KISHKINDA UNIVERSITY,

Ballari

Mini project Report

On

"Sports Event Marketing Planner"

Department of CSE

Submitted by:

Kannika **KUB23CSE061**

Table of Contents

1. INTRODUCTION

1.1 Background Information of the project

1.1.1 Problem Statement:

To address the problem statement "Sports Event Marketing Planner POC" using Python programming, we'll follow the guidelines to create an appropriate system. The system will include classes representing the main entities, specifically handling marketing campaigns with functionalities such as create, read, update, and delete (CRUD) operations. Additionally, methods to plan marketing strategies for specific sports events and track the campaigns' effectiveness will be implemented.

1.1.2 Proposed system:

- Centralized database: Store campaign data, strategies, and effectiveness metrics in a structured database (e.g., PostgreSQL, SQLite).
- Object-Oriented Programming (OOP): Use Python classes to represent marketing campaigns, strategies, and effectiveness data, promoting modularity and reusability.
- CRUD operations: Implement Create, Read, Update, and Delete functionalities for marketing campaigns.
- Marketing strategy planning: Allow users to define and execute marketing strategies tailored to specific sports events.
- Campaign effectiveness tracking: Track key metrics (e.g., impressions, clicks, conversions) and analyze campaign performance.

1.1.3 Scope of the project:

1. CRUD Operations:

- The program will include the ability to Create, Read, Update, and Delete (CRUD) marketing campaigns and related entities.
- This allows for dynamically managing marketing campaigns for different sports events, modifying details as needed, and removing outdated or irrelevant campaigns.
- Example methods might include:

- **create_ campaign()**

2.Marketing Strategy Planning:

- Methods to plan marketing strategies for specific sports events will be implemented, including defining target audiences, budgeting, and setting timelines.
- The system can generate marketing suggestions based on event details (e.g., audience demographics or regional preferences).

3. Campaign Effectiveness Tracking:

- The program will track key performance indicators (KPIs) such as reach, engagement, and conversions.
- The Campaign Tracker class will maintain information such as:
- reach
- click-through-rate
- conversion_ rate
- It will include methods to update and analyze the effectiveness of the campaigns, providing insights for future planning.

4. Event-Specific Customization:

- The system will be able to handle different types of sports events and their unique marketing needs.
- The planner will offer flexibility to adapt to small-scale local events or large-scale international sports competitions.

5.User Interface (UI) or Command-line Interface (CLI):

- The program could be built with a simple command-line interface (CLI) where users input data for the campaigns and events, or it could eventually evolve into a web-based or graphical UI for easier us
-

□

```
print(planner.delete_campaign(1))
```

1.1.4 Limitations of the project:

1. No Real-time Data Integration:

- The system may not connect to live data sources like real-time analytics or social media feeds. Instead, campaign effectiveness would have to be tracked using manually entered or static data.

2. Simplistic Data Models:

- The program's data model may be simple and may not account for complex marketing metrics (e.g., multivariate analysis, machine learning-based predictions).
- Budget forecasting, audience segmentation, and advanced targeting techniques may be limited in scope.

3. Limited External Integrations:

- The proof of concept (POC) will likely not integrate with external advertising platforms (e.g., Facebook Ads, Google Ads) or CRM systems, meaning campaign tracking and execution would need to be manual.

4.Scalability Constraints:

- This POC is designed for small- to medium-scale operations. If intended for large-scale sports events with massive audiences, the system might need optimization for performance and scalability.

5. No AI-Driven Recommendations:

- While the system may provide predefined strategies and basic suggestions, it will not include advanced AI-driven marketing strategies based on data insights.

6.Basic User Management:

- The POC may not implement complex user management systems such as role-based access control, meaning that it will be limited to a single user or administrative interface.

