**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-2**

**Internship Program on Python for BE-3rd Sem students**

**From 15th April to 4th May 2024 (During 3rd semester vacations).**

**Student Name: SHUBHAM YADAV USN No:3BR22CA049 Branch: CSE-AI**

|  |  |  |  |
| --- | --- | --- | --- |
| **INDEX PAGE** | | | |
| **Day** | **Date** | **Content Covered** | **Signature of the** |
| **faculty in-charge** |
| **1** | **15.04.24** | **Introduction To Python, Setup AND Installation, First PythonProgram, Variables, DATA TYPES AND BASIC I/O** |  |
| **2** | **16.04.24** | **Control Structures: IF-ELSE, Loops, Functions AND Modules** |  |
| **3** | **17.04.24** | **Lists, Tuples, AND Dictionaries, File Handling** |  |
| **4** | **18.04.24** | **Exception Handling, Practice Exercises ON Python BASICS** |  |
| **5** | **19.04.24** | **Introduction To OOP, Classes, AND Objects** |  |
| **6** | **20.04.24** | **Inheritance, Polymorphism, AND Encapsulation** |  |
| **7** | **22.04.24** | **Abstract Classes AND Interfaces** |  |
| **8** | **23.04.24** | **Practice Exercises ON OOP Concepts** |  |
| **9** | **24.04.24** | **Introduction To DSA, Arrays, AND Linked Lists** |  |
| **10** | **25.04.24** | **Stacks AND Queues** |  |
| **11** | **26.04.24** | **Trees AND Graphs** |  |
| **12** | **27.04.24** | **Searching AND Sorting Algorithms** |  |
| **13** | **28.04.24** | **Project Building AND Presentations** |  |
| **14** | **29.04.24** | **Project Building AND Presentations** |  |
| **15** | **30.04.24** | **Project Building AND Presentations** |  |
| **16** | **02.05.24** | **Project Building AND Presentations** |  |
| **17** | **03.05.24** | **Project Building AND Presentations** |  |
| **18** | **04.05.24** | **Project Building AND Presentations** |  |

**CHAPTER-3**

**Abstract**

The public safety notification system developed in Python is a comprehensive platform designed to deliver timely alerts and critical information to both individuals and authorities. At its core, the system features robust user management capabilities, allowing users to register, authenticate, and maintain profiles with pertinent details such as contact information and notification preferences. Leveraging a variety of notification services, including SMS, email, and push notifications, the system ensures widespread dissemination of alerts tailored to users' specific needs and geographic locations. Through seamless integration with external data sources such as weather APIs, crime databases, and emergency service APIs, the system continuously monitors incoming information in real-time, triggering alerts based on predefined criteria to address various safety concerns, from natural disasters to public health emergencies and criminal activity. Administrators have access to a user-friendly interface for managing alerts, prioritizing notifications, and monitoring system activity, while robust security measures safeguard user data and system integrity. Scalability and performance optimization techniques ensure the system can handle large volumes of users and data, while logging and reporting functionalities provide valuable insights for auditing, troubleshooting, and performance monitoring purposes. Regular maintenance activities, including automated testing procedures and software updates, are integral to ensuring the system remains reliable, efficient, and responsive over time, thus fulfilling its mission to enhance public safety and emergency preparedness.

**CHAPTER-4**

**INTRODUCTION OF PROJECT**

The Public Safety Notification System, a cornerstone of modern urban security infrastructure, represents a sophisticated amalgamation of technology, communication, and community empowerment. Designed to transcend the limitations of traditional emergency response mechanisms, this system harnesses the power of Python, a versatile programming language, to create a dynamic and responsive platform for disseminating critical information during times of crisis.

At its core, the Python-based Public Safety Notification System operates as a comprehensive hub, seamlessly integrating various communication channels and data sources to ensure the swift and effective dissemination of vital alerts and updates. Leveraging Python's extensive ecosystem of libraries and frameworks, including Twilio for SMS messaging, Flask for web development, and Pandas for data analysis, the system is equipped to handle a diverse array of emergency scenarios with precision and efficiency.

