SIMPLE BLOGGING PLATFORM USING PYTHON

A PROJECT REPORT

Submitted by

VISHAL A (2303811710621124)

in partial fulfillment for the completion of the course

ECA1121- PYTHON PROGRAMMING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, Affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

JUNE, 2024

K.RAMAKRISHNAN
COLLEGE OF TECHNOLOGY
An Autonomous Institution
Affiliated to Anna University Chennai, Approved by AICTE New Delhi,
ISO 9001:2015 & ISO 14001:2015 Certified Institution, Accredited with 'A+' grade by NAAC
Samayapuram, Tiruchirappalli – 621 112, Tamilnadu, India.

Incubating Minds
Catalyzing Careers

BONAFIDE CERTIFICATE

Certified that this project report titled **"SIMPLE BLOGGING PLATFORM USING PYTHON"** is the bonafide work of **VISHAL A (2303811710621124)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a course was conferred on an earlier occasion on this or any other candidate.

15/6/24

15/6/24

SIGNATURE

Dr.S.SYEDAKBAR, M.E.Ph.D

HEAD OF DEPARTMENT

ASSISTANT PROFESSOR

Department of Electronics and Communication Engineering

K Ramakrishnan College of Technology (Autonomous), Samayapuram- 621 112

SIGNATURE

Mrs.P.SUDHA, M.E., (Ph.D)

SUPERVISOR

ASSISTANT PROFESSOR

Department of Electronics and Communication Engineering

K Ramakrishnan College of Technology (Autonomous), Samaypuram- 621 112

Submitted for the viva-voce examination held on 15.06.2024

DECLARATION

I declare that the project report on **"SIMPLE BLOGGING PLATFORM USING PYTHON"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **ECA1121- PYTHON PROGRAMMING**.

Signature

VISHAL A

Place: Samayapuram

Date: 15.06.2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **DR. A. SYEDAKBAR, M.E., Ph.D.,** Head of the department, **ELECTRONICS AND COMMUNICATION ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project guide **MRS. K. KARPOORA SUNDARI, M.E.,(Ph.D).,** Department of **ELECTRONICS AND COMMUNICATION ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To emerge as a leader among the top institutions in the field of technical education.

MISSION OF THE INSTITUTION

- Produce smart technocrats with empirical knowledge who can surmount the global challenges.
- Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students.
- Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations.

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences

3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations

4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities

   with an understanding of the limitations

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

This project, titled "Simple Blogging Platform Using Python," is a command-line application that enables users to create and manage blog posts. The platform provides a straightforward approach to user account management, allowing users to create accounts, log in, and post content. Each user can create unique blog posts that include a title and content, and these posts are associated with the user who created them. Additionally, users can view their profile, which displays all posts they have authored. The system also supports logging out to ensure secure session management. This project serves as a fundamental example of implementing user authentication, session management, and content creation in Python.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS LIST OF ABBREVIATIONS

IDE	-	Integrated Development Environment
VS Code	-	Visual Studio Code
re	-	Regular Expression
iOS	-	Iphone Operating System
NLP	-	Natural Language Processing
GUI	-	Graphical User Interface
APIs	-	Application Programming Interface

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION TO PYTHON

1.1.1. Overview

Python is a widely-used, high-level programming language renowned for its readability and simplicity, making it an ideal choice for both novice and seasoned programmers. Created by Guido van Rossum and released in 1991, Python's core philosophy emphasizes code readability and straightforward syntax, allowing developers to write clear and concise code more efficiently compared to other languages like C++ or Java.

1.1.2. Programming Paradigms

Python supports various programming paradigms, including procedural, object-oriented, and functional programming. This flexibility, combined with a dynamic type system and automatic memory management, facilitates the development of a wide range of applications, from simple scripts to complex software systems.

1.1.3. Standard Library

The language's comprehensive standard library, often referred to as "batteries-included," provides built-in modules and functions for handling many programming tasks, such as file I/O, system calls, and even web services. This extensive library helps streamline the development process by offering ready-to-use solutions for common programming challenges.

