**A Project report on**

**Cricketers Details Management System**

SUBMITTED BY

Aadhithya R.K-220701002

Aaditya S -220701003

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING IN

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS) THANDALAM

CHENNAI-602105

2023 – 24

**BONAFIDE CERTIFICATE**

Certified that this project report “**Cricketers Details Management System”** is the bonafide work of **“AADHITHYA R.K -220701002, AADITYA S-220701003”**who carried out the project work under my supervision

**Submitted for the Practical Examination held on**

**SIGNATURE SIGNATURE Dr.R.SABITHA Ms.S.VINOTHINI**

Professor Assistant Professor

Computer Science And Engineering Computer Science And Engineering Rajalakshmi Engineering College, Rajalakshmi Engineering College

Chennai - 602 105 Chennai - 602 105

# **ABSTRACT**

# Cricket Players Detail Management System (CPDMS) is a comprehensive software solution designed to streamline the management of cricket players' information across various levels of the sport. The system aims to provide a centralized platform for storing, accessing, and updating essential details related to players, including personal information, performance statistics, medical records, contractual agreements, and more.

# The CPDMS is developed to cater to the needs of different stakeholders in the cricketing ecosystem, including players, coaches, team managers, administrators, and talent scouts. By digitizing and organizing player data, the system enhances efficiency, facilitates data-driven decision-making, and promotes transparency within cricket organizations.

# Key features of the CPDMS include:

# 1. Player Profiles: The system maintains comprehensive profiles for each player, encompassing their personal details, contact information, playing history, achievements, and any relevant documents.

# 2. Performance Tracking: CPDMS enables the recording and analysis of player performance metrics, such as batting averages, bowling figures, fielding statistics, fitness assessments, and match reports. This feature assists coaches and selectors in evaluating player performance and making informed team selections.

# 3. Talent Identification and Scouting: The system supports talent identification and scouting activities by maintaining a database of potential prospects, tracking their progress, and generating reports for talent scouts and talent development programs.

# In summary, Cricket Players Detail Management System (CPDMS) offers a comprehensive solution for managing cricket players' information effectively, promoting data-driven decision-making, and enhancing collaboration within cricket organizations. By leveraging technology to centralize player data and streamline administrative processes, CPDMS contributes to the development and success of cricket at all levels.

**TABLE OF CONTENTS**

**Chapter 1**

**1. Introduction** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1.1 Introduction ………………………………………..………………… 1

1.2 Objective ………………………………………..…………………….1

1.3 Module ……………………………………………………...…..……..1

**Chapter 2**

**2. Survey of Technologies** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

2.1 Software description …………….………….…………………. …….2

2.2 Languages …….……………………...…………………………….….2

2.2.1 HTML ………….…………………...……………………………….2

2.2.2 CSS .………………………………….…………………………..….2

2.2.3 PHP …………………………………………………………………3

2.2.4 MySQL ………………………..…………………………...………..3

2.2.5 Bootstrap…………………………………………….……………….3

2.2.6.Ajax…………………………………...……………………………...3

2.2.7.JavaScript……………………………………..……………………...3

**Chapter 3 3. Requirements and analysis \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

3.1 Requirement Specification …..………………………….…………….. 4

3.2 Hardware and Software Requirements ….……………………………...4

3.3 Architecture Diagram ……………..………………………………..….11

3.4. ER-Diagram …………………………...…………………………..….11

**Chapter 4**

**4. Program code \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

4.1 Code Details and Code Efficiency ………..………………………..…17

**Chapter 5**

**5.Conclusion** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

5.1 conclusion ……………………………………………..………………4

# Cricket players Management System

**INTRODUCTION**

**1.1 OBJECTIVES**:

* The main objective of the project is to design and develop a user friendly-system • Easy to use and an efficient computerized system.
* To develop an accurate and flexible system, it will eliminate data redundancy.
* To study the functioning of cricket players management System.
* To make a software fast in processing, with good user interface.
* To make software with good user interface so that user can change it and it should be used for a long time without error and maintenance.
* To provide synchronized and centralized farmer and seller database.
* Computerization can be helpful as a means of saving time and money.
* To provide better Graphical User Interface (GUI).
* Less chances of information leakage.
* Provides Security to the data by using login and password method.
* To provide immediate storage and retrieval of data and information.
* Improving arrangements for cricket players coordination.
* Reducing paperwork.

**1.2 MODULE:**

-LOGIN PAGE

-PLAYER INFORMATION PAGE

-TICKET BOOKING PAGE

-MATCH MANAGEMENT PAGE

-PLAYER PERFORMANCE ANALYSIS PAGE

**LOGIN PAGE**:

The login page module of our Cricket Players Management System is designed to ensure secure and streamlined access for authorized users. Upon accessing the system, users are prompted to enter their credentials, typically a username and password. The module verifies these credentials against stored user data, granting access only to authenticated users with valid credentials. Through robust authentication mechanisms and error handling, the login page module safeguards against unauthorized access attempts and ensures the confidentiality and integrity of user accounts. Once authenticated, users are granted access to the system's features and functionalities based on their assigned roles and permissions, facilitating efficient management of cricket players and related activities.