Code of the given statement:-

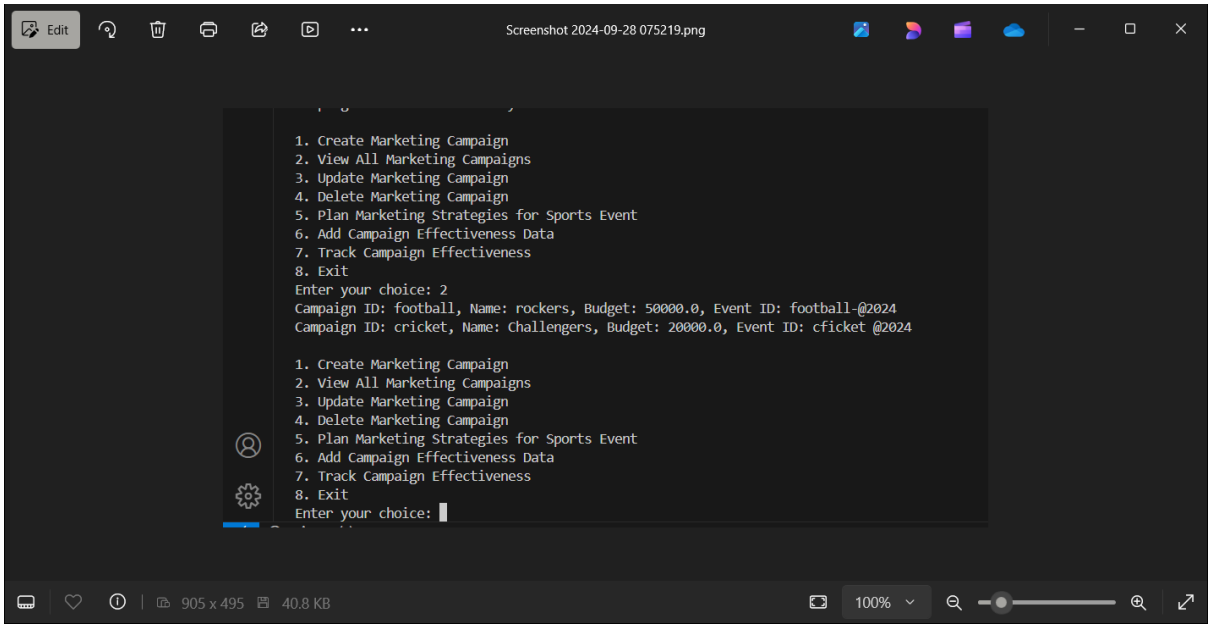
```
    class SportsEvent:
    def __init__(self, event_id, name, date, location):
        self.event_id = event_id
        self.name = name
        self.date = date
        self.location = location


class MarketingCampaign:
    campaigns = []


    def __init__(self, campaign_id, event_id, name, budget, start_date, end_date):
        self.campaign_id = campaign_id
        self.event_id = event_id
        self.name = name
        self.budget = budget
        self.start_date = start_date
        self.end_date = end_date


    # Create Campaign
    @classmethod
    def create_campaign(cls):
        campaign_id = input("Enter Campaign ID: ")
        event_id = input("Enter Event ID: ")
        name = input("Enter Campaign Name: ")
        budget = float(input("Enter Campaign Budget: "))
        start_date = input("Enter Campaign Start Date (YYYY-MM-DD): ")
        end_date = input("Enter Campaign End Date (YYYY-MM-DD): ")
        new_campaign = MarketingCampaign(campaign_id, event_id, name, budget, start_date,
end_date)
        cls.campaigns.append(new_campaign)
```

```
        print("Campaign Created Successfully!")

    # Read/View Campaigns
    @classmethod
    def read_campaigns(cls):
        if cls.campaigns:
            for campaign in cls.campaigns:
                print(f"Campaign ID: {campaign.campaign_id}, Name: {campaign.name}, Budget: {campaign.budget}, Event ID: {campaign.event_id}")
        else:
            print("No Campaigns Available")

    # Update Campaign
    @classmethod
    def update_campaign(cls, campaign_id):
        for campaign in cls.campaigns:
            if campaign.campaign_id == campaign_id:
                campaign.name = input(f"Enter new name (current: {campaign.name}): ")
                campaign.budget = float(input(f"Enter new budget (current: {campaign.budget}): "))
                campaign.start_date = input(f"Enter new start date (current: {campaign.start_date}): ")
                campaign.end_date = input(f"Enter new end date (current: {campaign.end_date}): ")
                print("Campaign Updated Successfully!")
                return
        print("Campaign Not Found")

    # Delete Campaign
    @classmethod
    def delete_campaign(cls, campaign_id):
        for campaign in cls.campaigns:
            if campaign.campaign_id == campaign_id:
```

```
                cls.campaigns.remove(campaign)
                print("Campaign Deleted Successfully!")
                return
        print("Campaign Not Found")
class CampaignMetrics:
    metrics = []
    def __init__(self, effectiveness_id, campaign_id, reach, engagement, conversions):
        self.effectiveness_id = effectiveness_id
        self.campaign_id = campaign_id
        self.reach = reach
        self.engagement = engagement
        self.conversions = conversions
