|  |  |  |
| --- | --- | --- |
| **Chapter no:** | **Topic** | **Page No:** |
| **1** | **Company Profile** | **2** |
| **2** | **Abstract** | **3** |
| **3** | **Introduction to project** | **4** |
| **4** | **Daily work progress** | **5** |
| **5** | **Module description** | **6-7** |
| **6** | **Flowchart** | **8** |
| **7** | **Outputs** | **10-12** |
| **8** | **Conclusion** | **13** |
| **9** | **Reference** | **14** |

**INDEX**

**Chapter 1:**

**COMPANY PROFILE**

**Company Name: EZ Trainings and Technologies Pvt. Ltd.**

**Introduction:**

EZ Trainings and Technologies Pvt. Ltd. is a dynamic and innovative organization dedicated to providing comprehensive training solutions and expert development services. Established with a vision to bridge the gap between academic learning and industry requirements, we specialize in college trainings for students, focusing on preparing them for successful placements. Additionally, we excel in undertaking development projects, leveraging cutting-edge technologies to bring ideas to life.

**Mission:**

Our mission is to empower the next generation of professionals by imparting relevant skills and knowledge through specialized training programs. We strive to be a catalyst in the career growth of students and contribute to the technological advancement of businesses through our development projects.

**Services:**

**College Trainings:**

• Tailored training programs designed to enhance the employability of students.

• Industry-aligned curriculum covering technical and soft skills.

• Placement assistance and career guidance.

**Development Projects:**

• End-to-end development services, from ideation to execution.

• Expertise in diverse technologies and frameworks.

• Custom solutions to meet specific business needs.

**Locations:** Hyderabad | Delhi NCR

**Chapter 2:**

**Abstract:**

The user account management system project presents a comprehensive solution for efficiently creating, managing, and deleting user accounts within an organization or application. Leveraging object-oriented programming principles, the system offers a user-friendly interface with essential functionalities to enhance security, productivity, and user experience.

Key features of the system include secure authentication mechanisms, efficient account management capabilities, and persistent storage of user account data. Users can seamlessly create new accounts, log in to existing accounts, log out securely, and delete accounts as needed. User information is stored in a text file (**accounts.txt**), ensuring data persistence across multiple sessions.

With a focus on scalability and flexibility, the system can accommodate future enhancements or modifications, such as account recovery options or role-based permissions. Through clear prompts, feedback messages, and robust security measures, the system aims to provide organizations and applications with a reliable and user-centric solution for managing user accounts while maintaining data security and integrity.

Top of Form

**Chapter 3:**

**Introduction of the Project**

The user account management system project addresses the fundamental need for efficiently managing user accounts within organizations or applications. In today's digital landscape, where security and user experience are paramount, effective account management is essential for ensuring data integrity, protecting sensitive information, and providing a seamless user experience.

This project aims to develop a robust system that simplifies the process of creating, logging in to, managing, and deleting user accounts. By leveraging object-oriented programming principles, the system offers a user-friendly interface with intuitive functionalities, catering to both end-users and administrators alike.

Through secure authentication mechanisms, stringent password management, and persistent storage of user data, the system prioritizes data security and privacy. Users can confidently create accounts, log in securely, and manage their account settings with ease, while administrators can efficiently oversee user management tasks.

With scalability and flexibility in mind, the system is designed to accommodate future enhancements and modifications, adapting to the evolving needs of organizations and applications. By providing a reliable and user-centric solution for user account management, this project aims to contribute to the overall efficiency, security, and user satisfaction of digital platforms.

**Chapter 5:**

**Module description**

**User Class**:

The User class represents a user of the system. It has attributes such as name, email, password, role, and login time. The constructor init initializes these attributes when a new user object is created.

**Account Manager Class**:

The Account Manager class manages user accounts. It initializes with an empty dictionary user to store user objects, with email addresses as keys for faster lookup.

**create account Method**:

This method prompts the user to input details like name, email, password, and role to create a new account. It checks if the email is already in use. If yes, it prints a message indicating that the account already exists. If the email is not in use, it creates a new User object with the provided details, adds it to the users dictionary, and appends the account details to the "accounts.txt" file.

**login Method**:

This method allows users to log in by providing their email and password. It checks if the provided email and password match an existing user's credentials. If the credentials are correct, it updates the user's login time attribute with the current time and returns the corresponding user object. If the credentials are incorrect, it prints a message indicating invalid credentials.

**logout Method**:

This method logs out the user by resetting the login time attribute to None. It also prints the login and logout times, along with a farewell message.