**PLAYER INFORMATION PAGE**:

The Player Info Module serves as the central repository for comprehensive player profiles within our Cricket Players Management System. It enables efficient management and organization of player data, allowing administrators to add, edit, and view detailed information about each player. From personal details like name, date of birth, and nationality to cricket-specific attributes such as batting and bowling styles, this module captures all pertinent information. Moreover, it facilitates easy retrieval of player statistics, performance history, and injury records, empowering coaches and team management to make informed decisions. With intuitive search and filtering capabilities, the Player Info Module ensures quick access to relevant player information, promoting effective player management and strategic planning within the cricketing ecosystem.

**TICKET BOOKING PAGE**:

The Ticket Booking Module streamlines the process of reserving seats for cricket matches within our Cricket Players Management System. It offers users a user-friendly interface to browse available matches, select desired seating arrangements, and securely book tickets for themselves and others. Through intuitive filtering options, users can easily find matches based on date, venue, or opponent team. The module also provides real-time updates on seat availability, ensuring users can make informed decisions before finalizing their bookings. With secure payment processing and booking confirmation functionalities, users can seamlessly complete their transactions and receive instant confirmation of their ticket reservations. Overall, the Ticket Booking Module enhances the match-going experience for cricket enthusiasts by simplifying the ticket reservation process and providing a hassle-free way to secure seats for their favorite matches.

**MATCH MANAGEMENT PAGE:**

The Match Management Page is the control center within our Cricket Players Management System, offering administrators a comprehensive platform to oversee and organize cricket matches effectively. From scheduling upcoming matches to managing match details and outcomes, this module provides a centralized hub for all match-related activities. Administrators can easily create, edit, and delete match entries, specifying essential information such as match date, venue, opponent team, and match type. Real-time updates ensure accurate tracking of match statuses and results, empowering administrators to stay on top of the ever-changing cricketing landscape. With intuitive features for generating match schedules, viewing match histories, and analyzing performance data, the Match Management Page facilitates seamless coordination and strategic planning for cricket matches, ultimately enhancing the overall efficiency and success of the cricketing endeavor.

**PLAYER PERFORMANCE ANALYSIS PAGE;**

The Player Performance Analysis Module is a pivotal tool within our Cricket Players Management System, offering in-depth insights into player performance for informed decision-making and strategic planning. This module aggregates and analyzes player statistics from matches, providing comprehensive metrics such as batting averages, bowling figures, strike rates, and more. Through intuitive dashboards and customizable reports, coaches and team management can easily assess player strengths, weaknesses, and trends over time. Additionally, the module offers comparative analysis tools to benchmark individual player performance against teammates and competitors, enabling targeted training and performance improvement strategies. With its robust analytical capabilities, the Player Performance Analysis Module empowers teams to optimize player performance, refine game strategies, and achieve greater success on the cricket field.

**1.3 LIMITATIONS:**

* Time consumption in data entry as the records are to be manually maintained faculties a lot of time.
* Lot of paper work is involved as the records are maintained in the files and registers.
* Storage Requires as files and registers are used the storage space requirement is increased.
* Less Reliable use of papers for storing valuable data information is not at all reliable.
* Aadhar linkage with the official aadhar database has not been done.

**STUDY OF EXISTING SYSTEM**

**CASE STUDY**

The success of any organization such as School of Public Health, University of Ghana hinges on its ability to acquire accurate and timely data about its operations, to manage this data effectively, and to use it to analyze and guide its activities. Integrated student database system offer users (Student, Registrar, HOD) with a unified view of data from multiple sources. To provide a single consistent result for every object represented in these data sources, data fusion is concerned with resolving data inconsistency present in the heterogeneous sources of data. The main objective of this project is to build a rigid and robust integrated student database system that will track and store records of students. This easy-to-use, integrated database application is geared towards reducing time spent on administrative tasks. The system is intended to accept process and generate report accurately and any user can access the system at any point in time provided internet facility is available. The system is also intended to provide better services to users, provide meaningful, consistent, and timely data and information and finally promotes efficiency by converting paper processes to electronic form. The system was developed using technologies such as, HTML, CSS ,JS and MySQL. PYTHON- FLASK, HTML and CSS are used to build the user interface and database was built using MySQ DATABASE DESIGN

**SOFTWARE REQUIREMENTS SPECIFICATION**

**SOFTWARE REQUIREMENTS:**

Frontend- HTML, CSS, Java Script, Bootstrap

Backend-Python flask (Python 3.7) , SQLAlchemy,

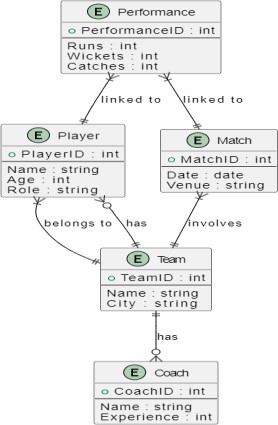
* Operating System: Windows 10
* Google Chrome/Internet Explorer
* XAMPP (Version-3.7)
* Python main editor (user interface): PyCharm Community
* workspace editor: Sublime text 3

**HARDWARE REQUIREMENTS:**

* Computer with a 1.1 GHz or faster processor
* Minimum 2GB of RAM or more
* 2.5 GB of available hard-disk space

**CONCEPTUAL DESIGN:**

**E-R DIAGRAM:**



**3.1 IMPLEMENTATION:**

An "implementation" of Python should be taken to mean a program or environment which provides support for the execution of programs written in the Python language, as represented by the CPython reference implementation.

There have been and are several distinct software packages providing of what we all recognize as Python, although some of those are more like distributions or variants of some existing implementation than a completely new implementation of the language.

