

**Table Of Contents**

<b><u>Chapter No.</u></b>	<b><u>TITLE</u></b>	<b><u>PAGE No.</u></b>
<b>1</b>	<b>INTRODUCTION</b>	
	1.1 Introduction of Project	<b>2</b>
	1.2 Project Analysis	<b>3</b>
<b>2</b>	<b>LITRATURE SURVEY</b>	
	2.1 Introduction to visual studio code	<b>4</b>
	2.2 Introduction to DJANGO Framework	<b>5</b>
	2.3 Introduction to SQLite	<b>6</b>
<b>3</b>	<b>SYSTEM REQUIREMENTS</b>	
	3.1 Hardware and Software Requirement	<b>8-9</b>
<b>4</b>	<b>SYSTEM STUDY</b>	
	4.1 Existing System v/s Proposed System	<b>10</b>
	4.2 Feasibility study	<b>10</b>
	4.3 Technical Feasibility	<b>10</b>
	4.4 Economic Feasibility	<b>11</b>
	4.5 Legal Feasibility	<b>11</b>
	4.6 Operational Feasibility	<b>11</b>
	4.7 Schedule Feasibility	<b>12</b>
	4.8 System Design	<b>12</b>
	4.8.1 Data Flow Diagram	<b>13</b>
	4.8.2 E-R Diagram	<b>15</b>
<b>5</b>	<b>IMPLEMENTATION</b>	<b>19</b>
<b>6</b>	<b>TESTING</b>	<b>55</b>
	6.1 Black box Testing	<b>45</b>
	6.2 White box Testing	<b>45</b>
	6.3 Levels of Testing	<b>56</b>
	6.4 Test Case of Project	<b>57</b>
<b>7</b>	<b>REPORTS</b>	<b>58-59</b>
<b>8</b>	<b>CONCLUSION</b>	<b>60</b>
<b>9</b>	<b>FUTURE ENHANCEMENT</b>	<b>61</b>
<b>10</b>	<b>BIBLIOGRAPHY</b>	<b>62</b>

## **CHAPTER-1**

### **INTRODUCTION**

#### **1.1 Introduction to project**

The Sports management is a module of the University Management System which is a web application built for the internal usage of the faculty and student of that university. A sports management project involves creating a comprehensive and well-structured plan for a specific sports. With the use of pertinent data including course descriptions, required. The web development project also gives students access to a result view of the sports been conducted each semester, the platform is meant to be simple to use and available to all students.

The university wants to enhance the total student experience and foster academic success by offering a centralized and user-friendly platform. The purpose of the sports management project is to demonstrate a students's ability to design effective and engaging sports that align with educational objectives and cater to the needs of their students.

In this Web application, the admin can enter a different form of data that has to be stored. the data entered are related to the sports management which can be edited or deleted by the admin. but the user can only view and print data.

#### **1.2 Project analysis**

This application consists of following modules:

1. Admin module
2. User module

#### **Admin Module**

Admin is able to perform different operations like Create, Update, Delete, and Read on the database of the lesson plan via the frontend provided. There are different lesson plan details that can be added and updated The admin can also view lesson plan details in a tabular format. This makes it easy to keep a tract of all the courses happening in the university.

### **User Module**

Users can only view and print all lesson plans. He may not be able to edit or delete data. With the use of pertinent data including course descriptions, requirements, and credit hours, the project seeks to give users a clear roadmap of the courses they must attend to finish their degree.

## **CHAPTER-2**

### **LITRATURE SURVEY INTRODUCTION TO VISUAL STUDIO CODE 2019**

Microsoft Visual Studio code is an integrated development environment [IDE] from Microsoft. It is used to develop computer programs as well websites, web apps, web services, and mobile apps. Visual studio code uses Microsoft Software development platform such as windows API, windows Forms, windows presentation foundation, windows and Microsoft Silverlight. It can provide both native and manage code. Visual Studio code includes a code editor supporting intelligence [The code completion Component] as well as code refactoring. The integrated debugger works both as source level debugger and machine level debugger, other built in tools Include code profiles, designer for GUI applications, web designer, class Designer and database scheme designer.

Visual Studio code supports 36 different programming languages and allows The code editor and debugger support nearly any programming languages Provided a language specific services exit in built-in language include C++, Visual basic, .Net, C#, F#, Java script, Typescript, xml, XSLT and CSS.

#### **2.1 INTRODUCTION TO VISUAL STUDIO CODE**

##### **IMPROVED SEARCH**

- Quick launch, new search experience faster and effective search result appear dynamically and result can often include key board shortcuts for command.
- Visual Studio 2016 is used to develop Android, iOS, and Windows mobile applications. Using C#, JavaScript and C++ it can build the mobile apps. It has high code reuse and native capabilities. We can test developed mobile apps easily.
- We can install it in a faster way compared to previous Visual Studio code versions. We can install it in a minimum of four minutes. It is a lightweight component.

##### **Some of the new features in visual studio code 2019:**

1. Code editor. Visual Studio (like any other IDE) includes a code editor that supports syntax highlighting and code completion using IntelliSense for variables, functions, methods, loops, and LINQ queries.

2. Debugger.
3. Designer.
4. Other tools.
5. Extensibility.
6. Previous products.
7. Community.
8. Professional.

## **2.2 INTRODUCTION DJANGO**

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Django was invented to meet fast-moving newsroom deadlines, while satisfying the tough requirements of experienced web developers. With Django, you can take web applications from concept to launch in a matter of hours. Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Django was designed to help developers take applications from concept to completion as quickly as possible. Django includes dozens of extras you can use to handle common web development tasks. Django takes care of user authentication, content administration, site maps, RSS feeds, and many more tasks — right out of the box. Django takes security seriously and helps developers avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery and clickjacking. Its user authentication system provides a secure way to manage user accounts and passwords. Some of the busiest sites on the planet use Django's ability to quickly and flexibly scale to meet the heaviest traffic demands.

Companies, organizations and governments have used Django to build all sorts of things — from content management systems to social networks to scientific computing platforms.

### **ARCHITECTURE OF DJANGO:**

Django, a free and open-source web application framework written in Python, offers an alternative to servlets, PHP, and JavaScript for developing the backend of web applications. The Django Software Foundation released the initial version of Django on July 15, 2005. The

recent version, 2.2.7 of the Django framework, was released on November 4, 2019. Now we will learn about Django architecture with MVT. Django's main advantages are making the creation of complicated databases, including web applications, as easy as possible, fast, many components are available implicitly, scalability, and good security. Now, getting into the architecture of Django, it follows MVT.

As mentioned, Django follows the MVT framework for architecture.

- M stands for Model
- V stands for View
- T stands for template

MVT is generally very similar to MVC, a Model, View, and Controller. The difference between MVC and MVT here is that Django itself does the work done by the controller part in the MVC architecture. Django does this work of controller by using templates. To be precise, the template file combines HTML and the Django Template Language (DTL).

### **FEATURES OF DJANGO:**

- Python Web-framework
- SEO Optimized
- High Scalability
- Excellent Documentation
- Versatile in Nature
- Offers High Security
- Provides Rapid Development
- Thoroughly Tested

### **2.3 INTRODUCTION TO SQLite**

SQLite is a database engine written in the C programming language. It is not a standalone app; rather, it is a library that software developers embed in their apps. As such, it belongs to the family of embedded databases. It is the most widely deployed database engine, as it is used by several of the top web browsers, operating systems, mobile phones, and other embedded systems.

