin_only
def delete_post(post_id):
    post_to_delete = db.get_or_404(BlogPost, post_id)
```

```
    db.session.delete(post_to_delete)
    db.session.commit()
    return redirect(url_for('get_all_posts'))

@app.route("/about")
def about():
    return render_template("about.html", current_user=current_user)
@app.route("/contact")
def contact():
    return render_template("contact.html", current_user=current_user)

if __name__ == "__main__":
    app.run(debug=True, port=5000)

################### forms.py ####################################
from flask_wtf import FlaskForm
from wtforms import StringField, SubmitField, PasswordField, TextAreaField
from wtforms.validators import DataRequired, URL
from flask_ckeditor import CKEditorField


# WTForm for creating a blog post
class CreatePostForm(FlaskForm):
    title = StringField("Blog Post Title", validators=[DataRequired()])
    subtitle = StringField("Subtitle", validators=[DataRequired()])
    img_url = StringField("Blog Image URL", validators=[DataRequired(),
URL()])
    body = CKEditorField("Blog Content", validators=[DataRequired()])
    submit = SubmitField("Submit Post")
```

```
# Create a RegisterForm to register new users
class RegisterForm(FlaskForm):
    email = StringField("Email", validators=[DataRequired()])
    password = PasswordField("Password", validators=[DataRequired()])
    name = StringField("Name", validators=[DataRequired()])
    question = StringField("Question", validators=[DataRequired()])
    answer = PasswordField("Answer", validators=[DataRequired()])
    submit = SubmitField("Submit to Register")


# Create a LoginForm to login existing users
class LoginForm(FlaskForm):
    email = StringField("Email", validators=[DataRequired()])
    password = PasswordField("Password", validators=[DataRequired()])
    submit = SubmitField("Let Me In!")

# Create a Subjective Questions to the user
class SubjectiveQuestion(FlaskForm):

    question = TextAreaField("Question", validators=[DataRequired()])
    answer = PasswordField("Answer", validators=[DataRequired()])
    submit = SubmitField("Submit answer")

# Create a CommentForm so users can leave comments below posts
class CommentForm(FlaskForm):
    comment_text = CKEditorField("Comment", validators=[DataRequired()])
    submit = SubmitField("Submit Comment")
```

```
##############################index.html##############################

{% include "header.html" %}

<!-- Page Header-->
<header
  class="masthead"
  style="background-image: url('../static/assets/img/home-bg.jpg')"
>
  <div class="container position-relative px-4 px-lg-5">
    <div class="row gx-4 gx-lg-5 justify-content-center">
      <div class="col-md-10 col-lg-8 col-xl-7">
        <div class="site-heading">
          <h1> Welcome {{current_user.name}} :)</h1>
          <span class="subheading">A collection of random musings.</span>
        </div>
      </div>
    </div>
  </div>
</header>
<!-- Main Content-->
<div class="container px-4 px-lg-5">
  <div class="row gx-4 gx-lg-5 justify-content-center">
    <div class="col-md-10 col-lg-8 col-xl-7">
      <!-- Post preview-->
      {% for post in all_posts %}
      <div class="post-preview">
        <a href="{{ url_for('show_post', post_id=post.id) }}">
          <h2 class="post-title">{{ post.title }}</h2>
          <h3 class="post-subtitle">{{ post.subtitle }}</h3>
```

```
  </a>
  <p class="post-meta">
    Posted by
    <!-- post.author is now a User object -->
    <a href="#">{{post.author.name}}</a>
    on {{post.date}}
    <!-- Only show delete button if user id is 1 (admin user) -->
    {% if current_user.id == 1: %}
    <a href="{{url_for('delete_post', post_id=post.id) }}">✗</a>
    {% endif %}
  </p>
</div>
<!-- Divider-->
<hr class="my-4" />
{% endfor %}

<!-- New Post -->
<!-- Only show Create Post button if user id is 1 (admin user) -->
{% if current_user.id == 1: %}
<div class="d-flex justify-content-end mb-4">
  <a
    class="btn btn-primary float-right"
    href="{{url_for('add_new_post')}}"
    >Create New Post</a >
</div>
{% endif %}
<!-- Pager-->
<div class="d-flex justify-content-end mb-4">
  <a class="btn btn-secondary text-uppercase" href="#!">Older Posts →</a>
```

```
      </div>
    </div>
  </div>
</div>
{% include "footer.html" %}
```

##########################login.html#################################

```
{% from "bootstrap5/form.html" import render_form %}
{% block content %}
{% include "header.html" %}

<!-- Page Header -->
<header
  class="masthead"
  style="background-image: url('../static/assets/img/login-bg.jpg')"
>
  <div class="container position-relative px-4 px-lg-5">
    <div class="row gx-4 gx-lg-5 justify-content-center">
      <div class="col-md-10 col-lg-8 col-xl-7">
        <div class="page-heading">
          <h1>Log In</h1>
          <span class="subheading">Welcome Back!</span>
        </div>
      </div>
    </div>
  </div>
</header>
```

