KISHKINDA UNIVERSITY

Mini Project Report

On

"Match Ticketing Analysis Tool"

Department of Computer Science and Engineering

Submitted By:

Jason Kenneth N	KUB23CSE045
Bharath Kumar J M	KUB23CSE025
K Bharath Kumar	KUB23CSE054
Shanawaz I H	BTech-CSE073
S Noor Basha	BTech-CSE080
A Suresh Rao	BTech-CSE018

Introduction

- The Match Ticketing Analysis Tool is a software application designed to manage and analyse ticket sales for various events. The tool aims to provide a comprehensive platform for event organizers to track ticket sales, analyse sales data, and predict future sales trends. This report provides an overview of the project, its objectives, methodology, and implementation details.

Project Overview:

- The Match Ticketing Analysis Tool is a Python-based application that utilizes Object-Oriented Programming (OOP) and Data Structures and Algorithms (DSA) principles. The tool is designed to handle CRUD (Create, Read, Update, Delete) operations for ticket sales and provides functions for analysing and predicting ticket sales trends.

Problem Statement:

- Event organizers often struggle to manage and analyse ticket sales data, leading to inefficient pricing and promotional strategies. The lack of a comprehensive platform to track and analyse ticket

sales data results in missed opportunities for revenue growth and customer engagement.

Objective

- The objective of this project is to design and develop a Match Ticketing Analysis Tool that provides a user-friendly interface for managing ticket sales data and analysing sales trends. The tool aims to help event organizers optimize their pricing and promotional strategies, leading to increased revenue and customer satisfaction.

Software Requirements:

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

Hardware Requirements:

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

Methodology

The development of the Match Ticketing Analysis Tool followed a structured approach, consisting of the following steps:

1. **Requirements Gathering**: This phase involved identifying the functional and non-functional requirements of the project. The requirements were gathered through a combination of techniques, including:
    - Literature review: Reviewing existing research and documentation on ticketing systems and sales analysis tools.
    - Stakeholder interviews: Conducting interviews with event organizers and ticketing system administrators to understand their needs and requirements.
    - User surveys: Conducting surveys to gather information on the features and functionalities required by event organizers and ticketing system administrators.
2. **System Design**: This phase involved designing the system architecture and data models. The design was based on the following principles:
    - Object-Oriented Programming (OOP): The system was designed using OOP principles to ensure modularity, reusability, and maintainability.

- Data Structures and Algorithms (DSA): The system was designed using DSA principles to ensure efficient data storage and retrieval.
- Scalability: The system was designed to be scalable to accommodate large volumes of ticket sales data.

3. **Implementation**: This phase involved implementing the system using Python, utilizing OOP and DSA principles. The implementation consisted of the following steps:
    - Developing the **Ticket** class: This class represented a single ticket and included attributes such as ticket ID, event ID, price, and sold status.
    - Developing the **Ticket Sales** class: This class managed a collection of tickets and included methods for adding, selling, and deleting tickets.
    - Developing the **analyse_ticket_sales_data** function: This function analysed ticket sales data and provided insights on sales trends.
    - Developing the **predict_future_sales_trends** function: This function predicted future sales trends based on historical sales data.
4. **Testing**: This phase involved conducting unit testing using the Unittest framework. The testing consisted of the following steps:

- Writing test cases: Writing test cases to cover all the functionalities of the system.
- Running test cases: Running the test cases to ensure that the system was working as expected.
- Debugging: Debugging the system to fix any errors or bugs that were identified during testing.

5. **Deployment**: This phase involved deploying the system to a production environment. The deployment consisted of the following steps:
    - Setting up the production environment: Setting up the production environment to ensure that the system was running smoothly.
    - Deploying the system: Deploying the system to the production environment.
    - Monitoring the system: Monitoring the system to ensure that it was working as expected

Create a tool to handle CRUD operations for ticket sales.

- Analyse ticket sales data to optimize pricing and promotions.
- Predict future ticket sales trends.
- It provides functions for analysing and predicting ticket sales trends.
- The tool uses Object-Oriented Programming (OOP) and Data Structures and Algorithms (DSA) principles

- CRUD: Create, Read, Update, Delete
- Helps us to modify the code
- In the project we can create a new ticket, Read the ticket, Update the price of the ticket and to delete the ticket when it is required
- It helps to keep the data in dynamic manner
- The Ticket Sales represents a collection of tickets

- **Efficient Ticket Management:** The Ticket Sales class allows for efficient management of ticket sales data, including adding, selling, and deleting tickets.
- **Improve Ticket Validation:** Enhance the class to include ticket validation to prevent duplicate or invalid tickets from being added or sold.
- **Easy Data Retrieval:** The class provides methods to retrieve ticket data, making it easy to access and analyze sales information.

- **Ticket Module:**
- Purpose: Encapsulate the details and behaviors of a ticket.
- **Ticket Sales Module:**
- Purpose: Manage the collection of tickets and provide functionalities to add, sell, delete, and analyze tickets.

- **Ticketless Module:**
- Purpose: Provide unit tests for the Ticket Sales class to ensure its methods work correctly.
- **Main Module:**
- Purpose: Serve as the entry point for running the ticket sales system and tests.

