**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

**ABSTRACT**

* The Lost and Found Management System Proof of Concept (POC) is designed to address the challenges associated with managing lost and found items in various settings such as educational institutions, corporate offices, public spaces, and more. The system provides a digital platform for efficiently logging, tracking, and retrieving lost items, thereby streamlining the overall process and enhancing user experience.
* The primary objective of the system is to create a centralized repository where users can report lost items, provide relevant descriptions, and search for their lost belongings. The POC demonstrates key functionalities including user registration, item logging, search capabilities, and administrative features.
* Users interact with the system through an intuitive web interface or mobile application. Upon encountering a lost item, users can easily submit details such as item name, description, location found, and contact information. The system then generates a unique identifier for each logged item, facilitating easy tracking and retrieval.
* Administrators, typically designated personnel responsible for managing lost and found operations, have access to additional features such as item categorization, advanced search filters, and reporting tools. They can efficiently monitor the status of lost items, coordinate retrieval efforts, and generate reports for analysis and audit purposes.
* The POC utilizes modern technologies such as cloud-based storage, relational databases, and responsive web design to ensure scalability, reliability, and accessibility across various devices and platforms. Integration with communication channels such as email notifications and SMS alerts further enhances user engagement and notification capabilities.
* Overall, the Lost and Found Management System POC serves as a robust solution to streamline the lost and found process, reduce administrative overhead, and improve user satisfaction. It lays the foundation for a comprehensive system that can be further developed and customized to meet the specific needs of diverse organizations and environments.

**INTRODUCTION**

* The Lost and Found Management System Proof of Concept (POC) for theaters presents an innovative approach to streamline the handling of lost items within theatrical venues.
* In bustling theater environments, instances of patrons misplacing personal belongings during performances are not uncommon, necessitating an organized method for managing and returning lost items efficiently.
* This POC introduces a digital platform that leverages CRUD (Create, Read, Update, Delete) operations to simplify the management process.
* Through this system, theater staff can effortlessly log newly discovered lost items (Create), update item details as necessary (Update), retrieve item information when patrons inquire (Read), and remove items from the database once they are reclaimed by their owners (Delete).
* By implementing these CRUD operations, the system facilitates seamless item tracking and ensures that lost items are promptly reunited with their rightful owners.
* Utilizing modern technologies and user-friendly interfaces, the Lost and Found Management System POC aims to enhance operational efficiency and customer satisfaction within theaters.
* Theater staff can easily navigate the system to input, update, and retrieve information about lost items, ultimately providing a more streamlined and reliable experience for both patrons and staff members alike.
* This initiative not only optimizes the management of lost items but also reinforces the theater's commitment to providing a secure and organized environment for all attendees.
* The system will aim to streamline the process of reporting and retrieving lost items, ensuring a smooth and efficient experience for both staff and patrons.
* By implementing this POC, we can explore features such as a user-friendly interface for reporting lost items, categorizing and cataloging found items, and a seamless communication system between staff and patrons.
* With this system in place, we'll make sure that lost items are reunited with their owners in no time, adding an extra level of convenience and peace of mind to the theater experience. Let's dive in and create a top-notch lost and found management system for the theater!

**MODULE DESCRIPTION**

Import Manager and User classes from the Theatre module.

**Define main() Function:**

Define a function named main () which serves as the entry point of the program.

Instantiate Manager Object:

Create an instance of the Manager class and store it in the variable manager\_system.

Welcome Message:

Call the welcome\_user () method of the manager\_system to display a welcome message to the user.

**User Input:**

Prompt the user with the question "Are you a User or a Manager?" and store their response in the variable choice.

**Check User or Manager:**

Call the check\_user\_or\_manager () method of the manager\_system to determine if the user is a User or a Manager. Store the returned system instance in the variable system.

**User Interaction Loop:**

If the system instance is of type User, enter a loop to interact with the user.

Display options for adding, updating, or deleting items.

Based on the user's input, call the corresponding methods (add\_item, update\_item, delete\_item) of the User instance.