```
<main class="mb-4">
  <div class="container">
    <div class="row">
      <!-- Adding Flash message here for users trying to register twice -->
      {% with messages = get_flashed_messages() %}
      {% if messages %}
      {% for message in messages %}
      <p class="flash">{{ message }}</p>
      {% endfor %}
      {% endif %}
      {% endwith %}
      <div class="col-lg-8 col-md-10 mx-auto">
        <!--Rendering login form here-->
        {{render_form(form, novalidate=True, button_map={"submit":
"primary"}) }}
      </div>
    </div>
  </div>
</main>
{% include "footer.html" %} {% endblock %}
```

##################other files are#####################################

//header.html, footer.html, register.html, subjective.html, makepost.html, about.html, contact.html., Are the template documents are there..are implemented using jinja template and Sqlalchemy used for db commend in python.

//bootstrap style sheet is used in this Module for css framework.

//assets include images and instance includes the database here

############# JavaScript working on the client side is ##################

```
window.addEventListener('DOMContentLoaded', () => {
   let scrollPos = 0;
   const mainNav = document.getElementById('mainNav');
   const headerHeight = mainNav.clientHeight;
   window.addEventListener('scroll', function() {
      const currentTop = document.body.getBoundingClientRect().top * -1;
      if ( currentTop < scrollPos) {
         // Scrolling Up
         if (currentTop > 0 && mainNav.classList.contains('is-fixed')) {
            mainNav.classList.add('is-visible');
         } else {
            console.log(123);
            mainNav.classList.remove('is-visible', 'is-fixed');
         }
      } else {
         // Scrolling Down
         mainNav.classList.remove(['is-visible']);
         if (currentTop > headerHeight && !mainNav.classList.contains('is-
fixed')) {
            mainNav.classList.add('is-fixed');
         } }
      scrollPos = currentTop; });
})
```

CHAPTER 8