**Database**

A Database Management System (DBMS) is computer software designed for the purpose of managing databases, a large set of structured data, and run operations on the data requested by numerous users. Typical examples of DBMSs include Oracle, DB2, Microsoft Access, Microsoft SQL Server, Firebird, PostgreSQL, MySQL, SQLite, FileMaker and Sybase Adaptive Server Enterprise. DBMSs are typically used by Database administrators in the creation of Database systems. Typical examples of DBMS use include accounting, human resources and customer support systems. Originally found only in large companies with the computer hardware needed to support large data sets, DBMSs have more recently emerged as a fairly standard part of any company back office.

A DBMS is a complex set of software programs that controls the organization, storage, management, and retrieval of data in a database. A DBMS includes:

* A modeling language to define the schema of each database hosted in the DBMS, according to the DBMS data model.

The dominant model in use today is the ad hoc one embedded in SQL, despite the objections of purists who believe this model is a corruption of the relational model, since it violates several of its fundamental principles for the sake of practicality and performance. Many DBMSs also support the Open Database Connectivity API that supports a standard way for programmers to access the DBMS

* Data structures (fields, records, files and objects) optimized to deal with very large amounts of data stored on a permanent data storage device (which implies relatively slow access compared to volatile main memory).A database query language and report writer to allow users to interactively interrogate the database, analyze its data and update it according to the users privileges on data.

* Data security prevents unauthorized users from viewing or updating the database. Using passwords, users are allowed access to the entire database or subsets of it called sub schemas. For example, an employee database can contain all the data about an individual employee, but one group of users may be authorized to view only payroll data, while others are allowed access to only work history and student data.
* If the DBMS provides a way to interactively enter and update the database, as well as interrogate it, this capability allows for managing personal databases. However, it may not leave an audit trail of actions or provide the kinds of controls necessary in a multi- user organization. These controls are only available when a set of application

□ A transaction mechanism, that ideally would guarantee the ACID properties, in order to ensure data integrity, despite concurrent user accesses (concurrency control), and faults (fault tolerance).

* It also maintains the integrity of the data in the database.
* The DBMS can maintain the integrity of the database by not allowing more than one user to update the same record at the same time. The DBMS can help prevent duplicate records via unique index constraints; for example, no two customers with the same customer numbers (key fields) can be entered into the database. See ACID properties for more information (Redundancy avoidance).

When a DBMS is used, information systems can be changed much more easily as the organization's information requirements change. to the Organizations may use one kind of DBMS for daily transaction processing and then move the detail onto another computer that uses another DBMS better suited for random inquiries and analysis. Overall systems design decisions are performed by data administrators and systems analysts. Detailed database design is performed by database administrators.

**SQL:**

Structured Query Language (SQL) is the language used to manipulate relational databases.

SQL is tied very closely with the relational model. • In the relational model, data is stored in structures called relations or tables*.*

SQL statements are issued for the purpose of:

• Data definition: Defining tables and structures in the database (DDL used to create, alter and drop schema objects such as tables and indexes)

.

## 4.2 : Stored Procedure

Routine name: proc Type: procedure

Definition: Select \* from register;

## 4.3 : Triggers

It is the special kind of stored procedure that automatically executes when an event occurs in the database. Triggers used :

1: Trigger name: on insert

Table: register

Time: after

Event: insert

INSERT INTO trig VALUES(null,NEW.rid,'Farmer Inserted',NOW())

2: Trigger name: on delete

Table: register

Time: after

Event: delete Definition:

INSERT INTO

trig

VALUES(null,O

LD.rid,'FARME

R

DELETED'NO

## 4.4 : Triggers

It is the special kind of stored procedure that automatically executes when an event occurs in the database. Triggers used :

1: Trigger name: on insert

Table: register

Time: after

Event: insert

INSERT INTO trig VALUES(null,NEW.rid,'Farmer Inserted',NOW())

2: Trigger name: on delete

Table: register

Time: after

Event: delete

3: Trigger name: on update

Table: register

Time: after

Event: update

Definition: INSERT INTO trig VALUES(null,NEW.rid,'FARMER UPDATED',NOW())

# Cricket players Management System

**PROGRAM CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta content="width=device-width, initial-scale=1.0" name="viewport">

<title>{% block title %}

{% endblock title %}</title>

<meta content="" name="description">

<meta content="" name="keywords">

{% block style %}

{% endblock style %}

<link href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,700,700i|Raleway:3 00,400,500,700,800" rel="stylesheet">

<!-- Vendor CSS Files -->

<link href="static/assets/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">

<link href="static/assets/vendor/venobox/venobox.css" rel="stylesheet">

<link href="static/assets/vendor/font-awesome/css/font-awesome.min.css" rel="stylesheet">

<link href="static/assets/vendor/owl.carousel/assets/owl.carousel.min.css" rel="stylesheet"> <link href="static/assets/vendor/aos/aos.css" rel="stylesheet">

