**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

At EZ Trainings and Technologies Pvt. Ltd., we believe in transforming potential into excellence

**CHAPTER-3**

**ABSTRACT**

In an age where security and privacy are paramount concerns, biometric data has emerged as a crucial tool for identity verification. However, the effective management of biometric data presents significant challenges. This abstract introduces a Biometric Data Management System (BDMS) designed to address these challenges by providing a comprehensive solution for the collection, storage, and authentication of biometric data. Key components of the BDMS include robust data encryption, real-time authentication, role-based access control, seamless integration with existing systems, scalability, and compliance with privacy regulations. The BDMS aims to streamline identity management processes while ensuring the privacy and security of individuals' biometric information.

In today's era of digitization and heightened security concerns, the need for robust biometric data management systems is paramount. Biometric technologies offer unique advantages in authentication and identification, but effective management of the associated data is essential to ensure privacy, security, and usability. This abstract outlines the framework and objectives of a Proof of Concept (POC) for a Biometric Data Management System (BDMS).

The proposed BDMS POC aims to address key challenges in biometric data handling, including data storage, access control, encryption, and integration with existing systems. By leveraging state-of-the-art biometric authentication methods such as fingerprint, facial recognition, iris scanning, and voice recognition, the BDMS offers a comprehensive solution for identity verification across various applications.

**CHAPTER-4**

**INTRODUCTION OF THE PROJECT**

“Biometric Data Management” stands at the forefront of modern security and authentication systems, revolutionizing the way individuals are identified and verified. Biometrics, such as fingerprints, facial features, iris patterns, and voiceprints, offer unique and immutable identifiers, making them invaluable in various sectors, including law enforcement, finance, healthcare, and access control. However, the effective management of biometric data presents a myriad of challenges, ranging from data security and privacy concerns to interoperability and scalability issues.

This introduction provides an overview of biometric data management, highlighting its importance in enhancing security, streamlining authentication processes, and safeguarding individual privacy. It explores the complexities associated with collecting, storing, and processing biometric information, as well as the technologies and strategies employed to mitigate risks and ensure compliance with regulatory standards. By addressing these challenges, organizations can harness the power of biometrics to strengthen identity verification systems and better protect sensitive information in an increasingly digital world.

**CRUD Operations for Biometric Data Management System:**

CRUD operations, standing for Create, Read, Update, and Delete, are fundamental to any data management system, including a biometric data management system. Here's how these operations would apply to a biometric data management system:

1. Create: This operation involves adding new biometric data records to the system. When a new individual enrolls in the system, their biometric data, such as fingerprints, facial features, or iris patterns, is captured and stored securely. The system assigns a unique identifier to each record for future reference.

2. Read: The read operation allows users to retrieve biometric data from the system. Authorized personnel, such as administrators or authenticated users, can query the system to access specific biometric records based on criteria such as individual identifiers or biometric characteristics.

3. Update: Biometric data may need to be updated over time due to changes in an individual's appearance or other factors. The update operation enables authorized users to modify existing biometric records to reflect any changes accurately. For example, if a person undergoes facial surgery or experiences a change in fingerprint patterns, their biometric data can be updated in the system accordingly.

4. Delete: The delete operation removes biometric data records from the system when they are no longer needed or when requested by the individual. This operation must be performed securely to ensure that the data is irreversibly removed from the system and cannot be recovered. Additionally, organizations must comply with data retention policies and regulatory requirements when deleting biometric data.

**CHAPTER-5**

**MODULE DESCRIPTION**

This Python code defines two classes: BiometricData and BiometricDataManager, along with a main() function for running the program. Let's break down the functionality and features of each part of the code:

**BiometricData class:**

This class represents individual biometric data records. It has attributes such as name, age, date of birth (DoB), nationality, ID number, biometric type, and visa number. The constructor (\_init\_) initializes these attributes.

**BiometricDataManager class:**

This class manages a collection of BiometricData objects. It has methods for loading data from a file, saving data to a file, creating new biometric data, deleting existing biometric data, updating existing biometric data, verifying data presence, displaying all data, and managing enroll data. When initialized with a file path, it loads existing data from that file.