SNAPSHOTS

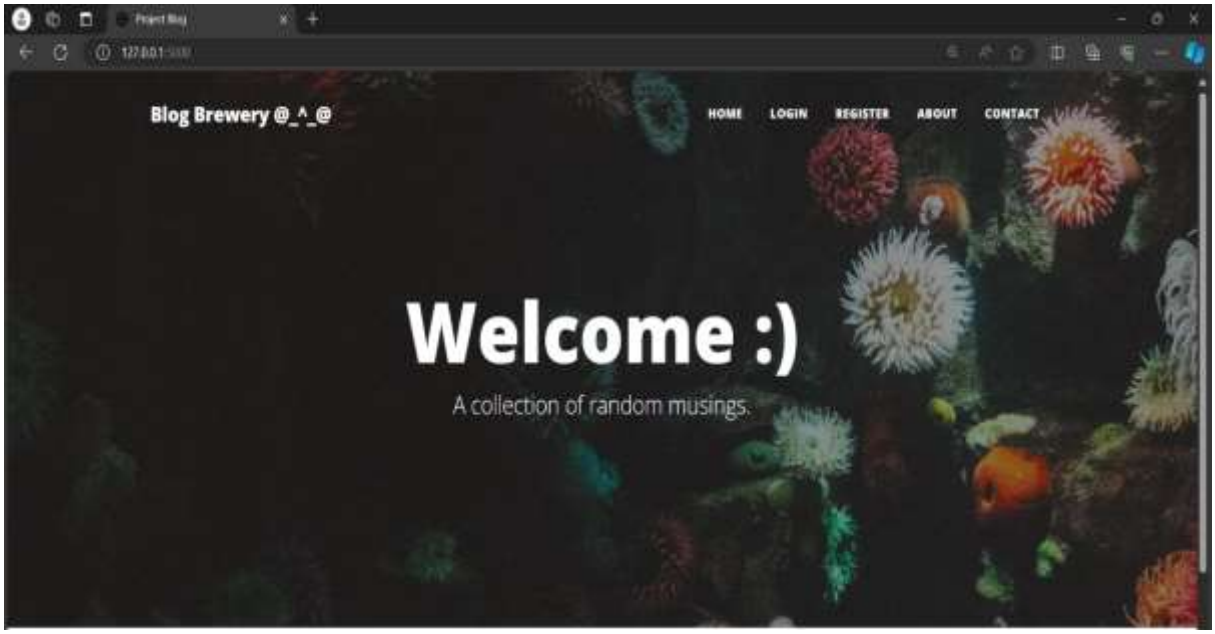

Fig. 8.1 Index page page of the site

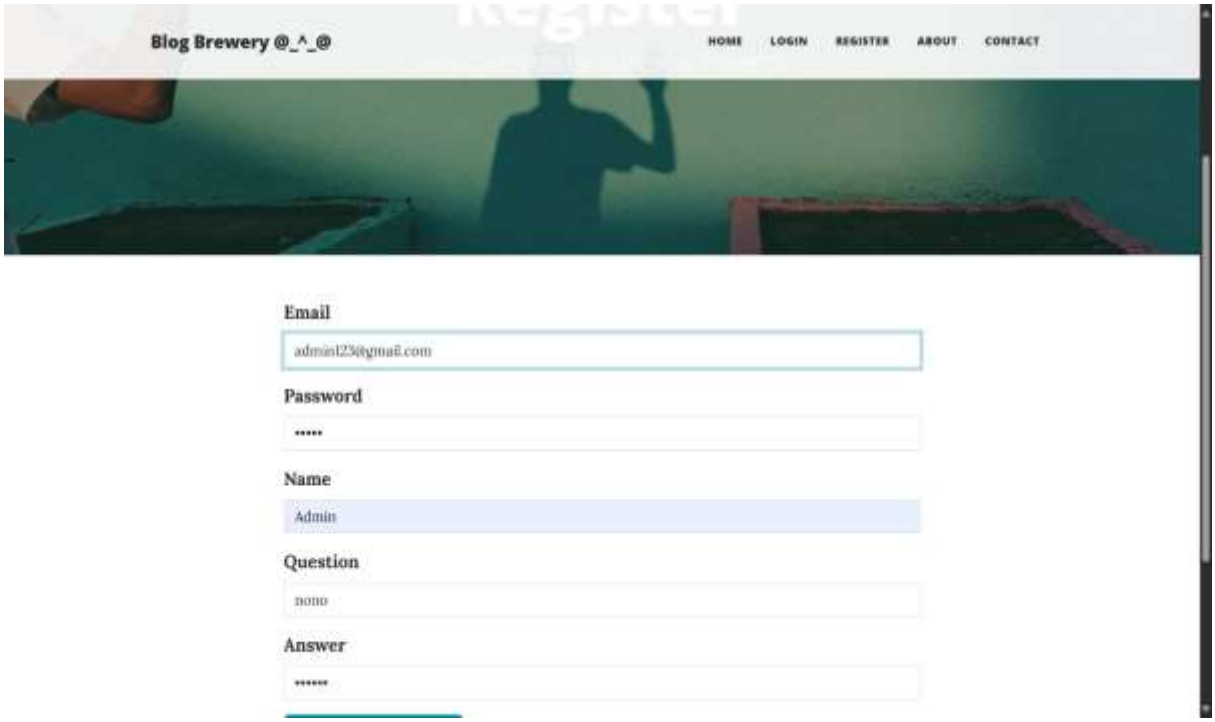

Fig. 8.2 click register button then register the 1st person as Admin

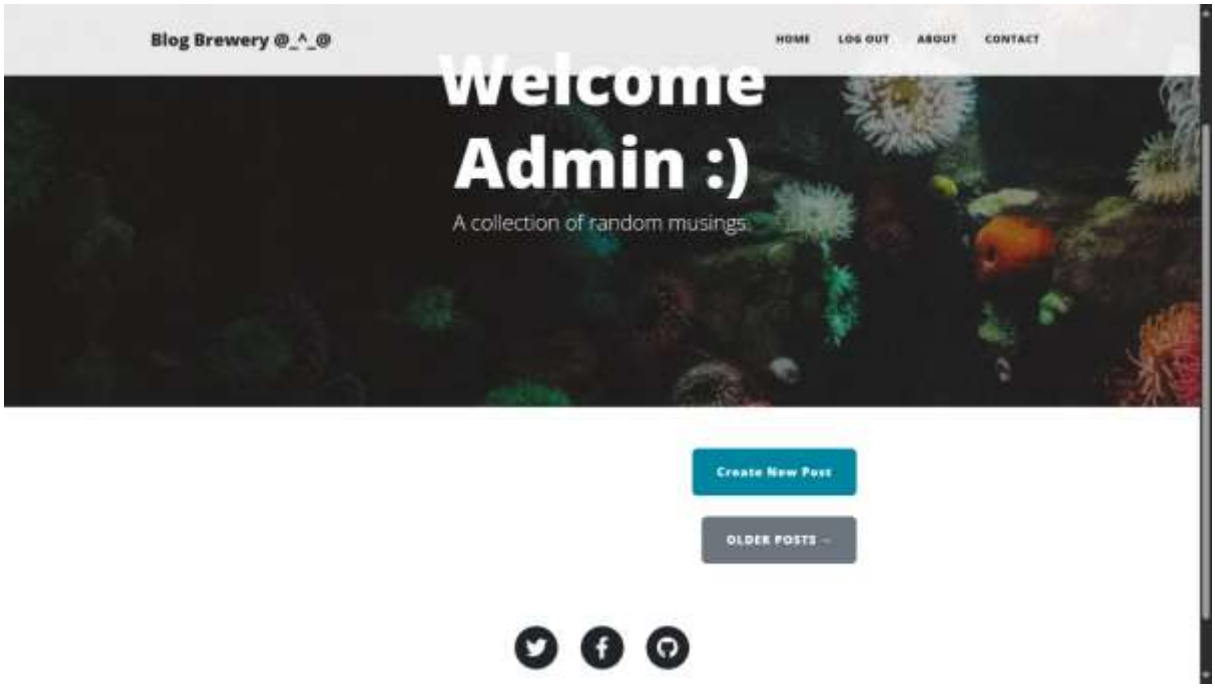

Fig. 8.3 After registration admin can able to see create post and older post as default

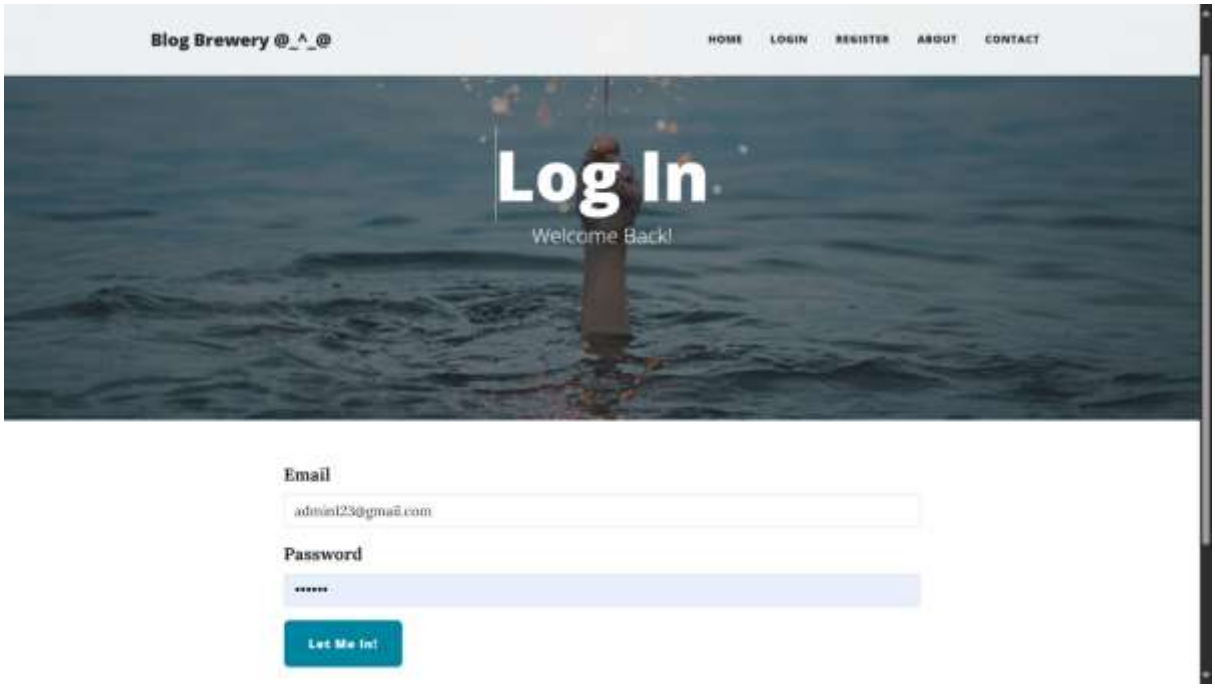

Fig. 8.4 After Clicking logout button we can able to get login page by pressing login button

Fig. 8.5 After the login process we can able to get additional login page we can enter the subjective question and answer which is entered when registration.

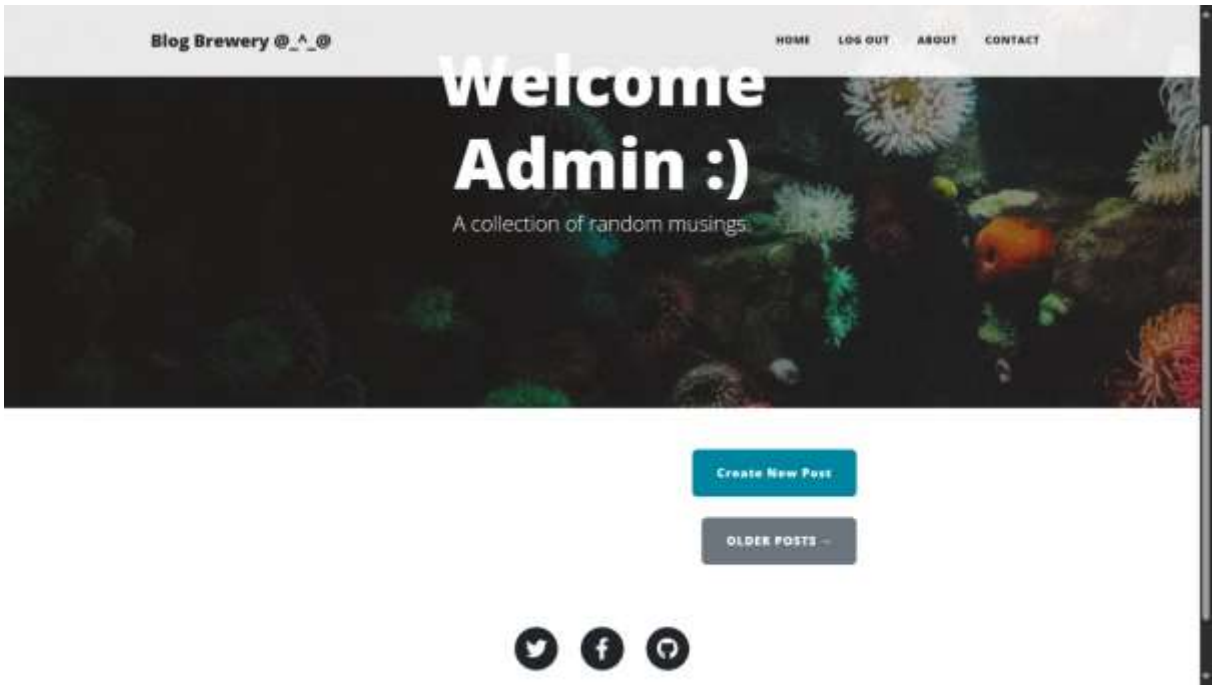

Fig. 8.6 After Clicking submit btn ., now we can able to access the priviliage for Admin (create, edit) blogs

Fig. 8.7 Admin creates a new blog

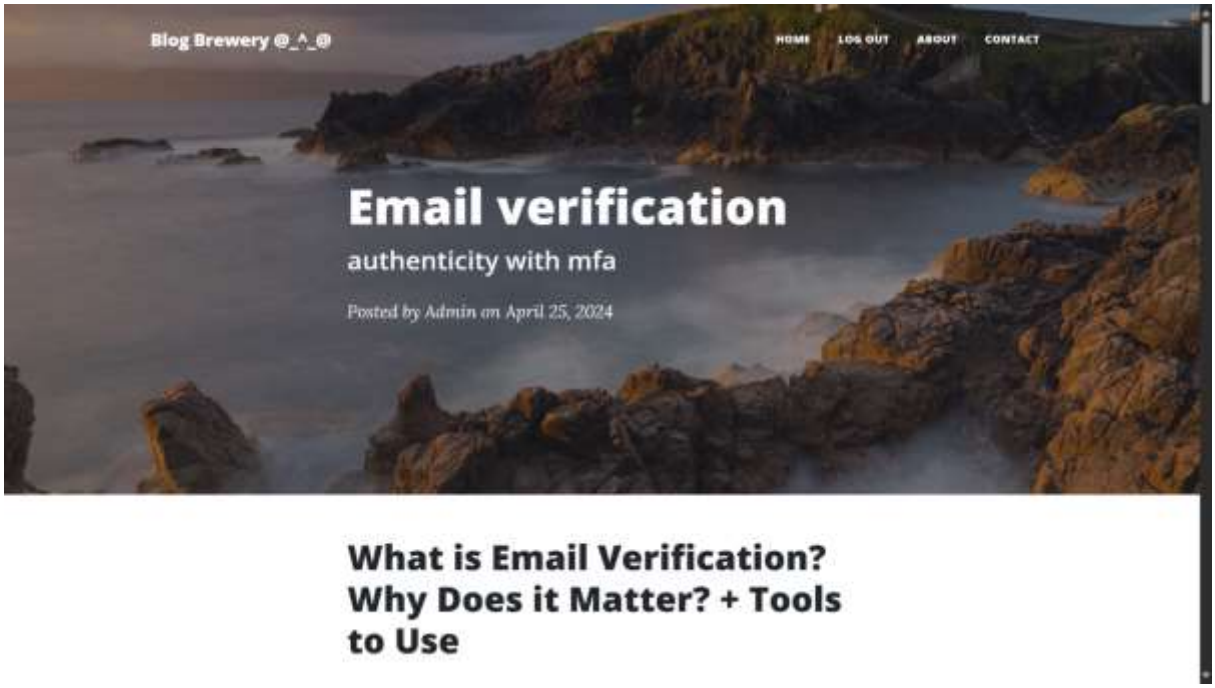

Fig. 8.8 The Blog post created by Admin

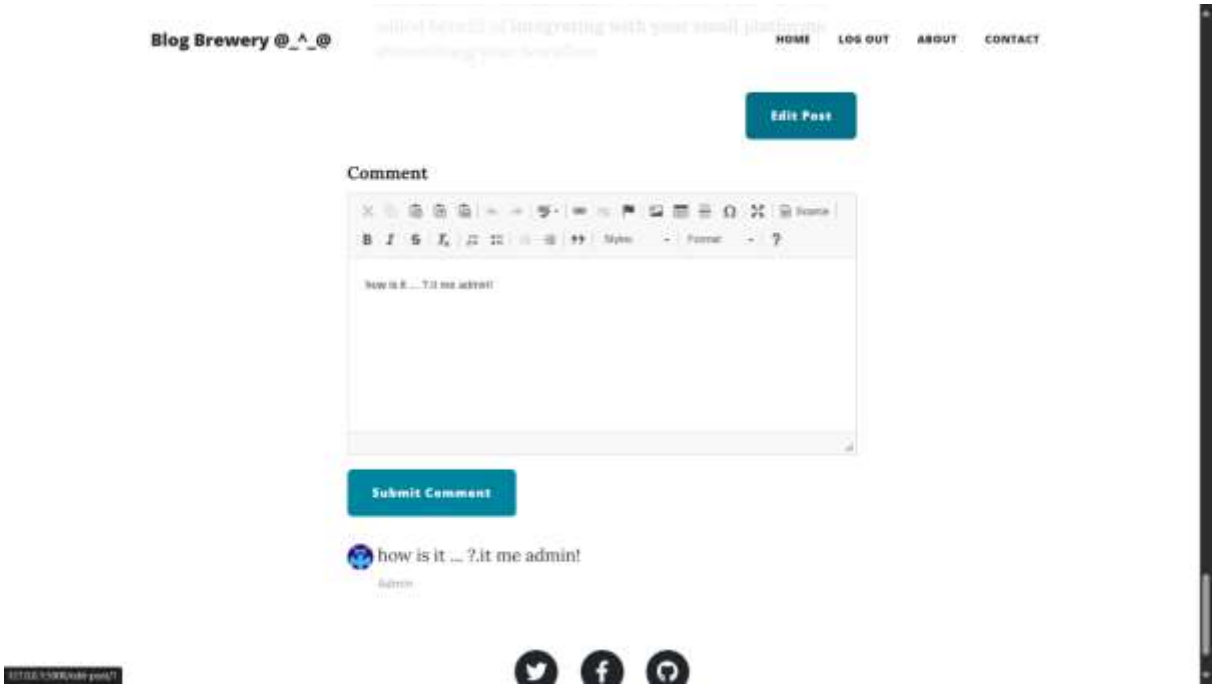

Fig. 8.9 In the Blog the admin can comment .