<!-- Template Main CSS File -->

<link href="static/assets/css/style.css" rel="stylesheet">

</head>

<body>

<!-- ======= Header ======= -->

<header id="header">

<div class="container">

<div id="logo" class="pull-left">

<a href="/" class="scrollto">S.M.S</a>

</div>

<nav id="nav-menu-container">

<ul class="nav-menu">

<li class="{% block home %}

{% endblock home %}"><a href="/">Home</a></li>

<li><a href="/addstudent">Students</a></li>

<li><a href="/addattendance">Attendance</a></li>

<li><a href="/department">Department</a></li>

<li><a href="/triggers">Records</a></li>

<li><a href="/studentdetails">Student Details</a></li>

<li><a href="/search">Search</a></li>

<li><a href="/about">About</a></li>

{% if current\_user.is\_authenticated %}

<li class="buy-tickets"><a href="">Welcome</a></li>

<li class="buy-tickets"><a href="/logout">Logout</a></li>

{% else %}

<li class="buy-tickets"><a href="/signup">Signin</a></li>

{% endif %}

</ul>

</nav><!-- #nav-menu-container -->

</div>

</header><!-- End Header -->

<!-- ======= Intro Section ======= -->

<section id="intro">

<div class="intro-container" data-aos="zoom-in" data-aos-delay="100">

<h1 class="mb-4 pb-0">STUDENT MANAGEMENT SYSTEM </span> </h1>

<p class="mb-4 pb-<a href="" class="about-btn scrollto">View More</a> </div>

</section><!-- End Intro Section -->

<main id="main">

{% block body %}

0">DBMS Mini Project Using Flask & MYSQL</p>

{% with messages=get\_flashed\_messages(with\_categories=true) %}

{% if messages %}

{% for category, message in messages %}

<div class="alert alert-{{category}} alert-dismissible fade show" role="alert"> {{message}}

</div>

{% endfor %}

{% endif %}

{% endwith %}

{% endblock body %}

<a href="#" class="back-to-top"><i class="fa fa-angle-up"></i></a>

<!-- Vendor JS Files -->

<script src="static/assets/vendor/jquery/jquery.min.js"></script>

<script src="static/assets/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

<script src="static/assets/vendor/jquery.easing/jquery.easing.min.js"></script>

<script src="static/assets/vendor/php-email-form/validate.js"></script>

<script src="static/assets/vendor/venobox/venobox.min.js"></script>

<script src="static/assets/vendor/owl.carousel/owl.carousel.min.js"></script>

<script src="static/assets/vendor/superfish/superfish.min.js"></script>

<script src="static/assets/vendor/hoverIntent/hoverIntent.js"></script>

<script src="static/assets/vendor/aos/aos.js"></script>

<!-- Template Main JS File -->

<script src="static/assets/js/main.js"></script>

</body>

</html>

2.cricket players.html

-- phpMyAdmin SQL Dump

-- version 5.0.2

-- https:[//www.phpmyadmin.net/](http://www.phpmyadmin.net/)

--

-- Host: 127.0.0.1

-- Generation Time: Jan 11, 2021 at 01:03 PM

-- Server version: 10.4.11-MariaDB -- PHP Version: 7.2.29

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

START TRANSACTION;

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- Database: `students`

--

--

-- Table structure for table `attendence`

--

CREATE TABLE `attendence` (

`aid` int(11) NOT NULL,

`rollno` varchar(20) NOT NULL,

`attendance` int(100) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--

-- Dumping data for table `attendence`

--

INSERT INTO `attendence` (`aid`, `rollno`, `attendance`) VALUES

(6, '1ve17cs012', 98);

--

-- Table structure for table `department`

--

CREATE TABLE `department` (

`cid` int(11) NOT NULL,

`branch` varchar(50) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--

-- Dumping data for table `department`

--

INSERT INTO `department` (`cid`, `branch`) VALUES

(2, 'Information Science'),

(3, 'Electronic and Communication'),

(4, 'Electrical & Electronic'),

(5, 'Civil '),

(7, 'computer science'),

(8, 'IOT');

--

-- Table structure for table `student` --

CREATE TABLE `student` (

`id` int(11) NOT NULL,

`rollno` varchar(20) NOT NULL,

`sname` varchar(50) NOT NULL,

`sem` int(20) NOT NULL,

`gender` varchar(50) NOT NULL,

`branch` varchar(50) NOT NULL,

`email` varchar(50) NOT NULL,

`number` varchar(12) NOT NULL,

`address` text NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--

-- Triggers `student`

--

DELIMITER $$

CREATE TRIGGER `DELETE` BEFORE DELETE ON `student` FOR EACH ROW INSERT INTO trig

VALUES(null,OLD.rollno,'STUDENT DELETED',NOW())

$$

DELIMITER ;

DELIMITER $$

CREATE TRIGGER `Insert` AFTER INSERT ON `student` FOR EACH ROW INSERT INTO trig

VALUES(null,NEW.rollno,'STUDENT INSERTED',NOW())

$$

DELIMITER ;

DELIMITER $$

CREATE TRIGGER `UPDATE` AFTER UPDATE ON `student` FOR EACH ROW INSERT INTO trig

VALUES(null,NEW.rollno,'STUDENT UPDATED',NOW())

$$ DELIMITER ;

--

-- Table structure for table `test` --

CREATE TABLE `test` ( `id` int(11) NOT NULL,

`name` varchar(52) NOT NULL,

`email` varchar(50) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--

-- Dumping data for table `test`

--

INSERT INTO `test` (`id`, `name`, `email`) VALUES

(1, 'aaa', 'aaa@gmail.com');

--

-- Table structure for table `trig` --

CREATE TABLE `trig` (

`tid` int(11) NOT NULL,

`rollno` varchar(50) NOT NULL,

`action` varchar(50) NOT NULL,

`timestamp` datetime NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--

-- Dumping data for table `trig`

--

INSERT INTO `trig` (`tid`, `rollno`, `action`, `timestamp`) VALUES

(7, '1ve17cs012', 'STUDENT INSERTED', '2021-01-10 19:19:56'),

(8, '1ve17cs012', 'STUDENT UPDATED', '2021-01-10 19:20:31'), (9, '1ve17cs012', 'STUDENT DELETED', '2021-01-10 19:21:23');