Many programming languages have bindings to the SQLite library. It generally follows PostgreSQL syntax but does not enforce type checking by default. This means that one can, for example, insert a string into a column defined as an integer.

### **Design of SQLite**

SQLite was designed to allow the program to be operated without installing a database management system or requiring a database administrator. Unlike client server database management systems, the SQLite engine has no standalone processes with which the application program communicates. Instead, a linker integrates the SQLite library — statically or dynamically into an application program which uses SQLite's functionality through simple function calls, reducing latency in database operations; for simple queries with little concurrency, SQLite performance profits from avoiding the overhead of inter-process communication.

Due to the serverless design, SQLite applications require less configuration than client–server databases. SQLite is called zero-conf because it does not require service management (such as startup scripts) or access control based on GRANT and passwords. Access control is handled by means of file-system permissions given to the database file itself. Databases in client–server systems use file-system permissions that give access to the database files only to the daemon process, which handles its locks internally, allowing concurrent writes from several processes.

SQLite stores the whole database (definitions, tables, indices, and the data itself) as a single cross-platform file on a host machine, allowing several processes or threads to access the same database concurrently. It implements this simple design by locking the database file during writing. Write access may fail with an error code, or it can be retried until a configurable timeout expires. SQLite read operations can be multitasked, though due to the serverless design, writes can only be performed sequentially. This concurrent access restriction does not apply to temporary tables, and it is relaxed in version 3.7 as write-ahead logging (WAL) enables concurrent reads and writes. Since SQLite must rely on file-system locks, it is not the preferred choice for write-intensive deployments.

## **CHAPTER-3**

### **SYSTEM REQUIREMENTS**

#### **3.1 Hardware and Software Requirements:**

##### **Hardware requirements:**

- RAM: 2 GB or Above.
- Hard disk: 256 GB or Above
- Processor: Intel Core i5 or more
- Peripheral Devices : Monitor, Keyboard, Mouse
- System type: 64/32-bit operating system

##### **Software requirements:**

- Front End: HTML, CSS, Bootstrap, JavaScript
- Back End: Python, Django Framework.
- Operating System: Windows 8 or above

### **FUNCTIONAL REQUIREMENTS**

A software requirements specification (SRS) is a description of a software system to be developed. It is modelled after business requirement specification (CONOPS) also known as stakeholder requirements specifications (SRS). The software requirements specification lays out functional and non-functional requirements, and it may include a set of use cases that describe user interaction that the software must provide to the user for perfect interaction.

Software requirements specification establishes the basic for an agreement between customer and supplier on how the software product should function (in a division). Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign. It should also provide a realistic basis for estimating product costs, risks and schedules. Used appropriately, Software requirements specifications can help prevent software project failure. The Software requirements specifications document list sufficient and necessary requirements for the project development. To derive the



requirements, the developer needs to have a clear and thorough understanding of the product under development. This is achieved through detailed and continuous communication with the project team and customer throughout the software development process. The SRS may be one of a contract's deliverable data item descriptions or have the other forms of organizationally-mandated content.

Typically a SRS is written by a technical writer, a system architect, or a software programmer.

### **The specific goals of SRS are as follows:**

- Facilitating reviews
- Describing the scope of work
- Providing a reference to software designers (i.e navigation aids, document structure)
- Providing a framework for testing primary and secondary use cases
- Including features to customer requirements
- Providing a platform for ongoing refinement (via incomplete) Providing a platform for ongoing refinement (via incomplete specs or questions)

## **CHAPTER-4**

### **SYSTEM STUDY**

The output of the system requirements analysis phase can be considered as an input to the system design phase. The architectural description of a System with details about its components and subcomponents is called System Design. The pictorial description of the system, modules, and subsystems gives a proper overview of the system and its design details. Through UML (**Unified Modelling Language**) diagrams we specify and visualize various aspects of a System and its architecture. Security, Reliability, being able to deliver desired output to end-users based on available resources, and that the system is responsible and dynamically changes are some of the important aspects ensured by the System Design.

#### **4.1 Existing system**

Provides the user with a dashboard that they can use to complete their daily tasks and functions related to managing, scheduling, analytics and inventory.

#### **Proposed System**

The proposed system is designed to improve the user can see results, they can download, print results and The main proposed of our system is to reduce paper work and your precious time and keep our sports records more securely in a proper systematic manner.

#### **4.2 Feasibility study**

For the purpose of well coordinate maintains of tool, we have collected the necessary information about id, team name, price, results. This made them manually and arrange them according to various streams. If any corrections or modification is required that is to be also done through manually.

#### **4.3 Technical feasibility**

This assessment is based on an outline design of system requirements, to determine whether the company has the technical expertise to handle completion of the project. At this level, the

concern is whether the proposal is both technically and legally feasible. The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. It is an evaluation of the hardware and software and how it meets the need of the proposed system.

### **4.4 Economic feasibility**

The purpose of the economic feasibility assessment is to determine the positive economic benefits to the organization that the proposed system will provide. It includes identification of all the benefits expected. This assessment typically involves a cost/ benefits analysis.

### **4.5 Legal feasibility**

Determines whether the proposed system conflicts with legal requirements, e.g. a data processing system must compile with the local data protection regulations and if the proposed venture is acceptable.

### **4.6 Operational Feasibility**

Operational feasibility is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. To ensure success, desired operational outcomes must be imparted during design and development. These include such design-dependent parameters such as reliability, maintainability, supportability, usability, disposability, sustainability, affordability and others. These parameters are required to be considered at the early stages of design if desired operational behaviours are to be realized.

## **4.7 Schedule Feasibility**

A project will fail if it takes too long to be completed before it is useful. Typically this means estimating how long the system will take to develop, and if it can be completed in a given time period using some methods like payback period. Schedule feasibility is a measure of how reasonable the project timetable is given.

## **4.8 SYSTEM DESIGN**

### **4.8.1 DATA FLOW DIAGRAM [DFD]:**

A data flow diagram (DFD) is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement. They are often elements of a formal methodology such as the Structured Systems Analysis and Design Method ([SSADM](#)).

#### **Advantages of DFD:**

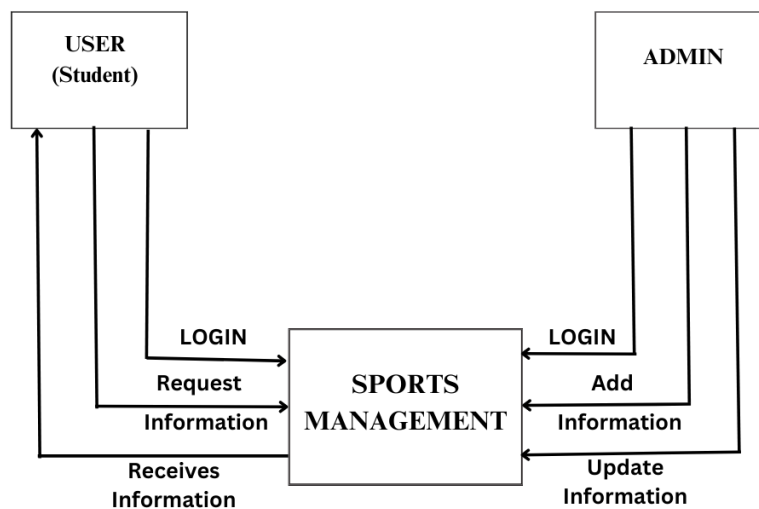
- It helps us to understand the functioning and the limits of a system.
- It is a graphical representation that is very easy to understand as it helps visualize contents.
- Data Flow Diagram represent detailed and well-explained diagram of system components.
- It is used as the part of system documentation file.
- Data Flow Diagrams can be understood by both technical and nontechnical person because they are very easy to understand

#### **Disadvantages of DFD:**

- At times DFD can confuse the programmers regarding the system.
- Data Flow Diagram takes long time to be generated, and many times due to these reasons analysts are denied permission to work on its design.

