BASAVARAJESWARI GROUP OF INSTITUTIONS

Ballari Institute of Technology & Management

AUTONOMOUS INSTITUTE UNDER VISVESVARAYA TECHNOLOGICAL UNIVERSITY JNANA SANGAMA, BELAGAVI 590018

INTERNSHIP

Report On

"SPORTS BOARDCAST SCHEDULING TOOLS"

Submitted in partial fulfillment of the requirements for the award of degree of

Bachelor of Engineering
IN
DEPARTMENT OF COMPUTER SCIENCE-DATA SCIENCE

Submittedby

VINUTHA V

3BR23CD105

Internship Carried Out
EZ TRAINING
& TECHNOLOGIES PVT.LTD
HYDERABAD

Internal Guide

PARAVATHI

ANUSHYA

External Guide

MR. VISHAL KUMAR

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT
NACC Accredited Institution*
(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belagavi)
"Jnana Gangotri" Campus, No.873/2, Ballari-Hospet Road, Allipur, Ballari-583104 (Karnataka) (India)
Ph: 08392–237100/237190, Fax: 08392–237197
2024-2025

BASAVARAJESWARI GROUP OF INSTITUTIONS

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

Autonomous institute under VISVESVARAYA TECHNOLOGICAL UNIVERSITY JNANA SANGAMA,

BELAGAVI 590018

NACC Accredited Institution*

(Recognized by Govt.of Karnataka, approved by AICTE, NewDelhi & Affiliated to Visvesvaraya Technological University, Belagavi.

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

Ph: 08392–237100/237190, Fax: 08392–237197

DEPARTMENT OF COMPUTER SCIENCE-DATA SCIENCE ENGINEERING

CERTIFICATE

This is to certify that the Internshipen titled **"SPORTS BOARDCAST SCHEDULING TOOLS"** has been successfully completed by **MS. VINUTHA V** bearing USN:- **3BR23CD105** bonafide student of Ballari Institute of Technology and Management, Ballari. For the partial fulfillment of the requirements for the **Bachelor's Degree in Department of COMPUTER SCIENCE-DATA SCIENCE ENGINEERING** of the VISVESVARAYA TECHNOLOGICAL UNIVERSITY, Belagavi during the academic year 2024-2025.

Signature of Internshp

Co-Ordinator

SignatureofHOD

Dr.ARADHANA

Prof and HOD(cse-ds)

DECLARATION

I, **VINUTHA V,** Third year student of **Computer science-data Science Engineering**, Ballari Institute of Technology, Ballari, declare that Internship entitled **"SPORTS BOARDCAST SCHEDULING TOOLS"** is a part of Internship Training successfully carried out by **EZ TECHNOLOGIES & TRAININGS PVT.LTD, HYDERBAD** at "**BITM, BALLARI".** This report is submitted in partial fulfillment of the requirements for the award of the degree, Bachelor of Engineering in Computer Science -Data science Engineering of the Visvesvaraya Technological University, Belagavi, Karnataka.

Date:28/09/2024 **Signature of the Student**
Place: BALLARI

ACKNOWLEDGEMENT

The satisfactions that we feel at the successful completion of my internship with a company "**EZ TECHNOLOGIES & TRAININGS PVT.LTD**" on "SPORTS BOARDCAST SCHEDULING TOOLS" would be incomplete without the mention of people who made it possible, whose noble gesture, affection, guidance, encouragement and support crowned my efforts with success. It is my privilege to express my gratitude and respect to all those who inspired me in the completion of my internship.

We are thankful lto **Dr.Aradhana,** H.O.D, Department of data science Engineering, for providing the facilities in the department to do this internship work.

Table of Contents

ABSTRACT:

Abstract

The **Broadcast Management System** is a comprehensive tool designed to streamline the scheduling and management of broadcast events in the dynamic media landscape. With the increasing demand for organized and timely broadcasting, this system addresses the complexities associated with managing various sports and entertainment events. Central to this system are two core components: the **Broadcast** class and the **BroadcastManager** class.

The **Broadcast** class encapsulates essential event information, including event ID, name, timing, channel, location, and duration. This structured representation allows for easy retrieval and manipulation of broadcast data, ensuring all critical details are accessible for both broadcasters and viewers.

The **BroadcastManager** class serves as the operational backbone of the system, facilitating Create, Read, Update, and Delete (CRUD) operations on broadcast events. It provides functionalities for scheduling events, updating broadcast details, and deleting obsolete broadcasts. Additionally, it manages rights information, ensuring that broadcasting rights are appropriately assigned and maintained.