    @classmethod
    def track_campaign_effectiveness(cls, effectiveness_id):
        for metric in cls.metrics:
            if metric.effectiveness_id == effectiveness_id:
                print(f"Effectiveness ID: {metric.effectiveness_id}")
                print(f"Campaign ID: {metric.campaign_id}")
                print(f"Reach: {metric.reach}")
                print(f"Engagement: {metric.engagement}")
                print(f"Conversions: {metric.conversions}")
                return
        print("Effectiveness Data Not Found")
    @classmethod
    def add_effectiveness_data(cls):
        effectiveness_id = input("Enter Effectiveness ID: ")
        campaign_id = input("Enter Campaign ID: ")
        reach = int(input("Enter Reach Count: "))
        engagement = int(input("Enter Engagement Count: "))
        conversions = int(input("Enter Conversion Count: "))
```

```
        new_metric = CampaignMetrics(effectiveness_id, campaign_id, reach, engagement, conversions)
        cls.metrics.append(new_metric)
        print("Campaign Effectiveness Data Added Successfully!")
def plan_marketing_strategies(event_id):
    print(f"Planning marketing strategy for Event ID: {event_id}")
    strategy = input("Enter the marketing strategy details: ")
    print(f"Strategy for Event ID {event_id}: {strategy}")
# Main Program Loop
def main():
    while True:
        print("\n1. Create Marketing Campaign")
        print("2. View All Marketing Campaigns")
        print("3. Update Marketing Campaign")
        print("4. Delete Marketing Campaign")
        print("5. Plan Marketing Strategies for Sports Event")
        print("6. Add Campaign Effectiveness Data")
        print("7. Track Campaign Effectiveness")
        print("8. Exit")
        choice = input("Enter your choice: ")
        if choice == '1':
            MarketingCampaign.create_campaign()
        elif choice == '2':
            MarketingCampaign.read_campaigns()
        elif choice == '3':
            campaign_id = input("Enter Campaign ID to update: ")
            MarketingCampaign.update_campaign(campaign_id)
        elif choice == '4':
            campaign_id = input("Enter Campaign ID to delete: ")
            MarketingCampaign.delete_campaign(campaign_id)
        elif choice == '5':
```

```
            event_id = input("Enter Event ID to plan strategy: ")
            plan_marketing_strategies(event_id)
        elif choice == '6':
            CampaignMetrics.add_effectiveness_data()
        elif choice == '7':
            effectiveness_id = input("Enter Effectiveness ID to track: ")
            CampaignMetrics.track_campaign_effectiveness(effectiveness_id)
        elif choice == '8':
            print("Exiting Program...")
            break
        else:
            print("Invalid Choice, Please Try Again")
if __name__ == "__main__":
    main()
```



