BASAVARAJESWARI GROUP OF INSTITUTIONS

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

AUTONOMOUS INSTITUTE UNDER VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI 590018

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Internship Report on

"SPORTS EVENT MARKETING PLANNER"

For the course: Python

Name:	USN:
UMESH.P	3BR23EC175
YASHWANTH SUHAS. SS	3BR23EC187
YASHWANTH.R	3BR23EC186
BALAJI NAGA V. N	3BR23EC022
V. AJAY KUMAR	3BR23EC176

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

NACC Accredited institution*

(Recognised by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belgavi)

"Jnana Gangotri" Campus, No.873/2, Ballari - Hospet Road, Allipur, Ballari-583104

Karnataka, India.

Ph: 08392-237100/23719, Fax:08392-237197

2023-2024

Introduction

The Sports Marketing System is a basic application that allows sports marketers to manage marketing campaigns for sports products like bikes, racing cars, and sports equipment. This system uses Object-Oriented Programming (OOP) concepts in Python to perform CRUD (Create, Read, Update, Delete) operations on marketing campaigns. The application is simple yet effective in handling multiple campaigns during sports events, keeping track of the products, audience, promotions, and activities involved in each campaign. This system allows for easy management of marketing data without relying on external databases.

Problem Statement

Sports companies often need to manage multiple marketing campaigns during large events, targeting specific audiences with tailored promotions and activities. Managing this data manually or with spreadsheets can become cumbersome and error-prone, especially when handling several products, offers, and activities.

The problem is to create a system that:

- Simplifies the management of sports marketing campaigns.
- Allows marketers to create, view, update, and delete campaigns easily.
- Organizes the information effectively using a simple, user-friendly interface.

Objectives

The main objectives of the Sports Marketing System are:

1. **Efficient Campaign Management:** Provide an easy way to handle sports marketing campaigns using CRUD operations.

2. **No Database Dependency:** Store campaign data in memory during runtime to avoid database complexity.

3. **Object-Oriented Approach:** Implement OOP concepts for modular and reusable code.

4. **User-Friendly Interface:** Allow marketers to interact with the system via a simple command-line interface

5. **Flexibility:** Enable easy updates and deletions of campaigns with minimal user effort.

Module Split-Up

1. Campaign Class:

Attributes:

- campaign_id
- product
- audience
- promotion
- activity

Methods:

- __init__: Initializes the campaign attributes.
- display: Displays the campaign details.

2. Marketing System Class:

Attributes:

- campaigns (list to store campaign objects)

Methods:

- **Create_campaign:** Takes user input to create a new campaign and adds it to the list.
- **read_campaigns:** Displays all campaigns in the

list.

- **Update_campaign:** Updates the details of an existing campaign by Campaign ID.
- **Delete_campaign:** Removes a campaign from the list by Campaign ID.
- **Run:** Handles the menu and user input to call respective CRUD methods.

Algorithm

Step-by-step Algorithm for CRUD Operations:

1. Start the Application:

- Display the main menu options (Create Campaign, Read Campaigns, Update Campaign, Delete Campaign, Exit).

2. Create Campaign:

- Accept user input for Campaign ID, Product, Audience, Promotion, and Activity.
- Create a Campaign object and add it to the list of

campaigns.

- Display success message.

3. Read Campaigns:

- Check if there are any campaigns stored.
- If yes, iterate through the list of Campaign objects and display their details.
- If no, show "No campaigns available."

4. Update Campaign:

- Ask the user for the Campaign ID.
- Search for the campaign in the list of campaigns by matching the ID.
- If found, allow the user to modify the product, audience, promotion, or activity.
- Save the updated details and display success message.

5. Delete Campaign:

- Ask the user for the Campaign ID.

- Search for the campaign and if found, remove it from the list.
- Display a success message or an error if the campaign is not found.

6. Exit:

- If the user selects "Exit," terminate the application.

PROGRAM:

```
class Campaign:
    def __init__(self, campaign_id, product,
audience, promotion, activity):
        self.campaign_id = campaign_id
        self.product = product
        self.audience = audience
        self.promotion = promotion
        self.activity = activity

    def display(self):
        print(f"Campaign ID: {self.campaign_id}")
        print(f"Product: {self.product}")
        print(f"Audience: {self.audience}")
        print(f"Promotion: {self.promotion}")
        print(f"Activity: {self.activity}")
        print('------------------------')


class MarketingSystem:
```

```
    def __init__(self):
        self.campaigns = []

    def create_campaign(self):
        campaign_id = input("Enter Campaign ID: ")
        product = input("Enter Product: ")
        audience = input("Enter Audience: ")
        promotion = input("Enter Promotion: ")
        activity = input("Enter Activity: ")

        # Create a new Campaign object
        campaign = Campaign(campaign_id, product,
audience, promotion, activity)
        self.campaigns.append(campaign)
        print("Campaign created successfully!")

    def read_campaigns(self):
        if not self.campaigns:
            print("No campaigns available.")
        else:
```

```
            for campaign in self.campaigns:
                campaign.display()

    def update_campaign(self):
        campaign_id = input("Enter Campaign ID to update: ")
        for campaign in self.campaigns:
            if campaign.campaign_id == campaign_id:
                print("Leave blank if no change is required.")
                campaign.product = input("Enter new Product: ") or campaign.product
                campaign.audience = input("Enter new Audience: ") or campaign.audience
                campaign.promotion = input("Enter new Promotion: ") or campaign.promotion
                campaign.activity = input("Enter new Activity: ") or campaign.activity
                print("Campaign updated successfully!")
                return
```

```
        print("Campaign ID not found.")

    def delete_campaign(self):
        campaign_id = input("Enter Campaign ID to
delete: ")
        for campaign in self.campaigns:
            if campaign.campaign_id == campaign_id:
                self.campaigns.remove(campaign)
                print(f"Campaign {campaign_id} deleted
successfully.")
                return
        print("Campaign ID not found.")

    def run(self):
        while True:
            print("\n--- Sports Marketing System ---")
            print("1. Create Campaign")
            print("2. Read Campaigns")
            print("3. Update Campaign")
            print("4. Delete Campaign")
```

```
            print("5. Exit")
            choice = input("Choose an option (1-5): ")

            if choice == '1':
                self.create_campaign()
            elif choice == '2':
                self.read_campaign()
            elif choice == '3':
                self.update_campaign()
            elif choice == '4':
                self.delete_campaign()
            elif choice == '5':
                print("Exiting the system.")
                break
            else:
                print("Invalid choice. Please try again.")


if __name__ == "__main__":
    system = MarketingSystem()
```

Conclusion:

The Sports Marketing System allows sports marketers to manage their campaigns effectively using a simple and intuitive interface. By leveraging Object-Oriented Programming concepts, the system maintains modularity, encapsulation, and ease of use. The application handles basic CRUD operations for sports marketing campaigns without the need for a database. Although the data is not stored persistently, the system provides an efficient way to manage campaigns during a session.

Future Enhancements

There are several ways to enhance this system to make it more robust and feature-rich:

- **Search Functionality:**

Implement search options to find campaigns based on product name, audience, or promotion.

- **Enhanced User Interface:**

Build a graphical user interface (GUI) using libraries like Tkinter or PyQt for better user experience.