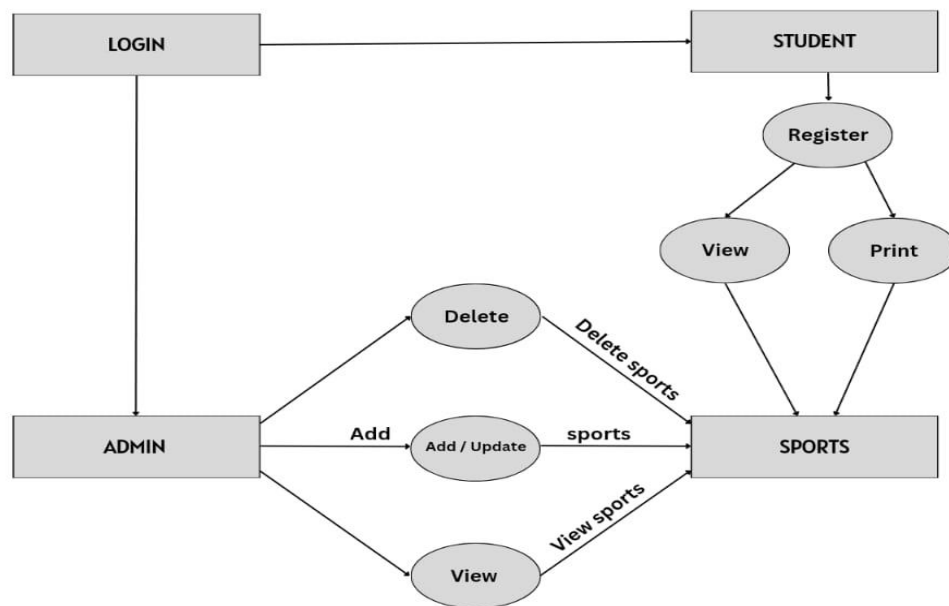
## DATA FLOW DIAGRAM

### LEVEL-0



## DATA FLOW DIAGRAM

### LEVEL-1



### **4.8.2 E-R Diagram**

An Entity-Relationship model (ER Model) describes the structure of a database with the help of a diagram, which is known as ENTITY RELATIONSHIP DIAGRAM (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: Entity set and Relationship set.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In term of DBMS, an entity is a table or attribute of a table in the database, so by showing relationship among the tables and their attributes, ER diagram shows the complete logical structure of a database.

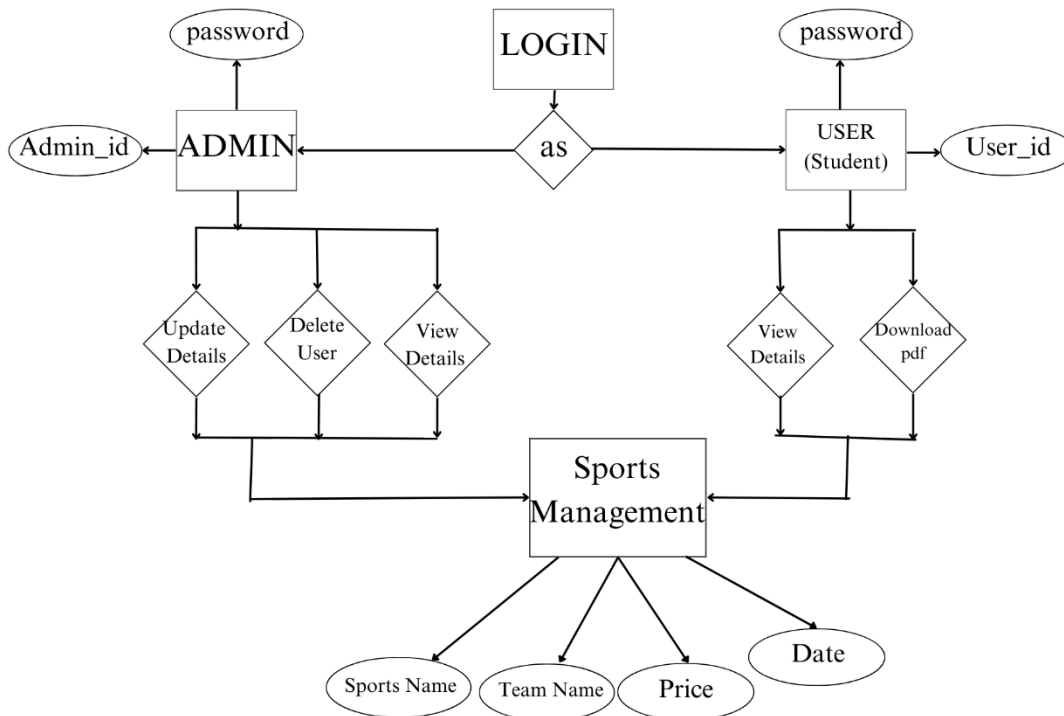
#### **Advantages of ER model:**

- Conceptually it is very simple: ER model is very simple because if we know relationship between entities and attributes, then we can easily draw an ER Diagram.
- Better visual representation: ER model is a diagrammatic representation of any logical structure of database. By seeing ER diagram, we can easily understand relationship among entities and relationship.
- Highly integrated with relational model: ER model can be easily converted into relational model by simply converting ER model into tables.
- Easy conversion to any data model: ER model can be easily converted into another data model like hierarchical data model, network model and so on.

#### **Disadvantages of ER model:**

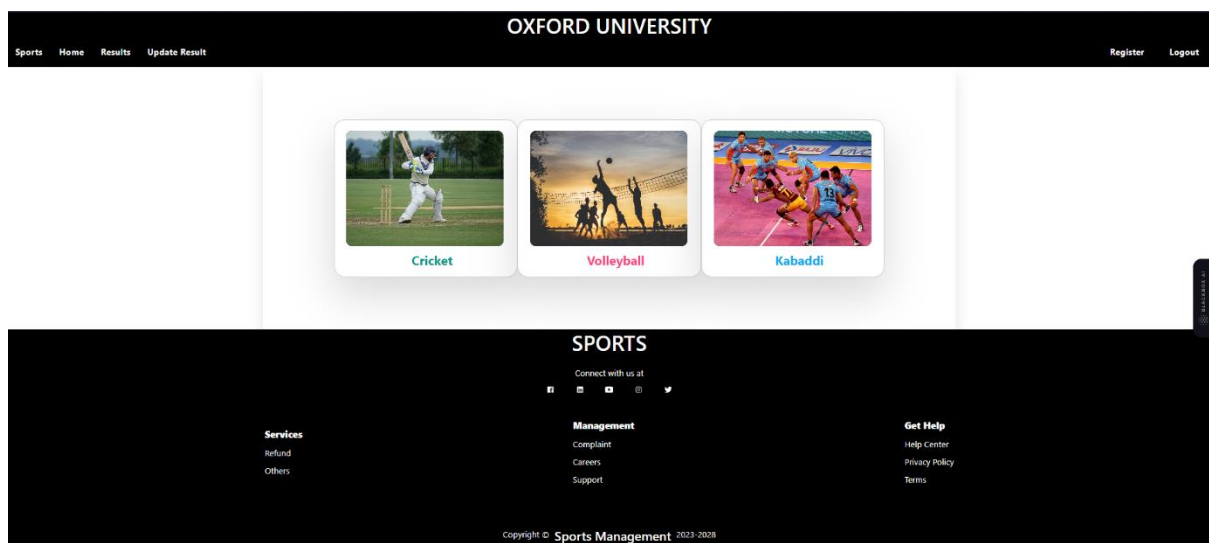
- Loss of information content: Some information be lost or hidden in ER model.
- Limited relationship representation: ER model represents limited relationship as compared to another data models like relational model etc.
- No representation of data manipulation: It is difficult to show data manipulation in ER model.