2. OBJECTIVES:

2.1.1 Develop an Object-Oriented System for Marketing Campaigns:

- Implement Python classes to represent core entities such as SportsEvent, MarketingCampaign, and CampaignTracker.
- Define attributes and methods within these classes that encapsulate the data and behavior required to manage sports event marketing.

2. Implement CRUD Operations for Campaign Management:

- Provide functionalities to Create, Read, Update, and Delete (CRUD) marketing campaigns.
- Allow the user to manage a list of marketing campaigns, enabling the ability to view details, edit campaigns, and remove obsolete ones.

3. Plan and Manage Marketing Strategies for Sports Events:

- Create methods to design marketing strategies specific to various types of sports events (e.g., local, national, international).
- Include options for defining campaign goals, budgets, timelines, and target audiences.
- Enable customization of strategies depending on the event scale, location, and target demographics.

4. Track and Measure Campaign Effectiveness:

- Implement methods to track key performance indicators (KPIs) such as reach, engagement, and conversion rates.
- Allow users to analyze the effectiveness of each marketing campaign over time, providing feedback for optimization.

5. Offer Flexibility for Different Sports Event Types:

- Design the system in a way that accommodates marketing campaigns for different sports events, ranging from local competitions to large-scale international tournaments.

- Ensure that marketing plans can be tailored to specific events' audiences, geography, and promotional needs.

6. Provide a Simple User Interface or Command-line Interface (CLI):

- Implement an interface (CLI or UI) that allows users to interact with the system to create, manage, and track campaigns.
- Ensure ease of use so that non-technical users can manage sports event marketing campaigns efficiently.

7. Support for Manual Data Entry and Static Analysis:

- The system will allow for the manual input of performance data (e.g., reach, conversion rates) and provide static reports on campaign effectiveness.
- This data can help users make informed decisions for future marketing strategies based on past performance.

8. Create a Reusable and Scalable Framework:

- Develop a framework that can be expanded in the future to include advanced features such as integration with external tools (e.g., ad platforms) or automated suggestions based on data insights.

3.Methodology:

1. **Object-Oriented Programming (OOP):**
    - OOP Principles such as encapsulation, abstraction, inheritance, and polymorphism are central to this program.
    - The program will use classes to model the real-world entities involved in sports event marketing:
        - Classes like SportsEvent, MarketingCampaign, and CampaignTracker will define attributes and methods for each entity.
        - Encapsulation ensures that each class handles its own data and logic.
        - Inheritance can be used if there are common behaviors between campaigns (e.g., digital marketing, offline marketing).

2. CRUD Design Pattern:

- CRUD operations (Create, Read, Update, Delete) will be implemented for managing the marketing campaigns.
  - Create: Adding new marketing campaigns with event-specific details.
  - Read: Retrieving and viewing campaign information.
  - Update: Modifying existing campaigns to adjust strategies or budgets.
  - Delete: Removing old or obsolete campaigns from the system.
- These operations will be encapsulated in methods within the MarketingCampaign class (e.g., create_campaign(), get_campaign(), update_campaign(), delete_campaign()).

3. Modular Programming:

- The system will be divided into distinct modules based on functionality:
  - Campaign Management Module for creating and managing campaigns.
  - Strategy Planning Module for defining the marketing plan for each event.
  - Tracking Module for monitoring the performance of campaigns.
- Modularization will make the code more maintainable and scalable, allowing future enhancements.

4. Design Patterns (Optional):

- Depending on complexity, design patterns like Factory Pattern (for creating different types of campaigns) or Observer Pattern (to track changes in campaign status) could be employed to improve system flexibility and maintainability.

5. Agile Development Methodology:

- The system can be developed using agile principles, where development occurs incrementally with continuous feedback and adjustments.

- Sprint-based iterations can help prioritize critical features (CRUD, strategy planning, performance tracking) for early delivery, while advanced features (e.g., analytics) can be developed in later iterations.

6. Data Management Techniques:

- The program will store marketing campaign data using basic data structures like lists or dictionaries.
- Persistent storage (e.g., using a database or file system) can be considered to ensure that campaigns are saved between sessions, but for a POC, a simple in-memory solution may suffice.
- Serialization (e.g., using JSON or Pickle) can be used to store and retrieve campaign data.

7. Test-Driven Development (TDD):

- Unit tests will be written to verify the correctness of individual methods (e.g., creating a campaign, updating it, deleting it).
- Integration tests will check the interaction between different components, such as campaign creation and tracking.
- This ensures that the system is robust and reliable.

8. Event-Driven Programming:

- The system may follow an event-driven approach where changes in the campaign's state (e.g., campaign starts, performance metrics update) trigger updates to the tracking module.