* The create\_Data() method adds new biometric data to the database.
* The delete\_Data() method removes biometric data from the database.
* The update\_Data() method allows for updating existing biometric data.
* The verify\_Data() method checks if a given ID number exists in the database.
* The display\_all() method prints all stored biometric data.
* The manage\_enroll\_data() method is used for managing enroll data, although it's not explicitly clear what this data is for without more context.

**File Handling:**

1. Loading Data from File (load\_data\_from\_file method):

* This method reads data from a file specified by ‘file\_path’ attribute.
* It uses a with statement to open the file in read mode ('r').
* It iterates over each line in the file.
* For each line, it splits the line using split(',') assuming that the data is comma-separated.
* It extracts the relevant fields like id\_number, name, age, DoB, nationality, biometric\_type, and visa\_number.
* It creates a new ‘BiometricData’ object with the extracted data and stores it in the data dictionary with id\_number as the key.

1. Saving Data to File (save\_data\_to\_file method):

* This method writes data to the file specified by ‘file\_path’ attribute.
* It uses a with statement to open the file in write mode ('w').
* It iterates over each ‘BiometricData’ object stored in the data dictionary.
* For each object, it writes a line to the file containing the comma-separated values of its attributes (name, age, DoB, nationality, id\_number, biometric\_type, visa\_number).

**Main function:**

The main() function serves as the entry point of the program. It creates an instance of ‘BiometricDataManager’, which initializes the data from a file.

It presents a menu to the user with options to create, delete, update, verify, or display biometric data, as well as managing enroll ‘data.Based’ on the user's choice, it calls the corresponding method of ‘BiometricDataManager’. It exits the program when the user chooses to do so.

**CHAPTER-6**

**ALGORITHM**

**1. Initialize Classes:**

* Define a class ‘BiometricData’ to represent individual biometric data records with attributes name, age, DoB, nationality, id\_number, biometric\_type, and visa\_number.
* Define a class ‘BiometricDataManager’ to manage a collection of ‘BiometricData’ objects with methods for CRUD operations and file handling.

**2. Main Function:**

* Define a main() function as the entry point of the program.
* Create an instance of ‘BiometricDataManager’.
* Enter a loop to present a menu to the user and perform actions until the user chooses to exit.

**3.Load Data from File:**

* Implement the ‘load\_data\_from\_file’ method in ‘BiometricDataManager’.
* Open the specified file in read mode.
* Read each line from the file.
* Split the line into fields using a delimiter.
* Create a ‘BiometricData’ object from the fields.
* Store the object in memory.

**4.Save Data to File:**

* Implement the ‘save\_data\_to\_file’ method in ‘BiometricDataManager’.
* Open the specified file in write mode.
* Iterate over the ‘BiometricData’ objects stored in memory.
* Write each object's attributes as a comma-separated line in the file.

**5. Create Biometric Data:**

* Implement the ‘create\_Data’ method in ‘BiometricDataManager’.
* Prompt the user for input to create a new ‘BiometricData’ object.
* Add the new object to memory.

**6. Delete Biometric Data:**

* Implement the ‘delete\_Data’ method in ‘BiometricDataManager’.
* Prompt the user for the ID number of the data to be deleted.
* Remove the corresponding object from memory.

**7. Update Biometric Data:**

* Implement the ‘update\_Data’ method in ‘BiometricDataManager’.
* Prompt the user for the ID number of the data to be updated.
* Prompt the user for updated attributes of the data.
* Update the corresponding object in memory.

**8. Verify Biometric Data:**

* Implement the ‘verify\_Data’ method in ‘BiometricDataManager’.
* Prompt the user for the ID number to be verified.
* Check if the ID number exists in memory and provide feedback.

**9. Display All Biometric Data:**

* Implement the ‘display\_all’ method in ‘BiometricDataManager’.
* Iterate over all stored ‘BiometricData’ objects.
* Print the attributes of each object.

**10. Manage Enroll Data:**

* Implement the ‘manage\_enroll\_data’ method in ‘BiometricDataManager’.
* Prompt the user for input to create a new ‘BiometricData’ object for enrollment.
* Add the new object to memory.