## Entity Relationship Diagram



### 4.8.3 Forms

#### Home Page





## Register Page

SportsHomeResultsUpdate ResultRegisterLogout

Register page

EVENT NAME

DATE

TIMINGS

TEAM NAME1

TEAM NAME2

PRICE

Register

## Registered Sports

SportsHomeResultsUpdate ResultRegisterLogout

Show all Registered Sports

Id	Event Name	Date	Timings	Team Name1	Team Name2	Price		
6	Volleyball	2023-08-20	10:30	ABC	XYZ	250	Edit	Delete
7	Volleyball	2023-08-20	10:30	ABC	XYZ	250	Edit	Delete
8	Cricket	2023-10-21	12:00	XXXX	YYYY	300	Edit	Delete

Download PDF

## Results page

### OXFORD UNIVERSITY

[Sports](#) [Home](#) [Results](#) [Update Result](#)

[Register](#) [Logout](#)

Show all Results

Id	Event Name	Date	Timings	Team Name1	Team Name2	Price	Result
6	Volleyball	2023-08-20	10:30	ABC	XYZ	250	
7	Volleyball	2023-08-20	10:30	ABC	XYZ	250	Win
8	Cricket	2023-10-21	12:00	XXXX	YYYY	300	

[Download PDF](#)

BUCKEYBOX AI

## Update Results page

### Update Results

SPORTSID

EVENT NAME

DATE

TIMINGS

TEAM NAME1

TEAM NAME2

PRICE

----result----

UPDATE

## **CHAPTER-5**

### **IMPLEMENTATION**

#### **❖ Admin Module**

#### **❖ User Module**

#### **ADMIN MODULE:**

Admin is able to perform different operations like Create, Update, Delete, and Read on the database of the guest talk, seminars etc. via the frontend provided.. The admin can also view events details in a tabular format. This makes it easy to keep a track of all the skill development programs happening in the university.

#### **USER MODULE:**

Users can only view all events. He may not be able to edit or delete data. With the use of persistent data including workshop, speakers, date, timings, the project seeks to give users a updates about skill development programs and college meetup's etc.

## Codes:

### •Views.py:

```
from django.shortcuts import render, redirect
from django.http import HttpResponse
from .models import sportsmanagement5

def showallsports(request):
    std = sportsmanagement5.objects.all()
    return render (request,"sportsmanagement/showallsports.html",{"std":std})

def result(request):
    std = sportsmanagement5.objects.all()
    return render (request,"sportsmanagement/result.html",{"std":std})

def showresult(request):
    std1 = sportsmanagement5.objects.all()
    return render (request,"sportsmanagement/showresult.html",{"allsports":std1})

def deletesports(request, id):
    s = sportsmanagement5.objects.get(pk=id)
    s.delete()
    return redirect('showallsports')

def updatesports(request, id):
    s = sportsmanagement5.objects.get(pk=id)
    return render(request,"sportsmanagement/updatesports.html",{"singlesports":s})
```

```
def updateresult(request, id):  
    s = sportsmanagement5.objects.get(pk=id)  
    return render(request, "sportsmanagement/updateresult.html", {"singlesports":s})
```

```
def doupdatesports(request, id):  
    updatedeventname = request.POST.get("eventname")  
    updateddate      = request.POST.get("date")  
    updatedtimings   = request.POST.get("timings")  
    updatedteamname1 = request.POST.get("teamname1")  
    updatedteamname2 = request.POST.get("teamname2")  
    updatedprice     = request.POST.get("price")
```

```
s = sportsmanagement5.objects.get(pk=id)  
s.eventname = updatedeventname  
s.date      = updateddate  
s.timings   = updatedtimings  
s.teamname1 = updatedteamname1  
s.teamname2 = updatedteamname2  
s.price     = updatedprice  
s.save()  
return redirect("showallsports")
```

```
def doupdateresult(request, id):  
    updatedeventname = request.POST.get("eventname")  
    updateddate      = request.POST.get("date")
```

```
updatedtimings = request.POST.get("timings")
updatedteamname1 = request.POST.get("teamname1")
updatedteamname2 = request.POST.get("teamname2")
updatedprice = request.POST.get("price")
updatedresult = request.POST.get("result")
s2 = sportsmanagement5.objects.get(pk=id)
s2.eventname = updatedeventname
s2.date = updateddate
s2.timings = updatedtimings
s2.teamname1 = updatedteamname1
s2.teamname2 = updatedteamname2
s2.price = updatedprice
s2.result = updatedresult
s2.save()
return redirect("showresult")
```

```
def showsportsregister(request):
```

```
    if request.method == "POST":
```

```
        # get all the values from the form . and store them
```

```
        eventname = request.POST.get("eventname")
```

```
        date = request.POST.get("date")
```

```
        timings = request.POST.get("timings")
```

```
        teamname1 = request.POST.get("teamname1")
```

```
        teamname2 = request.POST.get("teamname2")
```

```
        price = request.POST.get("price")
```

```
# make an object of your Model class, Student
s = sportsmanagement5()
s.eventname = eventname
s.date = date
s.timings = timings
s.teamname1 = teamname1
s.teamname2 = teamname2
s.price = price
s.save()

return redirect("showallsports")

return render(request,"sportsmanagement/showsportsregister.html")
```

```
def Home(request):
    return render(request,"sportsmanagement/Home.html")
```

- Urls.py:

```
from django.urls import path
from . import views

urlpatterns = [
    path("showallsports", views.showallsports, name="showallsports"),
    path("showresult", views.showresult, name="showresult"),
    path("showsportsregister/", views.showsportsregister,
name="showsportsregister"),
```

```
path("deletesports/<int:id>/", views.deletesports, name="deletesports"),
path("updatesports/<int:id>/", views.updatesports, name="updatesports"),
path("doupdatesports/<int:id>/", views.doupdatesports,
name="doupdatesports"),
path("updateresult/<int:id>/", views.updateresult, name="updateresult"),
path("result/", views.result, name="result"),
path("doupdateresult/<int:id>/", views.doupdateresult,
name="doupdateresult"),
path("", views.Home, name="Home"),
]
```

### • Models.py:

```
from django.db import models

# Create your models here.

class sportsmanagement5(models.Model):

    eventname = models.CharField(max_length=100)

    date = models.CharField(max_length=100)

    timings = models.CharField(max_length=100)

    teamname1 = models.CharField(max_length=100)

    teamname2 = models.CharField(max_length=100)

    price = models.CharField(max_length=100)
```



```
result = models.CharField(max_length=100)
```

### •Home.html:

```
{% extends "base.html" %}
{% block maincenter %}

<head>

  <meta charset="UTF-8" />

  <meta http-equiv="X-UA-Compatible" content="IE=edge"/>

  <meta name="viewport" content="width=device-width, initial-scale=1"/>

</head>

<body>

  <header>

    <b><h3>Outdoor Games</h3></b>

    <div class="container">

      <div class="card" style="--clr: #009688">

        <div class="img-box">

          </div>

          <div class="content">

            <h2>Cricket</h2>

            <p>
```

Cricket is a sport that doesn't only have great physical and mental benefits, but it also helps cultivate discipline, responsibility, and teamwork.