9. Performance Tracking and KPIs:

- The system will incorporate methods for tracking key performance indicators (KPIs) like reach, engagement, and conversion rates.
- This could be done using mathematical operations or simple data analysis techniques to calculate the effectiveness of each marketing campaign based on predefined goals.

Tools used in the program:

1. Python Programming Language:

- Python is the primary language for implementing the system due to its simplicity, readability, and vast ecosystem of libraries.
- It provides all necessary constructs for object-oriented programming (OOP), CRUD operations, and integration with data handling tools.

2. Integrated Development Environment (IDE):

- Tools like PyCharm, Visual Studio Code, or Jupyter Notebooks will be used to write, debug, and run the Python code.
- These IDEs provide features like syntax highlighting, code navigation, and debugging utilities, making development faster and more efficient.

3. Version Control System (Git):

- Git will be used for version control, allowing the team to track code changes, manage different branches, and collaborate on the project.
- Platforms like GitHub or GitLab can host the project repository for sharing and collaboration.

4. Object-Oriented Design Tools:

- UML (Unified Modeling Language) diagrams can be used to design class relationships, illustrating how entities like SportsEvent, MarketingCampaign, and CampaignTracker interact.
- Tools like Lucidchart, Draw.io, or StarUML can be used for designing and visualizing the system's structure before implementation.

5. Python Libraries for Data Management:

- Pandas (optional): If the program needs to handle complex data, libraries like Pandas can be used to manage and manipulate campaign data in tabular form.

- SQLite3: If persistent storage is required, the lightweight SQLite library can store marketing campaign information in a database.
- Pickle/JSON: These libraries can serialize objects (e.g., campaigns) for storing and retrieving data in files.

6. Command-Line Interface (CLI) Tools:

- The initial implementation might use Python's built-in argparse or click libraries to create a simple Command-Line Interface (CLI) for user interaction.
- These tools allow users to input parameters, interact with the system, and get feedback on campaign creation, updates, or analysis.

7. Testing Frameworks:

- unittest or pytest: Python's built-in unittest framework or the popular third-party pytest can be used to create unit tests and integration tests.
- These tools are crucial for the Test-Driven Development (TDD) approach, ensuring that the code is working as expected and can handle edge cases.

8. Performance Tracking and Analytics Libraries:

- NumPy: For basic mathematical operations like calculating reach, engagement, and conversions in the performance tracking module.
- Matplotlib or Seaborn (Optional): For simple visualizations of campaign performance metrics like reach or conversion trends.

4. Findings/results of the Sports Event Marketing Planner:

1. Functional CRUD Operations:

- The program successfully supports **Create**, **Read**, **Update**, and **Delete (CRUD)** functionalities for marketing campaigns.
  - **Create**: Users can create new campaigns by defining key attributes such as event name, target audience, budget, and timeline.
  - **Read**: Campaign details can be viewed at any time, providing insights into the event, strategy, and campaign progress.

- **Update**: Users can modify campaign details such as strategy changes or budget adjustments dynamically.
- **Delete**: Obsolete or completed campaigns can be removed from the system.

- **Finding**: The CRUD operations provide flexibility and efficient management of marketing campaigns, allowing users to make real-time changes as needed.

2. Structured Marketing Campaign Planning:

- The system allows users to plan and customize marketing strategies specific to each sports event. Users can define:
  - **Target audience** (e.g., demographics, geographic region)
  - **Budget allocation** (e.g., total budget, allocation across marketing channels)
  - **Campaign duration** (start and end dates)
  - **Marketing channels** (social media, digital ads, print media)
- **Finding**: The structured planning module helps users systematically design marketing strategies for each sports event, ensuring that campaigns are aligned with the event's goals and audience.

3. Performance Tracking and KPIs:

- The program tracks and measures the effectiveness of marketing campaigns based on key performance indicators (KPIs), such as:
  - **Reach**: Number of people exposed to the campaign.
  - **Engagement**: Interactions with the marketing material (e.g., likes, shares, comments).
  - **Conversion rates**: Number of people who took a desired action (e.g., ticket purchase, event registration).
- Users can manually input these metrics or simulate performance for the POC.
- **Finding**: By tracking campaign performance, users can gain insights into which strategies are working and which need adjustment. This feedback helps improve future campaigns by making data-driven decisions.