--

-- Table structure for table `user` --

CREATE TABLE `user` (

`id` int(11) NOT NULL,

`username` varchar(50) NOT NULL,

`email` varchar(50) NOT NULL,

`password` varchar(500) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--

-- Dumping data for table `user`

--

INSERT INTO `user` (`id`, `username`, `email`, `password`) VALUES

(4, 'anees', 'anees@gmail.com',

'pbkdf2:sha256:150000$1CSLss89$ef995dfc48121768b2070bfbe7a568871cd56fac85ac7c95a1e645c8806146e9');

--

-- Indexes for dumped tables

--

--

-- Indexes for table `attendence`

--

ALTER TABLE `attendence`

ADD PRIMARY KEY (`aid`);

--

-- Indexes for table `department`

--

ALTER TABLE `department`

ADD PRIMARY KEY (`cid`);

--

-- Indexes for table `student`

--

ALTER TABLE `student`

ADD PRIMARY KEY (`id`);

--

-- Indexes for table `test`

--

ALTER TABLE `test`

ADD PRIMARY KEY (`id`);

--

-- Indexes for table `trig` --

ALTER TABLE `trig`

ADD PRIMARY KEY (`tid`);

--

-- Indexes for table `user`

--

ALTER TABLE `user`

ADD PRIMARY KEY (`id`);

--

-- AUTO\_INCREMENT for dumped tables

--

--

-- AUTO\_INCREMENT for table `attendence`

--

ALTER TABLE `attendence`

MODIFY `aid` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=7;

--

-- AUTO\_INCREMENT for table `department`

--

ALTER TABLE `department`

MODIFY `cid` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=9;

--

-- AUTO\_INCREMENT for table `student`

--

ALTER TABLE `student`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=7;

--

-- AUTO\_INCREMENT for table `test`

--

ALTER TABLE `test`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=2;

--

-- AUTO\_INCREMENT for table `trig`

--

ALTER TABLE `trig`

MODIFY `tid` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=10;

--

-- AUTO\_INCREMENT for table `user`

--

ALTER TABLE `user`

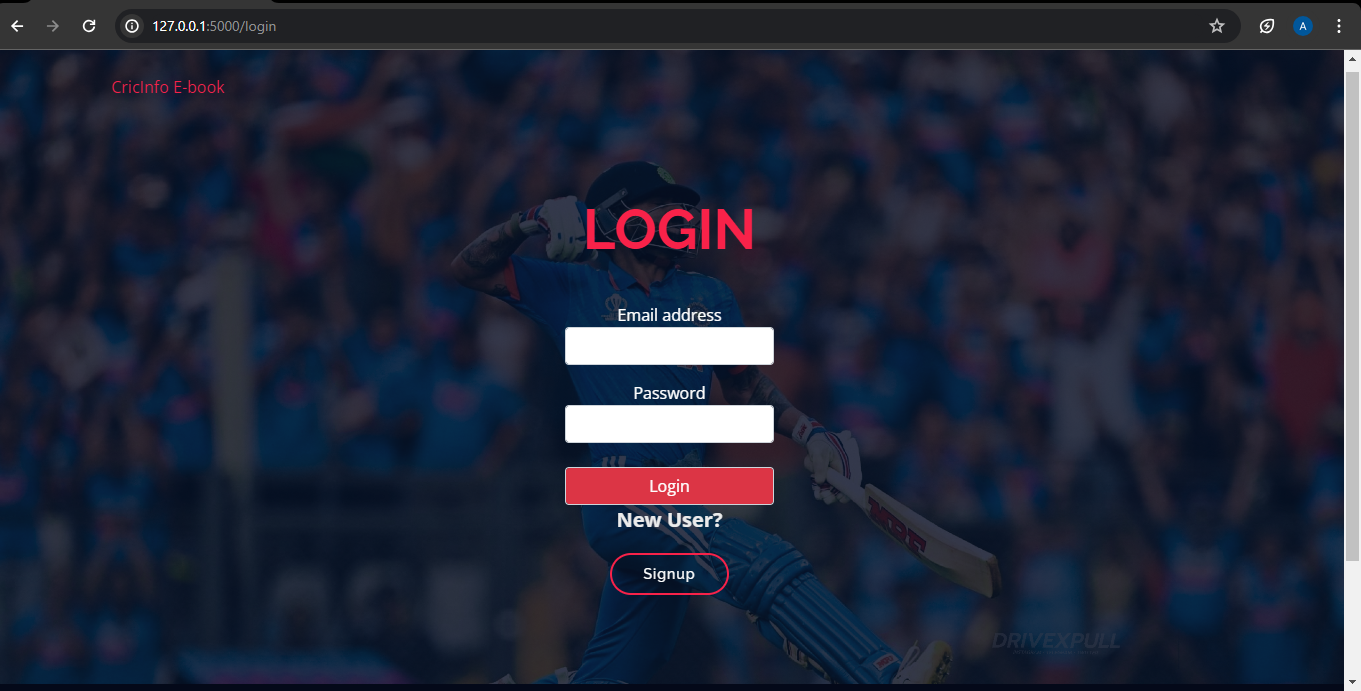
MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=5; COMMIT;