</p>

<a href="{% url 'showsportsregister' %}" class="btn btn-primary">REGISTER</a>

</div>

</div>

<div class="card" style="--clr: #FF3E7F">

<div class="img-box">



</div>

<div class="content">

<h2>Volleyball</h2>

<p>

Volleyball is a team sport in which two teams of six players are separated by a net. Each team tries to score points by grounding a ball on the other team's court under organized rules.

</p>

<a href="{% url 'showsportsregister' %}" class="btn btn-primary">REGISTER</a>

</div>

</div>

<div class="card" style="--clr: #03A9F4">

<div class="img-box">



&lt;/div&gt;

&lt;div class="content"&gt;

&lt;h2&gt;Kabaddi&lt;/h2&gt;

&lt;p&gt;

Kabaddi is played by two teams that both consist of twelve players each. However, only seven players per team are allowed on the field of play at any one time.

&lt;/p&gt;

<a href="{% url 'showsportsregister' %}" class="btn btn-primary">REGISTER</a>

&lt;/div&gt;

&lt;/div&gt;

&lt;/div&gt;

&lt;gap&gt;&lt;/gap&gt;

&lt;b&gt;&lt;h3&gt;Indoor Games&lt;/h3&gt;&lt;/b&gt;

&lt;div class="container"&gt;

&lt;div class="card" style="--clr: #FF4500"&gt;

&lt;div class="img-box"&gt;

&lt;img

src="https://media.istockphoto.com/id/155378373/photo/checkmate-move-chess-knight-is-checking-chess-king-chess-board.jpg?s=612x612&w=0&k=20&c=H5zxrmMMr1s-jr0QtNz0QdYC-AEQMrMAIF11g-LiDrM="/>

&lt;/div&gt;

&lt;div class="content"&gt;

&lt;h2&gt;Chess&lt;/h2&gt;

&lt;p&gt;

Chess is a board game for two players, called White and Black, each controlling an army of chess pieces in their color, with the objective to checkmate the opponent's king.

&lt;/p&gt;

<a href="{% url 'showsportsregister' %}" class="btn btn-primary">REGISTER</a>

```
</div>

</div>

<div class="card" style="--clr: #4B0082">

  <div class="img-box">

    </div>

    <div class="content">

      <h2>Carrom</h2>

      <p>

        Carrom is a tabletop game of Indian origin in which players flick discs,
attempting to knock them to the corners of the board. The game is very popular in the Indian
subcontinent.

      </p>

      <a href="{% url 'showsportsregister' %}" class="btn btn-
primary">REGISTER</a>

    </div>

  </div>

<div class="card" style="--clr: #C71585">

  <div class="img-box">

    </div>

    <div class="content">

      <h2>Table Tennis</h2>
```

<p>

Table tennis (also known as ping-pong) is a racket sport derived from tennis but distinguished by its playing surface being atop a stationary table, rather than the court on which players stand.

</p>

<a href="{% url 'showsportsregister' %}" class="btn btn-primary">REGISTER</a>

</div>

</div>

</div>

</div>

</div>

</header>

<section class="footer">

<div class="footer0">

<h1>SPORTS</h1>

</div>

<div class="footer1 ">

Connect with us at<div class="social-media">

<a href="#">

<ion-icon name="logo-facebook"></ion-icon>

</a>

<a href="#">

<ion-icon name="logo-linkedin"></ion-icon>

</a>

<a href="#">

<ion-icon name="logo-youtube"></ion-icon>

</a>

<a href="#">

```
<ion-icon name="logo-instagram"></ion-icon>

</a>

<a href="#">

  <ion-icon name="logo-twitter"></ion-icon>

</a>

</div>

</div>

<div class="footer2">

<div class="services">

  <div class="heading">Services</div>

  <div class="div">Refund</div>

  <div class="div">Others</div>

</div>

<div class="Company">

  <div class="heading">Management</div>

  <div class="div">Complaint</div>

  <div class="div">Careers</div>

  <div class="div">Support</div>

</div>

<div class="Get Help">

  <div class="heading">Get Help</div>

  <div class="div">Help Center</div>

  <div class="div">Privacy Policy</div>

  <div class="div">Terms</div>

</div>

</div>

<div class="footer3">Copyright © <h4>Sports Management</h4> 2023-
2028</div>
```

```
</section>
```

```
</body>
```

```
<style>
```

```
.h3{
```

```
  text-align: center
```

```
}
```

```
.container {
```

```
  position: relative;
```

```
  display: flex;
```

```
  justify-content: center;
```

```
  align-items: center;
```

```
  flex-wrap: wrap;
```

```
  gap: 100px px;
```

```
  padding: 100px 50px;
```

```
  box-shadow: 0 15px 25px rgba(0, 0, 0, 0.1);
```

```
  text-align: center;
```

```
}
```

```
.container .card {
```

```
  position: relative;
```

```
  display: flex;
```

```
  justify-content: center;
```

```
align-items: flex-start;
width: 350px;
max-width: 100%;
height: 300px;
background: white;
border-radius: 20px;
transition: 0.5s;
box-shadow: 0 35px 80px rgba(0, 0, 0, 0.15);
margin: 0 0.3em;
}
```

```
.container .card:hover {
  height: 400px;
}
```

```
.container .card .img-box {
  position: absolute;
  top: 20px;
  right: 25px;
  width: 300px;
  height: 220px;
  background: #000000;
  border-radius: 12px;
  overflow: hidden;
  transition: 0.5s;
}
```

```
.container .card:hover .img-box {
```



```
top: -100px;
left: 25px;
scale: 0.75;
box-shadow: 0 15px 45px rgba(253, 248, 248, 0.953);
}
```

```
.container .card .img-box img {
    position: absolute;
    top: 0;
    left: 0;
    right: 0;
    width: 100%;
    height: 100%;
    object-fit: cover;
}
```

```
.container .card .content {
    position: absolute;
    top: 252px;
    width: 100%;
    height: 35px;
    padding: 0 30px;
    text-align: center;
    align-items: center;
    overflow: hidden;
    transition: 0.5s;
}
```

```
.container .card:hover .content {  
    top: 130px;  
    height: 250px;  
}
```

```
.container .card .content h2 {  
    font-size: 1.5rem;  
    font-weight: 700;  
    color: var(--clr);  
}
```

```
.container .card .content p {  
    color: #335;  
}
```

```
.container .card .content a {  
    position: relative;  
    top: 15px;  
    display: inline-block;  
    padding: 12px 25px;  
    text-decoration: none;  
    background: var(--clr);  
    color: white;  
    font-weight: 500;  
}
```

```
.container .card .content a:hover {  
    opacity: 0.8;
```

```
background-color: black;
color: aqua;
}
```

```
.footer{
display: flex;
flex-direction: column;
background-color: black;
align-items: center;
color:white;
}
```

```
.footer1 {
display: flex;
flex-direction: column;
align-items: center;
color: white;
margin-top: 15px;
}
```

```
.social-media
{
display: flex;
justify-content: center;
color: white;
flex-wrap: wrap;
}
```

```
.social-media a
```

```
{
    color: white;
    margin: 20px;
    border-radius: 5px;
    margin-top: 10px;
    color: white;
}

.social-media a ion-icon
{
    color: white;
    background-color: black;
}

.social-media a:hover ion-icon
{
    color: rgb(29, 52, 204);
    transform: translateY(5px);
}

.footer2 {
display: flex;
width: 100%;
justify-content: space-evenly;
align-items: center;
text-decoration: none;
flex-wrap: wrap;
}

.footer0 {
font-weight: 1200;
transition-duration: 1s;
```

```
}  
.footer0:hover {  
color: rgb(39, 7, 243);  
}  
.footer2 .heading {  
font-weight: 900;  
font-size: 18px;  
}  
.footer3 {  
margin-top: 60px;  
margin-bottom: 20px;  
display: flex;  
flex-wrap: wrap;  
justify-content: center;  
}  
.footer2 .heading:hover {  
color: rgb(74, 7, 243);  
}  
.footer2 .div:hover {  
transform: scale(1.05);  
}  
.footer3 h4 {  
margin: 0 10px;  
}  
.footer2 div {  
margin: 10px;  
}
```

```
* {  
  margin: 0;  
  padding: 0;  
  box-sizing: border-box;  
}  
.ion-icon {  
  width: 50px;  
  height: 30px;  
  background-color: white;  
  color: black;  
}  
.ion-icon:hover {  
  cursor: pointer;  
}
```