**11. Exit Program:**

* Exit the loop and terminate the program when the user selects the exit option.

**CODE SNIPPET:**

class BiometricData:

def \_\_init\_\_(self, id\_number,name, age, DoB, nationality, biometric\_type, visa\_number):

self.name = name

self.age = age

self.DoB = DoB

self.nationality = nationality

self.id\_number = id\_number

self.biometric\_type = biometric\_type

self.visa\_number = visa\_number

class BiometricDataManager:

def \_\_init\_\_(self, file\_path):

self.data = {}

self.enroll\_data = []

self.file\_path = file\_path

self.load\_data\_from\_file()

def load\_data\_from\_file(self):

try:

with open(self.file\_path, 'r') as file:

for line in file:

data = line.strip().split(',')

id\_number = data[0]

name = data[1]

age = int(data[2])

DoB = data[3]

nationality = data[4]

biometric\_type = data[5]

visa\_number = data[6]

self.data[id\_number] = BiometricData(id\_number,name, age, DoB, nationality, biometric\_type, visa\_number)

except FileNotFoundError:

print("Data file not found. Starting with an empty database.")

def save\_data\_to\_file(self):

with open(self.file\_path, 'w') as file:

for data in self.data.values():

file.write(f"{data.id\_number},{data.name},{data.age},{data.DoB},{data.nationality},{data.biometric\_type},{data.visa\_number}\n")

def create\_Data(self, Data):

if Data.id\_number not in self.data:

self.data[Data.id\_number] = Data

self.save\_data\_to\_file()

print(f"Biometric data added for {Data.name}")

else:

print(f"id number {Data.id\_number} already exists in the database.")

def delete\_Data(self, id\_number):

if id\_number in self.data:

del self.data[id\_number]

self.save\_data\_to\_file()

print("Biometric data deleted successfully.")

else:

print("id number not found in the database.")

def update\_Data(self, id\_number):

if id\_number in self.data:

old\_data = self.data[id\_number]

name = input("Enter new name (leave it blank to keep it as it is): ")

age = input("Enter new age: ")

DoB = input("Enter new DoB(DD-MM-YY): ")

nationality = input("Enter new nationality: ")

biometric\_type = input("Enter new biometric\_type: ")

visa\_number = input("Enter new visa number: ")

if name == "":

name = old\_data.name

if age == "":

age = old\_data.age

else:

age = int(age)

if DoB == "":

DoB = old\_data.DoB

if nationality == "":

nationality = old\_data.nationality

if biometric\_type == "":

biometric\_type = old\_data.biometric\_type

if visa\_number == "":

visa\_number = old\_data.visa\_number

new\_Data = BiometricData(id\_number,name, age, DoB, nationality, biometric\_type, visa\_number)

self.data[id\_number] = new\_Data

self.save\_data\_to\_file()

print("Biometric data updated successfully.")

else:

print("id number not found in the database.")

def verify\_Data(self, id\_number):

if id\_number in self.data:

print("Biometric Data verified successfully.")

else:

print("Biometric Data not found in the database.")

def display\_all(self):

if self.data:

for id\_number, Data in self.data.items():

print(f"ID Number: {Data.id\_number}")

print(f"Name: {Data.name}")

print(f"Age: {Data.age}")

print(f"DoB: {Data.DoB}")

print(f"biometric\_type: {Data.biometric\_type}")

print(f"nationality: {Data.nationality}")

print(f"Visa Number: {Data.visa\_number}")

print()

else:

print("There is no Data to be displayed")

def manage\_enroll\_data(self, enroll\_data):

self.enroll\_data.append(enroll\_data)

print(f"Enroll data added for {enroll\_data.name}")

def main():

file\_path = "biometric\_data.txt"

manager = BiometricDataManager(file\_path)

while True:

print("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("\n~~~~~~~~Menu:~~~~~~~~~")

print("1. Create biometric data")

print("2. Delete biometric data")

print("3. Update biometric data")

print("4. Verify biometric data")

print("5. Display All")

print("6. Manage Enroll Data")

print("7. Exit")

choice = input("Enter your choice: ")