**delete\_account Method**:

This method allows users to delete their account by providing their email and password. It checks if the provided email and password match an existing user's credentials. If the credentials are correct, it removes the user from the user’s dictionary and updates the "accounts.txt" file by removing the corresponding line. If the credentials are incorrect or the account does not exist, it prints a message indicating the issue.

**\_update\_accounts\_file Method**:

This method is a helper method used by delete\_account to update the "accounts.txt" file. It reads all lines from the file, skips the line corresponding to the deleted user, and writes the remaining lines back to the file.

**main Function**:

The main function is the entry point of the program. It creates an instance of the Account Manager class and initializes a variable logged\_in\_user to keep track of the currently logged-in user. It presents a menu to the user with options to create an account, login, logout, delete an account, or exit the program. Depending on the user's choice, it calls the respective methods of the Account Manager class to perform the desired actions.

**Chapter 6:**

**Flow chart**

**Start**

|

|---(1) **Create Account**

| |--- Input name, email, password, role

| |--- Check if email already exists

| |--- Create new user object

| |--- Store user object in dictionary

| |--- Update accounts file

| |--- Print "Account created successfully"

|

|---(2**) Login**

| |--- Input email, password

| |--- Check if email exists and password matches

| |--- Update login time for user

| |--- Print "Login successful"

|

|---(3) **Logout**

| |--- Check if user is logged in

| |--- Update logout time for user

| |--- Print login and logout time

| |--- Print "Goodbye, [user name]!"

|

|---(4) **Delete Account**

| |--- Input email, password

| |--- Check if email exists and password matches

| |--- Remove user from dictionary

| |--- Update accounts file

| |--- Print "Account deleted successfully"

|

|---(5) **Exit**

| |--- Print "Exiting..."

| |--- End

|

**End**

**Chapter 7:**

**Output**

1. Create Account

2. Login

3. Logout

4. Delete Account

5. Exit

Enter your choice: 1

Enter name: John Doe

Enter email: john@example.com

Enter password: password123

Enter role (employee/manager/CEO): employee

Account created successfully.

1. Create Account

2. Login

3. Logout

4. Delete Account

5. Exit

Enter your choice: 2

Enter email: john@example.com

Enter password: password123

Login successful.

1. Create Account

2. Login

3. Logout

4. Delete Account

5. Exit

Enter your choice: 3

Logged in at: 2024-04-30 09:00:00

Logged out at: 2024-04-30 09:10:00

Goodbye, John Doe!

1. Create Account

2. Login

3. Logout

4. Delete Account

5. Exit

Enter your choice: 4

Enter email: john@example.com

Enter password: password123

Account deleted successfully.

1. Create Account

2. Login

3. Logout

4. Delete Account

5. Exit

Enter your choice: 5

Exiting...

**Updates to the "accounts.txt" file after each operation:**

Before creating an account:

**# Empty file**

After creating an account:

**John Doe, john@example.com, password123, employee**

After deleting the account:

**# Empty file**

**Chapter 8:**

**Conclusion**

In conclusion, the user account management system project provides a robust solution for creating, managing, and deleting user accounts within an organization or application. Through the implementation of object-oriented programming principles, we have developed a system that offers essential functionalities to enhance security, efficiency, and user experience.

Throughout the project, we focused on the following key aspects:

1. **User-Friendly Interface:** The system features a user-friendly interface that allows users to easily create, log in to, and manage their accounts. Clear prompts and feedback messages ensure a smooth user experience.
2. **Security Measures:** We implemented secure authentication mechanisms to protect user accounts from unauthorized access. Passwords are securely stored and verified during login, ensuring only authorized users can access their accounts.
3. **Efficient Account Management:** With the ability to create, log in to, log out from, and delete accounts, the system provides efficient account management capabilities. Users can perform these actions seamlessly, enhancing overall productivity.
4. **Data Persistence:** User account data is persisted in a text file (**accounts.txt**), ensuring that user information is retained even after the program is closed. This enables users to access their accounts across multiple sessions.
5. **Scalability and Flexibility:** The system is designed to be scalable and flexible, accommodating future enhancements or modifications as needed. Additional features, such as account recovery options or role-based permissions, can be easily integrated into the system.

**Chapter 9:**

**References:**

**Python documentation:**

<https://docs.python.org/3/tutorial/datastructures.html>

**Chat GPT:**

<https://chat.openai.com/c/9a1f5a95-bd14-4e72-be24-85b26d994dc9>

**GitHub:**

<https://github.com/>

**Jupyter Notebook**