</style>

```
<script src="https://unpkg.com/ionicons@4.5.10-0/dist/ionicons.js"></script>
```

```
<script
```

```
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.7/dist/umd/popper.min.js"
```

```
integrity="sha384-
```

```
zYPOMqeu1DAVkJHlqWBUTcbYfZ8osu1Nd6Z89ify25QV9guujx43ITvfi12/QExE"
```

```
crossorigin="anonymous">
```

```
</script>
```

```
{% endblock maincenter %}
```

•Register.html:

```
{% extends "base.html" %}

{% block header %}

<h3 class="text-center">Register page</h3>

{% endblock header %}

{% block maincenter %}

<form class="p-5 border" action="/showsportsregister/" method = "POST">

    {% csrf_token %}

    <div class="form-floating mb-3">

        <input          type="text"          class="form-control"          id="floatingInput"
placeholder="Eventname" name = "eventname" required>

        <label for="floatingInput">EVENT NAME</label>

    </div>

    <div class="form-floating mb-3">

        <input          type="text"          class="form-control"          id="floatingInput"
placeholder="DD/MM/YYYY" onfocus="this.type='date'" onblur="this.type='text'" name =
"date" required>

        <label for="floatingInput">DATE</label>

    </div>

    <div class="form-floating mb-3">

        <input          type="text"          class="form-control"          id="floatingInput"
placeholder="HH/MM"    onfocus="this.type='time'"    onblur="this.type='text'" name =
"timings" required>

        <label for="floatingInput">TIMINGS</label>

    </div>

</form>

</div>

</div>
```

</div>

<div class="form-floating mb-3">

<input type="text" class="form-control" id="floatingInput"  
placeholder="Teamname1" name="teamname1" required>

<label for="floatingInput">TEAM NAME1</label>

</div>

<div class="form-floating mb-3">

<input type="text" class="form-control" id="floatingInput"  
placeholder="Teamname2" name="teamname2" required>

<label for="floatingInput">TEAM NAME2</label>

</div>

<div class="form-floating mb-3">

<input type="text" class="form-control" id="floatingInput" placeholder="Price"  
name="price" required>

<label for="floatingInput">PRICE</label>

</div>

<div class="m-2">

<button type="submit" class="btn btn-primary">Register</button>

</div>

</form>

{% endblock maincenter %}

### •Results.html:



```
{% extends "base.html" %}

{% block header %}

<h3 class="text-decoration-underline">Update Result</h3>

{% endblock header %}


{% block maincenter %}

<style>

.table th{

    background-color: gray;

    color: black;

}

#container

{

    text-align: center;

}

</style>

<div id="container">

<table class="table" >

<thead>

<tr>

<th scope="col">Id</th>

<th scope="col">Event Name</th>

<th scope="col">Date</th>

<th scope="col">Timings</th>
```

```
<th scope="col">Team Name1</th>
<th scope="col">Team Name2</th>
<th scope="col">Price</th>

</tr>
</thead>
{% if std %}
{% for allsports in std %}
<tbody>
<tr>
<th scope="row">{{allsports.id}}</th>
<td>{{allsports.eventname}}</td>
<td>{{allsports.date}}</td>
<td>{{allsports.timings}}</td>
<td>{{allsports.teamname1}}</td>
<td>{{allsports.teamname2}}</td>
<td>{{allsports.price}}</td>
<td><a href = "{% url 'updateresult' allsports.id %}" class = "btn btn-sm btn-outline-
dark">UPDATE RESULT</a></td>
</tr>
{% endfor %}
{% endif %}
</tbody>
</table>
</div>
{% endblock maincenter %}
```

•Showallsports.html:

```
{% extends "base.html" %}

{% block header %}

<h3 class="text-decoration-underline">Show all Registered Sports</h3>

{% endblock header %}


{% block maincenter %}

<style>

.table th{

    background-color: gray;

    color: black;

}

#container

{

    text-align: center;

}

</style>


<script
src="https://cdnjs.cloudflare.com/ajax/libs/html2pdf.js/0.10.1/html2pdf.bundle.min.js"
integrity="sha512-
GsLIZN/3F2ErC5ifS5QtgpiJtWd43JWSulgh7mbzZ8zBps+dvLusV+eNQATqgA/HdeKFVgA5v3S/c
IrLF7Qnlg==" crossorigin="anonymous" referrerpolicy="no-referrer">

</script>
```

```
<div id="container">
<table class="table" >
  <thead>
    <tr>
      <th scope="col">Id</th>
      <th scope="col">Event Name</th>
      <th scope="col">Date</th>
      <th scope="col">Timings</th>
      <th scope="col">Team Name1</th>
      <th scope="col">Team Name2</th>
      <th scope="col">Price</th>

    </tr>
  </thead>
  {% if std %}
  {% for allsports in std %}
  <tbody>
    <tr>
      <th scope="row">{{allsports.id}}</th>
      <td>{{allsports.eventname}}</td>
      <td>{{allsports.date}}</td>
      <td>{{allsports.timings}}</td>
      <td>{{allsports.teamname1}}</td>
      <td>{{allsports.teamname2}}</td>
      <td>{{allsports.price}}</td>

      <td><a href = "{% url 'updatesports' allsports.id %}" class = "btn btn-sm btn-
outline-success">Edit</a></td>
```

```
<td><a href = "{% url 'deletesports' allsports.id %}" class = "btn btn-sm btn-
danger">Delete</a></td>

</tr>

{% endfor %}

{% endif %}

</tbody>

</table>

<button class="download btn btn-sm btn-outline-primary">Download PDF</button>

<button class="btn btn-sm btn-outline-primary" onclick = "{window.print()}">Print
Page</button>

</div>

<script>

let table = document.querySelector(".table");

let btn = document.querySelector(".download");

btn.addEventListener('click', () => {

    html2pdf().from(table).save()

})

</script>

{% endblock maincenter %}
```