This system enhances operational efficiency by minimizing scheduling conflicts and ensuring compliance with legal broadcasting requirements. By providing an organized approach to broadcast management, the Broadcast Management System not only aids broadcasters in maximizing viewership but also improves the overall experience for audiences seeking timely access to live events. Through this system, broadcasters can effectively navigate the complexities of event management while maintaining high standards of contentdelivery.

INTRODUCTION:

Introduction to Broadcast Management System

In today's fast-paced media landscape, efficient scheduling and management of broadcast events are crucial for maximizing viewership and ensuring a seamless experience for audiences. The **Broadcast Management System** serves as an essential tool for broadcasters, allowing them to organize, schedule, and manage various events effectively. This system focuses on two main components: the **Broadcast** class and the **BroadcastManager** class.

1. **Broadcast Class**:

- The **Broadcast** class represents individual events scheduled for broadcast. It encapsulates key information such as the **event ID**, **event name**, **timing**, **channel**, **location**, and **duration**. By organizing these attributes, the Broadcast class provides a structured format for storing and accessing event data, ensuring that all necessary details are readily available for both broadcasters and viewers.

2. **BroadcastManager Class**:

- The **BroadcastManager** class acts as the central hub for managing all broadcasts. It facilitates various operations such as creating, reading, updating, and deleting broadcasts (commonly known as CRUD operations). Additionally, it manages the scheduling of events and oversees rights management, ensuring that only authorized channels can broadcast specific events. This class simplifies the complexities of broadcast scheduling by allowing broadcasters to maintain an organized

database of events, thereby enhancing operational efficiency.

Importance of Broadcast Management

The importance of a Broadcast Management System cannot be overstated:

- It ensures that events are scheduled accurately and efficiently, minimizing scheduling conflicts and errors.

- It allows for easy updates to broadcast information, which is vital in the dynamic environment of live events.

- It provides a clear overview of all scheduled broadcasts, aiding in strategic planning and decision-making.

- It protects intellectual property by managing rights and permissions associated with each broadcast, ensuring compliance with legal requirements.

In summary, the Broadcast Management System plays a pivotal role in the broadcasting industry, streamlining processes and enhancing the overall viewer experience. This system is designed to meet the needs of broadcasters in managing their events while ensuring that audiences have access to timely and accurate information about upcoming broadcasts.

DESCRIPTION:

Description of the Broadcast Management System

The **Broadcast Management System** is a software solution designed to facilitate the efficient management of broadcast events, particularly in the context of sports, entertainment, and other live programming. This system is built around two primary components: the **Broadcast** class and the **BroadcastManager** class, each serving distinct roles in managing the lifecycle of broadcast events.

1. **Broadcast Class**

The **Broadcast** class represents an individual broadcast event and contains several key attributes:

- **Event ID**: A unique identifier for the broadcast, ensuring that each event can be distinctly referenced.
- **Event Name**: The title or name of the event, such as "Champions League Final" or "Super Bowl."
- **Timing**: The scheduled date and time when the event will occur, allowing for precise planning and promotion.
- **Channel**: The television channel responsible for broadcasting the event, which is crucial for viewer access.
- **Location**: The venue where the event is taking place, which may be of interest to fans and viewers.
- **Duration**: The expected length of the event, helping in scheduling and planning for both broadcasters and viewers.

This class provides a structured format for encapsulating all relevant details about a broadcast, allowing for easy retrieval and management of event data.

2. **BroadcastManager Class**

The **BroadcastManager** class serves as the operational control center for managing multiple broadcast events. It includes the following key functionalities:

- **Add Broadcast**: Allows users to input and store new broadcast events, ensuring that all relevant details are captured and stored in the system.
- **View All Broadcasts**: Provides a comprehensive list of all scheduled broadcasts, enabling users to quickly review upcoming events and their details.
- **Update Broadcast**: Enables modifications to existing broadcast details, such as changing the timing or channel, ensuring that information remains accurate and up-to-date.
- **Delete Broadcast**: Allows users to remove broadcasts from the system when they are no longer relevant or necessary, helping maintain an organized database of events.
- **Rights Management**: Manages the legal permissions associated with each broadcast, ensuring that only authorized channels are permitted to air specific events.

User Interface

The Broadcast Management System typically features a user-friendly interface built with GUI tools (e.g., Tkinter in Python) that allows users to interact easily with the system. Key components of the interface include:

- Input fields for entering broadcast details.
- Buttons for executing various actions (add, view, update, delete).
- A text area for displaying output messages and lists of broadcasts.