1.1.4. Third-Party Libraries And Frameworks

One of Python's significant strengths is its extensive ecosystem of third-party libraries and frameworks. Popular libraries such as NumPy and Pandas enable efficient data manipulation and analysis, while frameworks like Django and Flask streamline web development. In the realm of machine learning and artificial intelligence, libraries like TensorFlow and PyTorch are widely adopted for building and deploying sophisticated models.

1.1.5. Versions Of Python

Python has undergone significant evolution since its inception, with two major versions in use today:

Python 2: Released in 2000, Python 2.x series was a major milestone and widely used for many years. However, it reached its end of life on January 1, 2020, and is no longer maintained.

Python 3: Introduced in 2008, Python 3.x series brought substantial improvements and changes to the language, such as better Unicode support, a more consistent syntax, and enhanced standard libraries. Python 3 is the recommended version for all new projects.

1.1.6. Python Tools

Python's ecosystem includes numerous tools that enhance productivity and development experience:

- **IDEs and Code Editors:** Popular options include PyCharm, VS Code, and Jupyter Notebook, which offer features like syntax highlighting, code completion, and debugging.
- **Package Management:** Tools like pip and conda facilitate the installation and management of Python libraries and dependencies.
- **Virtual Environments**: virtualenv and venv allow developers to create isolated environments for different projects, ensuring dependency conflicts are avoided.
- **Testing Frameworks:** unit test, pytest, and nose are commonly used for writing and running tests to ensure code reliability and correctness.
- **Build Tools:** setup tools and wheel help in packaging Python projects, making them easy to distribute and install.
- **Documentation Generators:** Tools like Sphinx are used to create comprehensive documentation for Python projects.
- **Linters and Formatters:** pylint, flake8, and black help maintain code quality and consistency by enforcing coding standards and formatting.

1.1.7. Versatility And Adoption

Python's simplicity and versatility have led to its widespread adoption in various fields, including web development, data science, artificial intelligence, automation, and scientific computing. Its active community continually contributes to a rich repository of resources, tutorials, and documentation, making it easier for developers to learn and apply Python effectively.

CHAPTER 2
PROJECT DESCRIPTION

2.1. PROJECT INTRODUCTION

This project involves the creation of a simple blogging platform using Python. The platform allows users to create and share blog posts, comment on posts, and manage their profiles. This project aims to provide a basic yet functional environment for content creation and sharing, enabling users to engage with each other through posts and comments.

2.2. PROJECT OBJECTIVE

The primary objective of this project is to develop a basic command-line blogging platform in Python for user login, post creation, and exiting the platform.

To provide a user-friendly and engaging space where individuals and organizations can easily create, share, and interact with diverse content, fostering community, creativity, and the exchange of ideas.

2.3. PROBLEM STATEMENT

The Simple Blog project is a basic blogging system designed to provide users with a platform to create accounts, log in, write posts, and view their profiles. The application maintains user authentication through username-password pairs and allows logged-in users to create and manage their blog posts. Key features include account creation, secure login, post creation, and profile viewing, with each user able to view their own posts. The project highlights fundamental concepts of user management, session handling, and post management within a console-based interface. Future enhancements could incorporate data persistence, password security improvements, more robust validation, and additional user functionalities to elevate the overall user experience and security.

2.4. LIBRARIES USED

No libraries is used in this program

CHAPTER 3

SYSTEM ANALYSIS

3.1. EXISTING SYSTEM

The provided Simple Blog class implements a basic blogging system where users can create accounts, log in, make posts, view their profiles, and log out. The class maintains a dictionary to store user credentials and tracks the currently logged-in user. Upon creating an account, users can log in and post content, with each post being associated with the author. Users can view their own posts through their profile and log out when done. The system runs through a menu-driven interface that allows users to select options to perform these actions.