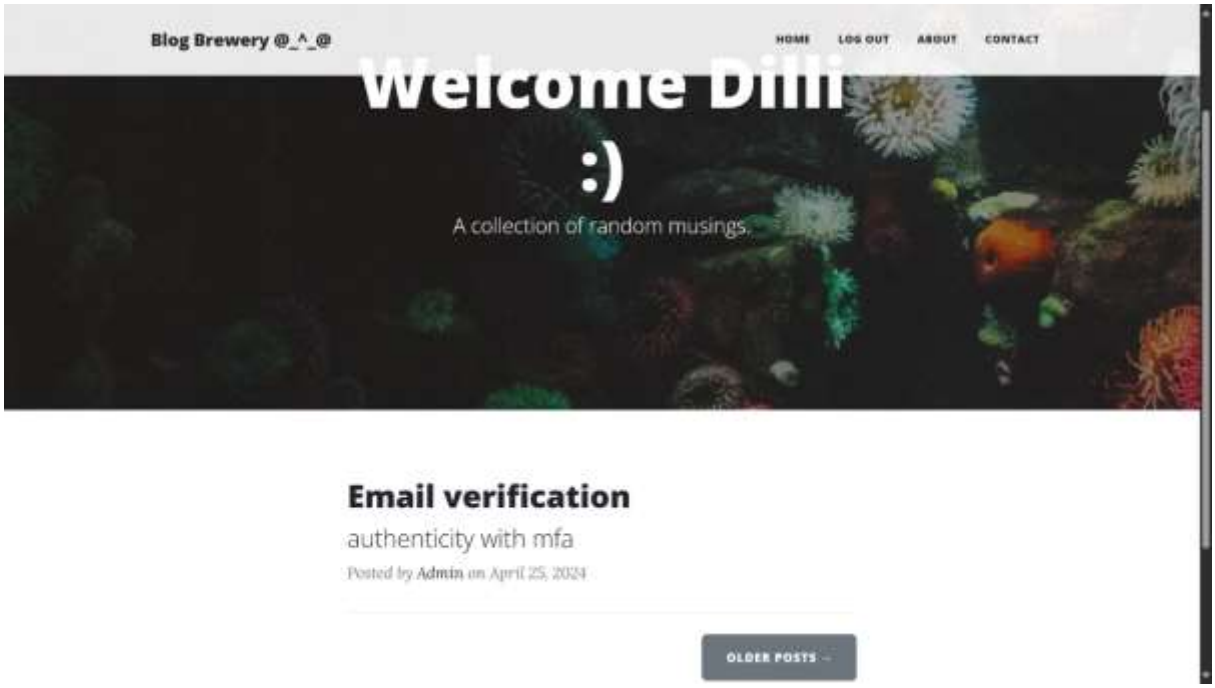

Fig. 8.10 The 2nd user register and logged in as considered as normal user the can't access the create and edit blog privilages in the site.

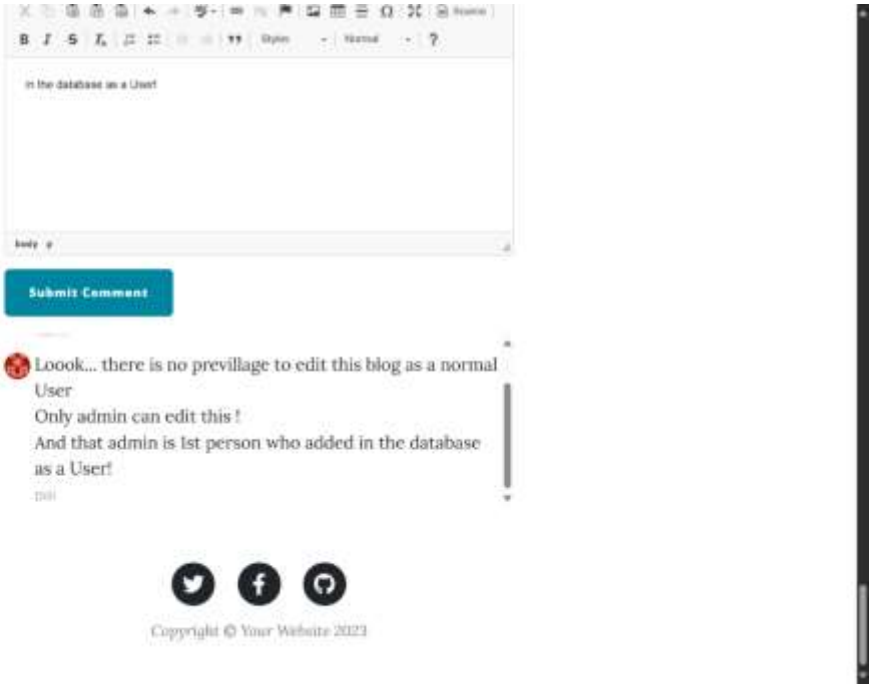

Fig. 8.11. The User can read the blog and comment each blog post like this.

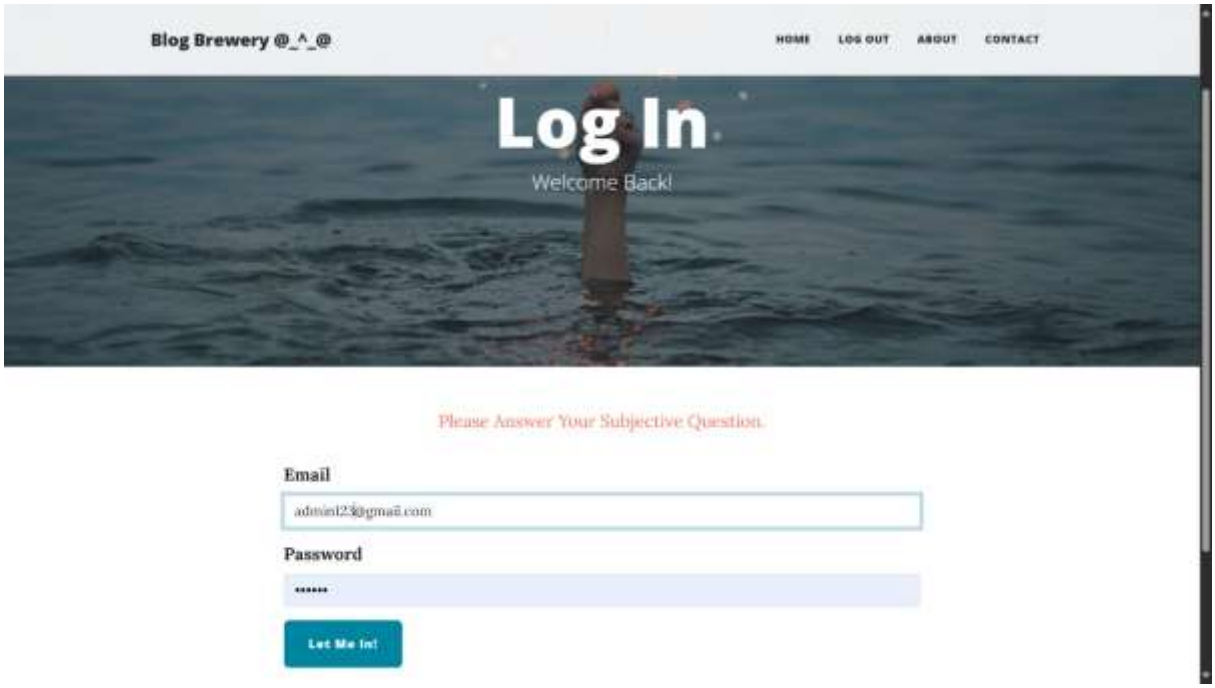

Fig. 8.12. If the user didn't enter he correct Subjective answer for Registered questions… then it will redirect to the index page afte the flash msg passed in the login page.

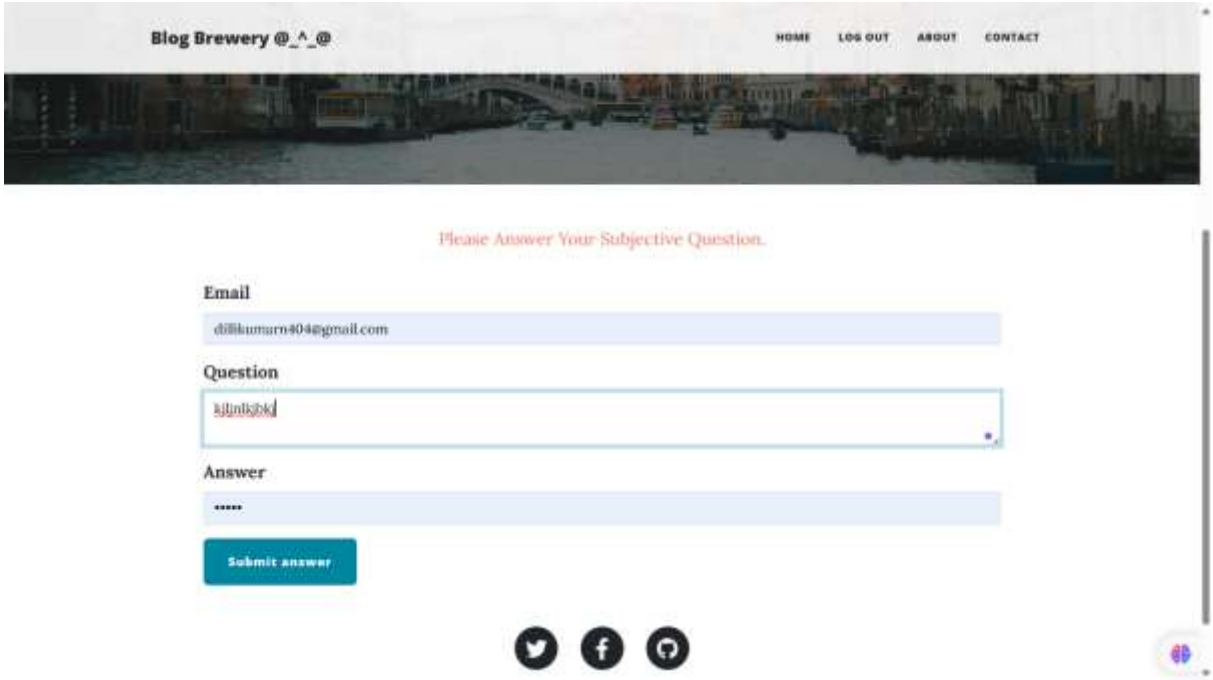

Fig. 8.13. Even if logged in if the next state of Subjective questions if not match with registerted.. then the user will be not allowed to account and mail sent to user email.

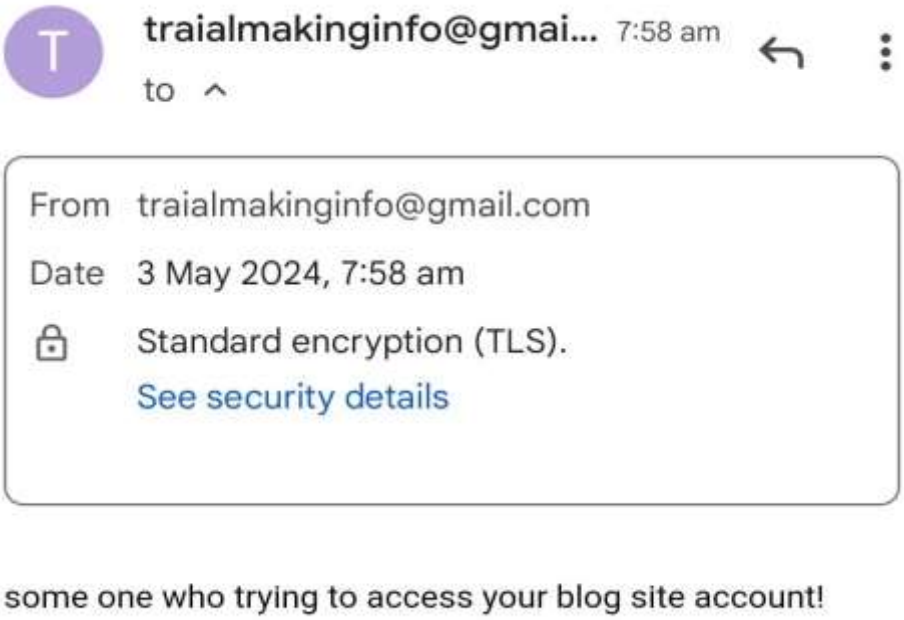

Fig. 8.13. Email alert message to the genuine user

CONCLUSION & FUTURE WORK

CONCLUSION