### •Showresult.html:

```
{% extends "base.html" %}
```

```
{% block header %}
```

<h3 class="text-decoration-underline">Show all Results</h3>

{% endblock header %}

{% block maincenter %}

<style>

```
.table th{
    background-color: gray;
    color: black;
}
```

#container

```
{
    text-align: center;
}
```

</style>

</style>

<script

```
src="https://cdnjs.cloudflare.com/ajax/libs/html2pdf.js/0.10.1/html2pdf.bundle.min.js"
integrity="sha512-
GsLIZN/3F2ErC5ifS5QtgpiJtWd43JWSulgh7mbzZ8zBps+dvLusV+eNQATqgA/HdeKFVgA5v3S/c
IrLF7Qnlg==" crossorigin="anonymous" referrerpolicy="no-referrer"></script>
```

<div id="container">

<table class="table">

<thead>

<tr>

```
<th scope="col">Id</th>
<th scope="col">Event Name</th>
<th scope="col">Date</th>
<th scope="col">Timings</th>
<th scope="col">Team Name1</th>
<th scope="col">Team Name2</th>
<th scope="col">Price</th>
<th scope="col">Winning Team</th>
</tr>
</thead>
{% if allsports %}
{% for eachsports in allsports %}
<tbody>
<tr>
<th scope="row">{{eachsports.id}}</th>
<td>{{eachsports.eventname}}</td>
<td>{{eachsports.date}}</td>
<td>{{eachsports.timings}}</td>
<td>{{eachsports.teamname1}}</td>
<td>{{eachsports.teamname2}}</td>
<td>{{eachsports.price}}</td>
<td>{{eachsports.result}}</td>
</tr>
</tbody>
{% endfor %}
{% endif %}
</table>

<button class="download btn btn-sm btn-outline-primary">Dowload PDF</button>
```

```
<button class="btn btn-sm btn-outline-primary" onclick = "{window.print()}">Print
Page</button>

</div>

<script>

let table = document.querySelector(".table");

let btn  = document.querySelector(".download");

btn.addEventListener('click', () => {

    html2pdf().from(table).save()

})

</script>

{% endblock maincenter %}
```

### •Showsportsregister.html:

```
{% extends "base.html" %}

{% block header %}

<h3 class="text-center">Register page</h3>

{% endblock header %}

{% block maincenter %}

<form class="p-5 border" action="/showsportsregister/" method = "POST">

    {% csrf_token %}

    <div class="form-floating mb-3">

        <input          type="text"          class="form-control"          id="floatingInput"
placeholder="Eventname" name = "eventname" required>

        <label for="floatingInput">EVENT NAME</label>
```



```
</div>

<div class="form-floating mb-3">

    <input      type="text"      class="form-control"      id="floatingInput"
placeholder="DD/MM/YYYY" onfocus="this.type='date'" onblur="this.type='text'" name =
"date" required>

    <label for="floatingInput">DATE</label>

</div>

<div class="form-floating mb-3">

    <input      type="text"      class="form-control"      id="floatingInput"
placeholder="HH/MM"   onfocus="this.type='time'"   onblur="this.type='text'" name   =
"timings" required>

    <label for="floatingInput">TIMINGS</label>

</div>

<div class="form-floating mb-3">

    <input      type="text"      class="form-control"      id="floatingInput"
placeholder="Teamname1" name = "teamname1" required>

    <label for="floatingInput">TEAM NAME1</label>

</div>

<div class="form-floating mb-3">

    <input      type="text"      class="form-control"      id="floatingInput"
placeholder="Teamname2" name = "teamname2" required>

    <label for="floatingInput">TEAM NAME2</label>

</div>

<div class="form-floating mb-3">

    <input type="text" class="form-control" id="floatingInput" placeholder="Price"
name="price" required>
```

```
<label for="floatingInput">PRICE</label>
</div>

<div class="m-2">
    <button type="submit" class="btn btn-primary">Register</button>
</div>
</form>

{% endblock maincenter %}
```

#### •updateresult.html:

```
{% extends "base.html" %}

{% block header %}
<h3 class="text-center">Update Results</h3>
{% endblock header %}

{% block maincenter %}

    <form class="p-5 border" action="/doupdateresult/{{singlesports.id}}/" method =
"POST">

        {% csrf_token %}

        <div class="form-floating mb-3">

            <input      type="text"      class="form-control"      id="floatingInput"
placeholder="Sportsid" name = "id" required value="{{singlesports.id}}">

            <label for="floatingInput">SPORTSID</label>

        </div>
```

```
<div class="form-floating mb-3">

    <input      type="text"      class="form-control"      id="floatingInput"
placeholder="Eventname"      name      =      "eventname"      required
value="{{singlesports.eventname}}">

    <label for="floatingInput">EVENT NAME</label>

</div>

<div class="form-floating mb-3">

    <input type="text" class="form-control" id="floatingInput" placeholder="Date"
name = "date" required value="{{singlesports.date}}">

    <label for="floatingInput">DATE</label>

</div>

<div class="form-floating mb-3">

    <input      type="text"      class="form-control"      id="floatingInput"
placeholder="Timings" name = "timings" required value="{{singlesports.timings}}">

    <label for="floatingInput">TIMINGS</label>

</div>

<div class="form-floating mb-3">

    <input      type="text"      class="form-control"      id="floatingInput"
placeholder="Teamname1"      name      = "teamname1"      required
value="{{singlesports.teamname}}">

    <label for="floatingInput">TEAM NAME1</label>

</div>

<div class="form-floating mb-3">

    <input      type="text"      class="form-control"      id="floatingInput"
placeholder="Teamname2" name = "teamname2" required value="{{singlesports.course}}">

    <label for="floatingInput">TEAM NAME2</label>

</div>
```

```
<div class="form-floating mb-3">
  <input type="text" class="form-control" id="floatingInput" placeholder="Price"
name="price" required value="{{singlesports.price}}">
  <label for="floatingInput">PRICE</label>
</div>
```

```
<select class="form-select" aria-label="Default select example" name="result">
  <option selected>----Winning Team----</option>
  <option value="TEAM1">Team1</option>
  <option value="TEAM2">Team2</option>
</select>
```

```
<div class="m-2">
  <button type="submit" class = "btn btn-primary">UPDATE</button>
</div>
</form>
```

```
{% endblock maincenter %}
```

### •updatesports.html:

```
{% extends "base.html" %}
{% block header %}
<h3 class="text-center">Update Sports</h3>
{% endblock header %}
```

{% block maincenter %}

```
<form class="p-5 border" action="/doupdatesports/{{singlesports.id}}/" method =  
"POST">
```

```
{% csrf_token %}
```

```
<div class="form-floating mb-3">
```

```
<input type="text" class="form-control" id="floatingInput"  
placeholder="Sportsid" name = "id" required value="{{singlesports.id}}" readonly>
```

```
<label for="floatingInput">SPORTSID</label>
```

```
</div>
```

```
<div class="form-floating mb-3">
```

```
<input type="text" class="form-control" id="floatingInput"  
placeholder="Eventname" name = "eventname" required  
value="{{singlesports.eventname}}">
```

```
<label for="floatingInput">EVENT NAME</label>
```

```
</div>
```

```
<div class="form-floating mb-3">
```

```
<input type="text" class="form-control" id="floatingInput" placeholder="Date"  
name = "date" required value="{{singlesports.date}}">
```

```
<label for="floatingInput">DATE</label>
```

```
</div>
```

```
<div class="form-floating mb-3">
```

```
<input type="text" class="form-control" id="floatingInput"  
placeholder="Timings" name = "timings" required value="{{singlesports.timings}}">
```

```
<label for="floatingInput">TIMINGS</label>
```

```
</div>
```

```
<div class="form-floating mb-3">
```

```
<input      type="text"      class="form-control"      id="floatingInput"
placeholder="Teamname1"      name      ="teamname1"      required
value="{{singlesports.teamname}}">

    <label for="floatingInput">TEAM NAME1</label>

</div>

<div class="form-floating mb-3">

    <input      type="text"      class="form-control"      id="floatingInput"
placeholder="Teamname2" name = "teamname2" required value="{{singlesports.course}}">

    <label for="floatingInput">TEAM NAME2</label>

</div>

<div class="form-floating mb-3">

    <input type="text" class="form-control" id="floatingInput" placeholder="Price"
name="price" required value="{{singlesports.price}}">

    <label for="floatingInput">PRICE</label>

</div>

<div class="m-2">

    <button type="submit" class = "btn btn-success">UPDATE</button>

</div>

</form>

{% endblock maincenter %}
```