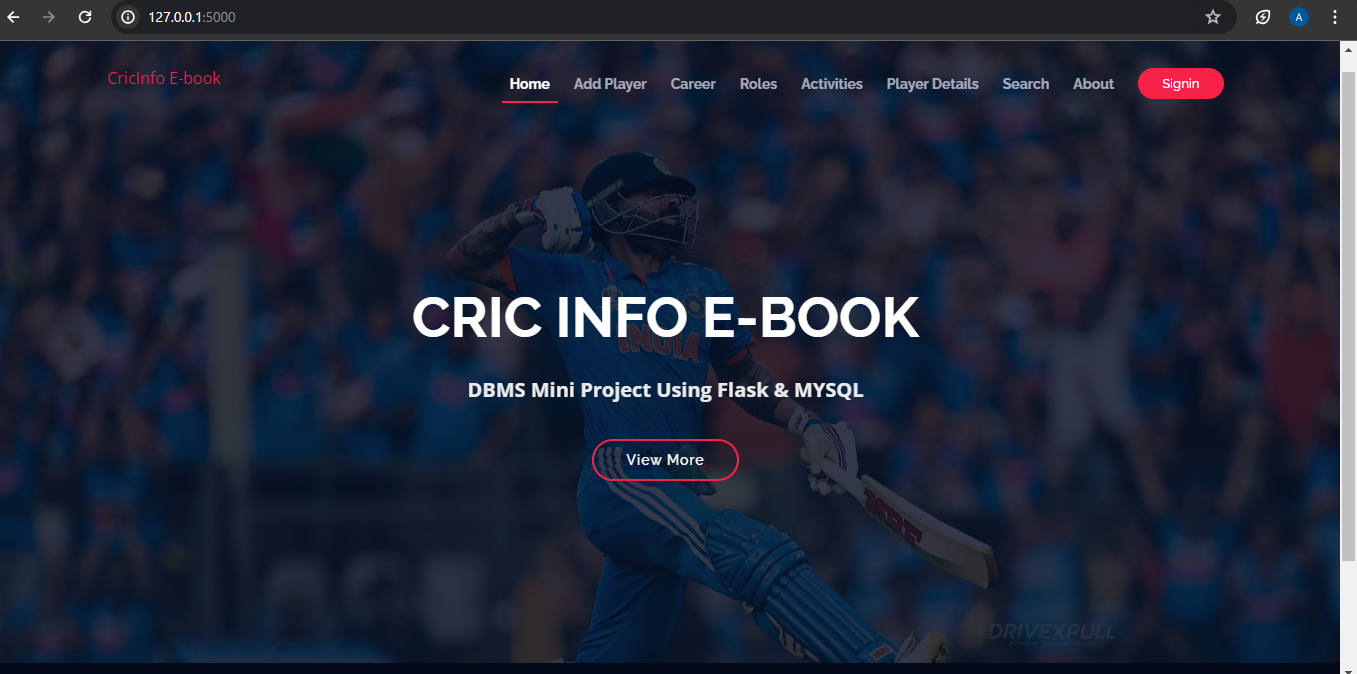
# Students Management System

**USER INTERFACE**

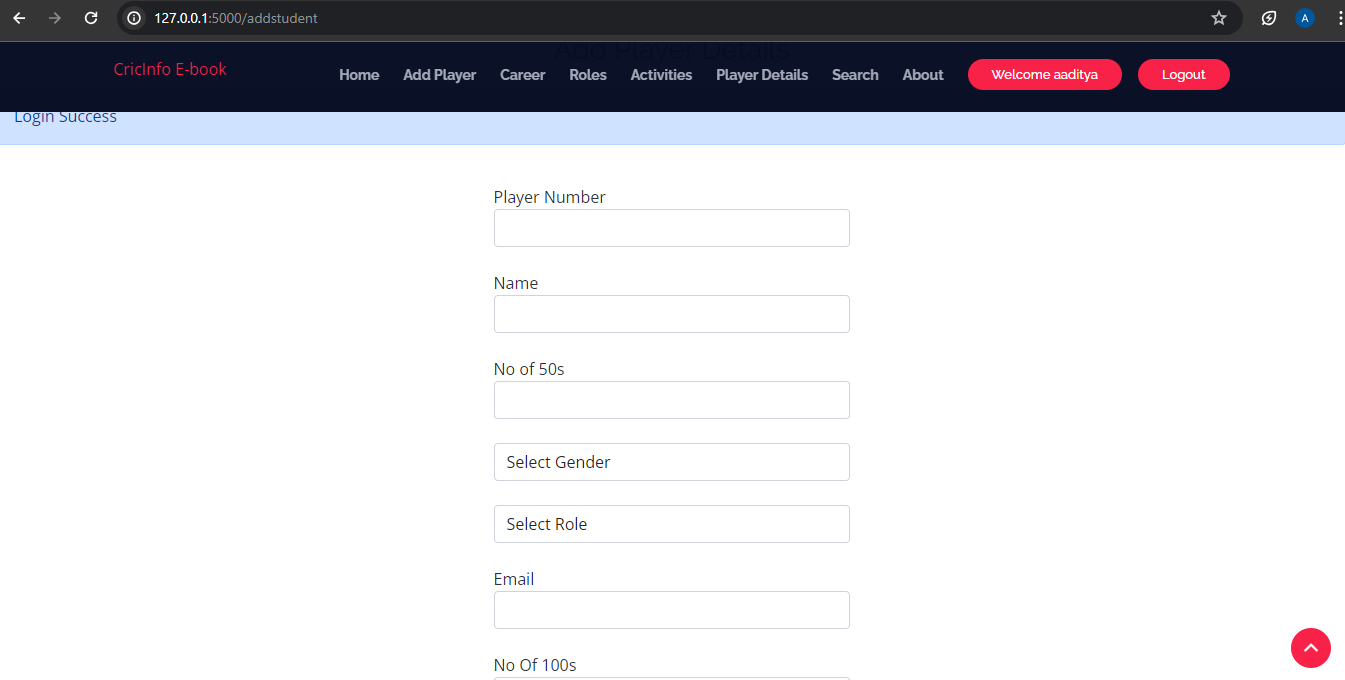
**4.1 SCREEN SHOTS**

**LOGIN PAGE:**



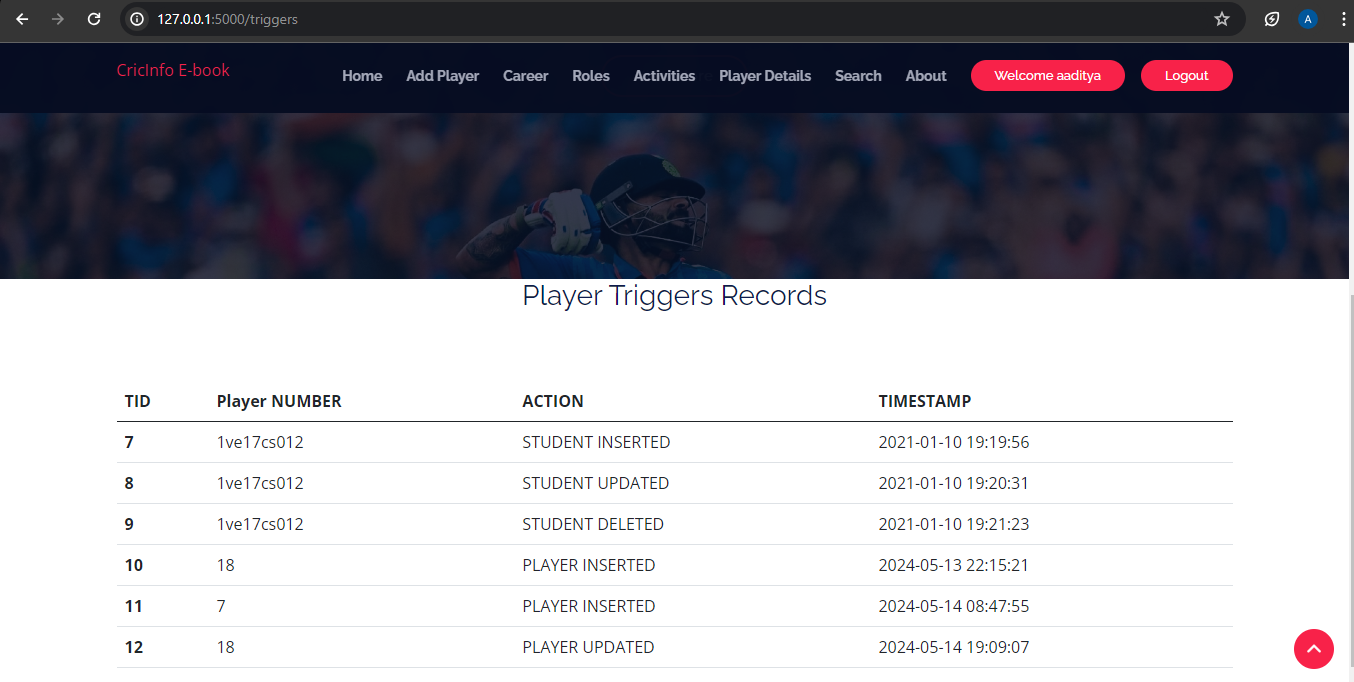


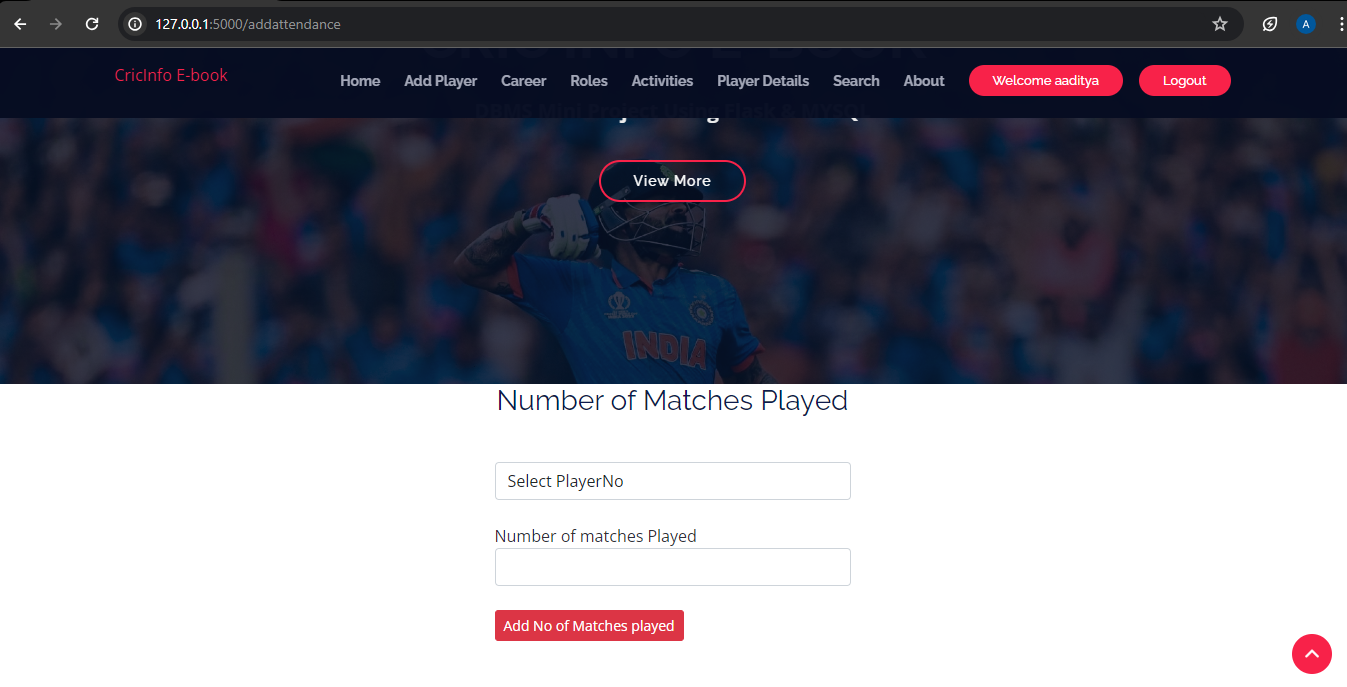
**ADD PLAYER INFO**

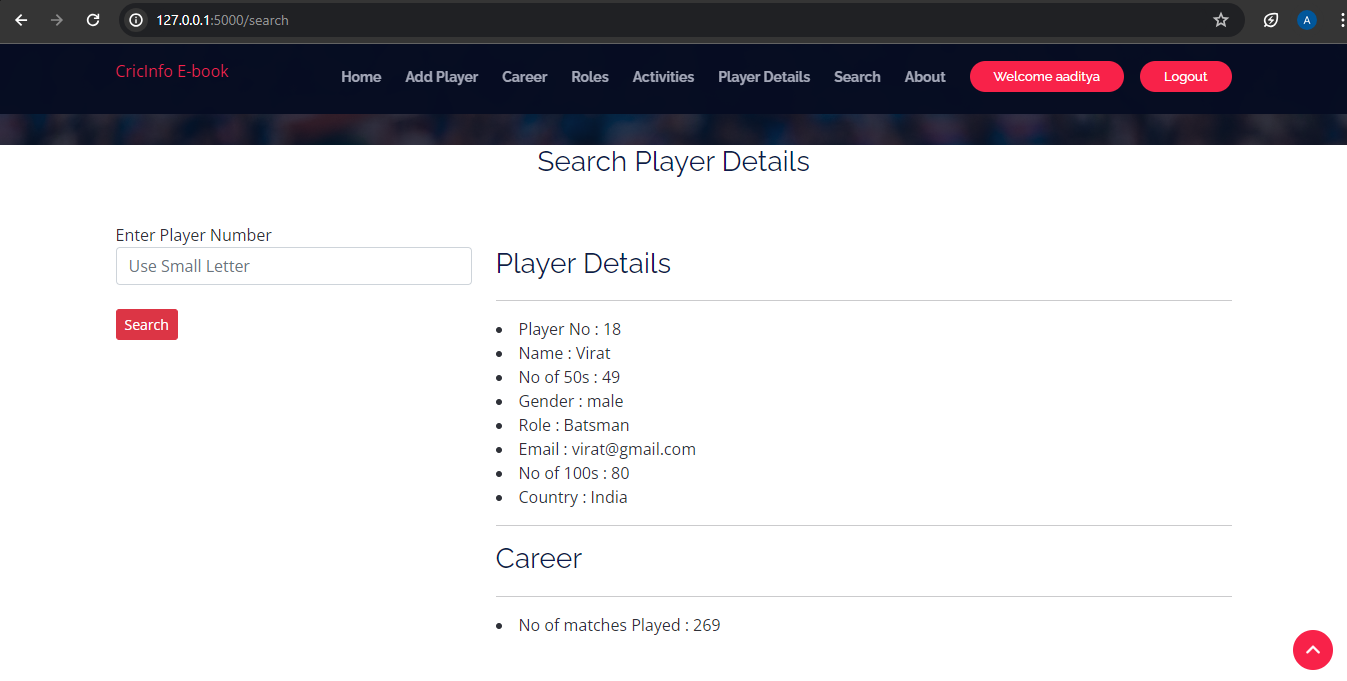




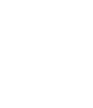
**TRIGGERS RECORDS**







**CONCLUSION**

 In conclusion, the Cricket Players Management System offers a streamlined approach to managing players' data, scheduling, and performance analysis. By leveraging technology, this system aims to enhance the efficiency and effectiveness of cricket team management. With its user-friendly interface and comprehensive features, it stands poised to revolutionize how teams strategize and play to win.

# **FUTURE ENHANCEMENT**

-Enhanced database storage facility

-Enhanced user friendly GUI

-more advanced results systems

-online feedbacks forms

# **REFERENCES**

- https://[www.youtube.com](http://www.youtube.com/)

- https://[www.google.com](http://www.google.com/)

* <http://www>[.getbootstrap.com□](http://www.getbootstrap.com/)