Benefits of the System

- **Efficiency**: Streamlines the process of scheduling and managing broadcasts, minimizing the potential for errors and conflicts.
- **Accessibility**: Provides quick access to broadcast information, enhancing transparency for both broadcasters and viewers.
- **Flexibility**: Allows for easy updates to broadcast details, accommodating changes in scheduling or channel assignments.
- **Legal Compliance**: Ensures that broadcasting rights are properly managed, protecting intellectual property and adhering to regulatory requirements.

.

ALGORITHM:

Here’s a structured algorithm for managing broadcasts using the **Broadcast** and **BroadcastManager** concepts. This algorithm outlines the steps involved in creating, updating, deleting, and viewing broadcasts.

Algorithm for Broadcast Management

1. **Data Structure Definition**
- **Broadcast Class**:
  - **Properties**:
    - `event_id`: Unique identifier for the broadcast.
    - `event_name`: Name of the event.
    - `timing`: Date and time of the event.
    - `channel`: TV channel broadcasting the event.
    - `location`: Where the event is taking place.
    - `duration`: Length of the event.

- **BroadcastManager Class**:
  - **Properties**:
    - `broadcasts`: A dictionary (or list) to store broadcast objects, using `event_id` as the key.

2. **Operations**

A. Add Broadcast
1. **Input**: Get `event_id`, `event_name`, `timing`, `channel`, `location`, and `duration` from user.
2. **Check**:
   - If `event_id` already exists in `broadcasts`:
     - **Output**: "Broadcast with event ID {event_id} already exists."
   - Else:
     - Create a new `Broadcast` object with the provided details.
     - Add it to `broadcasts` dictionary using `event_id` as the key.
     - **Output**: "Broadcast for {event_name} added."

B. View All Broadcasts
1. **Check**:
   - If `broadcasts` is empty:
     - **Output**: "No broadcasts available."
   - Else:
     - For each broadcast in `broadcasts`:
       - **Output**: Print details of each broadcast.

C. Update Broadcast
1. **Input**: Get `event_id`, `new_timing`, and/or `new_channel` from user.
2. **Check**:
   - If `event_id` exists in `broadcasts`:
     - Update the `timing` and/or `channel` of the corresponding `Broadcast` object.
     - **Output**: "Broadcast {event_id} updated."
   - Else:
     - **Output**: "Broadcast with event ID {event_id} not found."

D. Delete Broadcast
1. **Input**: Get `event_id` from user.
2. **Check**:
   - If `event_id` exists in `broadcasts`:
     - Remove the corresponding `Broadcast` object from `broadcasts`.
     - **Output**: "Broadcast with event ID {event_id} deleted."
   - Else:
     - **Output**: "Broadcast with event ID {event_id} not found."

3. **Rights Management** (Optional Extension)
- **Function**: Manage rights for a broadcast.
- **Input**: Get `event_id` and `rights_data` from user.
- **Check**:
  - If `event_id` exists in `broadcasts`:
    - Update the rights information for the broadcast.
    - **Output**: "Rights information updated for event {event_id}."
  - Else:
    - **Output**: "Broadcast with event ID {event_id} not found."

Conclusion

This algorithm provides a clear step-by-step approach for managing broadcasts using the defined classes and operations. It ensures that all functionalities (CRUD operations, scheduling, and rights management) are systematically organized and easy to follow.

CODE:

```
import tkinter as tk
from tkinter import scrolledtext

# Define the Broadcast class
class Broadcast:
    def __init__(self, event_id, event_name, timing, channel, location, duration):
        self.event_id = event_id # unique ID for the event
        self.event_name = event_name  # name of the event (e.g., FIFA World Cup Final)
        self.timing = timing # when the event will be broadcast (date/time)
        self.channel = channel # TV channel broadcasting the event
        self.location = location  # location of the event (e.g., stadium)
        self.duration = duration  # duration of the event (e.g., 2 hours)
        self.rights = None  # Rights information (if applicable)

    def __str__(self):
        return (f"Broadcast(ID: {self.event_id}, Event: {self.event_name}, Time: {self.timing}, "
                f"Channel: {self.channel}, Location: {self.location}, Duration: {self.duration}, "
                f"Rights: {self.rights})")

# Define the BroadcastManager class to handle the scheduling tool
class BroadcastManager:
    def __init__(self):
        self.broadcasts = {}

    # Add a new broadcast
```

```
    def add_broadcast(self, event_id, event_name, timing, channel,
location, duration):
        if event_id in self.broadcasts:
            return f"Broadcast with event ID {event_id} already exists."
        else:
            new_broadcast = Broadcast(event_id, event_name, timing,
channel, location, duration)
            self.broadcasts[event_id] = new_broadcast
            return f"Broadcast for {event_name} added."

    # View all broadcasts
    def view_all_broadcasts(self):
        if self.broadcasts:
            return "\n".join([str(broadcast) for broadcast in
self.broadcasts.values()])
        else:
            return "No broadcasts available."

    # Update an existing broadcast
    def update_broadcast(self, event_id, new_timing=None,
new_channel=None):
        if event_id in self.broadcasts:
            broadcast = self.broadcasts[event_id]
            if new_timing:
                broadcast.timing = new_timing
            if new_channel:
                broadcast.channel = new_channel
            return f"Broadcast {event_id} updated."
        else:
            return f"Broadcast with event ID {event_id} not found."

    # Delete an existing broadcast
```