3.1.1. DISADVANTAGES

a. Insecure Password Storage

The passwords are stored in plain text within the users dictionary. Storing passwords in plain text is highly insecure because if an attacker gains access to this data (e.g., through a system breach or unauthorized access to the code), they would have immediate access to all user passwords. Best practices recommend storing hashed passwords using strong hashing algorithms (like bcrypt, SHA-256, etc.) along with salt to protect user credentials.

b. Lack of Input Validation

The system does not validate user inputs thoroughly. For instance, it does not check if the username or password inputs are empty or contain invalid characters. This oversight can lead to issues such as allowing empty usernames or passwords, potentially leading to unexpected behavior or security vulnerabilities (e.g., SQL injection attacks if the system were connected to a database).

c. Single User Limitation

The system only supports one user logged in at a time (self.logged_in_user), which severely limits its usability in a real-world scenario where multiple users might need simultaneous access.

3.2. PROPOSED SYSTEM

The proposed system is a command-line-based blogging platform implemented in Python, offering essential functionalities such as account creation, login, post creation,

profile viewing, and logout. Users interact with the system through a menu-driven interface where they can create an account by providing a username and password, log in with existing credentials to access post creation capabilities, view their profile to see authored posts, and log out to end their session. Each post consists of a title, content, and is associated with the logged-in user's username, stored in a list within the system. This design ensures straightforward user interaction and basic blog management functionalities while maintaining simplicity and ease of use.

3.2.1. ADVANTAGES

a. Simplicity and Ease of Use

It provides a straightforward menu-driven interface, making it accessible even to users with minimal technical expertise.

b. Quick Setup and Deployment

Being a command-line application in Python, it is lightweight and easy to set up on various platforms without complex installation requirements.

c. Efficient Content Management

Users can create, view, and manage posts seamlessly through intuitive commands, enhancing productivity in managing blog content

d. Security

User authentication is implemented through username-password pairs stored in a dictionary, ensuring secure access to individual accounts.

e. Customization and Extensibility

The system can be extended with additional features or improvements easily due to its modular design and the flexibility of Python.

f. Offline Capability

As a command-line application, it can operate without requiring an active internet connection, allowing users to manage their blog posts offline.

