KISHKINDA UNIVERSITY,

Ballari



**Mini Project Report On**

**“MOVIE MARKET ANALYSIS SYSTEM POC”**

**Department of MCA**

**Submitted By:**

**S R Jyothi KUB23MCA014**

**C Rekha KUB23MCA003**

**Pallavi K KUB23MCA012**

**Bhemesh chowdary KUB23MCA002**

**Vishwanatha R S KUB23MCA019**

**KISHKINDA UNIVERSITY, Ballari**

**Project Overview:**

# Introduction

# The Movie Marketing Analysis Tool is designed to evaluate and optimize marketing strategies for film promotions. This tool leverages data analytics to provide insights into audience engagement, advertising effectiveness, social media trends, and box office performance. By analyzing various factors such as target demographics, campaign reach, online sentiment, and competitive benchmarking, the tool helps studios and marketers make data-driven decisions. Its purpose is to maximize marketing impact, increase audience anticipation, and improve the return on investment (ROI) for movie campaigns. Additionally, the tool tracks real-time campaign performance, enabling timely adjustments and ensuring marketing efforts align with audience preferences"

## Problem Statement:

* Aging Movie Marketing Analysis Tool to stay competitive and provide audiences with a better experience. Managing multiple renovation projects while tracking their impacts on audience behavior is challenging when done manually. There is a need for an automated system that can handle renovation schedules, store multiple renovation plans, and monitor the outcomes to assess the renovations' success.

# Objective

# The objective of a Movie Marketing Analysis Tool is to provide a comprehensive platform for analyzing the effectiveness of marketing strategies in the film industry. This tool aims to help stakeholders, such as production studios, marketing teams, and distributors, make data-driven decisions by evaluating key factors such as audience engagement, box office performance, social media impact, and marketing spend. The tool would enable users to optimize future marketing campaigns, predict audience behavior, and ultimately improve a movie's commercial success by providing insights into trends, demographics, and ROI on marketing efforts."

## Software Requirements:

* Python 3.12(64-bit)
* Visual Studio Code

## Hardware Requirements:

* Processor : Intel i5
* Ram : 16 GB
* Hard Disk : 500GB

**Classes Definition**

# Methodology

system for managing marketing campaigns through two main classes: Campaign and CampaignManager class. The Campaign class holds attributes related to a marketing campaign, such as campaign\_id, movie\_title, budget, revenue, impressions, and clicks.

1. Campaign Class:

Attributes include:

* campaign\_id: Unique identifier.
* movie\_title: Title of the associated movie.
* budget: Budget allocated for the campaign.
* revenue: Revenue generated from the campaign.
* clicks: Number of user interactions with the campaign.

2. Campaign Manager Class:

* add\_campaign(campaign): Adds a new campaign, raising an error if the ID already exists.
* get\_campaign(campaign\_id): Retrieves a campaign by its ID.
* update\_campaign(campaign\_id): Updates attributes of a campaign based on keyword arguments.
* delete\_campaign(campaign\_id): Removes a campaign by i

3. Unit Testing with unit test:

* + A test class Test Campaign Manager is defined to validate the functionality of Campaign Manager.
  + Tests include:
  + Adding and retrieving a campaign.
  + Updating a campaign's revenue.
  + Deleting a campaign.

**Key Features:**

1. Campaign Management: Create, update, and delete marketing campaigns.

2. Data Analytics: Track key performance indicators (KPIs) such as box office revenue, social media engagement, and trailer views.