## **CHAPTER-6**

### **TESTING**

Testing should be done through the implementation process. Even before and application is installed; it makes sense to verify that the basic platform is capable of achieving its design capabilities. System testing is a critical process. Testing is a process of executing a program with the explicit intention of finding errors that is making the program fail. This helps in finding the bottle neck in the system. Executing a program in a stimulated environment performs testing. The feedback from testing phase generally produces changes in the software to deal with errors and failures that are uncovered.

#### **6.1 BLACK BOX TESTING:**

In black box testing or functional testing test cases are decided. Test cases are decided on the basis of requirements or specifications of the program or module. Black box testing is done in the project to remove errors:

- Incorrect or missing function
- Interface errors.
- Errors in data structure or external database access.
- Behavioural or performance

#### **6.2 WHITE BOX TESTING:**

The White box testing or structural testing performs close operation of procedural details. They test the software logical path by having test cases exercising specific sets of condition and loops. White box testing is done in the project to remove errors:

- All modules path have been exercised at least once.
- Executed all loops at their boundaries and within their operational bounds.

## **6.3 LEVELS OF TESTING**

### **Unit Testing**

This is ideally our first level of testing software. How does it work? Here, specific lines of code, distinct functionalities, and desired procedures are isolated and tested. These lines of code, functionalities, and procedures are termed software units because they are combined to make up the software. They can also be referred to as components of the software.

### **Integration Testing**

This testing level involves combining all the components that make up the software and testing everything as a whole instead of individually as done during unit testing. Also, from this level, tests can be split into functional and non-functional types.

### **System Testing**

System testing has to do with verifying the required operations of the software and its compatibility with operating systems. In other words, we test both the technicalities and the business logic of the software; we run functional tests to check what the various functions of a system do, and non-functional tests to check how those functions work.

### **Acceptance Testing**

This level of software testing is similar to the system testing, but here, the test is carried out by some selected end-users. This is the only software testing stage that is carried out by users. This stage determines if the software is finally ready to be launched to the general public.



## 6.4 TEST CASE OF PROJECT

SI No.	ACTION	INPUT	Expected Output	Actual Output	Test Command
1	Login	Enter valid details	Redirects to home page	Redirected to home page	Pass
2	Enter Valid Details for Admin/user	Admin Id: admin Password: 123	Logs into the home page	Logged in	pass
3	Enter Invalid Log in details for Admin/User	Username: admin Password: admin	Invalid Admin Id or password	Invalid Admin Id or password	Pass
4	Home	Stay on home page	Stay on home page	Stay on home page	Pass
5	Register page	Enter all the Required Details	Adds the data to the sports table in database.	Adds the data to the table in database.	Pass
6	Sports page	Displays the registered sports	Redirects to sports registered page	Redirects to home page	Pass
7	Sports registered table page	Click on Action - edit	Redirected to Edit sports page	Redirected to Edit sports page	Pass
8	Edit sports registered details page	Change the details and click on submit.	Redirected to lesson plans page with changed details.	Redirected to sports page with changed details.	Pass
9	Edit sports details page	Click on Action – Delete	Deletes the selected sports.	Deletes the selected sports.	Pass
10	Results page	Click on	Redirected to User dashboard page	Redirected to User dashboard page	Pass
11	Update Results	Click on Update Results button	Redirected to Results page	Redirected to Results page.	Pass

## CHAPTER-7

### REPORTS

8/2/23, 3:52 PM

Sportsmanagement

OXFORD UNIVERSITY

Show all Registered Sports

	Event						
Id	Name	Date	Timings	Team Name1	Team Name2	Price	
6	Volleyball	2023-08-20	10:30	abc BCA first sem	xyz BSC 1st sem	250	<a href="#">Edit</a> <a href="#">Delete</a>
7	Volleyball	2023-08-20	10:30	ABC	XYZ	250	<a href="#">Edit</a> <a href="#">Delete</a>
8	Cricket	2023-10-21	12:00	XXXX	YYYY	300	<a href="#">Edit</a> <a href="#">Delete</a>
9	Basketball	2023-02-05	12:00	abc BCA first sem	xyz BSC first sem	250	<a href="#">Edit</a> <a href="#">Delete</a>

[Download PDF](#) [Print Page](#)

127.0.0.1:8000/showallsports

1/1

Print

1 page

Destination

Save as PDF

Pages

All

Layout

Portrait


More settings

[Save](#) [Cancel](#)

8/2/23, 9:13 PM

Sportmanagement

OXFORD UNIVERSITY



Show all Results

	Event						Winning
Id	Name	Date	Timings	Team Name1	Team Name2	Price	Team
6	Volleyball	2023-08-20	10:30	abc BCA first sem	xyz BSC 1st sem	250	TEAM1
7	Volleyball	2023-08-20	10:30	ABC	XYZ	250	Win
8	Cricket	2023-10-21	12:00	XXXX	YYYY	300	
9	Basketball	2023-02-05	12:00	abc BCA first sem	xyz BSC first sem	250	TEAM2

[Download PDF](#) [Print Page](#)


127.0.0.1:8000/showresult

1/1

Print

1 page

Destination

 Save as PDF

Pages

All

Layout

Portrait

More settings

Save

Cancel

## **CHAPTER-8**

### **CONCLUSION**

The project sought to develop an Sports management system that provides User/admin to record the details of the student development program. With a user-friendly user interface it becomes easy to record the data and view it.

The admin will have access to all the details in the system, he/she can view the sports results, print, and download pdf etc. Details easily through the data table and also add and update of the same.

There are different Sports details that can be added and updated. The admin can also view results details in a tabular format. This makes it easy to keep a tract of all the sports happening in the university.

Users can only view the sports events going on. He may not be able to edit or delete data.

With a user-friendly interface all the process becomes easy and fast, it is also easy to get the updates regarding the sports programmes.

## **CHAPTER-9**

### **FUTURE ENHANCEMENT**

- ✓ Add more forms and fields depending upon the requirement.
- ✓ More security for the user details/credentials with password encryption.
- ✓ Will update the application more attractively to attract the users.

## **CHAPTER-10**

### **BIBLIOGRAPHY**

Referred sources from Ecoder's tutor , Google and some other sources in Books.

- ❖ Ecoder's tutor
- ❖ [www.w3schools.com](http://www.w3schools.com)
- ❖ [getbootstrap.com](http://getbootstrap.com)
- ❖ GeeksForGreeks
- ❖ Youtube.com

#### **Books:**

- Advanced web Development with Django by –William s
- Visual Studio Code:End-To-End –Bruce Johnson