CHAPTER 4
SYSTEM DESIGN & MODULES

4.1. BLOCK DIAGRAM

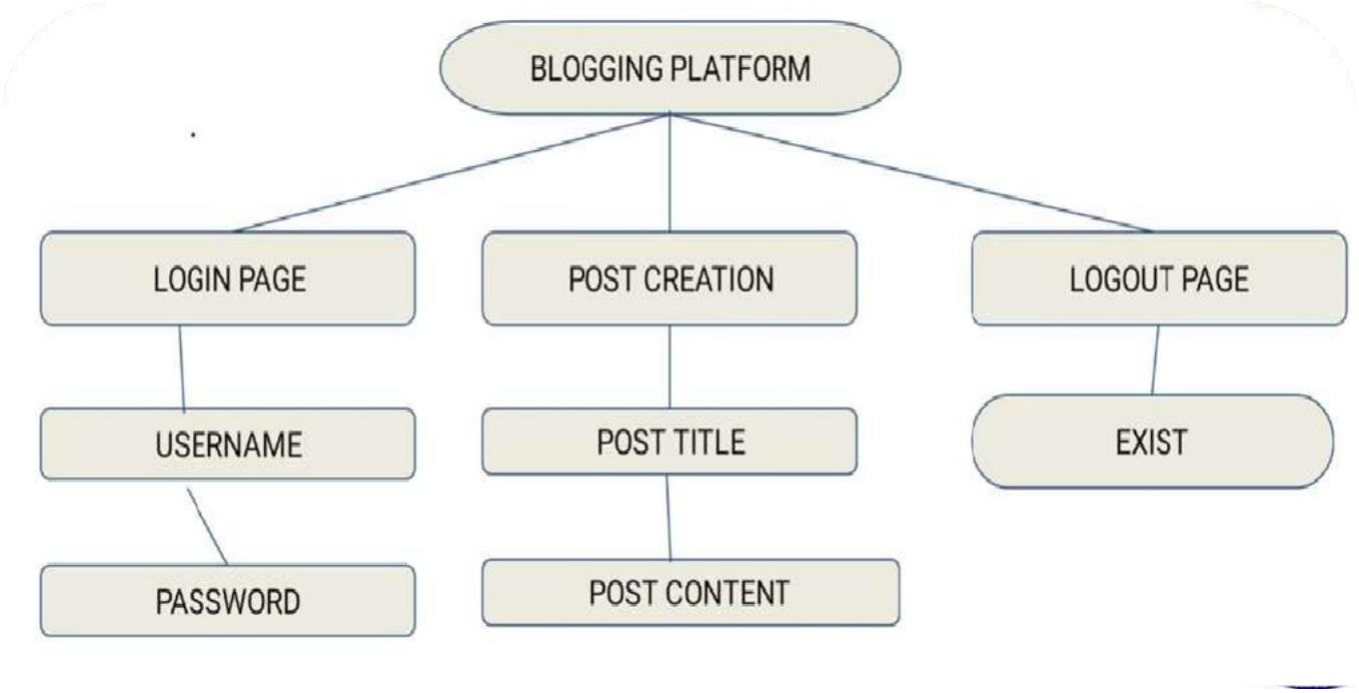

Fig. 4.1. Block Diagram

4.2. MODULE DESCRIPTION

4.2.1. CREATE ACCOUNT MODULE

Description: This module allows users to create a new account for the blog by providing a unique username and password.

Functionality: Users input their desired username and password, which are then stored securely in the blog's database.

4.2.2. LOGIN PAGE MODULE

Description: This module provides users with a login page to access their existing accounts.

Functionality: Users enter their username and password to authenticate themselves. If the credentials match an existing account, the user is granted access to the blog's features.

4.2.3. POST MODULE

Description: This module enables users to create and publish new posts on the blog.

Functionality:

I) POST TITLE: Users input the title of their post.

II) POST CONTENT: Users input the content of their post.

4.2.4. LOGOUT MODULE

Description: This module allows users to log out of their current session, terminating their access to the blog.

Functionality: Users can choose to log out, which clears their current session and returns them to the login page if they wish to access the blog again later.

CHAPTER 5
CONCLUSION & FUTURE ENHANCEMENT

5.1. CONCLUSION

The provided code defines a simple blogging system implemented in Python using classes. It allows users to create accounts, login, post content, view their profile, and logout. Each user can create multiple posts, and the posts are stored in a list within the blog instance. The system ensures that users must be logged in to perform certain actions like posting or viewing their profile. However, there are a few potential improvements that could enhance its functionality and security. Firstly, password encryption should be considered to protect user credentials. Additionally, input validation could be implemented to ensure that usernames are unique and that passwords meet certain criteria for strength. Furthermore, error handling could be enhanced to provide more informative messages to users in case of login failures or other issues. Overall, while the current implementation provides basic blogging functionality, further enhancements could improve its usability and security.

5.2. FUTURE ENHANCEMENT

In future enhancements, adding features like comment sections for posts could foster engagement within the blogging community. Implementing a search functionality would enable users to find posts based on keywords or topics of interest, enhancing the discoverability of content. Additionally, incorporating user roles and permissions could allow for different levels of access, such as administrators who can moderate content or manage user accounts. Integrating email notifications for new posts or comments could keep users informed and engaged with the platform. Furthermore, expanding the blog's capabilities to support multimedia content like images or videos would enrich the user experience. Lastly, considering scalability and performance improvements, such as optimizing database queries or implementing caching mechanisms, could ensure the system can handle increasing user traffic and content volume effectively. Overall, these future enhancements could elevate the blogging platform, making it more feature-rich, user-friendly, and scalable.