3. Benchmarking: Compare campaign performance against industry averages.

**CRUD Functionality:**

1. Create: Add new marketing campaigns, including setting goals, budget, and target audience.

2. Read: View campaign details, track KPIs, and analyze performance.

3. Update: Modify campaign settings, adjust budget, and update targeting.

4. Delete: Remove campaigns from the system.

**Data Structures:**

1. Lists: Store campaign data.

2. Dictionaries: Map campaign IDs to campaign objects.

3. Sets: Store unique campaign tags.

**OOPS Concepts:**

1. Classes: Define Campaign and KPI classes.

2. Objects: Instantiate campaign and KPI objects.

3. Inheritance: Create subclasses for specific campaign types (e.g., social media, TV).

4. Polymorphism: Implement methods for different campaign typ

**RESULT**

## Code:

class Campaign:

def \_init\_(self, campaign\_id, movie\_title, budget, revenue, impressions, clicks):

self.campaign\_id = campaign\_id

self.movie\_title = movie\_title

self.budget = budget

self.revenue = revenue

self.impressions = impressions

self.clicks = clicks

class CampaignManager:

def \_init\_(self):

self.campaigns = {}

def add\_campaign(self, campaign):

if campaign.campaign\_id in self.campaigns:

raise ValueError("Campaign ID already exists.")

self.campaigns[campaign.campaign\_id] = campaign

def get\_campaign(self, campaign\_id):

return self.campaigns.get(campaign\_id, None)

def update\_campaign(self, campaign\_id, \*\*kwargs):

campaign = self.get\_campaign(campaign\_id)

if not campaign:

raise ValueError("Campaign not found.")

for key, value in kwargs.items():

setattr(campaign, key, value)

def delete\_campaign(self, campaign\_id):

if campaign\_id in self.campaigns:

del self.campaigns[campaign\_id]

def analyze\_effectiveness\_of\_marketing(self, campaign\_id):

campaign = self.get\_campaign(campaign\_id)

if not campaign:

return None

ROI = (campaign.revenue - campaign.budget) / campaign.budget if campaign.budget else 0

CTR = campaign.clicks / campaign.impressions if campaign.impressions else 0

return {"ROI": ROI, "CTR": CTR}

def optimize\_marketing\_strategies(self, strategy\_data):

# Placeholder for optimization logic

return strategy\_data

# Main code to demonstrate functionality

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

# Create a CampaignManager instance

manager = CampaignManager()

# Add a new campaign

campaign1 = Campaign(1, "Movie A", 100000, 300000, 10000, 500)

manager.add\_campaign(campaign1)

# Retrieve and display the campaign

retrieved\_campaign = manager.get\_campaign(1)

print(f"Retrieved Campaign: {retrieved\_campaign.movie\_title}, Budget: {retrieved\_campaign.budget}, Revenue: {retrieved\_campaign.revenue}")

# Update the campaign's revenue

manager.update\_campaign(1, revenue=500000)

updated\_campaign = manager.get\_campaign(1)

print(f"Updated Campaign Revenue: {updated\_campaign.revenue}")

# Analyze the effectiveness of the marketing campaign

analysis = manager.analyze\_effectiveness\_of\_marketing(1)