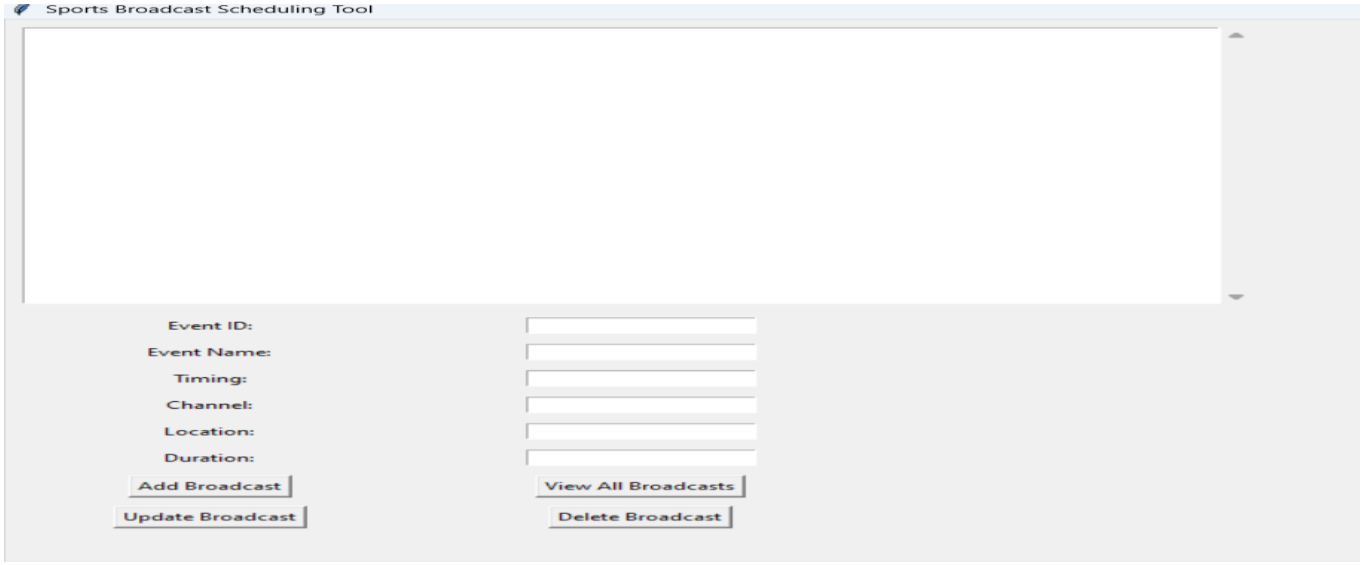
```
    def delete_broadcast(self, event_id):
        if event_id in self.broadcasts:
            del self.broadcasts[event_id]
            return f"Broadcast with event ID {event_id} deleted."
        else:
            return f"Broadcast with event ID {event_id} not found."



# Initialize Broadcast Manager
manager = BroadcastManager()

# --- Tkinter GUI Setup ---

# Initialize the main window
root = tk.Tk()
root.title("Sports Broadcast Scheduling Tool")
root.geometry("800x600")

# Add scrolled text area for output
output_text = scrolledtext.ScrolledText(root, wrap=tk.WORD,
width=80, height=20)
output_text.grid(row=0, column=0, columnspan=4, padx=10,
pady=10)

# Function to update output area
def update_output(message):
    output_text.delete(1.0, tk.END)
    output_text.insert(tk.END, message)

# Add a broadcast function
def add_broadcast():
```

```
    try:
        event_id = int(event_id_entry.get())  # Ensure event ID is a
number
        event_name = event_name_entry.get()
        timing = timing_entry.get()
        channel = channel_entry.get()
        location = location_entry.get()
        duration = duration_entry.get()

        result = manager.add_broadcast(event_id, event_name, timing,
channel, location, duration)
        update_output(result)
    except ValueError:
        update_output("Event ID must be a number.")

# View all broadcasts function
def view_broadcasts():
    result = manager.view_all_broadcasts()
    update_output(result)

# Update broadcast function
def update_broadcast():
    try:
        event_id = int(event_id_entry.get())  # Ensure event ID is a
number
        new_timing = timing_entry.get()
        new_channel = channel_entry.get()

        result = manager.update_broadcast(event_id, new_timing,
new_channel)
        update_output(result)
    except ValueError:
```