In future iterations, enhancing the blog's user interface with a modern and responsive design could improve user satisfaction and accessibility across various devices. Implementing social media integration would enable users to share posts easily on popular platforms, expanding the blog's reach and fostering community engagement. Additionally, introducing analytics tools to track user behavior and post performance could provide valuable insights for content creators to tailor their content to audience preferences. Incorporating features like scheduled posts or drafts would empower users to plan and manage their content more efficiently. Moreover, considering internationalization and localization features to support multiple languages and cultural preferences could attract a more diverse user base. Lastly, exploring monetization options such as advertisements, premium subscriptions, or affiliate marketing could provide opportunities for revenue generation and sustainable growth. By incorporating these enhancements, the blog could evolve into a robust and dynamic platform that meets the evolving needs of its users while also unlocking new opportunities for expansion and innovation.

APPENDICES

APPENDIX A-SOURCE CODE

```
class SimpleBlog:
    def _init_(self):
        self.users = {} # Dictionary to store username-password pairs
        self.logged_in_user = None
        self.posts = []
    def create_account(self):
        print("Create Account")
        username = input("Enter your username: ")
        password = input("Enter your password: ")
        self.users[username] = password
        print("Account created successfully!")
    def login(self):
        print("Login Page")
        username = input("Username: ")
        password = input("Password: ")

        if self.users.get(username) == password:
            self.logged_in_user = username
            print("Login successful!")
        else:
            print("Login failed!")
    def post(self):
        if not self.logged_in_user:
            print("You need to login first!")
            return
                print("New Post")
        title = input("Post Title: ")
        content = input("Post Content: ")
                new_post = {"title": title, "content": content, "author": self.logged_in_user}
        self.posts.append(new_post)
        print("Post added successfully!")
    def view_profile(self):
        if not self.logged_in_user:
            print("You need to login first!")
            return
                print(f"\nProfile of {self.logged_in_user}")
        user_posts = [post for post in self.posts if post['author'] == self.logged_in_user]
        if user_posts:
            for idx, post in enumerate(user_posts, 1):
```

```
            print(f"\nPost {idx}:")
            print(f"Title: {post['title']}")
            print(f"Content: {post['content']}")
        else:
            print("No posts found.")

    def logout(self):
        if self.logged_in_user:
            print(f"User {self.logged_in_user} logged out.")
            self.logged_in_user = None
        else:
            print("No user is currently logged in.")

    def display_menu(self):
        while True:
            print("\nMenu:")
            print("1. Create Account")
            print("2. Login")
            print("3.  Post")
            print("4. View Profile")
            print("5. Logout")
            print("6. Exit")
            choice = input("Choose an option: ")

            if choice == "1":
                self.create_account()
            elif choice == "2":
                self.login()
            elif choice == "3":
                self.post()
            elif choice == "4":
                self.view_profile()
            elif choice == "5":
                self.logout()
            elif choice == "6":
                print("Exiting...")
                break
            else:
                print("Invalid choice. Please try again.")
if _name_ == "_main_":
    blog = SimpleBlog()
    blog.display_menu(
```

APPENDIX B -SCREEN SHOTS

```
Menu:
1. Create Account
2. Login
3. Post
4. View Profile
5. Logout
6. Exit
Choose an option: 1
Create Account
Enter your username: user1
Enter your password: pass123
Account created successfully!

Menu:
1. Create Account
2. Login
3. Post
4. View Profile
5. Logout
6. Exit
Choose an option: 2
Login Page
Username: user1
Password: pass123
Login successful!

Menu:
1. Create Account
2. Login
3. Post
4. View Profile
5. Logout
6. Exit
Choose an option: 3
New Post
Post Title: My First Post
Post Content: This is the content of my first post.
Post added successfully!
```

```
Menu:
1. Create Account
2. Login
3. Post
4. View Profile
5. Logout
6. Exit
Choose an option: 4

Profile of user1

Post 1:
Title: My First Post
Content: This is the content of my first post.

Menu:
1. Create Account
2. Login
3. Post
4. View Profile
5. Logout
6. Exit
Choose an option: 5
User user1 logged out.

Menu:
1. Create Account
2. Login
3. Post
4. View Profile
5. Logout
6. Exit
Choose an option: 6
Exiting...
```