In our innovative authentication system, subjective-based questions take center stage. Users select personalized questions related to their memories, preferences, or opinions. These questions serve as unique authentication factors, enhancing security beyond traditional methods. The dynamic assignment of questions based on risk levels ensures efficient and effective authentication. Unlike knowledge-based approaches, which rely on shared secrets, our system avoids pre-agreed information vulnerable to social engineering attacks. By respecting user experiences and memories, we envision a robust yet engaging authentication landscape. Subjective questions are questions that require answers in the form of explanations. Unlike objective questions that have clear-cut answers, subjective questions involve personal feelings, opinions, and interpretations.

FUTURE WORK

For Future enhancement we ‘ve planned for to do the User Experience better which is to be enhance and to pick up, and make ready available for all end users from this project. As we look ahead, usability studies, machine learning integration, multi-factor authentication, and privacy enhancements will shape the future of this user-centric paradigm.

REFERENCES

[1]. P. C. Mondal, R. Deb and M. N. Huda, "*Know your customer (KYC) based authentication method for financial services through the internet*," 2016 19th International Conference on Computer and Information Technology (ICCIT), Dhaka, Bangladesh, pp. 535-540, 2016.

[2]. Z. Zulkifl et al., "*FBASHI: Fuzzy and Blockchain-Based Adaptive Security for Healthcare IoTs,*" in IEEE Access, vol. 10, pp. 15644-15656, 2022.

[3]. J. Lee, J. Kim, I. Kim and K. Han, "*Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles*," in IEEE Access, vol. 7, pp. 165607-165626, 2019.

[4]. A. B. Ajmal, M. A. Shah, C. Maple, M. N. Asghar and S. U. Islam, "*Offensive Security: Towards Proactive Threat Hunting via Adversary Emulation*," in IEEE Access, vol. 9, pp. 126023-126033, 2021.

[5]. M. S. Chishti, C. -T. King and A. Banerjee, "*Exploring Half-Duplex Communication of NFC Read/Write Mode for Secure Multi-Factor Authentic*ation," in IEEE Access, vol. 9, pp. 6344-6357, 2021.

[6]. N. A. Karim, O. A. Khashan, H. Kanaker, W. K. Abdulraheem, M. Alshinwan and A. -K. Al-Banna, "*Online Banking User Authentication Methods: A Systematic Literature Review*," in IEEE Access, vol. 12, pp. 741-757, 2024.

[7]. O.B. Lawal, A. Ibitola, O.B. Longe, "*Internet banking authentication methods in Nigeria Commercial Banks*," African Journal of Computing & ICT,Vol 6. No. 1, March 2013.

[8]. Shailesh Kumar Shivakumar, Babu Krishnamurthy, "*Advanced security design for financial applications*," External Document, 2016 [White Paper].

[9]. R. Di Pietro, Gianluigi Me , M. A. Strangio, "*A two-factor mobile authentication scheme for secure financial transactions*," [International Conference on Mobile Business (ICMB'05), pp. 2834, 2005].

[10]. Syeda Farha Shazmeen, Shyam Prasad, "*A practical approach for secure internet banking based on cryptography*," [International Journal of Scientific and Research Publications, Volume 2, Issue 12, December 2012].