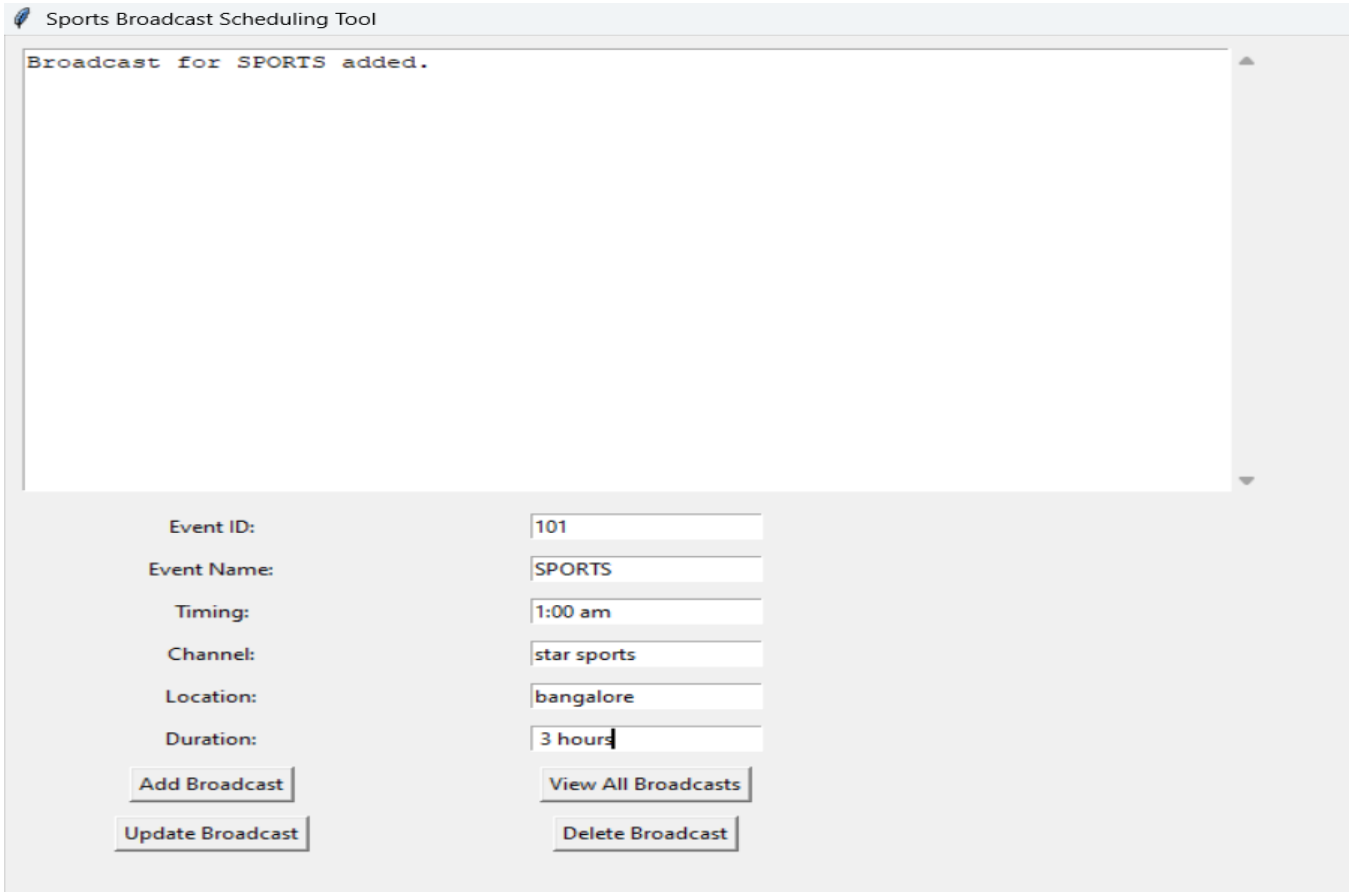
```
        update_output("Event ID must be a number.")

# Delete broadcast function
def delete_broadcast():
    try:
        event_id = int(event_id_entry.get())  # Ensure event ID is a
number
        result = manager.delete_broadcast(event_id)
        update_output(result)
    except ValueError:
        update_output("Event ID must be a number.")




# Entry labels and inputs
tk.Label(root, text="Event ID:").grid(row=1, column=0, padx=5,
pady=5)
event_id_entry = tk.Entry(root)
event_id_entry.grid(row=1, column=1, padx=5, pady=5)

tk.Label(root, text="Event Name:").grid(row=2, column=0, padx=5,
pady=5)
event_name_entry = tk.Entry(root)
event_name_entry.grid(row=2, column=1, padx=5, pady=5)

tk.Label(root, text="Timing:").grid(row=3, column=0, padx=5,
pady=5)
timing_entry = tk.Entry(root)
timing_entry.grid(row=3, column=1, padx=5, pady=5)

tk.Label(root, text="Channel:").grid(row=4, column=0, padx=5,
```

```
pady=5)
channel_entry = tk.Entry(root)
channel_entry.grid(row=4, column=1, padx=5, pady=5)

tk.Label(root, text="Location:").grid(row=5, column=0, padx=5,
pady=5)
location_entry = tk.Entry(root)
location_entry.grid(row=5, column=1, padx=5, pady=5)

tk.Label(root, text="Duration:").grid(row=6, column=0, padx=5,
pady=5)
duration_entry = tk.Entry(root)
duration_entry.grid(row=6, column=1, padx=5, pady=5)


# Buttons for actions
tk.Button(root, text="Add Broadcast",
command=add_broadcast).grid(row=8, column=0, padx=5, pady=5)
tk.Button(root, text="View All Broadcasts",
command=view_broadcasts).grid(row=8, column=1, padx=5,
pady=5)
tk.Button(root, text="Update Broadcast",
command=update_broadcast).grid(row=9, column=0, padx=5,
pady=5)
tk.Button(root, text="Delete Broadcast",
command=delete_broadcast).grid(row=9, column=1, padx=5,
pady=5)

# Start the Tkinter main loop
root.mainloop()
```