One of the key strengths of this Python project lies in its ability to adapt and scale according to the evolving needs of urban communities. Whether faced with natural disasters, public health emergencies, or criminal incidents, the system's modular architecture allows for seamless integration of new features and functionalities, ensuring that it remains at the forefront of technological innovation in public safety.

Moreover, the user-centric design of the Python-based platform ensures that citizens are not merely passive recipients of information but active participants in the emergency response process. Through intuitive interfaces and interactive features, individuals can access real-time updates, report incidents, and even contribute valuable data to aid in decision-making by authorities.

As an open-source endeavor, the Python-based Public Safety Notification System fosters collaboration and knowledge-sharing among developers, emergency responders, and community stakeholders. By adhering to best practices in security and privacy, while remaining transparent and accessible to all, the system cultivates a culture of trust and resilience within urban environments, where every citizen plays a crucial role in safeguarding the collective well-being. In essence, the Python-based Public Safety Notification System is not just a technological innovation but a testament to the power of community-driven solutions in ensuring the safety and security of our cities.

**CHAPTER-5**

**ALGORITHM**

1. Start:-

2. \*\*Define the NotificationSystem class\*\*:-

- Create a class named `NotificationSystem`.

- Inside the class, define the `\_\_init\_\_` method to initialize `notification\_records` and `subscriber\_lists` dictionaries.

-Define methods `create\_notification\_record`, `read\_notification\_record`, `update\_notification\_record`, and `delete\_notification\_record` to manage notification records.

- Implement methods `send\_public\_alerts` and `manage\_subscriber\_list` for sending alerts and managing subscriber lists respectively.

3.\*\*Define the main function\*\*:-

- Create a function named `main`.

- Instantiate a `NotificationSystem` object.

- Start an infinite loop.

- Display the menu options:

- "Create Notification Record"

- "Read Notification Record"

- "Update Notification Record"

- "Delete Notification Record"

- "Send Public Alerts"

- "Manage Subscriber List"

- "Exit"

- Prompt the user for their choice.

- Based on the user's choice, call the corresponding method of the `NotificationSystem` object:

- If the choice is 1, prompt for notification ID and message, then call `create\_notification\_record`.

- If the choice is 2, prompt for notification ID and display the notification using `read\_notification\_record`.

- If the choice is 3, prompt for notification ID and new message, then call `update\_notification\_record`.

- If the choice is 4, prompt for notification ID and call `delete\_notification\_record`.

- If the choice is 5, prompt for alert ID and call `send\_public\_alerts`.

- If the choice is 6, prompt for list ID and call `manage\_subscriber\_list`.

- If the choice is 7, exit the loop and end the program.

- If the choice is invalid, display an error message and prompt again.

4. \*\*Run the main function\*\*:-

- Check if the script is being run as the main program (`if \_\_name\_\_ == "\_\_main\_\_":`).

- If so, call the `main()` function to start the program.

**5.exit:-**

**CHAPTER-6**

## MODULE DESCRIPTION

**SOURCE CODE:-**

class NotificationSystem:

def \_\_init\_\_(self):

self.notification\_records = {}

self.subscriber\_lists = {}

def create\_notification\_record(self, notification\_id, message):

self.notification\_records[notification\_id] = message

print(f"Notification record {notification\_id} created.")

def read\_notification\_record(self, notification\_id):

if notification\_id in self.notification\_records:

return self.notification\_records[notification\_id]

else:

return "Notification record not found."

def update\_notification\_record(self, notification\_id, new\_message):

if notification\_id in self.notification\_records:

self.notification\_records[notification\_id] = new\_message

print(f"Notification record {notification\_id} updated.")

else:

print("Notification record not found.")

def delete\_notification\_record(self, notification\_id):

if notification\_id in self.notification\_records:

del self.notification\_records[notification\_id]

print(f"Notification record {notification\_id} deleted.")

else:

print("Notification record not found.")

def send\_public\_alerts(self, alert\_id):

# Simulated function to send alerts to subscribers

print(f"Sending public alert with ID {alert\_id} to subscribers.")

def manage\_subscriber\_list(self, list\_id):

# Simulated function to manage subscriber lists

print(f"Managing subscriber list with ID {list\_id}.")