**Manager Interaction Loop:**

If the system instance is of type Manager, enter a loop to interact with the manager.

Display options for searching items, displaying all items, or exiting.

Based on the user's input, call the corresponding methods (search\_item, display\_all\_items) of the Manager instance.

**Execution:**

Check if the script is executed directly (\_name\_ == "\_main\_").

If true, call the main () function to start the program.

**ALGORITHM**

**Define Abstract Classes:** The code begins by defining abstract classes AbstractUser and AbstractManager using Python's Abstract Base Classes (ABC) module. These classes contain abstract methods that must be implemented by their subclasses.

**Implement User and Manager Classes:** The code then implements concrete classes User and Manager, which inherit from AbstractUser and AbstractManager, respectively. These classes provide functionality for adding, updating, and deleting lost items for users, as well as searching for and displaying lost items for managers.

**Main Function:** The main() function serves as the entry point of the program. It creates an instance of Manager, welcomes the user, prompts the user to choose between being a user or a manager, and then calls the check\_user\_or\_manager() method to determine the type of user.

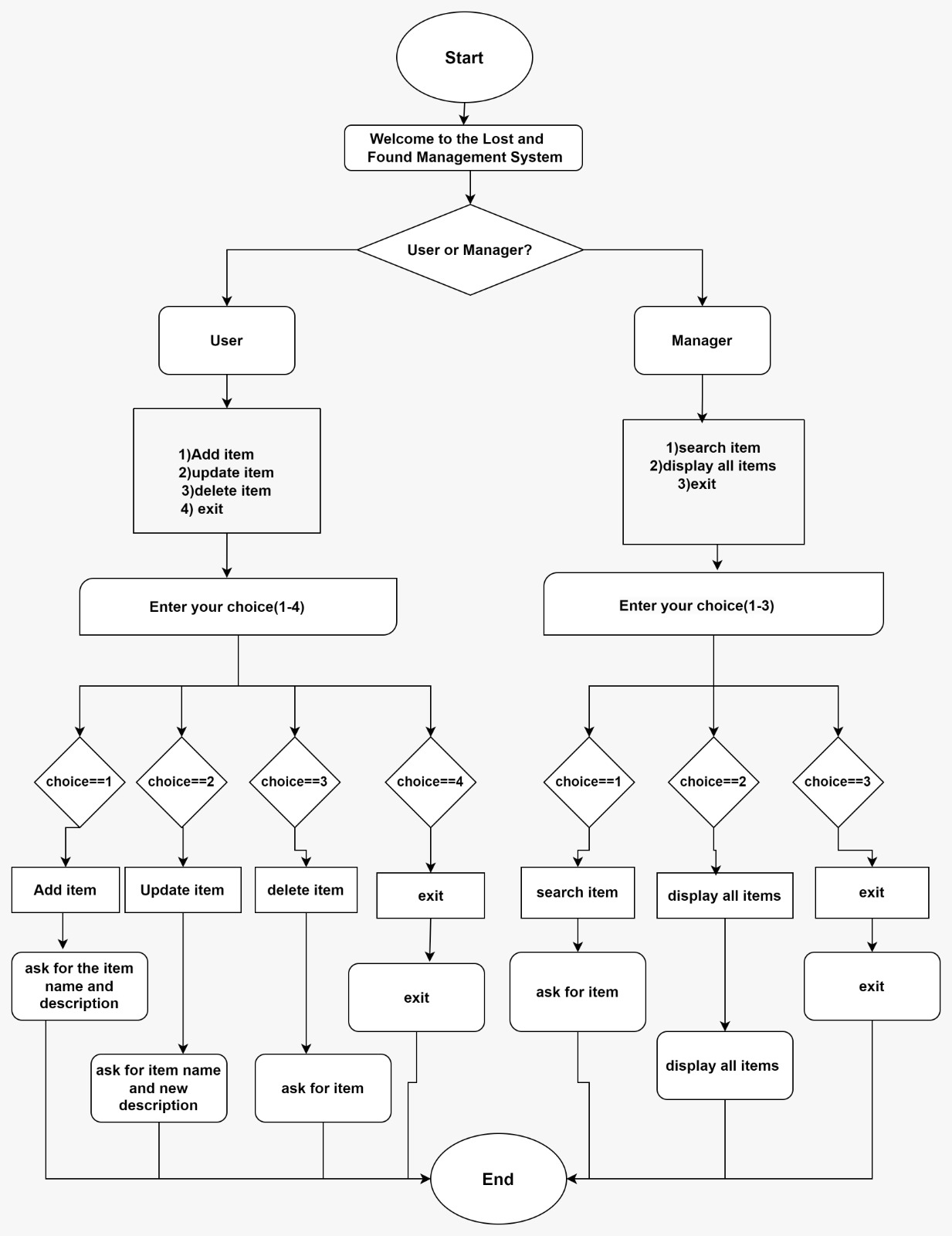
**User Interaction Loop:** If the user chooses to be a User, a loop is entered where the user is presented with options to add, update, or delete items. Depending on the chosen option, corresponding methods of the User class are called.