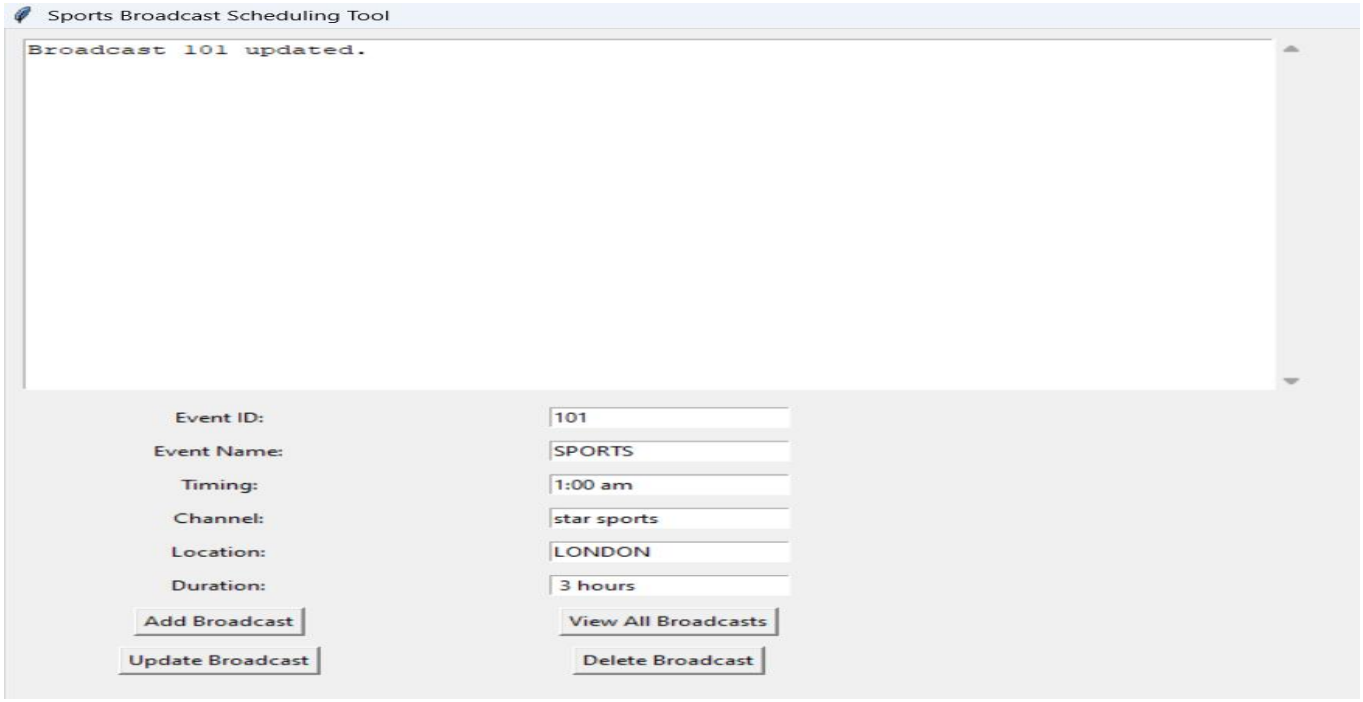
OUTPUT



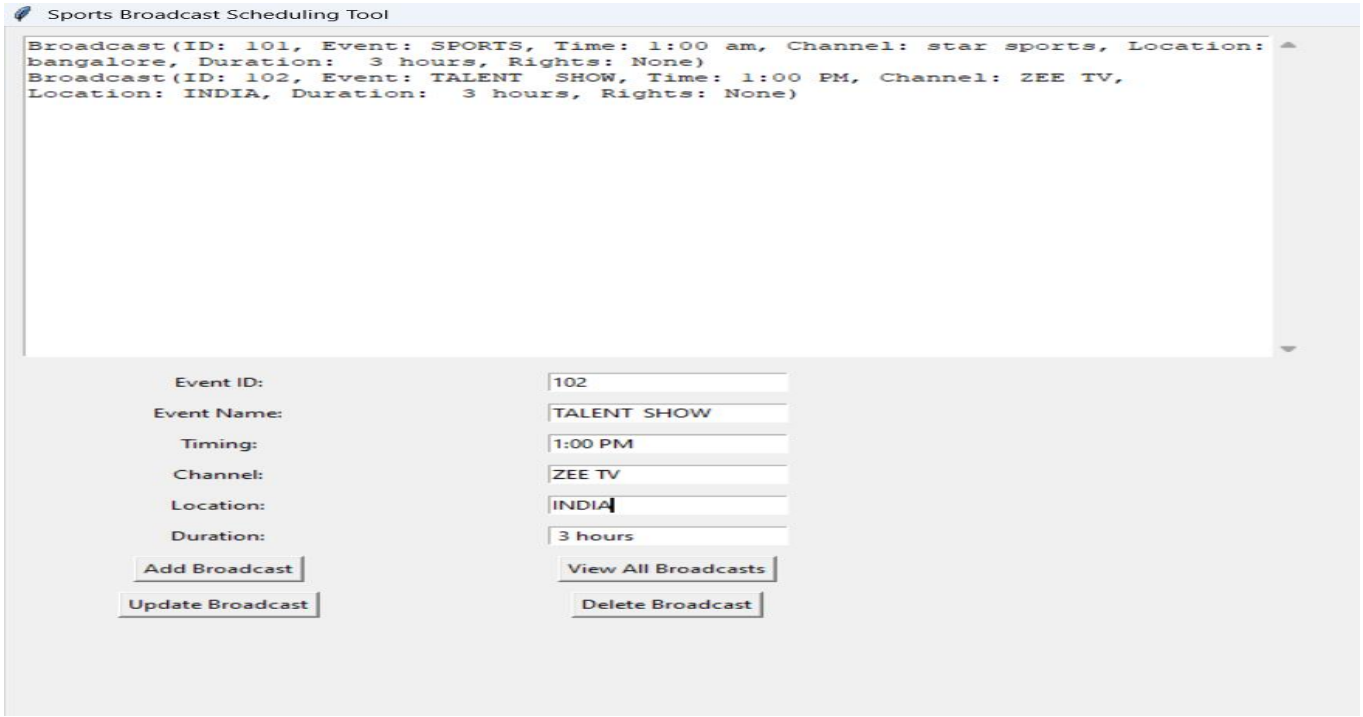
1.ADDING AN EVENT



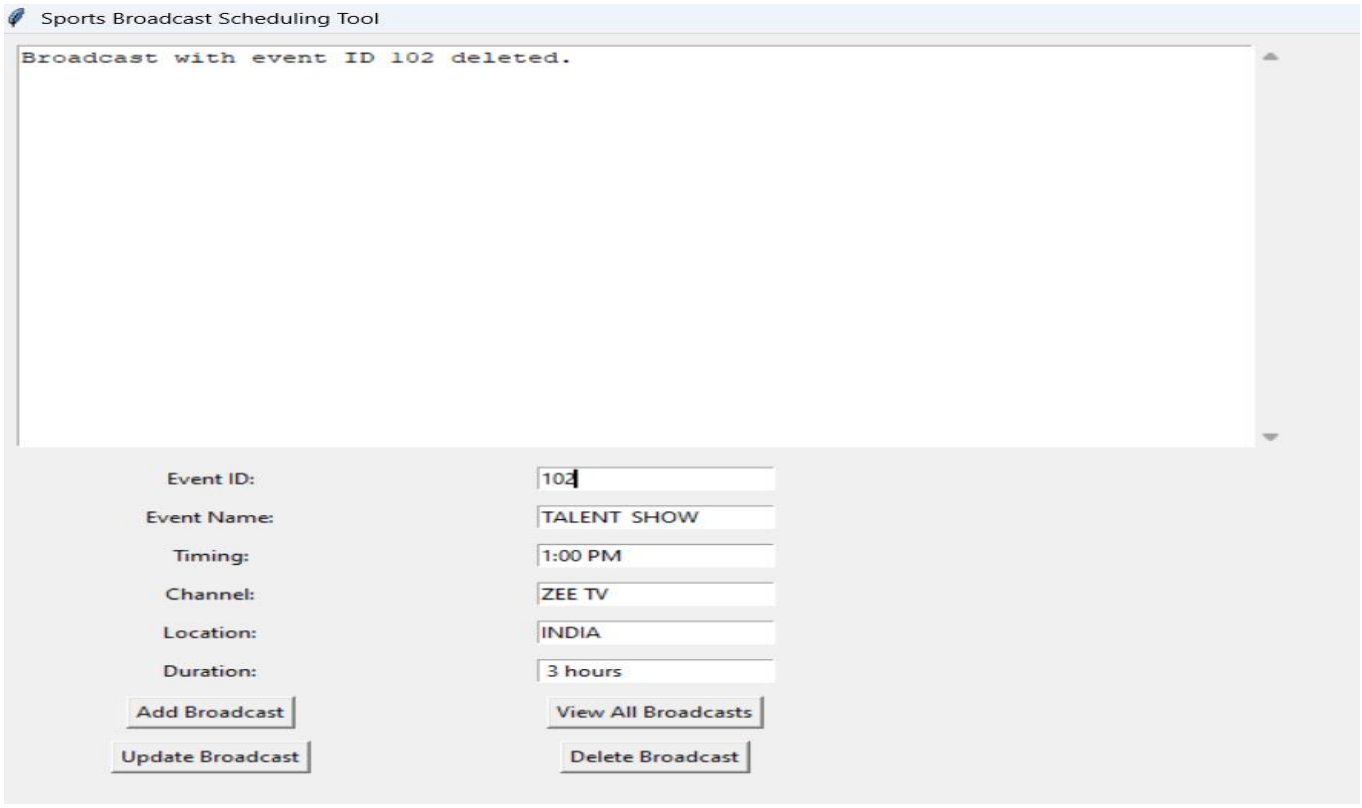
2.UPDATING EVENT



3. READ(VIEW ALL BOARDCAST)



4.DELETE EVENT



CONCLUSION:

1. **Definition of Broadcast:**
- A broadcast represents a scheduled event (e.g., a sports match) with essential details such as event ID, name, timing, and channel.

2. **Importance of Key Properties:**
- Each broadcast's unique attributes, such as event ID and timing, are crucial for effective scheduling and viewer engagement.

3. **Role of BroadcastManager:**
- Acts as the central hub for managing all broadcasts, ensuring accurate scheduling and updates.

4. **CRUD Operations:**
- Enables the creation, reading, updating, and deletion of broadcast events, allowing for flexibility in managing changing schedules.

5. **Scheduling Efficiency:**
- Streamlines the process of scheduling events, minimizing conflicts and ensuring timely broadcasts.

6. **Rights Management:**
- Addresses legal aspects of broadcasting, ensuring that the correct channels have permission to air specific events, thereby protecting intellectual property.

7. **Overall Significance:**
- Together, the Broadcast and BroadcastManager system enhance the organization and presentation of televised events, benefiting broadcasters and viewers alike.

REFERENCE

- Code:-https://chatgpt.com/share/66f63642-abf8-8006-a1f7-30c687048349

- Theory:-https://chatgpt.com/share/66f63642-abf8-8006-a1f7-30c687048349