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

if choice == '1':

id\_number = input("Enter ID number: ")

name = input("Enter name: ")

age = int(input("Enter age: "))

DoB = input("Enter DoB(DD-MM-YY): ")

nationality = input("Enter nationality: ")

biometric\_type = input("Enter biometric\_type: ")

visa\_number = input("Enter visa number: ")

Data = BiometricData( id\_number,name, age, DoB, nationality, biometric\_type, visa\_number)

manager.create\_Data(Data)

elif choice == '2':

id\_number = input("Enter id number to delete: ")

manager.delete\_Data(id\_number)

elif choice == '3':

id\_number = input("Enter ID number you want to update: ")

manager.update\_Data(id\_number)

elif choice == '4':

id\_number = input("Enter id number to verify: ")

manager.verify\_Data(id\_number)

elif choice == '5':

manager.display\_all()

elif choice == '6':

id\_number=int(input("Enter id :"))

name = input("Enter name: ")

age = int(input("Enter age: "))

DoB = input("Enter DoB(DD-MM-YY): ")

nationality = input("Enter nationality: ")

biometric\_type = input("Enter biometric\_type: ")

visa\_number = input("Enter visa number: ")

enroll\_data = BiometricData(id\_number,name, age, DoB, nationality,biometric\_type, visa\_number)

manager.manage\_enroll\_data(enroll\_data)

elif choice == '7':

print("Exiting...")

break

else:

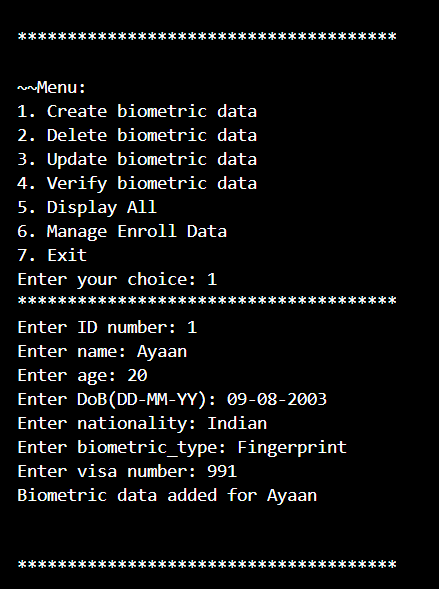
print("Invalid choice. Please enter a valid option.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**CHAPTER-7**

**OUTPUT**

****

**Fig 1. Create biometric data**

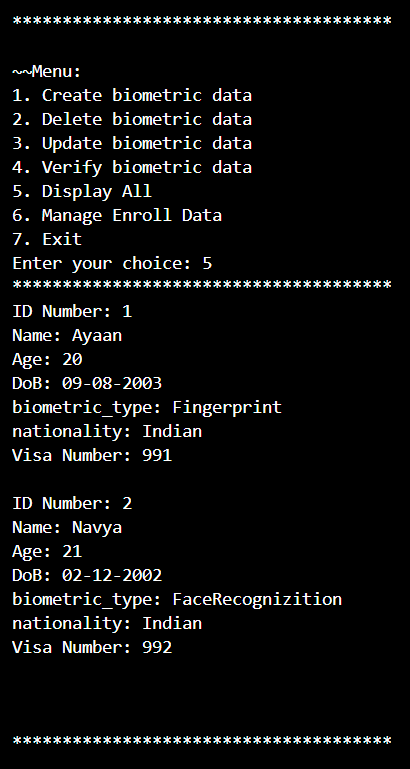


Fig 2. Display biometric data

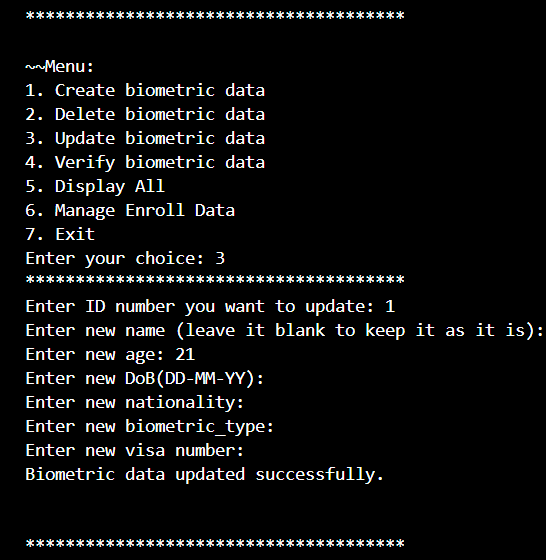


Fig 3. Update biometric data(case 1)