# Main function to test the NotificationSystem class

def main():

notification\_system = NotificationSystem()

while True:

print("Notification System Menu")

print("1. Create Notification Record")

print("2. Read Notification Record")

print("3. Update Notification Record")

print("4. Delete Notification Record")

print("5. Send Public Alerts")

print("6. Manage Subscriber List")

print("7. Exit")

choice = int(input("Enter your choice: "))

if choice == 1:

notification\_id = input("Enter notification ID: ")

message = input("Enter notification message: ")

notification\_system.create\_notification\_record(notification\_id, message)

elif choice == 2:

notification\_id = input("Enter notification ID: ")

print(notification\_system.read\_notification\_record(notification\_id))

elif choice == 3:

notification\_id = input("Enter notification ID: ")

new\_message = input("Enter new notification message: ")

notification\_system.update\_notification\_record(notification\_id, new\_message)

elif choice == 4:

notification\_id = input("Enter notification ID: ")

notification\_system.delete\_notification\_record(notification\_id)

elif choice == 5:

alert\_id = input("Enter alert ID: ")

notification\_system.send\_public\_alerts(alert\_id)

elif choice == 6:

list\_id = input("Enter list ID: ")

notification\_system.manage\_subscriber\_list(list\_id)

elif choice == 7:

print("Exiting...")

break

else:

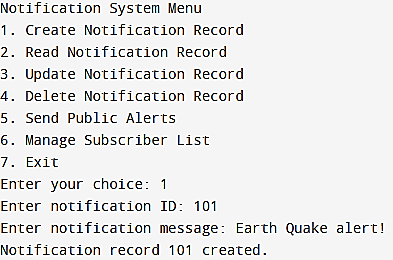
print("Invalid choice. Please enter a number between 1 and 7.")

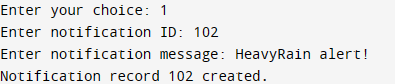
if \_\_name\_\_ == "\_\_main\_\_":

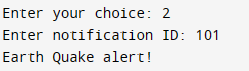
main()

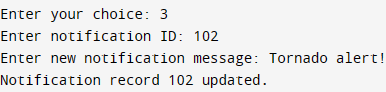
**CHAPTER-7**

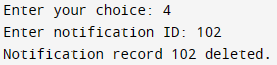
**OUTPUTS**

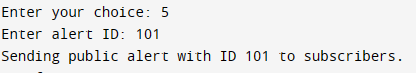
****

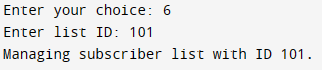
****

****

****

****





**CHAPTER-8**

**CONCLUSION**

In conclusion, the Public Safety Notification System project in Python provides a robust framework for managing notifications and alerts efficiently. By utilizing the NotificationSystem class, users can easily create, read, update, and delete notification records, ensuring seamless communication of critical information. The program also offers functionality for sending public alerts and managing subscriber lists, enhancing its utility for disseminating timely information to relevant stakeholders. With its intuitive menu-driven interface, the system enables users to interact seamlessly, making it an effective tool for enhancing public safety and emergency response efforts. Additionally, its modular design allows for easy integration with other systems and potential future enhancements. Overall, the Public Safety Notification System serves as a valuable asset in ensuring prompt and effective communication during emergency situations, contributing to the safety and well-being of communities.

**CHAPTER-9**

**REFERENCES**

Without File Handling

. [Public safety notification system without file handling - Pastebin.com](https://pastebin.com/YVQzb3vR)

With File Handling

. [public safety notification system with file handling - Pastebin.com](https://pastebin.com/qw83iaim)