- **Data-Driven Insights:** The analysis function provides valuable insights into ticket sales data, enabling informed decisions on pricing and promotions.
- **Predictive Capabilities:** The prediction function offers a basic trend analysis, which can be improved with machine learning algorithms to forecast future sales.
- **Enhance Prediction Model:** Develop a more sophisticated prediction model using machine learning algorithms to improve the accuracy of sales forecasts

- Sports Teams and Leagues
- Concert and Festival Promoters
- Theater and Performing Arts
- Travel and Tourism

- The Match Ticketing Analysis Tool POC is a functional prototype

- Future work includes integrating a machine learning algorithm for prediction
- Real-time Data Updates
- User Authentication and Authorization

RESULT:-

Code:

```
class Ticket:
    def __init__(self, event_id, price):
        self.event_id = event_id
        self.price = price
        self.sold = False

class TicketSales:
    def __init__(self):
        self.sales = {}

    def add_ticket(self, event_id, price):
        if event_id not in self.sales:
            self.sales[event_id] = []
        self.sales[event_id].append(Ticket(event_id, price))

    def sell_ticket(self, event_id):
        for ticket in self.sales.get(event_id, []):
            if not ticket.sold:
                ticket.sold = True
```

```
                return True
        return False


    def delete_ticket(self, event_id, index):
        if event_id in self.sales and 0 <= index <
len(self.sales[event_id]):
            del self.sales[event_id][index]


    def analyze_sales(self, event_id):
        tickets = self.sales.get(event_id, [])
        total_sold = sum(t.sold for t in tickets)
        avg_price = sum(t.price for t in tickets) / len(tickets) if tickets
else 0
        return {
            "total_sold": total_sold,
            "average_price": avg_price,
            "sold_percentage": (total_sold / len(tickets) * 100) if tickets
else 0
        }


import unittest


class TestTicketSales(unittest.TestCase):
    def setUp(self):
```

```
        self.ticket_sales = TicketSales()
        self.event_id = "E001"
        self.ticket_sales.add_ticket(self.event_id, 100)
        self.ticket_sales.add_ticket(self.event_id, 150)

    def test_sales(self):
        print("Initial tickets:")
        for i, ticket in enumerate(self.ticket_sales.sales[self.event_id]):
            print(f"Ticket {i+1}: Price={ticket.price},
Sold={ticket.sold}")

        print("\nSelling a ticket...")
        self.assertTrue(self.ticket_sales.sell_ticket(self.event_id))
        print("Tickets after selling one:")
        for i, ticket in enumerate(self.ticket_sales.sales[self.event_id]):
            print(f"Ticket {i+1}: Price={ticket.price},
Sold={ticket.sold}")

        print("\nAnalyzing sales...")
        sales_data = self.ticket_sales.analyze_sales(self.event_id)
        print(f"Total sold: {sales_data['total_sold']}")
        print(f"Average price: {sales_data['average_price']}")
        print(f"Sold percentage: {sales_data['sold_percentage']}%")
```

```
        print("\nDeleting a ticket...")
        self.ticket_sales.delete_ticket(self.event_id, 0)
        print("Tickets after deleting one:")
        for i, ticket in enumerate(self.ticket_sales.sales[self.event_id]):
            print(f"Ticket {i+1}: Price={ticket.price},
    Sold={ticket.sold}")

    if __name__ == '__main__':
unittest.main()
```

Output Screen Layouts:

```
Initial tickets:
Ticket 1: Price=100, Sold=False
Ticket 2: Price=150, Sold=False

Selling a ticket...
Tickets after selling one:
Ticket 1: Price=100, Sold=True
Ticket 2: Price=150, Sold=False

Analyzing sales...
Total sold: 1
Average price: 125.0
Sold percentage: 50.0%

Deleting a ticket...
Tickets after deleting one:
Ticket 1: Price=150, Sold=False
..
----------------------------------------------------------------------
Ran 2 tests in 0.000s

```

Conclusion

The Match Ticketing Analysis Tool provides a comprehensive platform for event organizers to manage and analyse ticket sales data. The tool's ability to analyse sales trends and predict future sales enables event organizers to optimize their pricing and promotional strategies, leading to increased revenue and customer satisfaction. The tool's scalability and flexibility make it an ideal solution for event organizers of all sizes.

Future Enhancement

1. Integration with Machine Learning Algorithms: Integrate machine learning algorithms to improve the accuracy of sales trend predictions.
2. User Interface: Develop a user-friendly interface for event organizers to interact with the tool.
3. Data Visualization: Integrate data visualization tools to provide a graphical representation of sales trends.
4. Scalability: Optimize the tool for large-scale event management.

References

- Python Documentation. (n.d.). Python Standard Library. Retrieved from https://docs.python.org/3/library/
- Unittest Framework. (n.d.). Unittest Framework. Retrieved from https://docs.python.org/3/library/unittest.html
- Object-Oriented Programming. (n.d.). Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Objectoriented_programming
- Data Structures and Algorithms. (n.d.). Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Data_structure