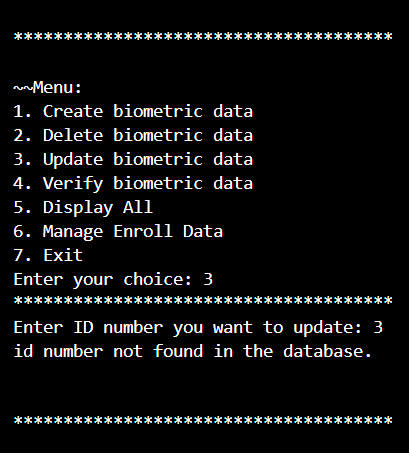


Fig 2.2. Update biometric data(case 2)

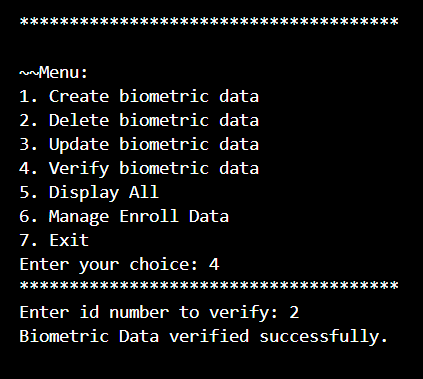


Fig 3. Verify biometric data(case 1)

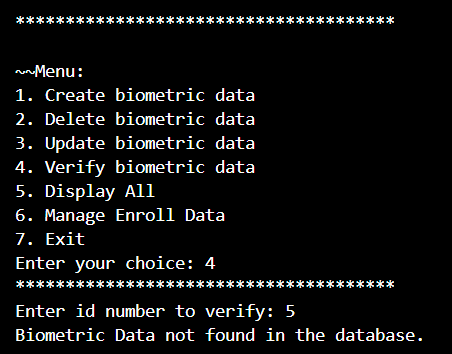


Fig 3.1. Verify biometric data(case 2)



Fig 4. Delete biometric data



Fig 5. Manage Enroll Data

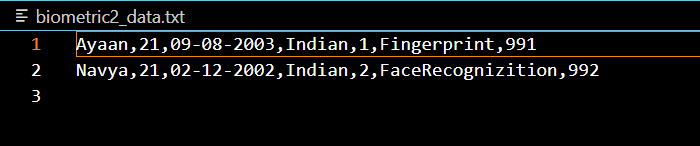


Fig 6. Data stored in the txt file.

**CHAPTER- 8**

**CONCLUSION**

Biometric Data Management systems are critical tools for ensuring secure and efficient identity verification across various sectors. They offer enhanced security through unique biometric traits, increased convenience for users, and robust fraud prevention measures. By complying with regulations and integrating scalable solutions, these systems contribute to improved access control, streamlined processes, and heightened data protection. The provided code exemplifies a foundational structure for managing biometric data, demonstrating key functionalities such as creation, deletion, updating, verification, and display, laying the groundwork for more sophisticated applications in diverse industries.

By following these guidelines, you can create a robust Proof of Concept for a Biometric Data Management System that supports CRUD operations, biometric enrollment management, and verification of biometric data. This POC will serve as a foundation for further development and refinement of the full-scale system.

**CHAPTER-9**

**REFERENCE**

* <https://medium.com/spidernitt/an-implementation-of-fingerprint-detection-with-python-f143d20c3a96>
* <https://vadic.vigyanashram.blog/2021/01/29/biometric-attendance-system-using-opencv-and-python/>
* <https://chat.openai.com/c/a20aaac4-19d1-4a0f-bf34-46de8837442c>
* Google , class notebook.