**Manager Interaction Loop:** If the user chooses to be a Manager, a loop is entered where the manager is presented with options to search for items, display all items, or exit. Depending on the chosen option, corresponding methods of the Manager class are called.

**Execution:** Finally, the main() function is called if the script is executed directly (\_name\_ == "\_main\_").

The code appears to be structured well and follows object-oriented principles. However, there's a small typo in the condition for checking if the script is executed directly. It should be \_name\_ == "\_main", not \_name == "main". Fixing this typo would ensure that the main() function is executed correctly when the script is run directly.

**FLOW CHART**

****

**IMPLEMENTATION**

**#Abstract.py**

from abc import ABC, abstractmethod

class AbstractUser (ABC):

@abstractmethod

def add\_item (self, item\_name, description):

pass

@abstractmethod

def update\_item (self, item\_name, new\_description):

pass

@abstractmethod

def delete\_item (self, item\_name):

pass

class AbstractManager (ABC):

@abstractmethod

def search\_item (self, user, item\_name):

pass

@abstractmethod

def display\_all\_items (self, user):

pass

@abstractmethod

def welcome\_user(self):

pass

@abstractmethod

def check\_user\_or\_manager (self, choice):

pass

**#Theatre.py**

from AbstractTheatre import AbstractUser, AbstractManager

class User(AbstractUser):

def \_init\_(self):

self.lost\_items = {"phone": "a black oneplus", "wallet": "a black leather wallet", "keys": "a set of house keys","SunGlasses":"Photochromatic","Water Bottle":"Milton"}

def add\_item(self, item\_name, description):

self.lost\_items[item\_name] = description

print("Item added successfully.")

def update\_item(self, item\_name, new\_description):

if item\_name in self.lost\_items:

self.lost\_items[item\_name] = new\_description

print("Item updated successfully.")

else:

print("Item not found.")

def delete\_item(self, item\_name):

if item\_name in self.lost\_items:

del self.lost\_items[item\_name]

print("Item deleted successfully.")

else:

print("Item not found.")

class Manager(AbstractManager):

def \_init\_(self):

self.found\_item={"phone": "a black oneplus", "wallet": "a black leather wallet", "keys": "a set of house keys","SunGlasses":"Photochromatic","Water Bottle":"Milton"}

pass

def search\_item(self, user, item\_name):

if item\_name in user.lost\_items:

print(f"Item: {item\_name}")

print(f"Description: {user.lost\_items[item\_name]}")

else:

print("Item not found.")

def display\_all\_items(self, user):

print("Lost Items:")

for item, description in user.lost\_items.items():

print(f"Item: {item} | Description: {description}")

def welcome\_user(self):

print("Welcome to the Lost and Found Management System.")

def check\_user\_or\_manager(self, choice):

if choice.lower() == "user":

return User()

elif choice.lower() == "manager":

return Manager()

else:

print("Invalid choice.")