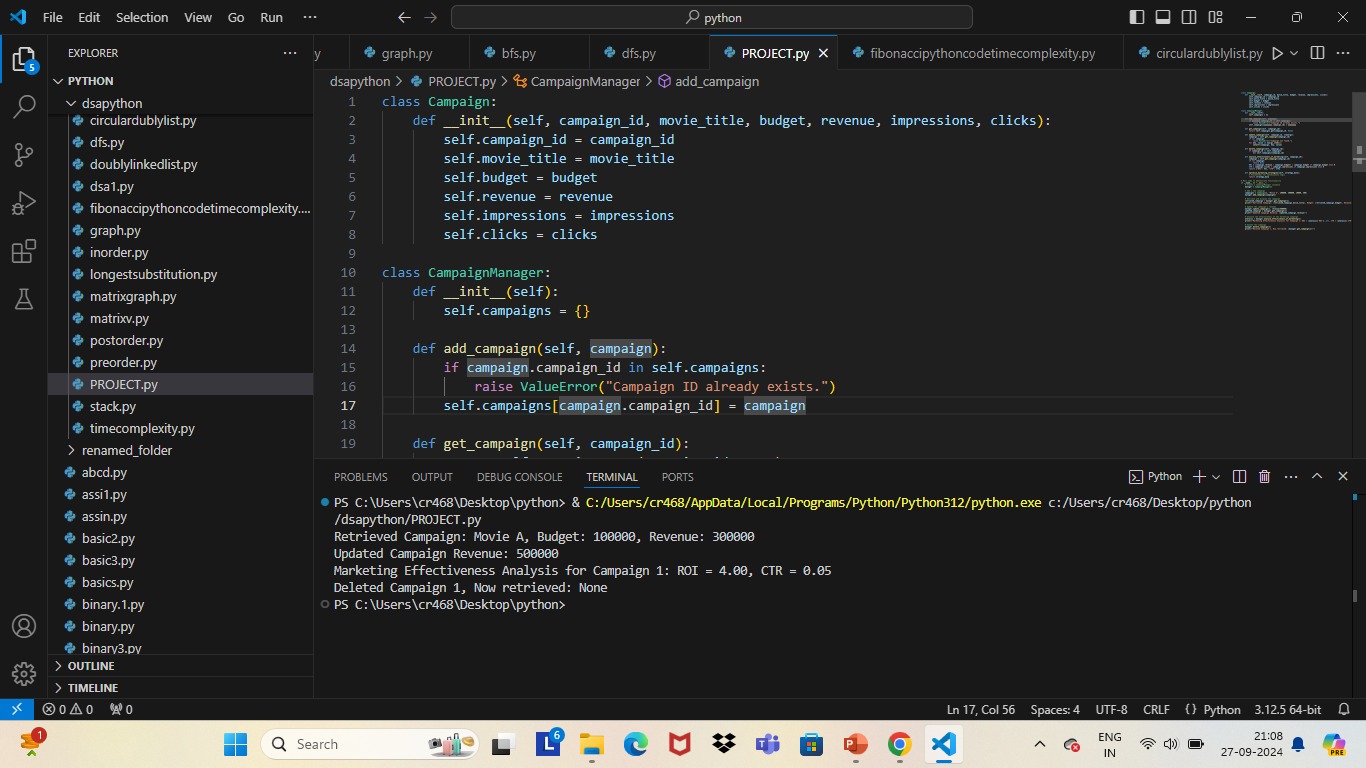
print(f"Marketing Effectiveness Analysis for Campaign 1: ROI = {analysis['ROI']:.2f}, CTR = {analysis['CTR']:.2f}")

# Delete the campaign

manager.delete\_campaign(1)

print(f"Deleted Campaign 1, Now retrieved: {manager.get\_campaign(1)}")

**Output Screen Layouts:**

****

# Conclusion

This project serves as a practical example of how to manage real-world data through effective programming techniques, making it suitable for further development in marketing analytics and campaign management tools.

Key Achievements:

1. Robust Campaign Management: The implementation of the `Campaign` and `Campaign Manager` classes allows users to seamlessly add, update, retrieve, and delete campaign data. This structured approach facilitates better organization and access to vital marketing information.

2. Performance Metrics: the tool empowers marketing teams to evaluate the effectiveness of their campaigns accurately. This data-driven analysis is crucial for informed decision-making.

3. Testing and Reliability: The inclusion of unit tests ensures that the functionalities are robust and reliable, minimizing the

# Future Enhancement

* Advanced Database Management: Use PostgreSQL or MongoDB for scalable data storage.
* AI for Predictive Maintenance: Implement machine learning to predict renovation needs based on trends.

# Implementing more detailed analytics and reporting features.

# Adding a user interface for easier campaign management.

# Expanding the optimization logic for marketing strategies

# References

* Python Software Foundation. Python Documentation. Available at: https://docs.python.org/
* Lutz, M. (2013). Learning Python, 5th Edition. O’Reilly Media.
* Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd Edition). The MIT Press.
* Python Testing with unittest. Available at: https://docs.python.org/3/library/unittest.html
* Sommerville, I. (2015). Software Engineering, 10th Edition. Pearson.
* Elmasri, R., & Navathe, S. B. (2015). Fundamentals of Database Systems, 7th Edition. Pearson.
* [https://www.greeksforgeeks.org](https://www.greeksforgeeks.org/)