4. Support for Multiple Sports Events:

- The system supports the creation and management of multiple campaigns for different sports events. Each campaign can be customized independently.
- **Finding**: The flexibility to manage multiple campaigns ensures the planner is adaptable to various event sizes, from local sports tournaments to large-scale international competitions.

5. Ease of Use (CLI or Simple Interface):

- The **Command-Line Interface (CLI)** provides an easy and intuitive way for users to interact with the system. Users can create, update, and track campaigns through simple commands.
- **Finding**: The CLI is effective for POC purposes, but a more advanced user interface (e.g., web or GUI) could further improve usability for non-technical users.

6. Modularity and Scalability:

- The program follows a modular structure, which makes it easy to extend and scale in the future. New features, such as integrations with external ad platforms (e.g., Facebook Ads), can be added without disrupting existing functionality.
- **Finding**: The modular design allows for future enhancements, such as connecting to real-time data sources or building advanced analytics and machine learning models for campaign optimization.

7. Limitations and Challenges:

- **Manual Data Input**: In the current version, performance metrics need to be input manually or simulated. There is no real-time integration with data sources like Google Analytics or social media platforms.
    - **Finding**: This is a limitation of the POC, but future versions could integrate APIs for real-time tracking.
- **Basic Analysis**: The program provides basic analysis of campaign performance (e.g., reach and conversion rates). More advanced analysis (e.g., ROI calculation, multi-channel attribution) is not implemented.

Minimum Software Requirements:-

- Python (Version 3.6 or Higher)
- Core programming language to build the system.
- Python 3.6 or higher is required for compatibility with modern libraries and syntax.
- **Download: Python Official Website**

Version Control: Git (Optional for Single-User Development)

- For individual projects, using version control is optional but recommended.
- **Git**: To manage versions of your code and track progress.

Minimum Hardware Requirements:

5. Conclusion:-

- **Strategic planning**: Sports event management requires strategic planning to ensure the event runs smoothly and on time.
- **Effective communication and Attention to details:** Both are essential for successful sports event management.
- **Embrace new trends:** Event managers should embrace new trends and technologies to create unforgettable experiences.

• **CPU**	Dual-core processor (Intel Core i3 or AMD equivalent)
• **RAM**	4 GB
• **Storage**	10 GB free disk space
• **Graphics**	Integrated graphics
• **Display**	1366x768 resolution
• **Operating System**	Windows 10/11, macOS, or Linux (64-bit)
• **Networking:**	Standard internet connection for downloading libraries

- **Event marketing is a powerful strategy**: Event marketing is a powerful way to engage with your target audience, Increase brand awareness, and generate leads.
- **Digital media:** Digital media can be an effective tool for sports event marketing

6.Future Enhancements:-

- **Integration with Social Media and Advertising Platforms:**

Integrate the system with social media platforms (e.g., Facebook, Instagram) and digital advertising platforms (e.g., Google Ads) to automatically create, manage, and track campaigns in real-time.

- **AI/ML-Based Marketing Optimization:**

Implement machine learning algorithms to predict campaign effectiveness, recommend strategies, and optimize marketing campaigns based on historical data and event characteristics.

- **Real-Time Analytics Dashboard:**

Add a real-time analytics dashboard that displays KPIs (Key Performance Indicators) such as conversion rates, audience engagement, and ROI for ongoing campaigns.

- **Campaign Scheduling and Automation:**

Introduce the ability to schedule campaigns in advance, automate certain actions (e.g., posting on social media at optimal times), and send reminders for campaign management tasks.

7.Bibilography/ References:-

- "**Python Crash Course" by Eric Matthes**: A hands-on guide to learning Python, useful for beginners and intermediate programmers.
- **"Fluent Python" by Luciano Ramalho**: A deep dive into Python features and libraries for more advanced understanding.
- **"Data Science from Scratch" by Joel Grus**: Covers data handling and analysis techniques in Python, which can be beneficial for analyzing campaign performance.
- www.google.com