return None

**#Main.py**

from Theatre import Manager, User

def main():

manager\_system = Manager()

manager\_system.welcome\_user()

choice = input("Are you a User or a Manager? ")

system = manager\_system.check\_user\_or\_manager(choice)

if isinstance(system, User):

while True:

print("\n1. Add Item")

print("2. Update Item")

print("3. Delete Item")

print("4. Print Lost Items") # Add option to print lost items

print("5. Exit")

option = input("Select an option: ")

if option == "1":

item\_name = input("Enter item name: ")

description = input("Enter item description: ")

system.add\_item(item\_name, description)

elif option == "2":

item\_name = input("Enter item name to update: ")

new\_description = input("Enter new description: ")

system.update\_item(item\_name, new\_description)

elif option == "3":

item\_name = input("Enter item name to delete: ")

system.delete\_item(item\_name)

elif option == "4":

print("Lost Items:")

for item, description in system.lost\_items.items(): # Print lost items

print(f"Item: {item} | Description: {description}")

elif option == "5":

print("Exit")

break

else:

print("Invalid option.")

elif isinstance(system, Manager):

user = User()

while True:

print("\n1. Search Item")

print("2.Display All Items")

print("3. Exit")

option = input("Select an option: ")

if option == "1":

item\_name = input("Enter item name to search: ")

system.search\_item(user, item\_name)

elif option == "2":

system.display\_all\_items(user)

elif option == "3":

print("Exit")

break

else:

print("Invalid option.")

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

main()

**OUTPUT:**

**USER OUTPUT**

Welcome to the Lost and Found Management System.

Are you a User or a Manager? user

1. Add Item

2. Update Item

3. Delete Item

4. Print Lost Items

5. Exit

Select an option: 1

Enter item name: wallet

Enter item description: a black leather wallet

Item added successfully.

1. Add Item

2. Update Item

3. Delete Item

4. Print Lost Items

5. Exit

Select an option: 2

Enter item name to update: bag

Enter new description: sky

Item not found.

1. Add Item

2. Update Item

3. Delete Item

4. Print Lost Items

5. Exit

Select an option: 2

Enter item name to update: keys

Enter new description: car key

Item updated successfully.

1. Add Item

2. Update Item

3. Delete Item

4. Print Lost Items

5. Exit

Select an option: 3

Enter item name to delete: phone

Item deleted successfully.

1. Add Item

2. Update Item

3. Delete Item

4. Print Lost Items

5. Exit

Select an option: 4

Lost Items:

Item: wallet | Description: a black leather wallet

Item: keys | Description: car key

Item: SunGlasses | Description: Photochromatic

Item: Water Bottle | Description: Milton

1. Add Item

2. Update Item

3. Delete Item

4. Print Lost Items

5. Exit

Select an option: 5

Exit

**MANAGER OUTPUT:**

Welcome to the Lost and Found Management System.

Are you a User or a Manager? manager

1. Search Item

2. Display All Items

3. Exit

Select an option: 1

Enter item name to search: purse

Item not found.

1. Search Item

2. Display All Items

3. Exit

Select an option: 1

Enter item name to search: Water Bottle

Item: Water Bottle

Description: Milton

1. Search Item

2. Display All Items

3. Exit

Select an option: 2

Lost Items:

Item: phone | Description: a black oneplus

Item: wallet | Description: a black leather wallet

Item: keys | Description: a set of house keys

Item: SunGlasses | Description: Photochromatic

Item: Water Bottle | Description: Milton

1. Search Item

2. Display All Items

3. Exit

Select an option: 3

Exit

**CONCLUSION**

This system offers a practical solution for organizations to effectively manage lost items and improve customer service. By automation, the system can streamline processes, reduce manual errors .The POC serves as a stepping stone towards the development of a fully functional lost and found management system that can benefit a wide range of industries and settings.

**REFERENCE**

1. <https://chat.openai.com/>
2. <https://iask.ai/>
3. <https://www.perplexity.ai/>